A Walkable Kademlia Network for Virtual Worlds

Matteo Varvello[†⋆], Christophe Diot[†], Ernst Biersack[⋆]
[†] Thomson, Paris, France
[⋆] Institut Eurecom, Sophia-Antopolis, France
{matteo.varvello,christophe.diot}@thomson.net, ernst.biersack@eurecom.fr

I. Introduction

Virtual worlds are digital lands populated by *objects*, where human-controlled *avatars* can live a virtual life. Commercial virtual worlds such as Second Life [6] are implemented using a Client/Server (C/S) architecture. A server stores a copy of all objects that reside on the virtual land. The clients run stateless applications that allow users to explore the virtual world through the eyes of their avatar. To do so, the clients send *range queries* to the server, i.e., requests for the objects which spatial coordinates are located within a given range. In practice, avatars identify the set of objects (e.g., trees, cars, etc.) located in their surroundings by sending to the server a range query with range equal to the avatar visibility area.

Range queries in virtual worlds can be divided into *local* and *non-local*. A local query consists in a request for objects located in the avatar surroundings. For example, avatars generate local queries when they walk, run or fly, in order to constantly update their visibility area. A non-local query is a request for objects located far away from the avatar coordinates. For example, avatars generate non-local queries when they suddenly cover a large distance via the teleport operation. Local queries must be answered quickly to ensure a good user experience. Conversely, a higher delay in answering non-local queries may be tolerable [6].

Both local and non-local queries in virtual worlds are easy to manage with a C/S architecture. However, this architecture exhibits poor scalability and high cost [7]. Peer-to-Peer (P2P) and Distributed Hash Tables (DHTs) are a cheap and scalable alternative [6]. DHTs are P2P architectures used to store and retrieve content. However, current DHT designs allow only to address content specifically. This limits the applicability of DHTs to new applications such as virtual worlds.

In this work, we design and evaluate Walkad, a P2P architecture for the management of range queries in virtual worlds. We design Walkad as an extension of Kademlia [3], a very popular DHT successfully adopted in eMule [1]. Walkad organizes the Kademlia keyspace in a *reverse binary trie*, i.e., a tree-based data structure where nodes of each level of the tree are labeled using the Gray Code [2].

We evaluate Walkad via emulation [5], and using object traces from Second Life [7]. Synthetic avatar traces are used to study different types of range queries. Our preliminary results show that Walkad is an efficient P2P design for virtual worlds. In fact, Walkad guarantees to its users a fast discovery of the virtual world, and fairly distributes load among peers.

II. Walkad in a Nutshell

In this Section, we design Walkad.

A. Key Indexing

We call a *cell* a portion of the virtual world, and a *cell-ID* the Kademlia key associated to a cell. We say that originally the virtual world is composed by a single cell. Then, the virtual world is recursively "split" into multiple cells as the number of objects becomes larger than a threshold D_{max}. We say that two cells are *neighbor* cells if: (1) they have a side in common, or (2) they are symmetric according to the axis used in previous split operations. Similarly, we say that two cell-IDs are *neighbors* when their cell-IDs have a Hamming distance of one, i.e., when they differ only by one bit. By definition, a cell-ID with l significant bits has l neighbor cell-IDs. To illustrate this, we consider an example of a one-dimensional virtual world (Figure 1). We denote the *i*-th cell/cell-ID generated by l splits respectively as C_i^l and k_i^l.

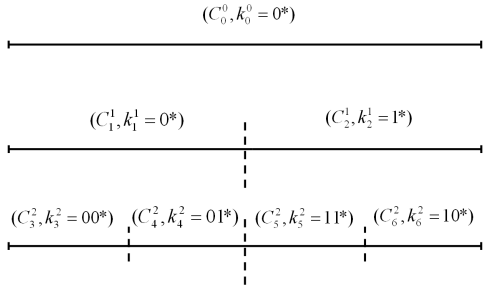

Fig. 1. 1-dimensional virtual world indexed by Walkad

The top portion of Figure 1 shows the initial cell organization. At this stage, there is only one cell, C_0^0, which covers the whole virtual world. The middle part of Figure 1 shows the virtual world configuration after the first split, where two new cells are created, C_1^1 and C_2^1. These two cells are adjacent and so neighbors. The bottom part of Figure 1 shows the result of splitting again both cells. Let's consider cell C_3^2. Its neighbor cells are cell C_4^2 which is adjacent and cell C_6^2 which is symmetric to C_3^2 according to the long dashed line.

Walkad organizes the cell-IDs in a *reverse binary trie* to associate neighbor cell-IDs to neighbor cells. In a reverse binary trie, the nodes of each level of the trie are labeled with the Gray Code [2], a binary numeral system where two successive values have Hamming distance equal to one.

We now explain how we organize the cell-IDs in a reverse binary trie considering the example of Figure 1. Cell C_0^0 is assigned cell-ID $k_0^0 = 0*$. When C_0^0 splits, we generate

the new cell-IDs by taking $k_0^0 = 0*$, and setting the least significant bit respectively to 0 and 1. We thus obtain two new cell-IDs, $k_1^1 = 0*$ and $k_2^1 = 1*$, which have a Hamming distance of one. The intuition is that to build a reverse binary trie in some cases we need to reverse the added bits (i.e., add a 1 to the left cell-ID). Figure 1 shows that splitting cell C_2^1 required setting the least significant bit to 0 for $k_6^2 = 10*$ and to 1 to $k_5^2 = 11*$ to guarantee that cells C_3^2 and C_6^2, which are neighbors, are assigned neighbor cell-IDs as well.

The indexing algorithm we described organizes the cell-IDs within the keyspace as a trie. Therefore, an unbalanced trie results in an unbalanced load among peers. In order to restore the uniform distribution of the cell-IDs, we divide the world in regions (as in Second Life), and we allocate to each region a *region-ID*. Then, we perform a XOR operation between the cell-IDs and the region-ID. In this way, the Hamming distance among cell-IDs of the same region is maintained and load balancing is achieved among cell-IDs of different regions.

B. Walkad and Kademlia

We call a *coordinator* the peer responsible for a cell. The coordinator for a cell C_i^l indexed by the cell-ID k_i^l is the XOR closest peer to k_i^l as defined by Kademlia. For each (cell/cell-ID) pair there are R coordinators. A Kademlia peer keeps for each $0 \leq i < 160$ bit of its identifier a *k-bucket*, i.e., a list of peers with distance $2^i \leq d < 2^{(i+1)}$ from itself. The entries in the n^{th} k-bucket have a different n^{th} bit from the peer identifier. Therefore, a coordinator for a cell-ID k_i^l stores in l different k-buckets the routing information towards the coordinators of l neighbor cell-IDs.

When a cell splits, its coordinators select the coordinators of the new sub-cells via a Kademlia lookup. Afterwards, they transfer to these coordinators a list of the existing neighbor (cell/cell-ID) pairs, and the set of objects located within the new cell. A peer selected to be a coordinator for a cell-ID k_i^l performs a Kademlia lookup for each existing neighbor cell-ID among the l cell-IDs with Hamming distance equal to 1 from k_i^l. This operation populates the coordinator k-buckets with the routing information towards the neighbor coordinators.

A range query submitted by a peer P is resolved as follows. P sends the range query to one of the coordinators for the cell where its own avatar is located. The coordinator answers the query or a portion of it according to the information it has about the neighbor cells. Then, it sends back to P the information it may know, i.e., routing information towards the coordinators for the cells intersecting with the query's range. In case a coordinator has not a complete view of the entire range, it forwards the query to the coordinators it knows that manage the closest cells to this range. This procedure is done iteratively until the range is completely covered. Finally, P contacts the coordinators responsible of the query's range to retrieve the objects located in this portion of the virtual world.

III. Experimental Evaluation

We evaluate Walkad on a local cluster using Modelnet [5] and a synthetic Internet topology generated by Inet [8]. We pick a classic Kademlia setup with k-bucket size $k = 20$ [3], and $R = 10$ [1]. We generate a realistic virtual world using object traces from five popular Second Life regions [7].

We evaluate first the *latency*, i.e., the time required to answer range queries, as a function of the network size N and type of range queries. To generate different range queries, we simulate avatar walking, running, flying and teleporting, via the Random Waypoint Mobility model [4]. In all these experiments, we set $D_{max} = 10$. Figure 2 shows that range queries generated by an avatar walking, running or flying are all resolved in about $100 - 130\ ms$ in average. In fact, all these movements generate local queries. Conversely, non local queries generated by an avatar teleporting require about twice the time, e.g., up to $200\ ms$ in the worst case. Figure 2 shows also that the overall latency only slightly increases with the size of the network N. In fact, the number of routing hops depends on the size of the virtual world rather than on the size of the network. However, when the network is very small, peers are coordinators of multiple cells, thus reducing the effective number of routing hops and latency as well.

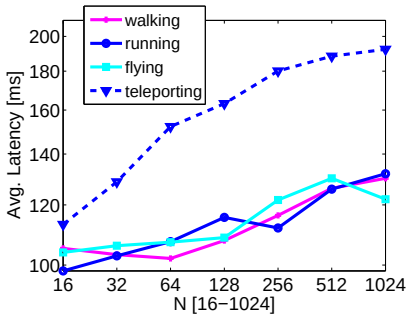

Fig. 2. Latency Evaluation ; $D_{max} = 10$

We now analyze load balancing, i.e., how the cell-IDs are distributed among peers in the experiments of Figure 2. We observe that the load is distributed fairly among peers as the network grows. For example, when $N \geq 256$, for 90% of the peers the difference in the fraction of cell-IDs they manage is smaller than 1%. The remaining 10% of the peers are responsible of a larger fraction of cell-IDs. This is due to the fact that we are considering a small virtual world composed by only five regions. Altought not shown for space limitations, we perform experiments with a virtual world composed by 100 Second Life regions. In this case, we observe that only 1% of the peers store a larger portion of the virtual world.

References

[1] eMule. http://www.emule.com/.

[2] F. Gray. Pulse Code Communication. U.S. Patent 2,632,05, March 1953.

[3] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In *IPTPS*, Cambridge, March 2002.

[4] Mobility Models. http://ica1www.epfl.ch/RandomTrip/.

[5] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kosti, J. Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *OSDI*, Boston, MA, USA, December 2002.

[6] M. Varvello, C. Diot, and E. Biersack. P2P Second Life: experimental validation using Kad. In *Infocom*, Rio De Janeiro, Brazil, April 2009.

[7] M. Varvello, F. Picconi, C. Diot, and E. Biersack. Is There Life in Second Life? In *Conext*, Madrid, Spain, Dec. 2008.

[8] J. Winick and S. Jamin. Inet-3.0: Internet Topology Generator. Technical Report CSE-TR-456-02, University of Michigan, 2002.