Validation of a LISP Simulator

Albert Cabellos-Aparicio, Jordi Domingo-Pascual
Technical University of Catalonia
Barcelona, Spain
Damien Saucez, Olivier Bonaventure
Université catholique de Louvain
Louvain-La-Neuve, Belgique

1.- Introduction

The 2006 Internet Architecture Board Routing and Addressing Workshop report [1] listed the alarming rate of growth of the Default Free Zone (DFZ) routing table the most important problem facing the Internet today. The scalability issues of the current Internet Routing architecture were previously recognized, and proposals already existed for future Internet architectures. Many of them are centered around the idea of separating the network node's identifier from its topological location. This report however sparked research for a solution, that would enable incremental deployment of the new proposed protocol, changing as little as possible current hardware, software and addressing schemes. The Locator/ID Separation Protocol (LISP) [2] is a promising proposal, pushed by Cisco and academia, that tries to meet the above mentioned goals.

In LISP, each end-host has two addresses: an Endpoint Identifier (EID) that is related with the identity of the node, and a Routing Locator (RLOC) that it is related with the topological position of the network where it is attached to. The EID is only routable inside the domain while the RLOC outside the domain. In order to provide bindings between EIDs and RLOC, LISP employs a special distributed database called Mapping System (MS).

Work on LISP and MS proposals is currently underway in the LISP WG of the IETF. The main protocol and LISP+ALT, one of the mapping systems, already have an experimental implementation for Cisco IOS, deployed in a testbed of about 20 nodes (lisp4.net). An open source implementation called OpenLISP [4] also exists for the FreeBSD kernel, implementing the main protocol and interoperable with the Cisco implementation.

We have developed a LISP simulator (CoreSim). There is a need for such simulation because the above mentioned implementations help validate the good functioning of the proposals, especially at micro level. Nevertheless, to evaluate them at macro level, to see how they scale to the size of the current Internet and if they are feasible, a simulator can be a useful complementary tool. CoreSim is an Internet-scale LISP deployment simulator. It is able to replay a packet trace and simulate the behavior of a LISP Ingress Tunnel Router (ITR) and the associated Mapping Resolver, on top of a topology based on measurements performed by the iPlane infrastructure [7]. It reports mapping lookup latency, the load imposed on each node of the MS and cache performance statistics. The simulator implements LISP-ALT [6] and LISP-DHT.

In this technical report we describe:

- The general architecture of CoreSim,
- An estimator for the latencies that does not report iPlane.
- A validation of the LISP-DHT implementation with the UCL's real DHT implementation

2.- A short overview on CoreSim

CoreSim simulates the operations of a iTR (input Tunnel Router) on an LISP-enabled Internet. On the one hand CoreSim emulates all the operations of an iTR (caching, buffering, lookup and forwarding) at a packet-level and uses an event-based simulation approach. On the other hand CoreSim simulates the performance of a LISP Mapping System (e.g. LISP-ALT and LISP-DHT) at an Internet-scale. This means that CoreSim's assumes that the Internet is running a given MS and thus, it is able to provide realistic values. Due to the large-scale of the Internet, the MS is simulated using a time-based approach.

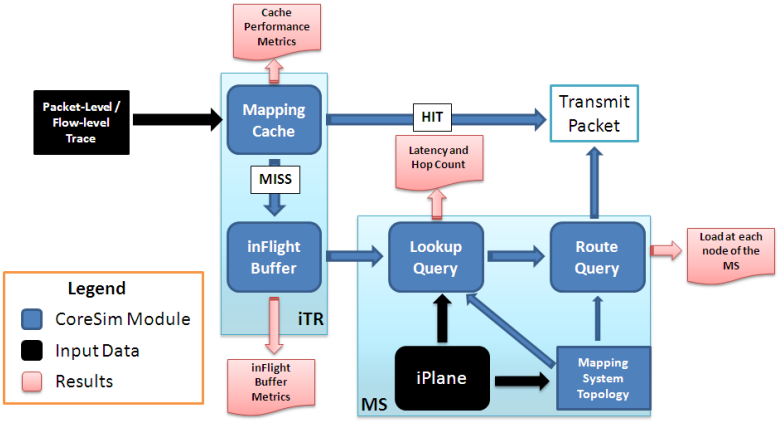

Figure 1.- CoreSim Architecture

Figure 1 presents the main architecture of CoreSim. As the figure shows the simulator is composed mainly by two blocks, namely iTR and MS. The iTR block uses as input a real packet-trace and, for each packet, first it checks if it has the EID-to-RLOC binding stored at the Mapping Cache. If the entry is already cached, the packet is transmitted, on the contrary, if the Mapping Cache produces a miss, then the packet is buffered at an inFlight Buffer. At the same time, the iTR generates a Mapping-Request for a given EID that it is processed by the MS block. Once the Mapping-Request has been resolved the iTR transmits the packet.

The MS block simulates the performance of a Internet-level MS. The topology of the MS is generated using real data, mainly BGP prefixes, which is provided by the iPlane project. This topology is generated only once (for each MS), and feed into the simulator. Based on this topology, the simulator routes the query through the MS-topology and accounts for the lookup latency and hop count. By accounting for the PoPs that have to route the query, CoreSim is able to provide an estimation of the latency using iPlane's measurements. Recall that iPlane has measured the latency between arbitrary pairs of PoPs. Unfortunately iPlane does not provide measurement for all pairs of PoPs. In this case CoreSim fills the gap estimating the latency based on the geographical distance between both PoPs. According to our results, roughly 65% of the latencies are resolved by iPlane, while the remaining ones by the estimator included into CoreSim.

In particular, each CoreSim's module operates as follows:

Mapping Cache: This module is responsible of maintaing the cached EID-to-RLOC mappings. Each Mapping-Reply is stored into the cache. The cache has a limited size and each query has a certain timeout. Further, the cache operates using a FIFO algorithm. This module outputs all the events related with the operation of the mapping cache.
inFlight Buffer: This buffer stores packets that have produced a miss at the Mapping Cache and that trigger a Mapping-Request. While the query is solved the packet is stored at a buffer. Again the module outputs all the events related with these operations.
Lookup Query: This module is responsible of estimating the latency of resolving a query. Each packet that produces a miss triggers a Maping-Requests. The module, taking into account the particular topology of the MS, routes the query among the responsible PoPs and estimates the latency and hop count. In particular it queries the iPlane database for the delay between each pair of PoP involved into resolving the query. As it has been mentioned before, if the iPlane has not measured the delay between a given pair of PoPs it, CoreSim estimates it using geographical information. The module outputs, for each query, the exact path that has followed the Mapping-Request along with the latency.
Route Query: Each query is routed through the MS and this module accounts for the amounts of queries routed by each node (i.e. PoP). Once the simulation finishes the module outputs this value for each PoP.

3.- Validation of tCoreSim

This subsection describes the validation of CoreSim.

3.1.- Latency Estimator

The latency estimator is used to provide values for the cases where iPlane does not provide a real measurement. In order to design the latency estimator we have used a dataset that contains roughly 200k of latencies between arbitrary pairs of PoPs. We have divided this dataset (randomly) into two sets, one for training and designing the estimator and the other one for validation purposes.

In order to design an estimator for the latencies we take into account the information that we can associate to each PoP. In particular we aim to correlate the the geographical distance between two PoPs and the latency between them. The geographical location is obtained using the MaxMind database. This database is open source and has an 99.8% accuracy at country level, 75% accuracy at city level (within a range of 25 miles), 22% accuracy at more than 25 miles, and 3% that the IP is not covered by such database.

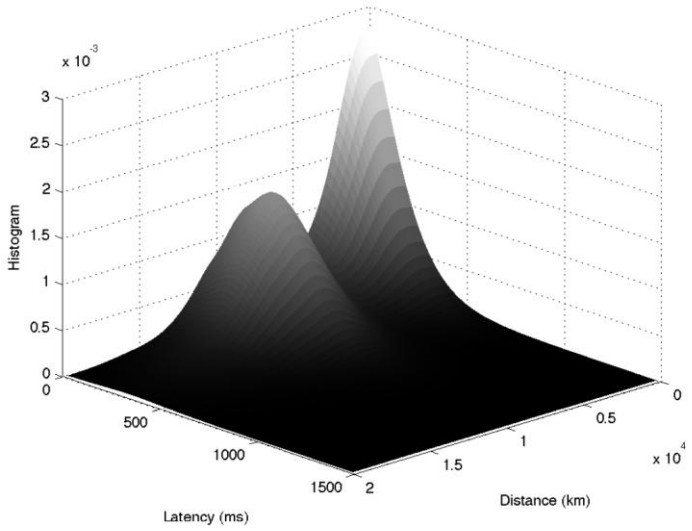

Figure 2.- 3D histogram (latency vs. distance)

First, in figure 2 we show a 3D histogram of the relation between the distance and the latency of the training dataset. As the figure plots mostly, the samples are of PoPs which are at less than 5000km, and between 10.000 and 15.000km.

Now, figure 3 shows a scatter plot of the distance vs. the latency, along with a linear regression (*latency=distance 0.017203+65.968*). As the figure shows there is a linear relation between both metrics, although the fit is not perfect. This is because the distance takes into account the propagation delay, however in a real path other terms affect the end-to-end latency, such as the queuing or processing delay. Unfortunately, at the best of our knowledge, there are no public data traces that include this information (i.e. number of hops) into the regression. Therefore we consider the linear regression as our first estimator for the latency.

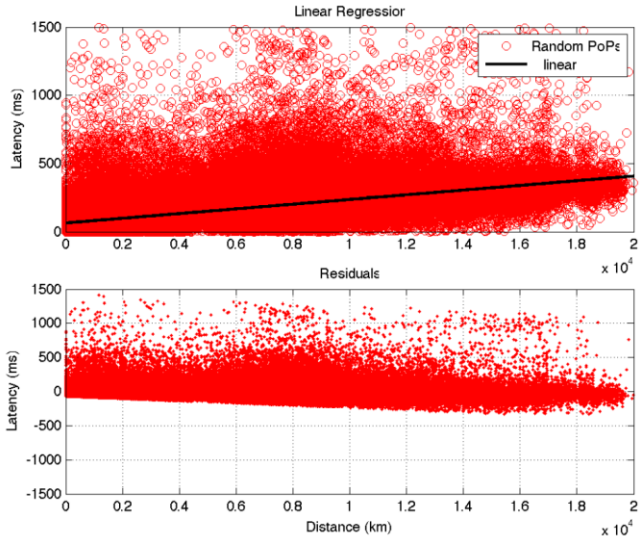

Figure 3.- Scatter Plot and Linear Regression (latency vs. distance)

We also consider another approach for the latency estimator. In this case we bin the pairs of PoPs depending on their distance and we compute the EDF (Empirical Distribution Function) of the latencies. Then, considering this training data, we estimate the delay of a pair of PoPs by first computing the distance between them, and then generating a random number that follows the EDF of the appropriate bin. In particular we consider two bin sizes: *(i)* (0-10km, 10-100km,100-1000km,1000-10.000km,10.000-20.000km) and *(ii)* (0-10km,10-100km,100-500km,500-1000km,1000-2500km,2500-5000km,5000-7500km,7500-10.000km,10.000-15.000km, 15.000-20.000km). With this approach we assume that there is a correlation between the bin size and the latency, for instance routers that are at a range of 10km may have the same amount of hops on their paths. Further, we also consider this approach taking into account the particularities of their location at a continent-level. It is clear that the topology of the Internet is different if we consider North America or Europe, this is because Europe is more densely populated and routers at the same distance may have a larger amount of hops in between.

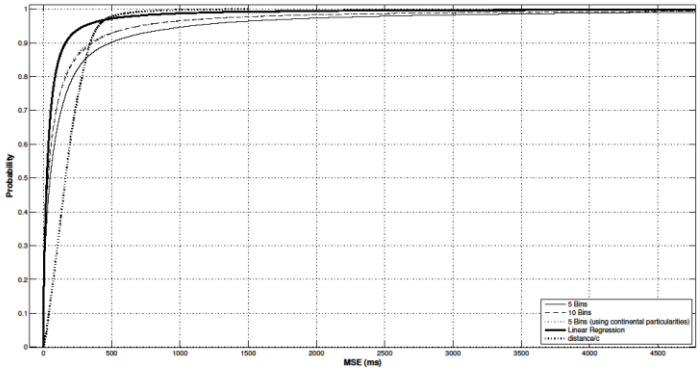

Figure 4.- Error of the estimators

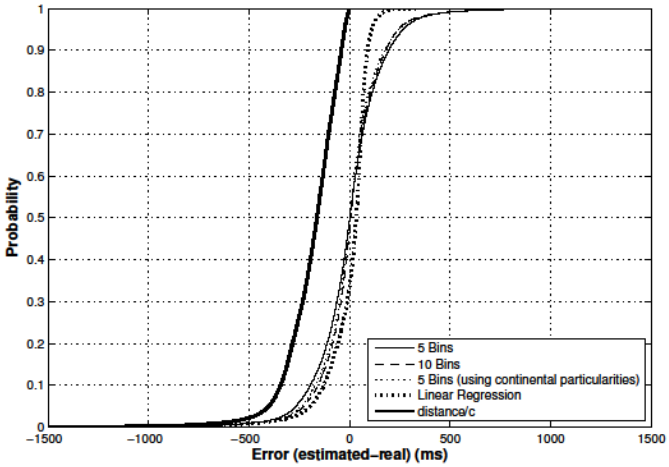

Figure 5.- MSE of the estimators

Figure 4 and 5 show the error and MSE (Mean Squared Error) of the above mentioned estimators. As both figure shows the performance of the estimators is similar, except for the *distance/c* estimator, which always produces under-estimations. This is because it only considers the propagation delay, and assume that end-to-end paths are just a fiber link. Regarding the rest of the estimators the linear regression is slightly more accurate than the rest of them. Further, this estimator is very fast, and will not slow down the simulator. It is important to note that generating random numbers that follow a certain EDF is computationally intensive. Also, as figure 4 the linear regression estimator is not biased, and since we plan to carry out large-scale experiments, with a high amount of repetitions, it will not impact the results.

4.- Validation of CoreSim

We validated our implementation of the Chord protocol for LISP-DHT using the LISP-DHT implementation developed at Universite Catholique de Louvain, which is based on OpenChord 1.0.5 with 50 nodes in steady state. Further, in order to validate the LISP+ALT and LISP-DHT implementations, we used a 35 second packet trace from UPC's Internet link, consisting of 2 million packets and we obtained the following results (figure 5,6 and 7) that were discussed with Prof. Olivier Bonaventure. The figure shows the lookup latency, hopcount and size of the packet buffer of the different MS implemented at CoreSim.

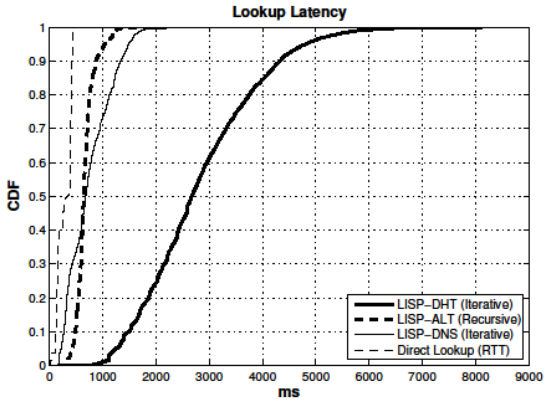

Figure 5.- Lookup Latency

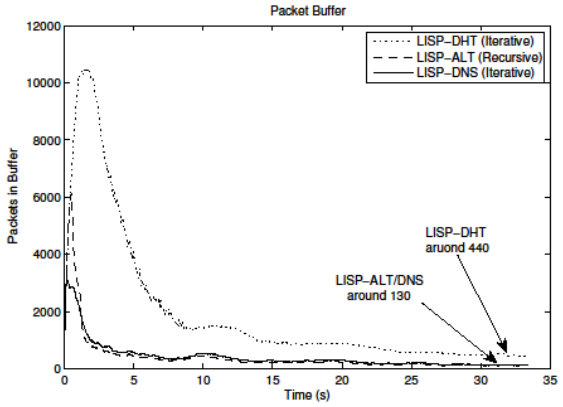

Figure 6.- Packet Buffer

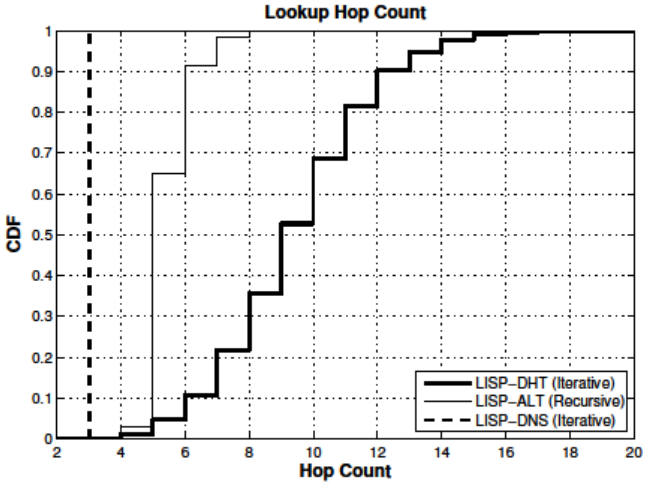

Figure 7.- Hop-Count

5.- Future Work

Researchers from the host and home institute are preparing a joint paper resulting from the STSM. The paper describes a qualitative and qualitative analysis of the performance of the different Mapping Systems.

References

[1] D. Meyer et al. “Report from the IAB Workshop on Routing and Addressing” RFC 4984, Sept. 2007.
[2] Farinacci, D., "Locator/ID Separation Protocol (LISP)", draft-farinacci-lisp-05 (work in progress), November 2007.
[3] Luigi Iannone and Olivier Bonaventure. On the Cost of Caching Locator/ID Mappings. December 2007. 3rd Annual CoNEXT Conference.
[4] OpenLISP (online) http://inl.info.ucl.ac.be/softwares/openlisp
[5] L. Mathy et al. “LISP-DHT: Towards a DHT to map identifiers onto locators”, draft-mathy-lisp dht-00 (work in progress), February 2008
[6] Farinacci, D., "LISP Alternative Topology (LISP-ALT)", draft-fuller-lisp-alt-01 (work in progress), November 2007.
[7] Harsha V. Madhyastha, Thomas Anderson, Arvind Krishnamurthy, Neil Spring and Arun Venkataramani “A Structural Approach to Latency Prediction”, in Proc. of IMC 2006