This documents summarizes the concepts and ideas on the Solipsis protocol, discussed in Emmanuelle's office on November 22, plus some details that I thought about while writing it.

1 Basic Definitions

1.1 Solipsis Objects

The Solipsis world consists of a set of *objects*. Each object is associated with a unique identifier (UID) and may be classified into groups: *roaming* or *static*[1]. Roaming objects may roam around the virtual world, and interact with it. Static objects on the other hand are associated with a fixed position in the virtual space. In this version of the specification, we consider only one type of roaming and one type of static objects, respectively, *avatars* and *sites*.

1.2 Solipsis Nodes

A *Solipsis node* is defined as a logical instance of the *Solipsis* application, running on a physical host. Each object in the solipsis world is associated with a unique node, referred to as its *owner*. Moreover, we require that each host be the owner of *at least one* site in the *Solipsis* world. Finally, as for *Solipsis* objects, each node is associated with a unique identifier (UID).

Objects may be described with several levels of details. In this version of the specification, we concentrate on a very coarse-grained subdivision between two such levels. Nonetheless, the protocol can easily support several levels of details with only minor modifications. The two levels we consider consist of a complete description of the object and of a an object descriptor. The former contains all the information necessary for the 3D visualization of an object, plus any sounds, videos, or multimedia content associated with it. From the point of view of this protocol, such a representation merely consists of a binary object that should be provided to the rendering component and of an associated version number. The object descriptor, on the other hand, consist of the minimal information required to display a rough representation of the object and includes the object's location, its shape, its size, and its visibility radius. The shape of the object is represented by an identifier that selects one from a set of predefined shapes; while its visibility radius identifies the distance from which the object should be visible in the absence of obstacles and without any visual aids.

2 Protocol Overview

The *Solipsis* protocol is responsible for managing the information regarding sites and avatars, for maintaining it persistent and for making it available to *Solipsis* nodes, according to the requests of their users. In the following we first describe

[1] **Davide:** These were movable and unmovable

the main components of our protocol. Then we discuss how they are employed to store information regarding sites and avatars; and finally describe the main operations of the protocol.

2.1 Main Protocol Components

The architecture of the protocol, depicted in Figure **??**, outlines three major components that provide the basis for the protocol's operation: RayNet, a DHT and a cadastre service.

RayNet Nodes are organized in a three-dimensional overlay network based on the Raynet protocol. The arrangement of nodes in this overlay is based on the locations of the corresponding sites in the *Solipsis* world. In particular, each node is responsible for a section of the world centered around its own site and comprising a region of space around it. In simple terms, two nodes are neighbors in the overlay if their corresponding sites are neighboring in the *Solipsis* world[2]. This, mapping between overlay and 3D-world positions allows RayNet to provide efficient means to address and communicate with the nodes responsible for the sites encountered by avatars roaming in the *Solipsis* world.

DHT The second component visible in the protocol architecture is the Distributed Hash Table (DHT). While the RayNet overlay allows nodes to access data by means of location information, the DHT allows them to retrieve information about objects with a given identifier. This is essential to access information regarding avatars that are not, by definition, tied to a specific location in the *Solipsis* world. Moreover, it is also useful to access information about sites when their complete description has not yet been retrieved.[3] The DHT associates each UID with the corresponding object's descriptor and with a list of the nodes that have cached a copy of the object's complete description as described in Section 2.2.

Cadastre Finally, the last component highlighted in the figure is a *cadastre* service, responsible for managing the locations that are available for new sites in the *Solipsis* world. In this version, we assume a centralized version of the cadastre service, but we are actively working[4] on a fully distributed implementation. Nodes may contact the service for to (i) obtain a list of the available locations for new sites; and (ii) to obtain a permit to create a new site with specified dimensions around a given 3D location.[5]

[2] **Davide:** This is not strictly true with raynet, right?

[3] **Davide:** Using the DHT for sites is convenient but not essential. Let's make sure we know why we are doing this.

[4] **Davide:** We are, aren't we?

[5] **Davide:** Some signature mechanism would be nice here... how about some P2P signing mechanism... is there any such a thing? How about using quorums for that? Does this make any sense?

2.2 Maintaining Information about Sites and Avatars

The *Solipsis* protocol provides an effective means to store information about sites and avatars. The automatic replication of popular data makes the protocol resilient to the disconnection or failure of *Solipsis* nodes, and augments the availability of data by making it available from several sources.

3D-Models and Object Descriptors As we mentioned above, we consider two levels of details in the description of objects. The first is a complete description comprising the object's 3D model and any necessary information to completely render the object. The second is an *object descriptor* containing the object's identifier and a minimal set of information that may be used for collision detection or to display a simple sketch of the object. Specifically, a minimal node description should include the following information.

- location
- shape
- size
- visibility

A key feature of the *Solipsis* protocol is the use of the RayNet overlay to organize peers in a topology that mimics the distribution of their sites in the 3D world. Specifically, a network host is associated with a nodes in the RayNet overlay characterized by virtual coordinates that are the same as those of its site in the 3D world. If a host owns multiple sites in the 3D world, then it appears in the RayNet overlay as multiple nodes. Note that the presence of a host in multiple points on the overlay does not consitute a problem for the small-world routing strategies exploited by RayNet. On the other hand, it allows the overlay to perform additional optimizations by taking shortcuts on the physical network that would not be available otherwise.

Maintaining a node's own cell Throughout its lifetime, each node maintains the necessary information to manage the interaction with the avatars that are visiting its site or the corresponding cell. This includes the complete description of the site[6] as well as a list of descriptors of the objects that are visible from anywhere in the cell.

The list is built using either a reactive or a proactive protocol. [7]

The list is built through a simple protocol. Specifically, nodes inform their neighbors whenever they detect a change

[6] **Davide:** Do we want to replicate to neighbors when the site goes beyond the border of a cell.

[7] **Davide:** describe the protocol

periodically[8] send a copy of their list to their nerighbors in the RayNet overlay. Receiving nodes, process this list and add to their own list any node that is also visible from their own cells. To manage the disappearance of objects from the visible list, nodes tag each entry in the list with the hopcount from the node that originated the

ls ll the time at which the entry was last updated. Entries that are not refreshed after a timeout t_v are removed from the visible list.

node may start from the list objects that are currently in its own cell. From this, it can compute which of the objects from the list are visible from outside the cell, using the visibility attribute in the object descriptors. The node may then forward this information to neighboring nodes, which may then use it to update their to their own list and possibly re-propagate the information with the next update. [9]

The complete description of a site may also be cached by other *Solipsis* nodes as we describe in Section **??**.

A note on RayNet cells It is important to observe that the position of a node in the RayNet overlay is solely determined by location of the corresponding site in the virtual world and *not* by the size of the site or by the presence of multiple hosts at nearby locations. As a result, it is possible that the RayNet cell associated with a node comprises a region that covers only part of the corresponding object in the 3D world. This has no influence on the behaviour of the protocol for two reasons. First, RayNet only considers the neighborhood relationship between nodes and does not associate any size or weight properties with overlay nodes. Second, according to the above descriptions *Solipsis* nodes maintain information regarding their own site and the objects visible from their own cell, regardless of its size.

Start by describing arrangement of nodes in raynet

Each node always maintains complete information for all the sites and avatars it owns, while the *descriptor* is instead recorded in the DHT, together with some bookkeeping information as we describe in the following.

REWRITE HERE AND NEXT PAR, IT IS NOT CLEAR say that each node maintains complete description of owned objs node descriptor of objects visible from its cell (CELL MUST BE DEFINED BEFORE THIS POINT)

Each node is the primary maintainer of any information associated with its own site. As a result, it is the only node that is entitled to modify either the complete or the short description of the site. To manage consistency in the presence of multiple copies, nodes mark their complete descriptions and object descriptions with two version numbers, one for each description, that is incremented whenever the description is updated.

The owner of an object is the only node that is entitled to modify either the complete or the short description of an object. As soon as it makes such

[8] **Davide:** or whenever they detect changes: this may be a nice point in the evaluation if we manage to characterize one or a few usage scenarios for the *Solipsis* world.

[9] **Davide:** This gives an idea but a correct description. It needs rewriting.

an update, the owner node also marks the modified data with a fresh version number. Nodes other than the owner can instead access either description at any time in order to display the object; moreover, they may cache the complete description and make it available to other *Solipsis* nodes. Specifically, each node maintains a LRT (Least Recently Used) cache in which it records a copy of all the complete descriptions of the objects it displayed. An entry in the cache is marked as the most recent whenever it is displayed by the node or it is provided to another node that requested it. In order to make the availability of such cached copies known, any node that caches the complete description of an object records its identifier in the DHT along with the description's version number.

2.3 Protocol Operations

Next, we describe the high-level operation of the protocol in response to some relevant events that occur while managing the 3D world. Throughout our description, we make use of the terms avatar node and site node to refer to the two roles that may be played by a *Solipsis* node: respectively, being responsible for an avatar or for a site.

Joining Solipsis: Creation of a Site Users may join the *Solipsis* network through the creation of a new site. Such a site is permanently associated with the node through which it was created[10] and determines the location of the node in the RayNet overlay. More precisely, a node joining the *Solipsis* network for the first time should contact the *cadastre* service to obtain a new location for its site. Based on this location the node determines its position in the RayNet overlay by computing the set of its neighbors. From this moment on, the node becomes responsible for a region of space containing, at least, its own site and some of the space separating it from neighboring sites. In the following, we will use the term *cell* to refer to the region of space assigned to a node by such process.

At each instant, each node maintains a complete description of the sites included in its cell (these may include other nodes' sites).[11]. In addition, it also maintains a list of all the objects that are visible from its cell. To build this list, a node may start from the list objects that are currently in its own cell. From this, it can compute which of the objects from the list are visible from outside the cell, using the visibility attribute in the object descriptors. The node may then forward this information to neighboring nodes, which may then use it to update their to their own list and possibly re-propagate the information with the next update. [12]

As more nodes enter the network, their sites may occupy positions that are close to those associated with previously existing sites. In this case, RayNet

[10]Relocation of sites to new nodes and movement to different locations are not covered by this version of the specification.

[11]**Davide:** We need to explain this better.

[12]**Davide:** This gives an idea but a correct description. It needs rewriting.

automatically resizes the cells associated with existing nodes so as to create a new cell that will be assigned to the joining node. In doing this, it is possible that the approximate voronoy cell associated with the owner of the smaller sites may end up containing a portion of the larger site. In this case, the owner of the smaller site will be notified by the owner of the larger one about the presence of the site.[13]

Movement of an Avatar Similar to sites, avatars are also associated with the node that created them. However, their location is not fixed and they can freely roam around the virtual world visiting sites. As an avatar visits a site, its node should be provided with all the necessary information to display the avatar's current view of the site. This necessarily includes all the multimedia content associated with the site as well as any other displayable information associated with avatars that are currently in it. In addition, the node should also be provided with information regarding sites and avatars that are visible from the avatar's position. In the following, we describe the details of the protocol providing such information.

An avatar node roaming around the 3D world constantly maintains contact with the site node responsible for its current location. This is achieved by means of periodic beacon messages sent by the avatar node to the site node. As soon as a site node receives a beacon message from an avatar that was not previously in its cell, it should provide the avatar with a list of all the object visible from its current position. These include all of the objects in its cell as well as visible objects in surrounding cells. If the site node has available resources, it may also provide the avatar node with a complete descriptions of all or some the sites in its cell.[14] The avatar node can immediately start building a scene using these descriptions while it downloads more descriptions. To achieve this, it looks up each of the remaining objects in the DHT and downloads its descriptor. If a rough rendering is not sufficient[15], then it selects one of the nodes that are listed as having a copy of the latest version of the object and downloads the required data.

As side effect of an avatar's entering its cell, a site node should also update the list of objects in its cell and propagate the update to all the avatars that may see the newly arrived avatar. To achieve this it informs all the avatars that are currently in its cell. In addition, it evaluates whether the new avatar will be visible from neighboring cells and if so, it informs all the nodes responsible for those that are within the visibility radius of the avatar. A simple way to compute whether an avatar should be visible from neighboring cells is to evaluate the presence of the new avatar as a variation in the number of avatars

[13] **Davide:** Do we care? Possibly not, I need to think about it more thoroughly.

[14] **Davide:** I put the direct download of descriptions as an optional step and I am not even sure we want to keep it. The problem is that an avatar that enters a cell might already have the descriptions of the objects in the cell, either because it retrieved them from farther away or because it has been moving in and out of the cell.

[15] **Davide:** Here we need more input from rendering people.

already in the cell. If such a variation corresponds to a very low percentage[16], then the change will most likely go unnoticed and propagation can be avoided. If, on the other hand, the percentage of change is greater than a threshold, then the information about the new avatar is propagated to neighboring cells. When receiving such an update from a neighboring cell, a site node computes whether the update is going to affect the scene seen by the avatars in its and in neighboring cells and propagates the update as necessary.[17]

Scalability and Fault Tolerance The peer-to-peer nature of the *Solipsis* protocol makes it easy to guarantee that (i) information is likely to be retrieved even after the disconnections of the nodes that were previously hosting it and (ii) it can be downloaded efficiently even in the presence of a very large number of requests.

To achieve both of these goals, we exploit replication when storing information about objects in the 3D world. Specifically, each node may cache the complete description of each site it displays and record this fact in the DHT, along with the version number associated with such information. ***Davide:*** *Here we need to determine whether we want to deal with partial views.* When downloading the complete description of an object, a node may therefore choose among the nodes that have an up-to-date replica, by examining the object's entry in the DHT. A possible metric for choosing the best node to download from may be the average load on nodes that have the desired data.

Disconnection of a Node Replication is also the key mechanism enabling the *Solipsis* protocol to maintain persistence in the event of a failure or disconnection of a node associated with one or more sites. Addressing such an event requires that some nodes replace the disconnected node in (i) providing a complete description about the sites[18] owned by the node; and (ii) managing the list of avatars that are currently visiting the failed node's cell.

Goal (i), can easily be achieved by exploiting the replication mechanism described above. First, we need to observe that the disconnection of a node may cause an automatic reconfiguration in the RayNet overlay, determining a change in the size of the cell associated with a node. As soon as a node detects such a change, it uses the DHT to retrieve the list of sites owned by the disconnected node. It should then go through this list to determine whether any of the sites are now located in its own region. If this is the case, it contacts one the nodes that are listed as having a copy of the sites and downloads it.

[16] **Davide:** need to define this
[17] **Davide:** Expand...
[18] **Davide:** What happens to the avatars?