The Impact of Address-Assigning on Network Routing Paths

Felipe A. Moura Miranda, Carlos A. dos Reis Filho and Rogério Esteves Salustiano
School of Electrical and Computer Engineering (FEEC)
State University of Campinas (UNICAMP)
Campinas, Brazil
{fmiranda, carlos_reis, rsalusti}@lpm.fee.unicamp.br

***Abstract*— The use of hierarchical identifiers as a mechanism of network addressing has proven to be inadequate as the Internet expands. Since the size of border gateway routing tables is in direct proportion with the growth of the Internet and with the increasing number of accesses, the latency time increases at a higher rate and processing capability is steadily more demanding. Therefore, a more efficient routing mechanism is unquestionably required for the future of the Internet. Peer-to-peer networks brought about a novel routing mechanism, based upon a cooperative behavior, which has shown remarkable results. On peer-to-peer networks, each node retains only a limited number of identifiers in a structure called DHT (Distributed Hash Table), which requires less memory as well as reduces the searching time. There is a key difference between the current Internet routing mechanism and existing peer-to-peer networks: while in the first the identifiers follow a hierarchical addressing, in the second the addressing is flat. By providing the flat nature of the second an appropriate ruling, the routing mechanism can be significantly improved. This paper addresses this particular problem by showing, through simulations, how address-assigning heuristics affect the number of hops on a routing path. It contributes by providing elements that can be used to establish criteria for choosing the appropriate heuristic.**

Keywords- Internet, Routing, Address-assigning heuristic

I. INTRODUCTION

The Internet evolution is marked by many changes on its structure, dimension and purposes [1] [2]. Since its beginning in 1969, Internet has experienced an outstanding success and development. Within a very short time, the Internet evolved from an specialized military communication tool into an omnipresent virtual road, which provides mankind countless possibilities of interactions. Technically, it implements a network of networks with worldwide dimensions using many varieties of backbones, devices and structures. In the 90's, many problems related with the Internet growth were revealed. Its colossal size, the new devices that can connect to it and the new necessities of the users revealed some fragile points of its architecture and of the TCP/IP model [1] [2] [3].

Instead of pursuing solutions to fix the Internet problems known at that moment, a paradigm called overlay networks, which was seldom used, bloomed as a means for developing new applications. These overlay networks [4] brought about some interesting mechanisms targeting to create abstractions of the Internet structure and topology making it easier for programmers to use it as a development substrate.

Peer-to-peer networks are an interesting outcome of this abstraction of the Internet structure, by this feature, peer-to-peer networks can implements its own topology, addressing structures and routing mechanisms. Some of the existing peer-to-peer networks use flat labels [6][7][8] to identify both services and nodes. Another property is the implementation of routing based on a cooperative approach among the network nodes. In cooperative routing there is no need for a node to store information about a large number of other nodes as it is done in BGP (Border Gateway Protocol) [9]. Instead, the cooperative routing uses a chain of short lists, called DHT (Distributed Hash Tables), that point to nodes, thus the processing burden per node is much smaller.

In order to establish a cooperation relationship between nodes toward accomplishing an aimed connection three protocols are normally used: Pastry [7], VRR [8] and Chord [6]. All of these protocols use a table lookup mechanism to find the next node to be included in the route. The two first protocols adopt a criterion that takes into account the physical and virtual proximity of the nodes to build the table and the elected members are called neighbors. The third protocol, called Chord differs from the two others in the way it chooses the members of the DHT. Instead of using the criterion of physical or virtual proximity, the members of the DHT are chosen with basis on a proprietary algorithm, which requires information of how the nodes identifiers are distributed as one of its arguments. The name given to the table members is *successor*. Therefore, it is to be expected that any change in the distribution of the node identifiers have an impact on the performance of the network. This paper addresses an evaluation, by means of simulations, of how the routing paths of a network are affected by changes in the distributions of its nodes identifiers.

II. CHORD

Chord [6] is a peer-to-peer DHT-based lookup protocol. According to [6], Chord provides support for just one operation: given a key, it maps a key onto a specific node. It permits that a distributed set of participants can agree on a single node as a *rendezvous* point for a given key. But, in order to reach this only goal, Chord has some interesting structures and mechanisms.

This work has been supported by Brazilian Agency for Personnel Development (CAPES - Brazil) and National Council for Scientific and Technological Development (CNPq - Brazil).

A. Chord Ring

Chord allocates both key and node identifiers in an *m*-bit circular space, called *Chord ring*. These identifiers are arranged in the Chord ring on an increasing clockwise order, from 0 to $2^m - 1$.

B. Successors

For a node in a Chord ring, the *successors* are its partners for achieving some goals in the network. A node can store routing information regarding a small number of other nodes because of the cooperative behavior between a node and its *successors*. The routing on a Chord network is made by the cooperative relationship of a node and its *successors*. Therefore, a node heavily depends on some successors in order to reach other nodes through a short routing path. The complete algorithm used to find the *successors* of a given node is described on [6].

C. Finger Table

Chord uses a structure called *finger table* to store routing information about other nodes in a Chord network. This structure keeps *m* entries to store routing information for the other nodes in the *Chord ring*, these nodes that have their routing information stored in the finger table are called *successors*. Differently from other routing tables, a finger table always preserves the same number of entries in its structure.

D. Lookup and Routing

A lookup operation depends directly of the routing information stored in the finger table. The next *hop* of a package exchange is always a *successor* of the node, following the clockwise movement. The selection of the *next hop* of a lookup operation is made by analyzing the *interval* field of the finger table of each traversed node in the routing path. According to [6], on a network with *N* nodes, lookup operations can be resolved traversing at most $O(\log N)$ nodes.

E. Identifiers

Chord has no constraints about the structure of the identifiers it looks up. Its identifier space is completely flat. Node identifiers are usually generated by a consistent hashing of their IP addresses [10]. Hereinafter, in this work the term "*node identifier*" will be changed by "*virtual address*" since it is a more suitable term.

III. Simulations Details

To analyze how the use of different types of addressing heuristics would actuate on the length of the routing paths of a small Chord network, a simulator was built. This simulator was implemented using the *Java* language [13] due to its *portability*, which allows the simulator to run on different operating systems, and for its enormous quantity of classes, which gives the possibility to expand the features of the simulator, and the analysis that it givens, in an easy way [11].

Another important *Java* feature for the simulator was its *BigInteger* class [12]. *BigInteger* objects can handle huge integer numbers. Without the *BigInteger* class features, any *Java* program would not be able to compute numbers bigger than $2^{63} - 1$, feature that will be very important for possible future works with longer virtual addresses.

A. Chord Implementation

In order to make the initial analysis, some characteristics presented in *Chord* protocol were used in the simulator: its model of *finger tables*; its simple and reliable *lookup* algorithm and its unrestricted structure for *virtual addressing*.

1) The Finger Table: Following Chord protocol, the quantity of available entries in the finger table of each simulated node is the same as the number of bits used in the keys/nodes identifiers of the simulated network. As IPv4, which is the current most used protocol in computers networks, has addresses of 32 bits, the length of the identifiers used by the simulated nodes had 32 bits too, implying on finger tables with 32 entries, permitting that a single simulated node stores routing information of up to 32 different nodes.

2) The Lookup Algorithm: The lookup was made following the clockwise movement in the Chord ring, from the initial virtual node to the target node, where any node between the sender and the target was responsible for find the next hop of the routing path. At each node traversed, the next hop of the path was dinamicaly calculated. According with [6], the Chord protocol can be implemented either in a recursive or in an iterative style, on this work, it was simulated using the recursive style.

3) Virtual Addressing: Since the long length of the virtual addresses, which uses 32 bits, and the unconstrained key/node identifier (virtual address) structure of Chord, the quantity of different virtual addresses that can be generated is larger than 4 billion. In order to work with virtual addresses, BigInteger objects were used, permitting that all the range of different virtual node identifiers used on the simulations (0 to 2^{32}) could be computed.

B. Routing Path Length

In order to calculate the length of the routing paths of a single node, all the routing paths linking this node to all the others were traced. The length of each routing path was determined by the quantity of *hops* made until reach the target node. After counting how many *hops* there were on each routing path, they were organized by their own length. In order to calculate the length of the routing paths of all simulated networks, these procedures were repeated until all the nodes were selected as the initial point of the routing paths.

Since *Chord* has no constraints about its *virtual addresses* structures, all the *virtual addresses* were made using the same structure: integer numbers using the *BigInteger objects*.

IV. Experiments and Results

In order to make an initial analysis of how the routing path length would be affected by the chosen heuristic that assigns the virtual addresses on the network, three different heuristics (*Contiguous*, *Fibonacci Sequence* and *Exponential Growth*) for the virtual addresses distribution were selected. As a kind of basis for comparison with all the other results, simulations of how the network would behave if all the virtual addresses were distributed randomly were made.

As an initial work, just small networks were simulated. The main reason for it was that one of the chosen heuristics, the *Exponential Growth*, only permits 32 different addresses using 32 bits, which is the number of bits on an IPv4 identifier.

The statistics presented at all the scenarios were made analyzing the number of hops on each routing path linking every single node to all the other nodes in the entire network.

A. Scenario I – Random Virtual Addressing

On this scenario, all the random virtual addresses used in the simulations were created by the method *randomAddressCreator()*. This method creates pseudorandom virtual addresses, uniformly distributed between 0 and 2^{32}, Fig. 1 shows an example of a network generated on this scenario. The value was limited to 2^{32} because IPv4, which is the currently most used network protocol, uses 32 bits for location and identification of the nodes.

Aiming for results with more reliable values, every scenario in the random virtual addressing was simulated 100 times, where each simulation was made generating a new pseudorandom virtual addressing distribution for all nodes in the network. After all this simulations, an arithmetic mean was calculated with the values taken from each one of the simulations.

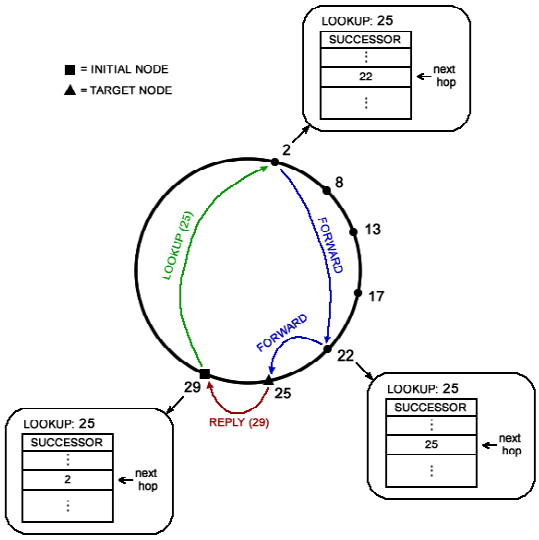

Figure 1. Example of a network generated on Scenario I

The results showed in Fig. 2 demonstrate that the random virtual addressing does not grant a uniform distribution between the routing path cases. At the simulated networks, most destinations would be reached using routing paths with one or two intermediary nodes between the initial node and the target node. But with the growth of the simulated network, there were a spreading of cases, where even results with six intermediary nodes were obtained.

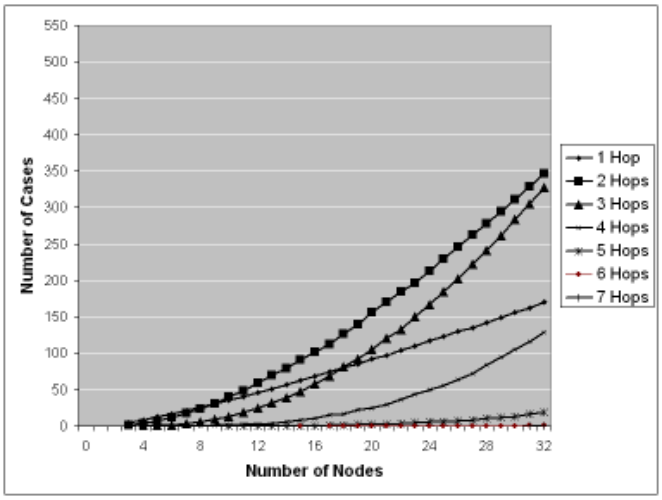

Figure 2. Random Virtual Addressing

B. Scenario II – Contiguous Virtual Addressing

On this scenario, all the virtual addresses were distributed contiguously on the initial part of the *Chord* ring, in a way that the nodes were virtually adjoined to each other (Fig. 3).

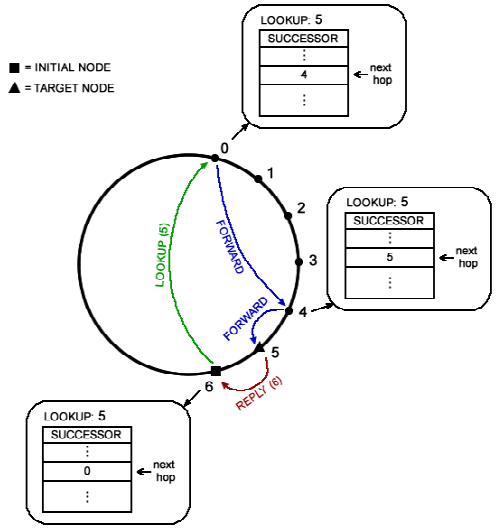

Figure 3. Example of a network with contiguous virtual addresses

On Fig. 4, the results of the simulations showed that the *contiguous virtual addressing* did not give great changes on the routing paths length. The results obtained by the *contiguous virtual addressing* were similar to the results obtained by the *random virtual addressing* simulations.

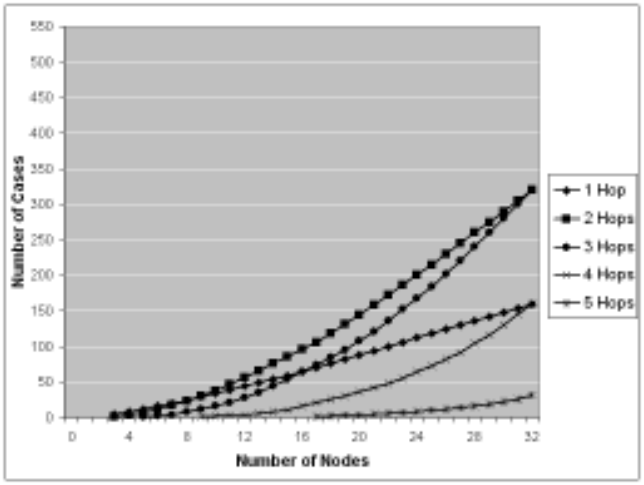

Figure 4. Contiguous Virtual Addressing

Similarly to the *random virtual addressing* simulations, most destinations would be reached passing by up to two intermediary nodes. But with this *contiguous* heuristic, the quantity of hops on the routing paths was more folded than in the *random* heuristic. The worst cases using this heuristic had only four nodes between the initial node and the target node.

C. Scenario III - Fibonacci Sequence Virtual Addressing

Using this heuristic, all the virtual addresses were generated and distributed following the *Fibonacci Sequence* (Fig. 5). The *Fibonacci Sequence* generates, in the interval from 0 to 2^{32}, exactly 46 different numbers, but aiming dimension equivalence between the simulated scenarios, only the firsts 32 *Fibonacci numbers* of the sequence were used on this scenario.

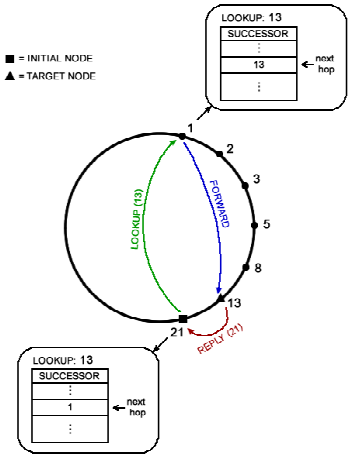

Figure 5. A network with seven nodes using Fibonacci virtual addresses

The simulations showed that the virtual addressing using *Fibonacci Sequence* (Fig. 6) had a great impact on the routing paths length. With this heuristic of virtual addressing, most routing paths could reach its destination using only one intermediary node, or they can reach its destination directly, without any intermediate node on its routing path. There was also an improvement on the distribution of the number of hops in the routing paths, the worst case had only two intermediary nodes between the sender and the target.

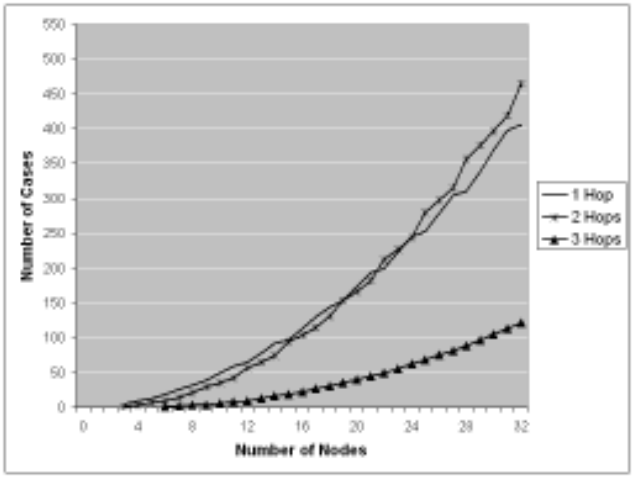

Figure 6. Fibonacci Sequence Virtual Addressing

D. Scenario IV - Exponential Growth Virtual Addressing

The virtual addresses in this scenario were created following an *exponential growth* with base 2. This kind of heuristic permits only 32 different virtual addresses using 32-bit identifiers (like IPv4), but, as an initial research, the goal of this work was simulate only small networks, so this quantity of different virtual addresses was adequate for the simulations.

The simulations using this heuristic generated exceptional results, present on Fig. 7. More than 50% of all the routing paths had no intermediary nodes between the starting point and its end point, in other words, most routing paths could link two nodes directly. On this scenario, the worst case had only one intermediate node between the initial node and the target node.

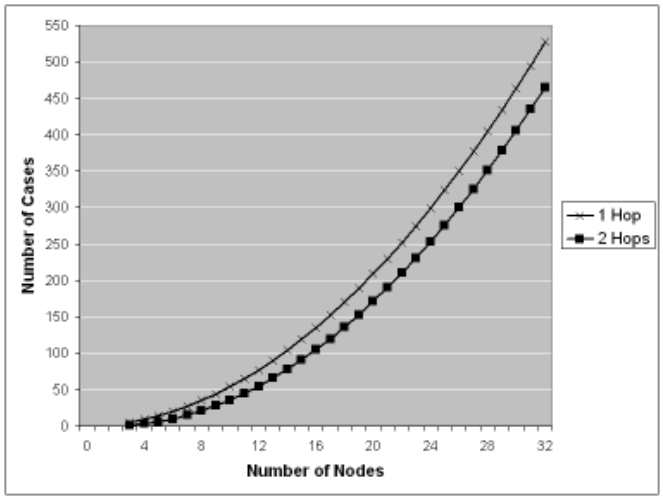

Figure 7. Exponential Growth Virtual Addressing

V. Conclusion

The simulations showed that there are differences on the lengths of routing paths of a network, depending on the how the nodes identifiers are distributed.

Results given by *Random virtual addressing* followed the maximum quantity of $O(\log N)$ hops in a routing path, but there were a few cases in which this limit was broken. *Contiguous*

virtual addressing did not make significant changes in comparison with the random scenario.

The most striking results were obtained by *Fibonacci* and *Exponential* virtual addressing. At the *Fibonacci* scenario, the longest routing path had just three hops until reach its target, and at the *Exponential* scenario all the lookups were resolved directly (one hop) or passing by just a single intermediate node.

From the simulation results, one can clearly conclude that the use of some heuristics can improve the connections between a node and its *successors*, since they lead to shorter routing paths in a lookup operation

The disadvantage of the use of a heuristic for virtual addressing is that it needs a kind of supervisor for the address-assign task, and the two heuristics presented on this work (*Fibonacci* and *Exponential*) had to discard some addresses by its nature. And as the key storage was not considered, the impact of the use of these heuristics could not be calculated.

ACKNOWLEGMENTS

Special thanks are due to Prof. Mauricio Magalhães, Ms. Catalina Zapata and Ms. Ellis Sandra S. Lima for suggestions and fruitful discussions.

REFERENCES

[1] B. M. Leiner et al. 10 Dec. 2003. Histories of the Internet: A Brief History of the Internet. Retrieved from: <http://www.isoc.org/internet/history/brief.shtml>.

[2] A. S. Tanenbaum, "Computer Networks," 4th edition, Prentice Hall, New Jersey, USA, 2003, 912p.

[3] V. Cerf and R. Kahn, "A Protocol for Packet Network Intercommunication," IEEE Trans. Comm., Vol. 22, No. 5, 1974, pp.637–648.

[4] R. Hagens, N. Hall and M. Rose. Use of the Internet as a Subnetwork for Experimentation with the OSI Network Layer. Internet Engineering Task Force, Feb 1989. RFC 1070.

[5] R. Schollmeier. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE 2002.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In Proc. ACM SIGCOMM'01, San Diego, CA, Aug. 2001.

[7] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Nov. 2001, pp. 329–350.

[8] M. Caesar, M. Castro, E. Nightingale, G. O'Shea, A. Rowstron, "Virtual ring routing: network routing inspired by DHTs," ACM SIGCOMM, September 2006.

[9] Y Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). Internet Engineering Task Force, 1995. RFC 1771.

[10] FIPS 180-1. Secure Hash Standard. U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, VA, Apr. 1995.

[11] H. M. Deitel and P. J. Deitel, "Java: how to program," 4th edition, Prentice-Hall, New Jersey, USA, 1999, 1568p.

[12] Sun Mycrosystems, Class BigInteger. <http://java.sun.com/j2se/1.5.0/docs/api/java/math/BigInteger.html>.

[13] J. Gosling, B. Joy, G. Steele and G. Bracha. The Java Language Specification. 2nd edition <http://java.sun.com/docs/books/jls/second_edition/html/jtitle.doc.html>