Using Spacial Locality and Replication to Increase P2P Network Performance in MMO Games

Ross Humphrey, Alexander Allan and Giuseppe Di Fatta
School of Systems Engineering
The University of Reading
Whiteknights, Reading, Berkshire, RG6 6AY, UK
gk004759@reading.ac.uk, A.J.M.Allan@student.reading.ac.uk, G.DiFatta@reading.ac.uk

***Abstract*—Massively Multiplayer Online Games (MMOGs) are increasing in scale and popularity, which is putting strain on the classical client-server(C/S) architecture. As a consequence there is a growing research interest in the adoption of Peer to Peer (P2P) architectures to spread the load throughout participating client machines. However this presents many challenges, amongst which is creating a shared virtual space between nodes, an area known as Interest Management. When using the Region-Based model of Interest Management the game world is mapped to a logical space which is broken into regions managed by peers. When a player's avatar moves through the game-space it moves through these regions, and must download content from the appropriate peer. Finding this peer can be handled by a look-up on a Distributed Hash Table with a circular key. This work explores the advantage of mapping Distributed Hash Table(DHT) keys using a locality preserving function instead of a conventional uniformity enforcing hash algorithm within a P2P protocol for MMOGs. Content retrieval robustness in terms of handling node failures is also explored with multiple data replication techniques analysed and compared. The performance difference is measured in terms of hop count, node stabilisation and node failure. Results show that using locality sensitive hashing and 12 node replication provided favourable performance across all three measurements used.**

Keywords-Peer-to-Peer Networks; Massively Multiplayer Online Games; Interest Management.

I. INTRODUCTION

A Massively Multiplayer Online Game (MMOG) allows large numbers of on-line players to interact with each other in the same persistent virtual space. Traditionally Client Server (C/S) architectures were employed as they were convenient in terms of implementation and security [1]. However, the popularity of MMOGs is rapidly increasing [2] which is putting increasing strain on the C/S architecture presenting, amongst other things, a large cost in terms of hardware and facilities, a single point of failure and a network bottleneck. This has led to a growing research interest towards architectures capable of harnessing the power of client machines in the form of Peer to Peer (P2P) MMOGs [3], [4], [5], [6].

A key problem when creating a P2P MMOG is maintaining a sense of shared space across all players. This can be solved by each player having a full copy of the game state and broadcasting any changes to all other players. However, this is not feasible as the number of messages needing to be sent over the network scales exponentially with the number of players.

A solution to this is to send players only relevant information, a process known as Interest Management [7]. This can be achieved by dividing the game up spatially. A fine grained approach to this is the *aura-nimbus* information model [8]. The *aura* bounds the presence of an object in space, with the *nimbus* (the area-of-interest) representing the boundary within which the object can perceive other objects.[9] While this allows very accurate Interest Management, it does not scale well due to the cost of computing the intersection between areas of interest and *auras* of objects [10].

Region-based interest management is an approximation which addresses the scalability issues of the pure aura-nimbus model by partitioning the game space into static regions [8], [11], [12], [13].

Traversal of these regions requires that a player query a lookup table of some kind in order to contact the node who is regional administrator. In a distributed environment this lookup can be achieved using a node ID as the index for a Distributed Hash Table (DHT). Chord offers one such scalable implementation of a DHT with a lookup performance of $O(log(n))$ and allows for index updates when nodes join and leave the network [14]. Chord was chosen as it represents a simple case of a distributed hash table in which to use as a test bed for experimentation relating to replication and region based interest management.

It is common practice when assigning DHT keys to do so in a manner which assigns IDs uniformly around the keyspace, a method known as consistent hashing. This is commonly achieved by using a cryptographic hash algorithm such as MD5[15] or SHA-1[16]. However, since in this application domain regions are traversed sequentially, a locality preserving hash function could be more efficient in terms of DHT hop count than a conventional one. This work presents a comparison of four different keyspace generation methods for a Chord DHT. Consistent hashing given by the MD2[17] and MD5 cryptographic algorithms is compared to linear and Hilbert curve based methods (outlined in Section II, Subsections C and D). The effectiveness of the four methods is measured in terms of hop count and node keyspace stabilisations within a simulation of a P2P protocol for MMOGs.

This work also uses the same simulation to look at the the effectiveness of implementing a replication algorithm to maintain the games' state in event of node loss.

 ISBN: 978-1-61208-238-7

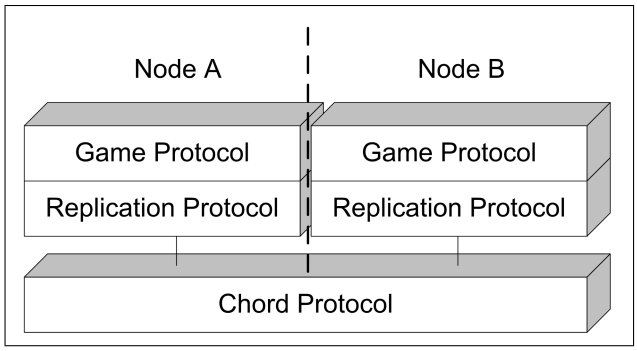

Fig. 1. Components of the simulation protocol stack

The rest of the paper is organised as follows. Section II provides additional detail on the ID mapping and replication techniques used within the simulated environment. Section III provides a description of the simulated P2P MMOG environment. Section IV describes the comparative analysis of the ID mapping algorithms and data replication techniques. Section V provides the experimental results and their interpretation. Concluding remarks are given in Section VI.

II. Overview of Techniques

A. Chord DHT

Chord was used as the underlying P2P protocol for this simulation. Chord is a Distributed Hash Table (DHT) which allows efficient mapping of keys onto nodes in a peer to peer environment [14]. By using Chord a node can locate another node within $O(log(n))$ hops. The key/value pairs are spread throughout the network giving Chord its ability to scale to a large number of peers.

The Chord protocol layer is comprised of a number of nodes. Each node contains a section of the DHT with the key/value pairs stored within it. The section held within each node is changed when a node leaves or joins the network. A stabilisation takes place when this table is changed. A stabilisation consumes both network and hardware resources as it involves querying the nodes on the network and changing values within a node dependent on queries made. The DHT within this simulation uses a 128-bit key space; this key space can be changed depending on the ID allocation used for the node.

Each node within a Chord ring has an identifier that indicates where the node is mapped in the logical ID space. The ID in a chord network is typically generated using a one way consistent hash function such as MD5. Two forms of consistent (non locality preserving and uniformity enforcing) hashing were implemented as comparison with the two locality preserving key generation algorithms. The two implementations decided on were MD5 and MD2 which both produce a 128-bit ID.

B. Consistent Hashing - Message Digest Algorithms MD2 and MD5

A message digest algorithm is a one way cryptographic hash function which outputs a string of fixed length (a hash) for an arbitrary size of input data.

The MD2 and MD5 algorithms are derivations of the popular Merkle-Damgård method [18] and output 128-bit words for any given input. These methods of consistent hashing are commonly used within the Chord protocol to map IP addresses into the key space and will be used in comparison with locality aware hashing methods.

C. Locality Preserving - X-Axis Linear curve

A simple method of embedding locality in a 1D index is by an ordered linear traversal along an axis. The 2Dimensional space is broken down into a grid of a finite granularity and the grid squares are assigned an index sequentially. The X-Axis method fixes the y axis for each traversal of the x axis. This has the effect that locality is preserved effectively in the x-coordinate at the expense of relatively poor locality preservation in the y-coordinate.

The ID allocation operates by assigning IDs to nodes in a linear fashion across the x-axis. The nine squares of the grid in Figure 2 are traversed in the following sequence of coordinates:

$$(0,0),(1,0),(2,0),(0,1),(1,1),(2,1),(0,2),(1,2),(2,2)$$

This method is compared to the consistent hashing (uniformity enforcing) method provided by MD2 and MD5, and with the locality preserving Hilbert curve as a method of key generation for a Chord DHT.

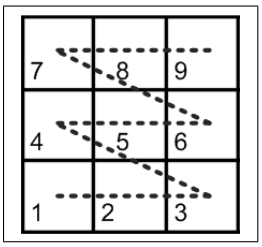

Fig. 2. X-axis linear curve in a 3x3 2-dimensional grid

D. Locality Preserving - Hilbert's Curve

Hilbert's curve is a continuous fractal space-filling curve of finite granularity. Giuseppe Peano (1858-1932) discovered a densely self intersecting curve in 1890 which passes through every point in a 2D space (and by extension in an n-dimensional hypercube) [19], [20]. This work was followed in 1891 by that of David Hilbert [21] who published his own version of the space-filling curve including illustrations for construction (Figure 3). Hilbert's variant proves to have performance advantages (in terms of how locality and how well 'compact regions' of 2D space are represented) over other space-filling curves [22], [23]. This explains its attraction

 ISBN: 978-1-61208-238-7

as a contemporary multi-dimensional indexing method. The Hilbert's variant proceeds through each step replacing the U shape with an upside down Y. Each corner in the diagram represents an additional number in the sequence. As a means of dimensionality reduction, it transforms the data from n to 1 dimension by assigning each point in space a number.

We compare the key space generated by the Hilbert Curve with those generated by the consistent hashing algorithms MD2 and MD5, and with the simple X-axis curve.

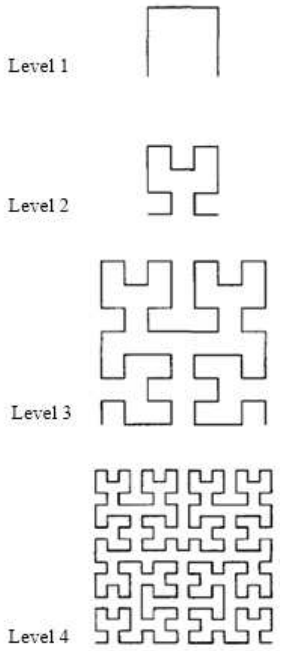

Fig. 3. The first 4 levels of Hilbert's curve in 2 dimensions

E. Replication

The elastic nature of multiplayer online games does not lend itself well to a static P2P network. As players log out spuriously or disconnect from the network data is lost and cannot be recovered until the peer next logs in to the network. This is a problem when objects in the game world go offline and other players wish to access them.

In order to solve this problem replication must be implemented so that when a player leaves the network, the persistent world in the game continues without the data being lost. A replication strategy was devised that would allow for data to be replicated to other nodes in the network. This replication strategy involves mirroring information using a set of replication nodes.

If for example node n queries node a for data d, but finds that a is no longer present in the network, n can then query a's replication nodes. As long as at least one of a's replication nodes remain online, n will be be able to retrieve d.

Formally each node n has a set of R_n replication nodes $Rep_n = \{r_0^n, ..r_i^n\}$ where $0 \leq i \leq R_n$ which each carry a copy of its data. One of these replication nodes is chosen randomly, the others are chosen from n's successor nodes on the DHT. The number of replication nodes used at node n, R_n, varies and is based on the popularity of the data at n. If data is requested more than once every 5 seconds from n then Rep_n will contain all DHT successor nodes and one random node.

When data is modified at node n or any member of Rep_n, a lock message is sent to n and each member of Rep_n. When the data is successfully modified, its new value is broadcast to n and members of Rep_n. When all these nodes have successfully received the new value, an unlock message is broadcast amongst replicated nodes.

III. Simulation of a P2P MMOG protocol

The proposed system contains a number of components that are used to simulate a P2P MMOG (Figure 1). The system adopts three separate protocols: the Chord DHT Protocol, the Game Protocol and the Replication Protocol.

1) The Chord DHT Protocol: is used to find the information being searched for by the user in the network. The layer uses the nodes provided by the simulated network, when searching for the requested data. The search data requested is provided by the Game Protocol that receives the requests from the Traffic Generator. The traffic generator builds requests based on coordinates. The coordinates are relative to the position within a game world. From this, a message is generated and sent from the sender node within the simulator.In a real MMO application a user would provide the requests. If an ID cannot be found in the network the modified protocol will research using the successor list of the node (where the node data is replicated to).

2) The Game Protocol: is used in the product to store game data within nodes. The Game Protocol layer is used to parse messages between layers as well as building initial request messages. The Game Protocol also has the functionality to store successful searches in a local cache at each node.

The Game Protocol receives initial messages from the Traffic Generator. Valid message types can be created in the traffic generator and parsed to the Game Protocol layer where they are automatically handled depending on their type. The Game Protocol also receives replication message instructions from the Replication Protocol to spread data in the network to reduce failures.

3) The Replication Protocol: is a set of methods containing logic to replicate data around the network. Locking and unlocking of data is set up within this layer of the product as well as logic to ensure consistency of data throughout the network (such as pushing new data to all replication nodes). The Replication Protocol layer interacts directly with the data held within the Game Protocol layer. The Replication protocol is explained in greater depth in Section III. A locking and unlocking mechanism is also employed. The locking and unlocking messages can be sent by any node holding a replication with a time released unlock being employed in the event of a node failing whilst the data is locked.

 ISBN: 978-1-61208-238-7

These protocols interact with each other to perform simulated user actions created by the Traffic Generator. These user actions are in the form of object lookup requests given coordinates in the game space which simulate the user moving through a physical 2D space.[24]

The network actions are dealt with by the P2P simulation environment PeerSim [25]. This allows simulation of a real time dynamic P2P network environment with adjustable node failure and churn rate.

The list of game objects are assigned IDs based on four different mapping techniques: MD2, MD5, X-Axis linear curves and Hilbert Curves. The methods are explained in more detail in the following section.

IV. Comparative Analysis of ID Mapping Algorithms

A comparative analysis of the ID mapping and replication techniques described was conducted within the simulation described in section II. The effectiveness is measured in terms of Chord hops, node failures and node stabilisations:

4) Hops: The number of hops that take place within the simulation model from the sender node to the receiver node determines search time. The simulator calculates three metrics for performance comparison:

- **Max Hop** Maximum number of hops taken by the simulator to find data at a receiver node in the network.
- **Min Hop** Minimum number of hops taken by the simulator to find data at a receiver node in the network.
- **Mean Hop** Average number of hops over a complete simulation.

The lower the number of hops the faster data can be retrieved in the network. This results in higher performance.

5) Stabilisations: A stabilisation takes place in the network when data within the DHT is changed by a node. The DHT is changed when a node leaves the network or a new node joins the network. A stabilisation consumes both network and hardware resources (CPU cycles, memory) as it involves querying the nodes on the network and changing values within a node dependent on queries made.

6) Failures: A failure occurs in the network when the sender node fails to find the data being queried. Failures can occur on the network when:

- Data has not been replicated and the master node has left the network.
- A collision in the ID space has occurred (only observed when hash collisions have occurred).
- The ID cannot be found within the network ID space.
- A failure in the (simulated) physical network occurs.

Having a high number of failures is undesirable as it gives a player an inaccurate representation of the MMO game world. In having a high number of failures a player may not be able to retrieve character information leading to a poor overall game experience.

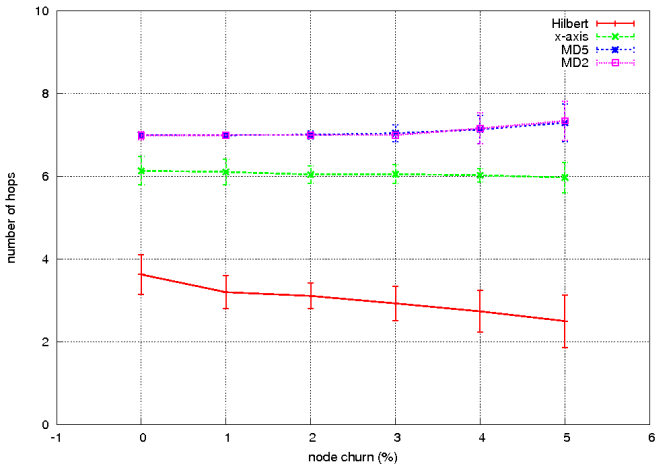

Fig. 4. Average Hop rate per cycle vs churn rate

A. Simulation 1

A network of 10,000 nodes was created and the network initialised on top of this. The node ID was created by MD2, MD5, X-Axis Locality and Hilbert indexing to produce four different data sets. For each data set the churn rate was increased from 0-500 nodes (0-5%) per cycle in increments of 100. At each of these increments the hop count and node stabilisations were recorded. The results from each data set were gathered 100 times and were averaged for each variable.

B. Results for Simulation 1 Scenario

In using X-axis locality, the average number of hops required to find data on the P2P network is reduced (see Figure 4), and in using Hilberts curve the number of stabilisations can be reduced (see Figure 6). In reducing stabilisations the amount of network traffic is also reduced which in a real physical network would result in better performance (less bandwidth) and a decrease in physical resource utilization at each node (memory, CPU cycles).

When searching within a local area within the game application the performance of Hilberts curve increases on average (see Figure 4), this is a result of the locality preserved within the logical ID space[26][27]. A local cache of network addresses is used which increases in size over time. This is independent of churn resulting in a decrease of hops over time.

C. Simulation 2

A network of 10,000 nodes was created and the network initialised on top of this. The node ID was created by MD2, MD5, X-Axis Locality and Hilbert indexing to produce four different data sets. For each data set the churn rate was increased from 0-500 nodes (0-5%) per cycle in increments of 100. At each of these increments the failure rate was recorded. The results from each data set were gathered 100 times and were averaged for each variable.

D. Failure Rate

Each ID generation method was tested where no replication mechanism is used (figure 5). These results show that when

 ISBN: 978-1-61208-238-7

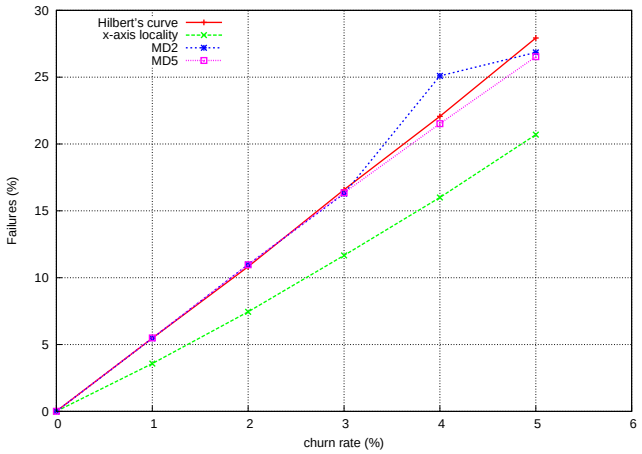

Fig. 5. Average failure rate per cycle vs churn rate: no replication strategy

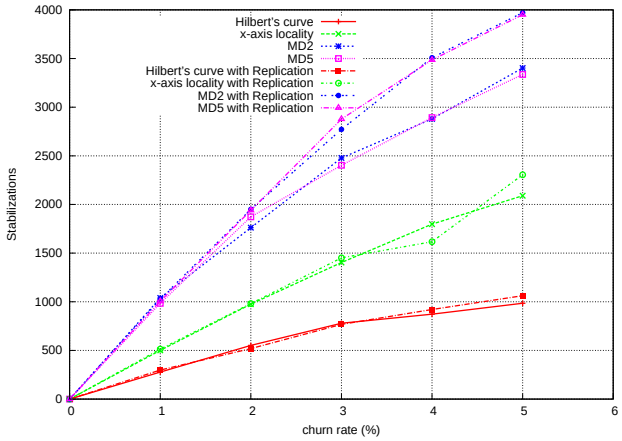

Fig. 6. Average stabilisations per cycle vs churn rate

using the Chord protocol without replication strategy the number of failed queries increases linearly with the churn rate.

When using replication in the network the number of failures is reduced to a negligible number for all methods of ID generation, regardless of the churn rate (up to 50%).

Replication also improved the hop performance of using Hilberts curve to map IDs in a logical space when using a random query. In doing so Hilberts curve performs as well as X-axis locality (see Figure 4). By searching a local area, performance is still significantly faster (also seen in Figure 6). Stabilisations are unaffected by the use of replication, with Hilberts curve ID mapping still offering the most significant performance increase. When using replication a higher number of messages is sent to distribute data in the network which results in more data being sent across the network. The initial testing was carried out in a random manner to test performance in a generalized application. Later testing using geographical proximity was used to test whether locality had a bearing on the lookup time of data in a P2P network.

In using locality aware IDs, the number of hops in a P2P network can be reduced as well as the number of stabilisations required. Within a game application locality offers a significant performance enhancement (see Figure 4). The failure rate however is mostly unaffected and requires another technique to improve performance.

The performance of a P2P network can be beyond that of random IDs with and without replication. This is done using a combination of locality aware ID mapping in a logical ID space and replication.

V. Conclusion and Future Work

This paper explored two aspects of the implementation of a MMOG over P2P protocols. We showed how using a locality aware ID mapping when performing Interest Management in a logical space can have a positive effect on the stability and efficiency of a DHT based P2P protocol. This is measured in terms of number of hops and stabilisations. Specifically, we show that by using Hilberts curve to map IDs with respect to the regions of interest within a logical game space hop performance in a DHT can be doubled (on average). We additionally outline and implement a data replication scheme and show how it leads to the number of failures in a P2P network being reduced. Replication also reduced the number of hops (on average) when using; Hilberts curve, MD2 and MD5 to map IDs to a logical space. In general, when using Hilberts curve as ID generation for Interest Management, in combination with the replication method we propose, the network performance is improved (over the other implementations explored). This should be of consideration to developers of P2P MMOGs who are in need of a node ID scheme for a DHT implementation and data replication strategy. Future work will look at a more extensive implementation of MMOG features such as global game-state management and player communication and the problems these pose for a P2P implementation.

References

[1] J. Mulligan and B. Patrovsky, *Developing online games: An insider's guide*. New Jersey: New Riders, March 2003.

[2] B. Woodcock, "An analysis of MMOG subscription growth," *MMOGCHART.COM [On-line]*, vol. 24, 2008, available = http://tinyurl.com/d6kpl5s [May 24, 2012].

[3] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*. IEEE, March 2004.

[4] T. Hampel, T. Bopp, and R. Hinn, "A peer-to-peer architecture for massive multiplayer online games," in *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*. ACM, 2006, pp. 30–31.

[5] S. Douglas, E. Tanin, A. Harwood, and S. Karunasekera, "Enabling massively multi-player online gaming applications on a p2p architecture," in *Proceedings of the IEEE international conference on information and automation*. IEEE, 2005, pp. 7–12.

[6] M. Ratti, S.and Pakravan and S. Shirmohammadi, "A distributed latency-aware architecture for massively multi-user virtual environments," in *Haptic Audio visual Environments and Games, 2008. HAVE 2008. IEEE International Workshop on*. IEEE, 2008, pp. 53–58.

[7] K. Morse, "Interest management in large-scale distributed simulations," Department of Information and Computer Science, University of California, Irvine, California, Tech. Rep. 96-27, 1996.

[8] S. Fiedler, M. Wallner, and M. Weber, "A communication architecture for massive multiplayer games," in *Proceedings of the 1st workshop on Network and system support for games*. ACM, 2002, pp. 14–22.

 ISBN: 978-1-61208-238-7

[9] J. Lee, H. Lee, S. Kangm, S. Kim, and J. Song, "Ciss: An efficient object clustering framework for dht-based peer-to-peer applications," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 51, no. 4, pp. 1072–1094, 2007.

[10] S. Singhal and M. Zyda, "Networked virtual environments: design and implementation," *Recherche*, vol. 67, p. 2, 1999.

[11] M. Macedonia, M. Zyda, D. Pratt, D. Brutzman, and P. Barham, "Exploiting reality with multicast groups," *Computer Graphics and Applications, IEEE*, vol. 15, no. 5, pp. 38–45, 1995.

[12] T. Funkhouser, "Ring: a client-server system for multi-user virtual environments," in *Proceedings of the 1995 symposium on Interactive 3D graphics*. ACM, 1995, pp. 85–95.

[13] H. Abrams, K. Watsen, and M. Zyda, "Three-tiered interest management for large-scale virtual environments," in *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, 1998, pp. 125–129.

[14] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[15] R. Rivest, "The md5 message-digest algorithm," *Internet activities board*, vol. 143, 1992.

[16] F. PUB, "Secure hash standard," *Public Law*, vol. 100, p. 235, 1995.

[17] B. Kaliski, "The md2 message-digest algorithm," RSA Laboratories, Cambridge, Massachusetts, Tech. Rep. 1319, 1992.

[18] R. Merkle, "Secrecy, authentication, and public key systems." *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, 1979.

[19] G. Peano, "Sur une courbe, qui remplit toute une aire plane," *Mathematische Annalen*, vol. 36, no. 1, pp. 157–460, 1890.

[20] A. Butz, "Space filling curves and mathematical programming," *Information and Control*, vol. 12, pp. 314–330, 1968.

[21] D. Hilbert, "Ueber die stetige abbildung einer line auf ein fluchenstuck," *Mathematische Annalen*, vol. 38, pp. 459–460, 1891.

[22] G. C. and M. Lindenbaum, "On the metric properties of discrete space-filling curves," *IEEE Transactions on Image Processing [On-line]*, vol. 5, no. 1, pp. 794–797, Jan 1996, available = http://tinyurl.com/c9qse9k [May 24, 2012].

[23] B. Moon, H. Jagadish, C. Faloutsos, and J. Saltz, "Analysis of the clustering properties of the hilbert space-filling curve," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 1, pp. 124–141, Jan 2001.

[24] A. Montresor and M. Jelasity, "A walkable kademlia network for virtual worlds," in *INFOCOM'09 Proceedings of the 28th IEEE international conference on Computer Communications Workshops*. IEEE, 2009, pp. 313–314.

[25] M. A. and M. Jelasity, "Peersim: A scalable p2p simulator," in *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*. IEEE, 2009, pp. 99–100.

[26] Z. Gharib, M. Barzegar and J. Habibi, "A novel method for supporting locality in peer-to-peer overlays using hypercube topology," in *ISMS '10 Proceedings of the 2010 International Conference on Intelligent Systems, Modelling and Simulation*. ACM, 2010, pp. 391–395.

[27] B. Ratti, S. Hariri and S. Shirmohammadi, "Nl-dht: A non-uniform locality sensitive dht architecture for massively multi-user virtual environment applications," in *ICPADS '08 Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems*. ACM, 2008, pp. 793 –798.

 ISBN: 978-1-61208-238-7