# Chapter 5

# Large-Scale Peer-to-Peer Game Applications

## 5.1. Introduction

Massively multiplayer online games (MMOG) recently emerged as a popular class of applications with up to millions of users, spread over the world, connected through the Internet to play together. Most of these games provide a *virtual environment* in which players evolve, and interact with each other. When a player moves, moves an object, or performs any operation that has an impact on the virtual environment, players around him can see his actions.

In a MMOG, each player runs a piece of client software on a local device (e.g. a personal computer or a game station) called *node* thereafter. The local client software is responsible for maintaining an up-to-date version of the state of the virtual world surrounding the player and to offer him the ability to perform operations on it (i.e. to play). To offer an acceptable gaming experience, the distributed application needs to render the virtual world surrounding the player with minimal latency.

### 5.1.1. *Limitations of the client-server paradigm*

Current most popular MMOG such as World of Warcraft [ENT 11] or Second Life [LAB 11] are based on the client-server paradigm. Using this model, a server is responsible for keeping the state of the virtual world up to date. Each time a player plays (performs an action), he has to notify the server in order for his actions to be taken into account. The server then computes the player's neighborhood, composed of

Chapter written by Sébastien MONNET and Gaël THOMAS.

all the players that are potentially impacted by the performed action. It then notifies all the impacted nodes about the changes in their virtual environment. The nodes locally render the modification to the human player. Within a game, users are continuously moving, and therefore their neighborhoods are continuously changing. Players also keep modifying their virtual environment while playing. The server thus continuously receives a large amount of notifications. It has to compute many neighborhoods and must send even more notifications.

This client-server model is unfortunately not scalable for this kind of application [KUM 08]. In fact, using the currently available client-server-based MMOG implementations, the millions of registered users are not really playing at the same game instance at the same time: these MMOG introduce new playing rules in order to limit the number of participants within a single game instance. The game is partitioned into *sub-games*. For instance, the Second Life world is not contiguous: it is composed by multiple *islands*; World of Warcraft is split into many *realms* [PIT 07], etc. At a given time, they are no more than a few hundreds players interacting in a same sub-game connected to the same server.

Furthermore, the client-server paradigm implies an expensive financial cost for the provider [ALV 07]. Servers need to provide high computing performances. They are usually replicated to support failures or disconnections for maintenance. Generally, a game instance (e.g. a Second Life island) is served by a cluster of high-performance nodes rather than a single server. These clusters have to be sufficiently powerful to support usage peaks (e.g. when an important event occurs within an game, like a concert for instance). The cost for building and for maintaining these clusters is high.

### 5.1.2. *The decentralized model*

To circumvent these limitations, a new generation of decentralized *networked virtual environments* (NVE) based on peer-to-peer overlays has emerged [KEL 03, BEA 07a, BEA 07b, HU 06, FRE 08, LEG 10, VAR 09b, VAR 09a, IIM 04, BHA 06, BHA 08, BHA 04] (most of these works are discussed later in the chapter). In addition to scalability and cost, these new solutions offer better availability as it is possible to keep playing even when many players join and leave the game. In fact, several players can play together as soon as they are connected. These new systems also offer more freedom to the player: it becomes possible to have a game without any central organization/company controlling it.

However, distributing such applications using the peer-to-peer paradigm is really challenging. Most existing peer-to-peer overlays do not fulfill these emergent application requirements. Their communication latencies are way too high, they are not able to offer the player an acceptable gaming experience.

Building large-scale peer-to-peer game applications requires new kinds of overlays, tailored for this class of application: as they are particularly dynamic, they require that the underlying overlay adapts itself to keep communications latencies low, while communication patterns evolve.

Section 5.2 describes the requirements of this particular class of application. Then section 5.3 explains in more details why classical overlays are not suitable for large-scale game distributed applications and presents a new class of peer-to-peer network overlays that fulfill these applications requirements. Then, several state-of-the-art solutions are presented for the two main classes of multi-player games: first person shooters (FPS) [BHA 04, BHA 08, BHA 06] in section 5.4 and life simulations [KEL 03, FRE 08] in section 5.5.

## 5.2. Large-scale game applications: model and specific requirements

This section defines the application model. It provides several notions, and details the requirements for large-scale game applications on the underlying distributed system. Finally it presents the main characteristics of such applications that could be used while designing distributed systems for them.

### 5.2.1. *Application model*

The emergent MMOGs systems use different terminologies. We present here the most important notions and the terminology used in the remainder of this chapter.

*Virtual environment*

MMOGs applications are based on the concept of a *virtual environment*. A virtual environment is an n-dimensional applicative naming space in which players evolve. Usually the dimensions are mapped on the coordinates of a virtual world (e.g. the x, y and possibly z axis of a three-dimensional map), but it is also possible to use other notions of semantic distance (e.g. players' contact lists, hidden gates that allow the player to jump somewhere else in the virtual world, etc.).

*Avatar*

Each player node manages an entity representing the player in the game. This entity is a particular object of the virtual environment, usually called an *avatar*. Each object, and thus each avatar, is assigned coordinates in the virtual world. When a player plays, the player's avatar moves (and potentially moves objects) and its coordinates in the virtual environment change.

*Object manager*

Regarding scalability concerns, it is not possible for a node to store and maintain up-to-date information retaining to the whole virtual environment: a virtual world could contain up to hundreds of thousands of players at a given time (even millions). Furthermore, it is continuously changing, therefore, each node would have to continuously receive information from all other nodes, which is clearly not scalable. This is exactly what happens using the client-server paradigm. Thus, in a *distributed* large-scale game application, the virtual world is distributed over the nodes.

Each object is therefore managed by a set of nodes called *the managers of the object*. For robustness or scalability, most of the time, each object is replicated on more than one manager. When a node wants to read the state of an object, it must therefore find and then contact one, some, or all the managers of the object, depending on the design choices. Furthermore, if the node wants to modify the state of the object, it delegates this modification to the manager, which agree on the new state of the object.

*Knowledge area*

Each node therefore manages a set of objects. We define this set as the *knowledge area* of the node, also called "responsibility zone". Each time something changes in its knowledge area, a node has to update it, either by being notified of changes by other players or by periodically probing nodes that the avatar plays in its knowledge area, depending on design choices.

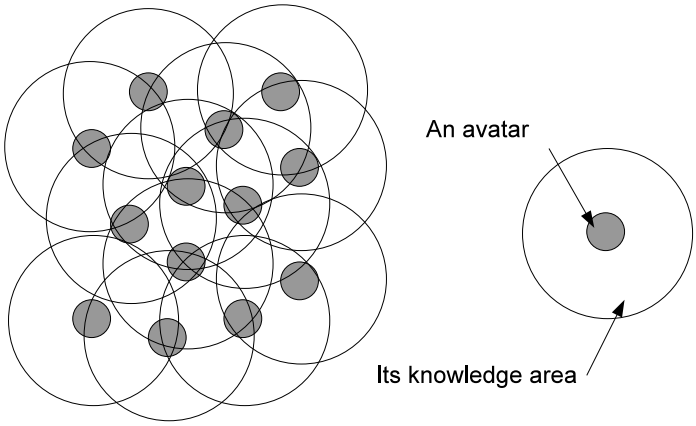


**Figure 5.1.** *Knowledge areas intersect: the virtual world is replicated*

As objects are replicated among their object managers, the intersection of the knowledge areas of two nodes contains the objects that the two nodes manage: knowledge areas are shared between nodes. With simple replication protocols, the replication granularity is the knowledge area. With more advanced replication protocols, the replication granularity is the object. Figure 5.1 illustrates an advanced protocol where the knowledge area of a node depends on its avatar position [FRE 08]. The figure presents avatars and their knowledge areas. In this example, a portion of the virtual environment is often part of two or more knowledge areas: it is replicated on the nodes running the corresponding avatars. In simple and advanced replication protocols, the virtual environment is distributed (and replicated) among all the participating nodes.

As no node hosts a complete copy of the virtual environment, they have to collaborate in order to give to the players the illusion of a unique continuous virtual world. The distributed MMOG has to ensure that every portion of the virtual world is hosted by at least one node in the system. This means that the whole virtual world has to remain the union of the knowledge areas.

*Playing area*

The client software (running on player nodes) needs to maintain an up-to-date *playing area* for the player. The playing area is the zone surrounding the avatar that needs to be rendered/displayed to the human player. Figure 5.2 illustrates this notion. When a player plays, its avatar modifies the local playing area. Changes are propagated to nodes that manage objects in this playing area: these are the nodes whose knowledge area contains parts of the modified playing area.

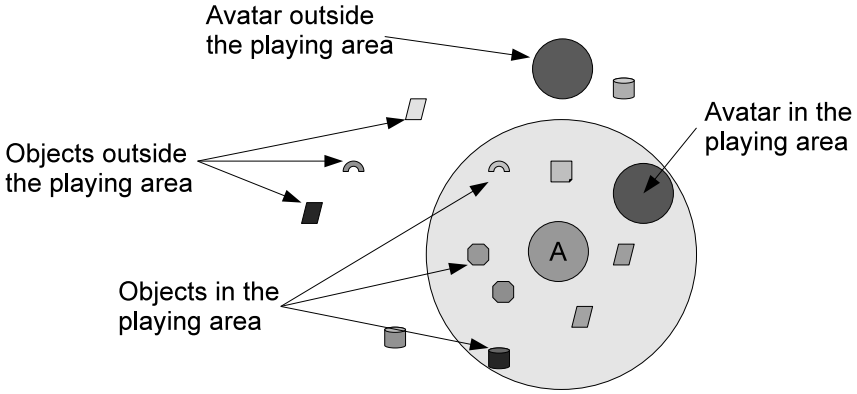


**Figure 5.2.** *Avatar A's playing area*

*Elders of a playing area*

To simplify the presentation, we introduce also the notion of elders [LEG 10]. The elders of a playing area are the nodes that manage the objects in the area. In other

words, the elders of a node N are the nodes E whose knowledge areas intersect the playing area of N. The elders of a playing area are the servers of the objects located inside the playing area. We also define the elders of a node as the elders of the playing area.

*Playing area/knowledge area relations*

There are two main design choices.

First, on a node, the playing area can be included in the knowledge area. In this case, the knowledge area of a node is around its avatar and the limits of this area change as the avatar moves. With this design, elders of a node are close to the node's avatar inside the virtual environment. To ensure that the whole virtual environment is replicated even if a player is alone in its playing area, knowledge areas are bigger than playing areas. This is the model chosen by the Solipsis system [KEL 03] detailed in section 5.5.

Conversely, the world can also be statically partitioned in responsibility zones (i.e. knowledge areas), which are then attributed to/replicated on several peers. Therefore, in this model a peer is a server for a part of the virtual environment, its knowledge area; but it acts as a client: its avatar may evolve in a totally different zone and its playing area may be far from its knowledge area in the virtual environment. In this case the knowledge area boundaries are static and fixed while the playing areas are changing with avatars movements. This is usually the case in systems based on peer-to-peer distributed hash tables (DHTs) [VAR 09b, VAR 09a, IIM 04] (see section 3.2.2).

*Summary of the terminology*

The terminology used in large-scale game applications is summarized below:

– *virtual environment*: the n-dimensional applicative naming space;

– *avatar*: a special object that represents a player in the virtual environment;

– *object managers*: the set of nodes that manages an object;

– *knowledge area*: the set of objects managed by a node;

– *playing area*: the set of objects the player is interested in;

– *elders* of a playing area: the object managers of the objects located in the playing area.

### 5.2.2. *Main requirements for large-scale game applications*

There are two main requirements in large-scale game applications: playability and consistency.

*Playability*

To be used, a game has to provide the user with a comfortable gaming experience. The player should have the illusion that there is no latency at all between the moment another player moves and the moment the player sees it. At the distributed infrastructure level, this implies that latency should remain very low (under certain limits). Luckily, many graphical tricks exist at the application level to hide the inevitable network latency [PAN 02, PAN 07].

The admitted latency offered by the peer-to-peer system largely depends on the kind of game application. For instance, a life-simulation game, or a virtual visit of a town/museum may tolerate greater latencies than FPS games. In the first case, high latencies may induce slow movements, slightly degrading the playing experience, while in the second case, a player may still be playing while its avatar is already dead, making the game totally unplayable.

Therefore, the goal of the emergent peer-to-peer overlays for large-scale distributed game applications is to keep communication delays as low as possible. This is challenging because while players play, their avatars move, their playing areas change, which in turn changes the communication patterns among nodes.

*Consistency*

The whole game, i.e. the whole virtual world, has to remain available. As it is dynamic and spread all over the participating nodes, replication mechanisms have to be implemented in order to ensure that a copy of each portion of the virtual world is available at any time. Furthermore, copies of the same portions of the virtual world have to be mutually consistent. Therefore, nodes have to somehow synchronize their knowledge areas.

### 5.2.3. *Influence of the avatar mobility on the playability*

Within large-scale game applications, a player shows and acts on a limited part of the virtual environment: the objects of the playing area. Therefore, a node participating in a distributed large-scale game application is interacting with a limited number of other nodes at a given time: its elders.

To increase the playability, it is thus possible to efficiently predict the communication patterns using the avatars coordinates in the virtual environment: avatar coordinates define their playing areas which directly defines their elders. However, during the game, the mobility of the avatar induces continuous changes in the coordinates and thus of the elders and of the communication patterns.

To increase the playability, it is therefore important to characterize how avatars are moving.

*Characterization of avatar mobility*

The characteristics of avatar mobility may be learned from real traces. However, all existing commercial MMOG projects are based on a client-server architecture which, as stated above, usually implies poor scalability. Therefore, each trace of a real game session involves at most a few hundred avatars simultaneously. Moreover, the number of available real traces is small because they are difficult to obtain [LA 08]. Therefore, there are few real traces.

Recently, some research effort has been applied to characterizing avatar distribution and avatar mobility in large-scale game applications [LEG 10, RHE 08, LIA 08, LA 08, MIL 09]. Avatars in virtual environments usually have total freedom of movement. Resulting virtual environments are then very dynamic: data representing objects and avatars may not be uniformly distributed all over the universe. Recent studies of existing popular large-scale games, such as Second Life [LAB 11] and World of Warcraft [ENT 11], have shown that the distribution of avatars was extremely disparate [LA 08, PIT 07]: most of the avatars were gathered around a few hot-spots of interest, while large parts of the virtual environment were almost deserted. Figure 5.3 shows the distribution of avatars on a Second Life island.

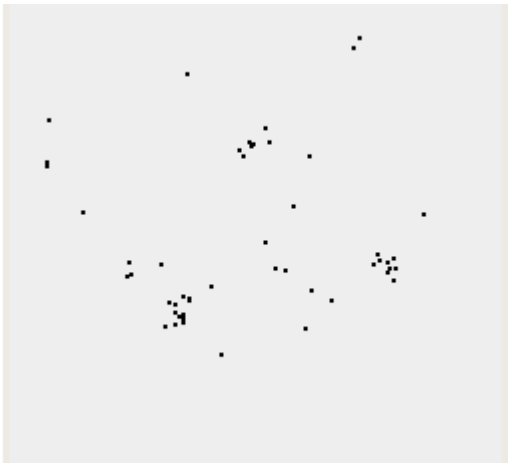

**Figure 5.3.** *A sample avatar distribution in Second Life*

This kind of distribution with hot-spots also corresponds to real density distributions of human populations, such as the European *blue banana* [BRU 02] that covers one of the world's highest concentrations of populations around the cities of London, Brussels, Amsterdam, Cologne, Frankfurt, and Milan with approximately 20% of the European population.

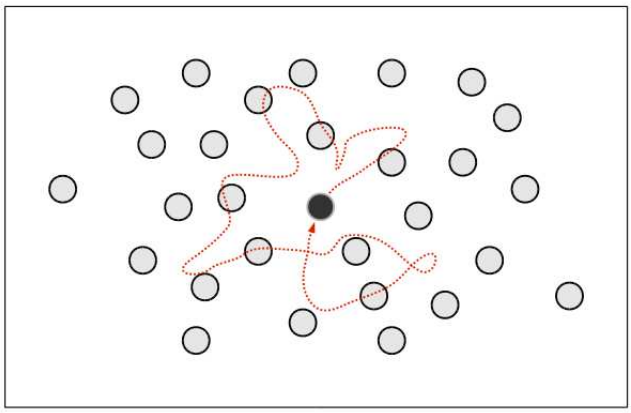

**Figure 5.4.** *Avatar movement within high density zones*

Moreover, regarding player mobility in virtual environments, studies have shown that it is quite similar to human mobility in the real world [RHE 08, LA 08]. The mobility patterns of avatars has been shown to be highly non-uniform: avatars move slowly and chaotically within the hot-spots as illustrated in Figure 5.4, whereas the movement between the hot-spots is straight and fast [LIA 08]. This mobility pattern is usually modeled with Lévy flights [RHE 08, LA 08]. Indeed, Lévy flights are a particular sort of random walk in which the increments are distributed according to a "heavy-tailed" probability distribution [CHE 06] with short and chaotic movements, and sometimes, long and straight movements. Therefore, Lévy flights naturally differentiate the two observed behaviors of avatars: periods of travel and periods of exploration with chaotic movements. However, Lévy flights do no help to model hot-spots because they do not ensure that avatars stay grouped around hot-spots and that density around hot-spots remains the same whatever avatar mobility is. Recent research effort has focused on designing models able to generate realistic avatar movements on a large scale in order to be able to test really large-scale game applications prototypes [LEG 10].

*Influence of game rules on avatar mobility*

Finally, the game rules can also influence avatar movement. For instance in war games, or quest games, avatars may be enrolled in fellowships. All the avatars belonging to the same fellowship (or a same army) may move *together* in the same direction with similar speeds, generating *group movement* [MIL 09], as illustrated in Figure 5.5. An avatar within a moving group changes its playing area, but the objects of its playing area remain almost the same because avatars located in its playing area move along with him.

This kind of group movement is particularly efficient for distributed MMOGs where the playing area is included in the knowledge area. In this case, the elders of a player within the group also remain around the player. Only avatars located on the front line of the group have to obtain new elders ahead of the movement.

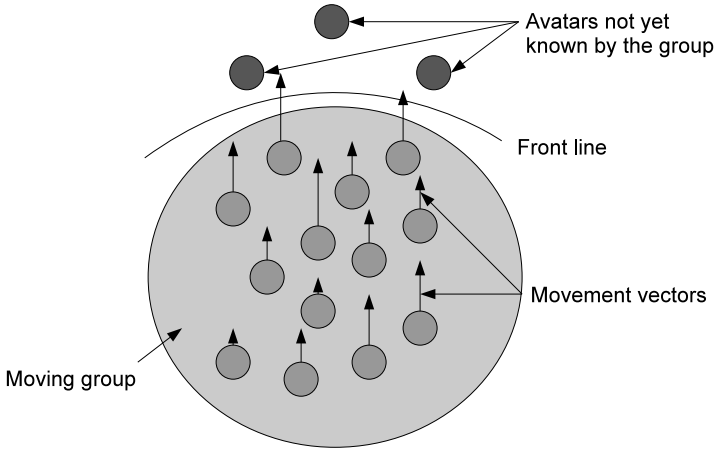


**Figure 5.5.** *Avatar group movements*

These characteristics can be considered while designing distributed systems for large-scale game applications. By taking them into account, it becomes possible to offer relatively low latency communications between peers that need to interact. The next section presents peer-to-peer overlay networks tailored for large-scale game applications based on these characteristics.

## 5.3. Overview of peer-to-peer overlays for large-scale game applications

As stated in the previous section, a client of a large-scale MMOG must only have a partial knowledge of the whole virtual environment: its playing area. The peer-to-peer model suits this kind of application well where each node only has a local view, and where the union of the local views offers a global view.

However, even if the peer-to-peer model seems well adapted, all peer-to-peer overlays are not suitable for MMOGs because they must ensure the playability of the game: communications between a node and its elders should be efficient. The underlying peer-to-peer network overlay must, therefore, limit the number of hops between a node and its elders.

In this section we present an overview of the different overlays, from the most *inflexible* to the most *malleable* and highlight their qualities and faults for playability.

### 5.3.1. *Unadaptable/inflexible overlays*

Considerable research effort has been conducted in the past decade in the field of peer-to-peer overlay networks. As presented in the previous chapter, these overlays have mainly been designed for *one* specific target application: large-scale read-only file-sharing. These overlays are supposed to build a graph that connects all the nodes together and permits efficient random search operations. Nodes choose the function of their neighbor's hash number if the overlay is based on a DHT or randomly if the overlay is unstructured [ORA 01, STO 01]. Except in case of failure, the neighbors of a node do not evolve over time because read-only file-sharing applications do not have this need: searches are supposed random and a fixed random overlay is sufficient.

Building a distributed virtual environment over of a random inflexible overlay is a hard task. First, a node and its elders have no reason to be close in the overlay while they communicate. Moreover, as presented in the previous section, the elders of a node evolve with the movement of the node's avatar. In these inflexible overlays, the neighborhood of a node does not change and a node can, therefore, not choose the closest neighbors to its elders in terms of number of hops.

Varvello *et al*. [VAR 09b] implemented a virtual environment over a DHT. The virtual world is partitioned into knowledge areas that are replicated among the DHT. The authors show that the responsiveness of the DHT is acceptable with minimal avatar mobility but not if mobility increases. In this case, implementing a reverse binary tree on top of a DHT could help to lower the latency [VAR 09a].

Colyseus [BHA 06], a decentralized architecture to support MMOGs with tight latency constraints (typically FPS games), which is detailed in section 5.4, is also based on a DHT for virtual object discovery. At the storage level, Colyseus prefetches objects.

Donnybrook [BHA 08], the sequel of Colyseus, also implements advanced dead reckoning techniques. These techniques predict the position of an avatar based upon its previously determined location. Dead reckoning techniques avoid a lot of messages to update avatar locations and, therefore, decrease the network load. Donnybrook is described in more detail in section 5.4.

### 5.3.2. *Overlays reacting to application needs*

Recent works have focused on dynamically adapting the overlay to better satisfy application needs. For instance, semantic overlays [VOU 04] build links between semantically close peers. These allow semantically close peers to be close in the overlay

(in terms of number of hops), which is a good because they are likely to interact to exchange data. Few recent overlays adapt their structure function to the application needs [MON 06, VOU 06, HU 06]: they react to the evolution of the application, generally by detecting communication between nodes.

Recently, some research effort influenced by these works has focused on building overlays tailored for distributed virtual environments. In such systems, the logical neighborhood of a node in the overlay is determined by the playing area of its avatar in the virtual environment: the overlay tries to keep the elders of the node in the neighborhood of the node. As an avatar moves in the virtual environment, the playing area of its node evolves and the overlay reacts by choosing the new relevant elders as neighbors. Figure 5.6 illustrates how a peer-to-peer overlay adapts itself when an avatar moves in the virtual world. The top of the figure presents the virtual environment with the avatars, and the bottom, the overlays with the nodes. Each node is connected to the avatars located in its playing area. When node A moves, its playing area changes and it therefore chooses new neighbors in the overlay.

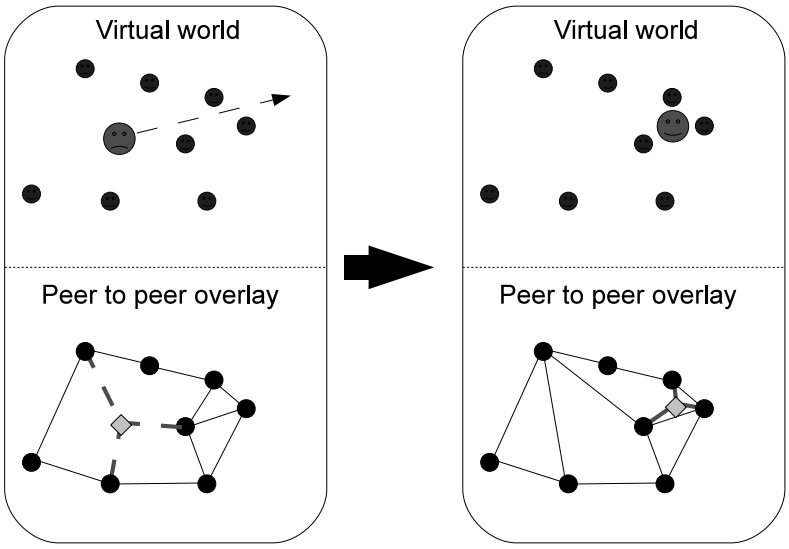


**Figure 5.6.** *Peer-to-peer overlay network malleability*

Recently, several of these overlays have been designed. Two main families have been proposed. The first is based on an unstructured overlay. This is the case for Solipsis [KEL 03], which is detailed in section 5.5.

The second is based on a structured overlay based on the virtual environment. They use a Voronoi tessellations-based overlay. The n-dimensional applicative naming space is divided into tiles centred around each avatar. Figure 5.7 presents a Voronoi

tiling in a two-dimensional space where a point represents an avatar. The sizes of the tiles are dynamically computed function of the object density: the number of objects in each tile is approximately the same. VoroNet/RayNet or VON [BEA 07a, BEA 07b, HU 06, HU 04] are MMOG overlays based on Voronoi. A tile is the knowledge area of a node. The main strength of these overlays is the efficiency of search queries of objects around a given coordinate: the query is forwarded along the shortest possible path. However, maintaining this structure is costly because avatars are always moving.

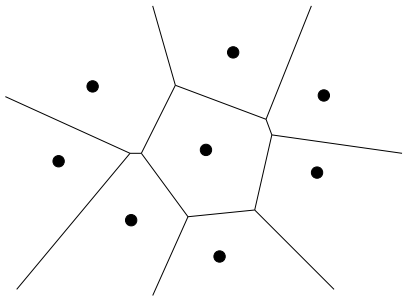

**Figure 5.7.** *Voronoi tessellations arround players' avatars*

#### 5.3.3. *Anticipating avatar movement to pro-actively adapt the overlays*

Previous overlays react to avatar movements but they do not try to anticipate these movements to pro-actively adapt the overlay.

An avatar movement anticipation module called Blue Banana has been proposed in [LEG 10]. This anticipation mechanism can be implemented on top of the distributed MMOGs presented in the last subsection. Blue Banana detects the movement of an avatar, and tries to prefetch neighbors located ahead of the anticipated movement. Blue Banana is detailed in section 5.5

### 5.4. Overlays for FPS games

FPS, such as Quake, require very low latency: in FPS games, the fact of winning or loosing is often based on player reactivity. If a player dies before the avatar that shot him enters in its playing area, the game becomes unplayable.

However, this kind of game rarely involves a few thousand participants. The scale is usually tens or hundreds than rather thousands and a DHT is sufficient.

This presents two main research efforts in this direction: Colyseus [BHA 06] and Donnybrook [BHA 08]. These systems are based on DHTs that support range queries, we first introduce Mercury [BHA 04] which is used by Colyseus.

### 5.4.1. *Malleable overlay for range queries*

Classical peer-to-peer DHT, as presented in section 3.2.2, allow efficient put/get operations. However, the key-based storage is a bit simplistic: it does not provide either range queries, or multi-attributes queries.

Mercury [BHA 04] is a peer-to-peer system allowing both range and multi-attributes queries. To support multi-attributes queries, Mercury organizes peers in multiple logical DHTs: one per attribute. Then, to handle range queries easily within each logical ring, Mercury arbitrary assign ranges to peers. Therefore, each peer becomes responsible for a sub-portion of the semantic space.

Within a virtual environment, objects are not likely to be distributed uniformly in the space. Thus, a peer may quickly become overloaded if the portion it is responsible for contains too many objects. Therefore, Mercury proposes a smart load balancing mechanism: each peer periodically probes the global naming space to identify high-density zones. Underloaded peers change their hash number to join high-density zones. Peer distribution thus converges to the objects distributed in the virtual environment. This overlay is *malleable*: it adapts automatically to the object distribution over the DHT.

However, when the distribution changes (e.g. when objects move) the load may change quickly. This design is not tailored for high variations.

### 5.4.2. *Colyseus*

Colyseus [BHA 06] is built on top of the Mercury DHT. It is a peer-to-peer system tailored for FPS games. It is based on a publish/subscribe mechanism: each peer publishes its avatar state in the DHT, and each peer subscribes to the area they are interested in (their playing area). Each time an avatar moves, the avatars in its playing area are notified because they have subscribed to the zone in which it moves.

However, as lookups in DHTs may be too slow, Colyseus offers (i) a caching mechanism and (ii) a prefetch mechanism. Therefore, objects may be discovered quickly: based on locality and predictability in data access patterns, Colyseus speculatively prefetches objects. Furthermore, this mechanism is only used for discovery. Once discovered, objects are cached and direct links are drawn between both the primary copies (in the DHT) and the caches.

Colyseus has been tested with Quake II [1] with hundreds of players.

1. http://www.idsoftware.com/business/index.php

### 5.4.3. *Donnybrook*

Donnybrook [BHA 08] has been designed based on Colyseus. It is also a peer-to-peer system tailored for FPS games.

It is based on the fact that the human brain cannot pay attention to too many objects simultaneously. Each peer, at each period, selects only five objects in its playing area. Then these objects are kept up-to-date with high frequency in order to ensure a good freshness quality until the next period. During the next period, a new subset will be elected.

The selection of these five objects is based on three criteria that allows the computation of the objects requiring high-fidelity rendering:

– physical proximity: a close object is more likely to attract attention than a distant one;

– visibility: an object in the center of the screen attracts the player's attention more;

– temporal proximity: an object having already attracted the player attention will probably also attract his attention during the next period.

Other objects located in the player's playing area are also kept up-to-date but with a lazy mechanism: they are synchronized every second with the primary copy in the Donnybrook system. Such a period may lead to inconsistencies: objects move continuously, they not make "jumps" each second. To deal with these inconsistencies, Donnybrook uses *dead reckoning* techniques [PAN 02]: the player's nodes simulate behavior of each object between two synchronizations. Therefore, the behavior of these objects is approximate, but it is not critical: they are only secondary objects which are not in the center of the player's attention.

These approximations enable the network to have a reduced load and to enhance the whole scalability of the system. They allow Donnybrook to support hundreds, and even thousand of simultaneous players in Quake II.

To summarize, DHT can be efficient to build overlays for MMOGs if (i) the number of player remains limited, (ii) each object in a playing area of a node is cached, (iii) only important objects are accurately updated, and (iv) the movement of other objects are predicted.

## 5.5. Overlays for online life-simulation games

Life-simulation games are characterized by (i) their potentially very large scalability; (ii) the lower importance of offering small latencies. Indeed, hundreds of thousands of players may play simultaneously, being spread all over the world. To offer a

pleasant gaming experience, latencies should remain low, but this is not as critical as it is for FPS games.

Peer-to-peer systems for life simulation can be classified in two main classes:

1) overlays where the knowledge area of a node does not depend on its coordinate. Knowledge areas are usually a statically distributed function of their hash key on a DTH [IIM 04, VAR 09b, VAR 09a];

2) overlays where the knowledge area of a node is centered around its coordinate. They use either unstructured overlays [KEL 03] or Voronoi tessellations [BEA 07a, BEA 07b, HU 06, HU 04]. In these systems, the playing area is included in, or coincides with, the knowledge area.

This section focuses on one of the state-of-the-art overlays for online life simulation games: Solipsis [KEL 03] and its extension with Blue Banana [LEG 10].

### 5.5.1. *Solipsis overview*

Solipsis is an overlay designed to sustain a distributed virtual environment. Each node of the Solipsis overlay is responsible for one avatar. In Solipsis, the knowledge area and the playing area coincide. The objects of a playing area are, therefore, replicated on the nodes that manage the avatars within this playing area. In Solipsis, elders of a node are in its playing area: they are the nodes that manage the objects in the playing area.

Solipsis maintains a set of direct neighbors for each node. Nodes communicate using message passing through the overlay: latency increases with the distance (measured in number of hops) in the overlay. Solipsis tries to maintain its elders in its neighborhood to communicate efficiently. If two avatars $A$ and $B$ are neighbors in the virtual environment, the Solipsis overlay adapts itself so that $B$ will *eventually* be in $A$'s neighborhood and vice versa. In order to ensure that behavior, Solipsis is based on two fundamental rules:

1) *local awareness*: an avatar $a$ has a circular playing area $\omega_a$ centered on the avatar. If another avatar $b$ is inside $\omega_a$, the nodes of $a$ and $b$ must be neighbors in the overlay. The size of $\omega_a$ is dynamically adjusted to ensure that $a$ has a number of neighbors contained between a minimum and a maximum limit. When a node enters a high-density zone, $\omega_a$ increases, and when it enters a low-density zone, $\omega_a$ decreases;

2) *global connectivity*: let $N_c$ be the neighbor set of a node $c$ in the overlay. The avatar of $c$ must be located inside the convex hull of the set formed by avatars of $N_c$. Figure 5.8 illustrates this property. On the left, $c$ is located inside the convex hull and on the right, after a move of the avatar, the property is broken. This property ensures that the avatar does not neglect a part of its playing area, causing (i) inconsistent

rendering and, (ii) possibly partitioning the Solipsis overlay graph (and, therefore, making parts of the virtual world unreachable).

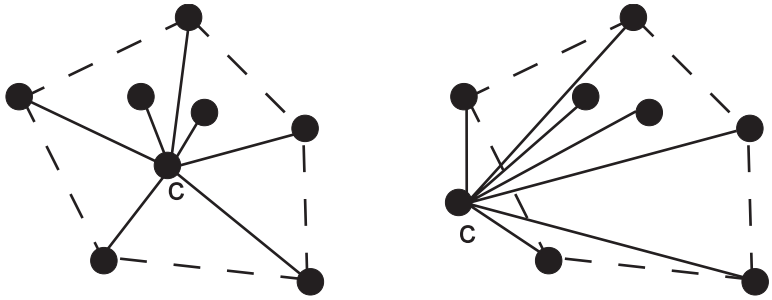


**Figure 5.8.** *While a peer moves its convex hull has to be reconstructed*

To ensure these rules, Solipsis implements a mechanism called *spontaneous collaboration*. At each moment, thanks to periodic updates, a node sends its coordinates and the size of its knowledge areas to its neighbors. When a node detects that one of its neighbors enters the knowledge area of another of its neighbors, it sends a message to both entities to warn them that the local awareness rule is about to be broken.

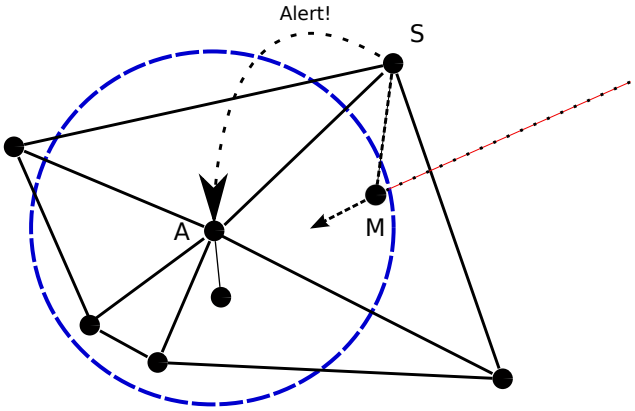


**Figure 5.9.** *Solipsis spontaneous collaboration mechanism*

This algorithm is illustrated in Figure 5.9. A and M are initially in the knowledge area of S. S sends an alert to node A and M to warn them when it detects that M enters the knowledge area of A. As they receive that message, the two entities (A and M in Figure 5.9) become neighbors. Simulations showed that this technique is very efficient: most of the time, a node receives a warning message and does not have to initiate a costly new-neighbor query. The global connectivity rule ensures that

a node is always surrounded by its neighbor set, making spontaneous collaboration more efficient.

To conclude, if the local awareness rule is violated for a node $n$, it means that an avatar has arrived in the playing area of $n$ and is not yet included to the local knowledge of $n$, causing a transient failure. If the global connectivity rule is violated for a node $n$, it means that $n$ is not surrounded by its neighbor set. It will then not receive spontaneous data updates for a part of its playing area, which will necessarily lead to transient failures.

An avatar keeps breaking fundamental rules as long as it moves because the spontaneous collaboration mechanism is not always able to react on time. For that reason, a more efficient anticipation mechanism is required.

### 5.5.2. *Anticipating avatar mobility to pro-actively adapt the overlay*

Blue Banana [LEG 10] is an extension of Solipsis that pro-actively adds new neighbors "ahead" of the player by anticipating its movements. For this purpose, it finds nodes in the direction of the avatar's movement and add them to a new set called the prefeteched set. Once the moving avatar approaches a prefetched node, the prefetched node is added to the regular neighbor set of Solipsis that defines the overlay topology. Hence, Blue Banana substantially helps native algorithms in Solipsis to maintain the fundamental rules, minimizing resulting transient failures.

*Algorithm description*

Technically, if the algorithm observes that the avatar of a node B (for Blue Banana) is moving fast and straight, and if the prefetched neighbor set is not full, B starts searching for new prefetched neighbors in the movement direction. It sends a message to its neighbor that is closest to its *movement vector* as illustrated by Figure 5.10. The message contains the number of prefetched neighbors that B is willing to retrieve (called the TTL), the position of B, its direction, and its speed.

As B is moving during the message transfer, upon receipt of a prefetching request on a node R (for Receptor), R estimates the probable position p of B when B will receive a response from R. For this purpose, R estimates the network latency and uses the initial position and the speed of the avatar. It then builds a probability cone that begins at the apex p and that has the direction of B. The shape of the cone decreases with B's speed. R responds to B with all its neighbors (itself included) that are in this cone. R then sufficiently decreases the TTL and if the TTL remains greater than zero, R forwards the request to its neighbor closest to B's vector movement.

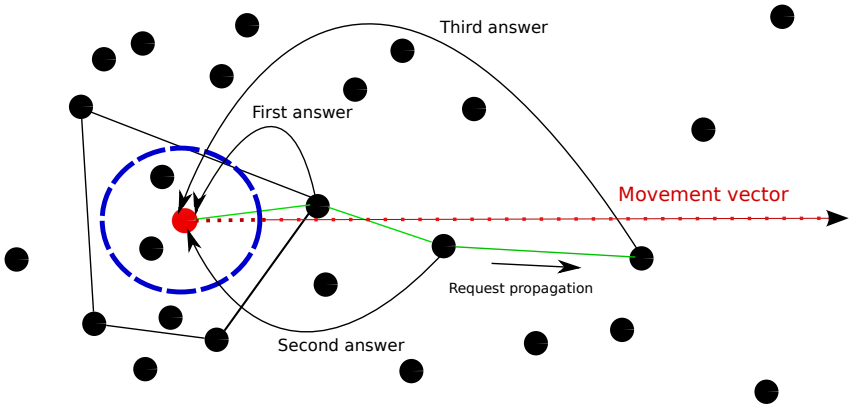


**Figure 5.10.** *Blue Banana prefetching algorithm*

*Network overhead*

Blue Banana does not interfere with the maintenance protocol of Solipsis: the prefetched neighbors are not placed in the regular Solipsis neighbor set, but in a separate one. Therefore, Solipsis does not use network resources on maintaining links with prefetched neighbors. Blue Banana does not spend network resources to maintain a link with a node that is useless *in the present* as it is not yet in the playing area.

As a consequence, once inserted in the prefetched neighbor set, the position of a node's avatar is not updated, while it can move outside the probability cone. Blue Banana automatically removes useless prefetched neighbors (i) when they have been overtaken by the moving avatar, (ii) when the avatar changes its direction or (iii) when the avatar slows down. It is possible to consider another policy by periodically updating the state of the prefetched neighbors. However, there is a risk of spending network resources to update possibly useless nodes.

In order to compensate the small network overhead, Blue Banana nodes take advantage of the high predictability of the avatar movement in desert zones. In Solipsis, a node periodically propagates the coordinates of its avatar to all the members of its neighbor set, so the neighbor-nodes can update their view of the virtual environment. Blue Banana doubles the period of such updates for nodes when it detects the avatar is moving fast and straight. The neighbors of that node simply predict the position of the avatar between two updates by using its initial position and its speed. This technique is a simple form of *dead reckoning* [2], but it could easily be enhanced with more sophisticated mechanisms widely used in online gaming [BHA 08, PAN 02, PAN 07].

2. Recall that dead reckoning is the process of estimating one's current position based upon a previously determined position.

To summarize, life-simulation games do not exhibits hard constraints in term of game reactivity. However, the number of players involved in this kind of game can reach hundreds of thousands. The scalability therefore becomes a problem and a DHT becomes inefficient. Overlays tailored for MMOGs attempt to solve this scalability problem by (i) building knowledge areas that correspond to player areas and by (ii) using node's elders as neighbors in the network. These topologies require frequent updates when the avatars move and, therefore, require the addition of neighbor prefetching mechanisms to increase the gaming experience.

## 5.6. Conclusion

Nowadays game applications become important both in terms of number of the users and in term of the financial impact. Successful games are now multi-player: hundreds of thousands of players are spread all over the world [LAB 11, ENT 11]. Building the software infrastructure for these applications is still challenging.

The currently deployed systems are based on the classical "client-server" paradigm and do not scale with the number of nodes. Additional rules have been introduced to limit the number of players interacting on the same set of servers at a given time (e.g. Second Life "islands" or World of Warcraft "realms" [PIT 07]).

Classical overlays built for file-sharing applications also do not scale with the number of players. However, FPS games that exhibit real-time constraints that are successfully built on DHT.

Only really recent research work has proposed overlays tailored for MMOGs. These overlays try to minimize the latencies between nodes and their elders by moving them closer in the overlay. For this purpose, they identify the knowledge and the playing areas. These overlays react to player movements. Observing avatar movements to prefetch neighbors in the movement direction, which minimizes the reaction time.

*Open challenges*

New overlays tailored for MMOGs minimize the communication latencies but these systems only provide *best effort* "guarantees". Therefore, how are cheaters dealt with in such environments? How is players collusion avoided? It is mandatory to ensure (i) privacy: unauthorized information must not be obtain about another avatar; (ii) integrity: the game rules have to be respected; (iii) availability: the system has to remain responsive. Few recent research works focus on these issues, however, these are still open issues [NEU 07].

We believe that the next generation of peer-to-peer overlays for large-scale game applications will be multi/complex overlays: malleable overlays such as Solipsis or, such as those based on Voronoi tessellations are well adapted for local interactions,

however they are not suitable for global search or publish/subscribe. However, such "global" operations are likely to be needed: e.g. to maintain virtual bank accounts, to count player "credits", or more simply, to find information within a virtual environment.

Research in this field is still very active; new systems will emerge in the coming years.

## 5.7. Bibliography

[ALV 07] ALVES R. T., ROQUE L., "Because players pay: the business model influence on MMOG design", AKIRA B., Ed., *Situated Play: Proc. of the 2007 Digital Games Research Association Conference*, Tokyo, The University of Tokyo, p. 658–663, September 2007.

[BEA 07a] BEAUMONT O., KERMARREC A.-M., MARCHAL L., RIVIERE E., "VoroNet: a scalable object network based on Voronoi tessellations", *21th International Parallel and Distributed Proc. Symposium (IPDPS 2007), Long Beach, USA*, p. 26-30, March 2007.

[BEA 07b] BEAUMONT O., KERMARREC A.-M., RIVIERE E., "Peer to Peer Multidimensional Overlays: Approximating Complex Structures", TOVAR E., TSIGAS P., FOUCHAL H., Eds., *OPODIS*, vol. 4878 of *LNCS*, Springer, p. 315-328, 2007.

[BHA 04] BHARAMBE A. R., AGRAWAL M., SESHAN S., "Mercury: supporting scalable multi-attribute range queries", *SIGCOMM Comput. Commun. Rev.*, vol. 34, num. 4, p. 353–366, ACM, 2004.

[BHA 06] BHARAMBE A., PANG J., SESHAN S., "Colyseus: a distributed architecture for online multiplayer games", *NSDI'06: Proceedings of the 3rd conference on Networked Systems Design & Implementation*, Berkeley, CA, USA, USENIX Association, p. 155-168, 2006.

[BHA 08] BHARAMBE A. R., DOUCEUR J. R., LORCH J. R., MOSCIBRODA T., PANG J., SESHAN S., ZHUANG X., "Donnybrook: enabling large-scale, high-speed, peer-to-peer games", BAHL V., WETHERALL D., SAVAGE S., STOICA I., Eds., *SIGCOMM*, ACM, p. 389-400, 2008.

[BRU 02] BRUNET R., "Lignes de force de l'espace Européen", *Mappemonde*, vol. 66, p. 14-19, 2002.

[CHE 06] CHECHKIN A., GONCHAR V., KLAFTER J., METZLER R., "Fundamentals of Lévy flight processes", *Advances in chemical physics*, vol. 133B, p. 439-496, 2006.

[ENT 11] BLIZZARD ENTERTAINMENT INC, "World of Warcraft", http://us.battle.net/wow/en/, January 2011.

[FRE 08] FREY D., ROYAN J., PIEGAY R., KERMARREC A., ANCEAUME E., FESSANT F. L., "Solipsis: a decentralized architecture for virtual environments", *The Second International Workshop on Massively Multiuser Virtual Environments at IEEE Virtual Reality (MMVE' 09 )*, Lafayette, USA, March 2008.

[HU 04] HU S.-Y., LIAO G.-M., "Scalable peer-to-peer networked virtual environment", *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, New York, NY, USA, ACM, p. 129–133, 2004.

[HU 06] HU S.-Y., CHEN J.-F., CHEN T.-H., "VON: a scalable peer-to-peer network for virtual environments", *IEEE Network*, vol. 20, num. 4, p. 22-31, Jully 2006.

[IIM 04] IIMURA T., HAZEYAMA H., KADOBAYASHI Y., "Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games", *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, New York, NY, USA, ACM, p. 116–120, 2004.

[KEL 03] KELLER J., SIMON G., "Solipsis: a massively multi-participant virtual world", *International Conference on Parallel and Distributed Processing Techniques and Applications – PDPTA*, p. 262-268, 2003.

[KUM 08] KUMAR S., CHHUGANI J., KIM C., KIM D., NGUYEN A., DUBEY P., BIENIA C., KIM Y., "Second Life and the new generation of virtual worlds", *Computer*, vol. 41, num. 9, p. 46-53, IEEE Computer Society, 2008.

[LA 08] LA C.-A., MICHIARDI P., "Characterizing user mobility in Second Life", *SIGCOMM 2008, ACM Workshop on Online Social Networks, August 18-22, 2008, Seattle, USA*, August 2008.

[LAB 11] LAB L., "Second Life", http://secondlife.com/, January 2011.

[LEG 10] LEGTCHENKO S., MONNET S., THOMAS G., "Blue Banana: resilience to avatar mobility in distributed MMOGs", *The 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010)*, Chicago, USA, July 2010.

[LIA 08] LIANG H., TAY I., NEO M. F., OOI W. T., MOTANI M., "Avatar mobility in networked virtual environments: measurements, analysis, and implications", *CoRR*, vol. abs/0807.2328, 2008.

[MIL 09] MILLER J. L., CROWCROFT J., "Avatar movement in world of warcraft battlegrounds", *Netgames 2009*, IEEE, November 2009.

[MON 06] MONNET S., MORALES R., ANTONIU G., GUPTA I., "MOve: Design of An Application-Malleable Overlay", *Symposium on Reliable Distributed Systems 2006 (SRDS 2006)*, Leeds, UK, p. 355-364, October 2006.

[NEU 07] NEUMANN C., PRIGENT N., VARVELLO M., SUH K., "Challenges in peer-to-peer gaming", *SIGCOMM Comput. Commun. Rev.*, vol. 37, num. 1, p. 79–82, ACM, 2007.

[ORA 01] ORAM A., "*Peer-to-Peer: harnessing the power of disruptive technologies*", Chapter Gnutella, p. 94-122, O'Reilly, May 2001.

[PAN 02] PANTEL L., WOLF L. C., "On the suitability of dead reckoning schemes for games", WOLF L. C., Ed., *NETGAMES*, ACM, p. 79-84, 2002.

[PAN 07] PANG J., UYEDA F., LORCH J. R., "Scaling peer-to-peer games in low-bandwidth environments", *IPTPS '07: Proc. of the 6th International Workshop on Peer-to-Peer Systems*, February 2007.

[PIT 07] PITTMAN D., GAUTHIERDICKEY C., "A measurement study of virtual populations in massively multiplayer online games", *NetGames '07: Proc. of the 6th ACM SIGCOMM*

*workshop on Network and system support for games*, New York, NY, USA, ACM, p. 25–30, 2007.

[RHE 08] RHEE I., SHIN M., HONG S., LEE K., CHONG S., "On the Levy-Walk nature of human mobility", *INFOCOM*, IEEE, p. 924-932, 2008.

[STO 01] STOICA I., MORRIS R., KARGER D., KAASHOEK F., BALAKRISHNAN H., "Chord: a scalable peer-to-peer lookup service for internet applications", *Proceedings of the Symposium on Communications Architectures and Protocols (SIGCOMM '01)*, San Diego, USA, p. 149-160, August 2001.

[VAR 09a] VARVELLO M., DIOT C., BIERSACK E. W., "A walkable Kademlia network for virtual worlds", *Infocom 2009, 28th IEEE Conference on Computer Communications, April 19-25, 2009, Rio de Janeiro, Brazil*, 04 2009.

[VAR 09b] VARVELLO M., DIOT C., BIERSACK E. W., "P2P Second Life: experimental validation using Kad", *Infocom 2009, 28th IEEE Conference on Computer Communications*, Rio de Janeiro, Brazil, p. 19-25, April 2009.

[VOU 04] VOULGARIS S., KERMARREC A. M., MASSOULIE L., VAN STEEN M., "Exploiting semantic proximity in peer-to-peer content searching", *10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004)*, Suzhou, China, May 2004.

[VOU 06] VOULGARIS S., RIVIERE E., KERMARREC A.-M., VAN STEEN M., "SUB-2-SUB: self-organizing content-based publish and subscribe for dynamic and large scale collborative networks", *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS)*, Santa Barbara, USA, February 2006.