Automatic construction of dynamic 3D metaphoric worlds: An application to network management

P. Abel[a], P. Gros*[a], C. Russo Dos Santos[a], D. Loisel[a], J.P. Paris[b]
[a]Eurecom Institute, Sophia-Antipolis, 06904, France
[b]France Telecom-CNET, Sophia-Antipolis, 06921, France

ABSTRACT

In this paper we present the CyberNet research project. The aim of this project is to study how 3D visualization may help the user in the process of monitoring large amounts of dynamic information. The project focuses on a specific application domain, network management, but it is designed so that it can be applied to other domains. The paper presents how information is collected and structured in order to define services that the user wants to monitor. We also present the impact of the dynamic nature of information on the system. Each 3D world represents a service, that is a model that groups and structures all the information related to a specific aspect of the network. Each modification of the service has a real time impact on the virtual world. This representation is done within the context of a 3D metaphor that defines the rules used to map services onto graphical elements (layout managers and glyphs). It also defines how information is mapped onto the visual parameters (color, size, shape, etc.) of these graphical elements. The metaphor also handles suitable navigation and interaction mechanisms to allow the user to explore and manage the virtual world.

Keywords: Dynamic information visualization, 3D Metaphors, Tuplespaces, Network management, Interaction

1. INTRODUCTION

Nowadays there are domains, such as the Stock Exchange, where it is necessary to monitor large amounts of dynamic information. The goal of our project is to propose a generic system to solve the problems encountered in this context: how to collect, structure and visualize data that the user wants to see in real-time? In this paper, we focus particularly on the visualization part of this project, that studies how 3D metaphoric worlds with enhanced user interface can help the user visualize large amounts of information

Our system is presently used in a network management environment, in a project called CyberNet. This is an interesting domain of application because network managers have to monitor distributed applications running over networks made of thousands of nodes. In addition, information needs to be structured in terms of services rather in terms of devices. The term service is very comprehensive and may cover Topology, Connectivity, Routing, DBMS, NFS, Mailing, etc. The status of a service can generally be understood by analyzing large amounts of data distributed on the network devices. Thus there is a need for a tool that helps monitor and analyze these services. Each visualization represents the current status of a monitored service and continuously evolves in order to reflect any change. Some prior work has been already done in the field of using virtual world for network management, namely in [9].

We have defined a distributed object framework in order to cope with the problems that we were faced with: distributed nature of the data, dynamic nature of the data, structuring raw data and visualization using a simple VRML-enabled Web browser. This framework is composed of three distinctive parts that are presented in this paper.
The collecting layer is used to gather raw network data from the monitored devices.
The structuring layer is the kernel of the system. It structures raw information according to the services being monitored.
The visualization layer defines 3D metaphors that allow to construct a visualization and provide user interaction. It is also in charge of mapping the structured information onto the chosen metaphor.

* Correspondence: Email: gros@eurecom.fr; WWW: http://www.eurecom.fr/~abel/cybernet/

2. COLLECTING INFORMATION

The first task of the CyberNet system is to collect the information required to describe the monitored service. In network management, that information is spread over a large number of network devices. Traditional collecting techniques (e.g. SNMP) require the manager to pull the data from the collecting agents in the network. They are not suited to our needs since they would largely increase the network bandwidth, requiring that the data be pulled at extremely short intervals in order to present a real-time visualization to the user.

In order to reduce the generated traffic, we had to develop our own collection strategy. The collecting process is implemented using distributed agents located on or near the observed devices (see figure 1). These agents are designed so that they "push" information to subscribers according to predefined policies. These policies are based on frequency, freshness, percentage of modifications, etc. Each agent is registered in a Yellow Pages service with the kind of information it can provide. When a user wants to monitor a service, the structuring layer locates informators responsible for collecting the necessary data (via a Yellow Pages mechanism) in order to monitor that particular service and subscribe to these informators according to the defined policies. Once the configuration is completed, informators start "pushing" every network modification falling into these policies criteria to the structuring layer. These agents may use any method (SNMP, HTTP, scripts...) for retrieving information, but the important thing to be retained is that they push relevant-only information to the upper part of the system, conforming to the notification policy.

This agent-based mechanism can be generalized and applied to other application domains. The only change is the way the information is collected. For instance, in a Stock Exchange visualization application, one can imagine agents that examine some kind of share and send any modifications to its subscribers.

Data that are pushed to the structuring layer are not raw data. Informators push *entities*, which are the first form of structured information in our system as we will see in the next section.

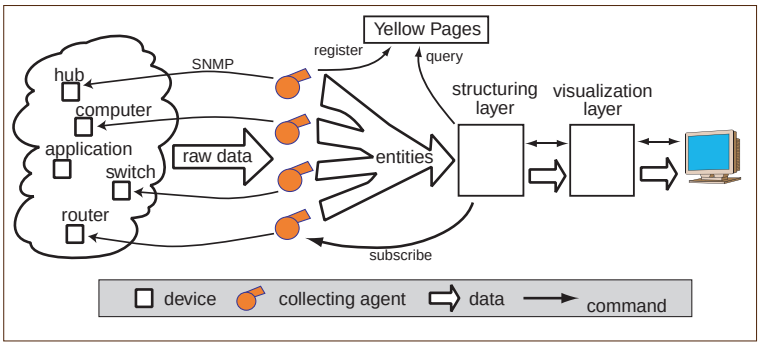


Figure 1: collecting information

3. STRUCTURING DYNAMIC INFORMATION

Structuring information is of prime importance since 3D representations will automatically be built from the structured data. The goal of the structuring layer is to build a service tree that will be directly exploited by the visualization layer. The reason to limit the topology of the service graph to a tree is that it is easier to understand[5] and its translation into a 3D scene hierarchy can be straightforward. This requires several steps of *data transformation* according to the terminology of Card et al[6]. We will first describe these structuring mechanisms in a static environment. We will then introduce the additional requirements of a dynamic information environment.

3.1. Static environment

Several structuring operations on raw data are required before the service tree can be built. These data transformation stages are based on 3 layers that allow the data to be grouped and filtered (see figure 2). In a first stage, agents structure raw data as entities. In a second stage, entities can be filtered and grouped to form relations. Service trees are finally built using relations.

Entities group the collected information. There are several *types* of entities that can represent physical devices (e.g., hubs, routers, computers, etc.) or conceptual items (e.g., connections, processes, etc.). Each entity groups all the collected values necessary to describe the current state of the element it represents. For example, a workstation entity may include the machine name and IP address, its available memory, the CPU load, the internal disk size, etc.

Relations are groups of entities of the same type defined through filtering. Selection criteria can be defined for each attribute of an entity. For example, it is possible to define a relation grouping all the processes of a user on a network or all the printers on a given storey of a building. A relation may involve any number of entities and an entity may be involved in any number of relations. Therefore, entities can be shared effectively between several services.

Services use entities to build an acyclic graph, i.e. a tree, and organize the information to be visualized. Every node in the tree has got, at least, one relation and therefore is aware of the entities involved in that relation. Depending on the entities contained in a relation, a node may decide to add a new node to the tree. Upon receiving a workstation entity, a service may for example create a child node in charge of building a new relation containing all the users of that particular workstation. The relationship between a node and one of its children may be structural (e.g. a process is related to the workstation it is running on) or functional (e.g. a client/server relationship between two workstations.)

In a context of static information, the only requirement for the good operation of this system is a global repository where all entities are referenced by type and attributes. This repository is used by the relations that query the entities they need.

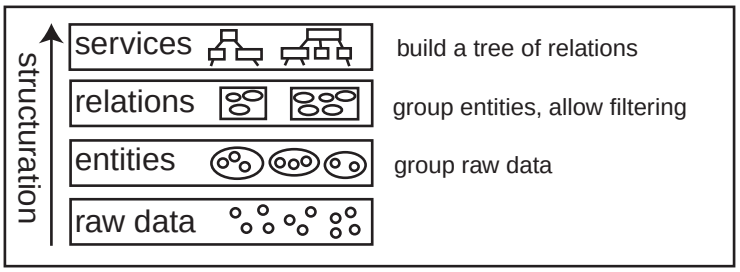


Figure 2: structuring information

3.2. Dynamic environment

Things are different in a dynamic environment. As raw data constantly change, this has an impact on each element of the structuring layer :
Entities : New entities are created and existing entities die. In addition, dynamic attributes of an entity evolves with time.
Relations : The evolution of the entities (creation, death) result in changed relations.
Services : Services depend on relations and therefore the service's tree evolves according to current state of relations.

As with agents, we found it necessary to use push mechanisms in order to optimize the operation of the system. Therefore, relations do not only query the global repository, they also subscribe to it in order to be sent every new entity meeting their selection criteria. Likewise, each node in the service subscribes to the entities of its relation(s) to be notified when their dynamic attributes change or when they die. These mechanisms are shown in figure 3.

Our system is only designed to manage the entities required by running services, that is the entities queried by the relations derived from these services. Every time a service creates a relation, the system must know if it is necessary to contact a new agents or if the entities needed are already in the system. To handle this problem , we use a special component, called the AgentManager, that is notified each time a relation is created. The Agent Manager knows what types of entities are currently being sent to the system by agents and can therefore decide whether a new agent should be invoked.

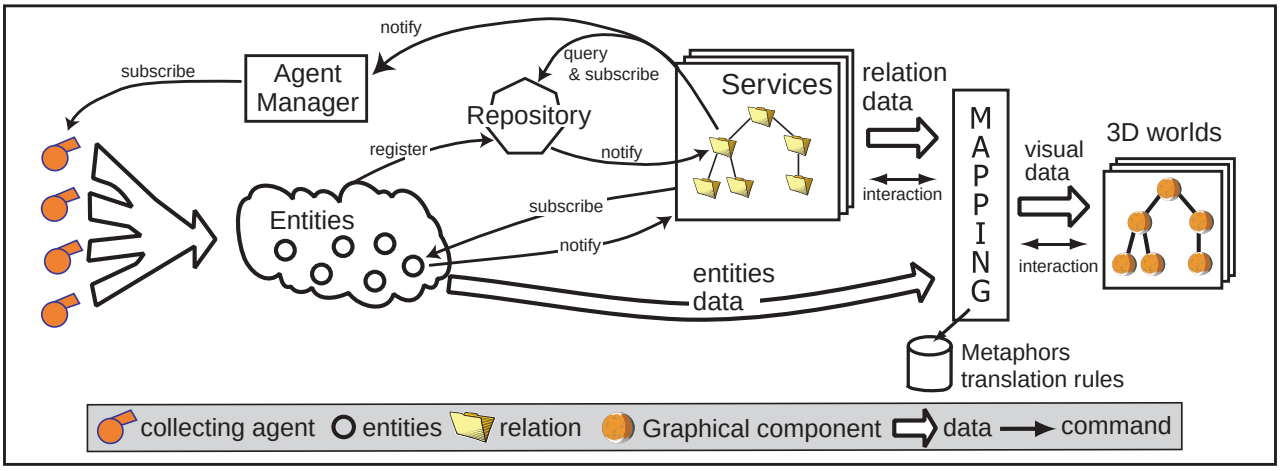


Figure 3: overview of the framework

4. VISUALISATION

Our goal is to provide 3D dynamic interactive visualization of services as defined in the previous section. Our visual structure is based upon the concept of 3D metaphors such as a city or solar system. A metaphor is not designed for a specified service. In this context, a service can be visualized using any metaphor provided it offers sufficient capabilities for viewing its information. This allows the user to select the metaphor best suited to his needs. Because of the dynamic nature of the system, each change in a service is reflected by its visualization in order to visually show these changes. In this section, we present the components of a metaphor that allow visualizing information, and how the mapping between a service and a metaphor is achieved. Finally, we present the interaction mechanisms provided by the virtual world.

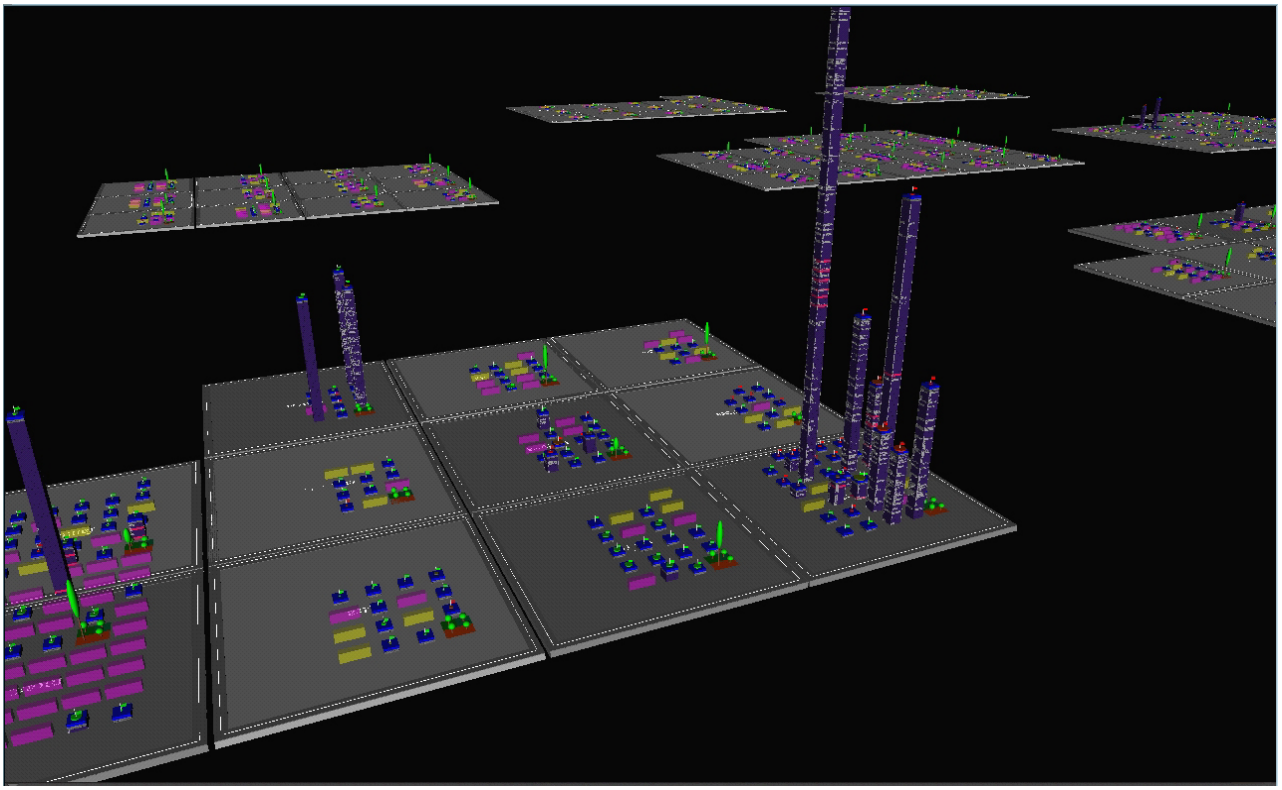
4.1. Graphical components

In the CyberNet system, the visual structure of a metaphor is composed of graphical components (GCs) from which metaphoric virtual worlds can be built. Each graphical component has visual parameters that can be modified in order to visualize information. There are two kind of graphical components: 3D glyphs and layout managers.

- 3D Glyphs (3DG) are 3D objects whose appearance may be controlled in real time through visual parameters. 3DGs represents data through visual parameters that are either retinal (e.g., color and size), or temporal[8]. For example, a box is a 3DG that could offers three retinal parameters (color, radius and height) and one temporal (rotation around itself). The level of complexity of a 3DG can be measured from the number of visual parameters it offers for modification and hence to the dimension of data that can be displayed. Some examples of complex 3DG can be found in [8,13].

- Layout managers (LM) are responsible for one of the most important visual choice: the use of space. They are used as containers and may contain 3DGs or other LMs. They organize their children in space (position and orientation) according to some built-in policy and may also animate these positions along paths. For instance, a LM may organize elements in orbit around its center, on a plane in rows/columns, in Russian puppet boxes, or use a cone tree model[10]. Layout managers can also add some visual elements like semitransparent bounding boxes (see [7] for the use of semitransparency) around their children to enhance the visualization. These additional visual elements can also be used to show information as 3D glyphs do.

Using these two kinds of graphical components, one can build a 3D-scene hierarchy. Each internal node of this hierarchy is a LM and each leaf is a 3DG. Specific graphical components must be defined for each metaphor. For example, in a city metaphor the minimal requirement is 3DGs of houses, buildings or roads and a LM that position these elements in different districts. A solar system metaphor could be composed of a star and planet glyphs, and an orbital layout manager. The more



Figures 4 and 5 : NFS related information of the Eurecom network mapped on a city meptahor. Each district is a workstation where buildings represent disks. Each storey of a building represents a NFS client which is mounting this disk. The number of windows of a floor represents the number of file handles of the client. More details about mapping can be found on our Web site (www.eurecom.fr/~abel/cybernet/)

GCs are used in a metaphor, the more numerous the ways to visualize the data. Some metaphors cannot be used to visualize a service because of a lack of graphical components to show information.
These graphical components are not only passive, they may also handle user interaction as will be shown in the interaction subsection. In a first time, we will see how to use them to construct a 3D visualization from a service.

4.2. Mapping

Mapping is the process that constructs a 3D metaphoric world from the information contained in a service. It results in a graphical hierarchy where internal nodes are layout managers and leaves are 3D glyphs. So far, the mapping process is hard-coded but we are currently developing an automatic mapping process based on information and visual parameters characterization.

4.2.1. Static mapping

In our current implementation, each metaphor owns a set of GCs and contains a set of association rules for each service that can be visualized with this metaphor. These rules specify which GC is associated to which element (entity or relation) of the service according to the type of this element. Rules can also take into account the position of the element in the service's tree. In particular, since an entity may be a part of several relations, it may have several visual counterparts in the presentation domain.

These rules also define how information contained in the service element is mapped onto the visual parameters of the associated GC. A trivial example could be the mapping of a process entity on a box glyph: the CPU used could be mapped on the color of the box and the memory used on the size of the box. All the rules of a metaphor respect some conventions in order to preserve metaphor coherency and identification purposes. For example, if one rule uses the color for identification purposes, other rules must not use this visual parameter to represent a value.

Generally, a relation is associated with a LM and an entity is associated with a 3D glyph. Thus, the graphical hierarchy follows the structure of the service. But we have found that it is sometimes useful to add special nodes in the graphical hierarchy in order to achieve functional mechanisms at a graphical level (e.g a 'fit' node that constrain the size of its children).

Different types of metaphors have been implemented such as buildings (figure 7) , cities (figures 4 and 5), solar system (system 9) and cone trees[10] (figure 8) and less metaphoric visualization such as landscapes[11] (figure 10). From these experiments, we have acquired some experience that helps us in developing automatic mapping.

4.2.2. Automatic mapping

The goal of the automatic mapping process is to generate the rules described above on the fly. The generated mapping rules are defined based on an effectiveness criterion: finding the best mapping considering the visual parameters attributes and their appropriateness for visually encoding information.

To this effect, visual parameters are ranked according to their capabilities of encoding the different types of data. An extension to this strategy is encoding the most relevant information in the most efficient manner. In other words, when deciding how to encode data of the same type, the most important data will be encoded in the most efficient manner. The ranking of the data relevancy is dependent on the type of service.

The above described mapping process involves several steps of characterization. To start with, data need to be characterized according to their type (e.g., quantitative or qualitative), their temporal behavior (e.g., static or dynamic) and other factors. On the one hand, the more detailed we get in this characterization process, the more selective we can afford to be when mapping information onto the visual parameters, up to a certain degree. On the other hand, this involves a loss of versatility and the risk of being too dependent on the application field. However, it must be pointed out that characterization is critical, along with the temporal dimension, as our system must take into account highly dynamic data.

Another step is the characterization of visual parameters according to their ability to visually encode different types of data. For example, some visual parameters may be appropriate for mapping qualitative data but are ill suited for mapping

quantitative data (e.g., shape). The characterization of visual parameters is essential in meeting the effectiveness criterion of trying to find the best mapping. Work has already been done in this area, namely in [12], as regards 2D visualization. However, the use of 3D visual parameters (e.g., transparency, use of light) for encoding information is still to be studied.

Finally, to complete the mapping process always bearing in mind the effectiveness criterion, information has to be characterized according to its importance within a given service. In other words, information is ranked by order of importance to that service. This allows for choices to be made when we have more than one piece of information with the same data type, that we want to encode using the visual parameters. If information is classified by order of importance, then the most significant information gets the best encoding (i.e., is coded with the visual parameters best suited for that particular data type), whereas less significant information eventually gets a not so good encoding, to preserve the effectiveness criterion. Ordering information on the basis of its significance is, as already mentioned, service-context dependent. The same piece of information may be more or less significant according to the service being visualized. For instance, knowing if a disk is hard or soft-mounted on a machine might be crucial for an NFS service, but have less importance in a service that only analyses that particular machine.

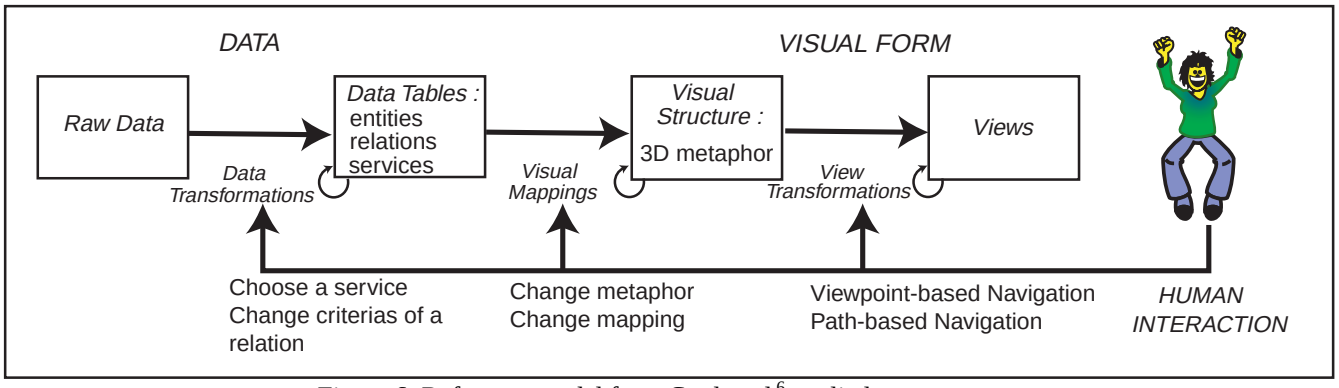


Figure 6: Reference model from Card et al [6] applied to our system

4.3. Interaction

Interaction is fundamental in our project for two reasons. Firstly, we are in the context of virtual environments (VE), where users need to navigate to explore information. Secondly, as the amount of information can be very large one must provide mechanisms to optimize the visualization to the user. The first group of interaction tools has an impact on the view transformations, i.e. on the parameters of the camera, and the second group has an impact on the mapping and the service. Note that since our system use non-immersive VE, we will only discuss 2D interaction with a 3D world. When we talk about a 'button', it can be a mere button outside the VE or a button displayed in a HUD (head up display , i.e. a display superimposed on the graphics).

4.3.1. Navigation

Navigation is one of the main problems that a user is faced with when interacting with a 3D world. Traditional 3D VRML browsers feature navigation mechanisms[15] which allow the user to move freely in the 3D world (i.e. change its position and orientation). Another mode provided is the examine mode that exploits the virtual trackball[14] to examine an object. The user can also jump or fly to a predefined viewpoint. Any user that has experienced these navigation tools will agree on the fact that they are far from sufficient and that he often gets lost in space or lose time making very basic navigation movements. Our goal concerning navigation is to add navigation capabilities to these traditional mechanisms.

4.3.1.1. Advanced Viewpoint-based Navigation

In order to achieve enhanced navigation, we have been working with viewpoints (VP) in the VRML sense, that is a set of specific locations from which the user can view the scene. The metaphor places VPs in every spot of interest until the complete set of VPs covers all visible information in the world. The VPs are structured in relation to the metaphor, which allows the user to navigate between them logically. Each VP has an associated GC from which it can retrieve information such as its name. Navigation using VPs can be used either absolutely or relatively.

In absolute navigation, the user can select a VP and get to it. There are two different modes of interaction. The first type of interaction is complementary to traditional navigation. The user clicks a GC that is in his field of vision to move to its associated VP. This differs from the "Seek" function available in VRML browsers in that the spot (i.e. the VP) where the user arrives is precisely defined. The second mode of interaction allows the user to select from all VPs in the world, not only those he can see at a given time. Contrary to the mode described above, the user interacts with elements external to the metaphor. He is in fact presented with a map showing the VPs in a structured form. This map can take various forms depending on the interface. For instance, the user can be presented with a hierarchical menu containing all the VPs in the world, or with a hyperbolic tree. A more visual option would be a virtual map of the world containing all VPs.

In relative navigation between VPs, the system must know the current position of the user in the metaphor. The user can choose between such operations as "Up" or "Next". Suppose we use the metaphor of a building, "Next" would take the user to the next room and "Up" to the start of the street. Relative navigation does not rely on a simple 3D-scene hierarchy traversal. It stronly relies on the kind of the metaphor and is implemented inside the metaphor.

4.3.1.2. Path-based Navigation

In all navigation methods previously discussed, the user is not constrained to a given path to go from one spot to another. Our idea is to offer the user a navigation method taking full advantage of the structure of the metaphor. In the metaphor of a town, the user would not go directly through the houses to get from one point to another, but would follow the streets. In the metaphor of a building, he would use the elevator or the stairs to go from one storey to another instead of going right through the ceiling (see figure 7 for an example of interface).

This mode of navigation requires a knowledge of the user's current position in the world as well as a routing mechanism between VPs defined by the metaphor. It should be noted that this routing mechanism depends on the visual layout of the metaphor. In path-based navigation, a user going from one VP to another passes by sites of interest that would not have been visible with traditional straight-line navigation methods. Another advantage is that the user knows where he is and does not feel disoriented. Path-based navigation can also allow the user to define "rounds" by specifying some key VPs. The system then defines a loop passing through these VPs. This path can be traveled non-stop in order to continuously monitor the information of sites passed by.

4.3.2. Interaction with the mapping

All interactions used by the navigation merely imply a change in the parameters of the camera in the VE. To help users manage the visualization, we should offer them a mode of interaction allowing them to change the mapping.

The most obvious way to affect the mapping is to change the kind of metaphor used and obtain a new visualization. It can be useful because some metaphors are better suited for some kind of information, and thus provide a more effective visualization.

Another option is to change the mapping of visual parameters while keeping the current metaphor. This can be done by changing the mapping rules as described previously in the mapping section, or in case of automatic mapping by changing the effectiveness criterion.

We can also offer users metaphor-dependent tools allowing them to change some visual parameters that do not provide information from the service but optimize visualization for a specific goal. For example, in a building metaphor, it can be useful to hide the walls (or make them transparent) to help users in navigating and to prevent them from getting lost (see figure 7).

Showing all the information at once is useless as users cannot visualize it all at the same time. Therefore, another interesting technique is to provide a visual overview of a part of the world. To do this, the service provide summary entities that summarize the information contained in a given part of the service, and the visualization display this summary entity instead of giving a thorough representation of all the information.
This is what you do, in real life, when looking down at a city from a plane: you recreate an automatic overview of the town, dividing the town into several districts and analyzing general patterns in each district. In a virtual world, we can execute this

task for the user, i.e. create a visual overview showing general patterns for each district. This relieves the user from the cognitive load that is necessary to analyze the district. In theory (although of course not in practice), one could even imagine a visualization boiling down to only one glyph roughly summarizing all the information. This interaction allows the user to manage the density of information by expanding or summarizing some parts of the world. It is specially usefull in a large world.

4.3.3. Interaction with the service

Interaction can also have an impact on the service. We offer the user a way to obtain the numerical and textual information from each graphical component on demand. This can be useful as it is often necessary to obtain detailed values after a problem has been detected through visualization. These values are displayed in a 2D table outside the virtual world because we did not found an efficient way for displaying several lines of text in a 3D world. This information is also automatically displayed when the user navigates and clicks an object of interest to go to the associated VP.

Currently, we are studying how a kind of dynamic query[16] could be applied to the parameters of a service's relation that is visually in the user's focus. In addition, a way to select objects in a view is under study. This mechanism can be useful to correlate objects between two different views displaying the same data with a different metaphor.

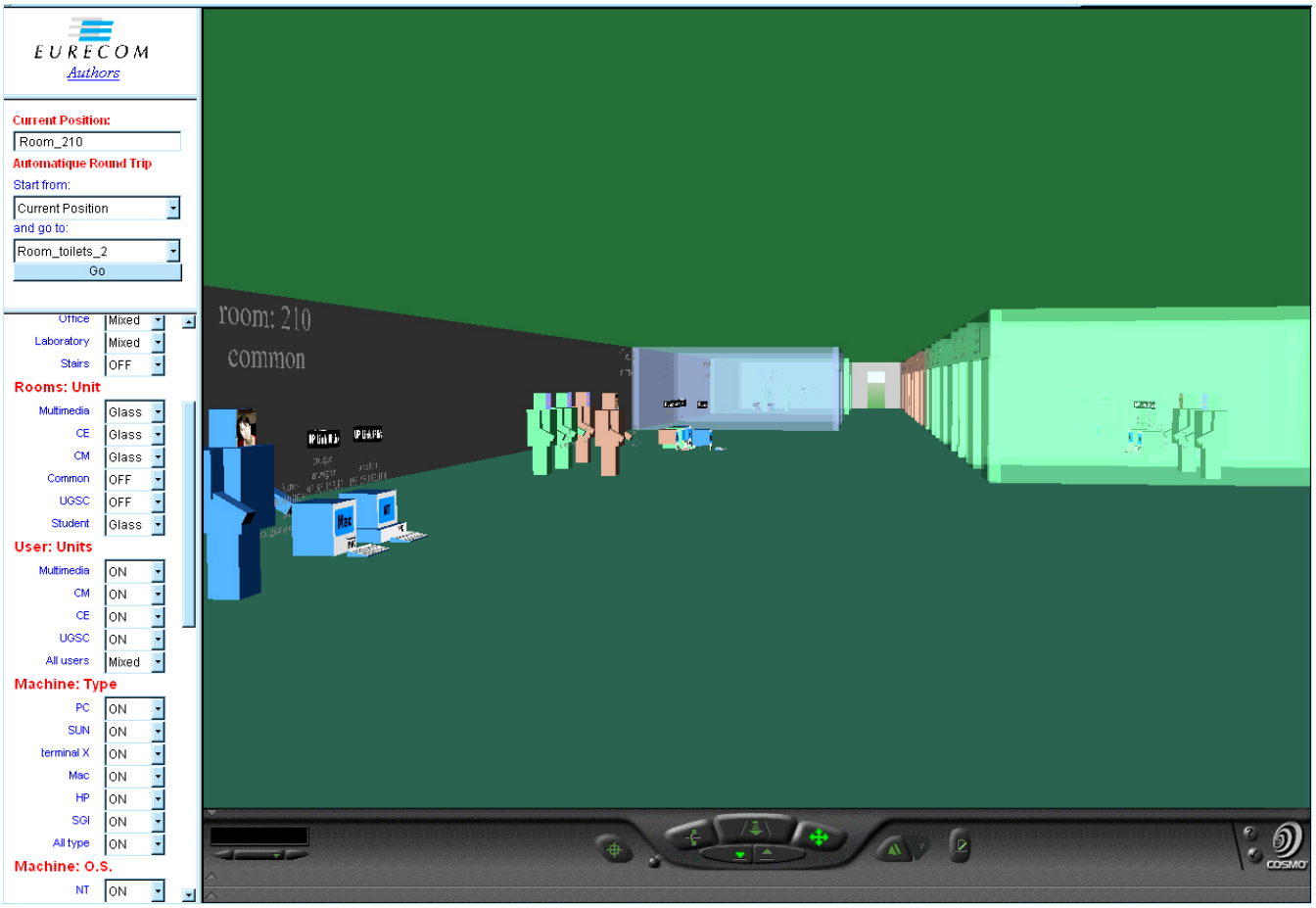


Figure 7: This static world represents the Eurecom Institute building with the different department, people and their workstation in their office. All the components of this world can be hidden or shown using menus on the lower-left. In addition, office and walls can be also semitransparent. In this picture, some rooms have been hidden whereas others are semitransparent. Path-based navigation is implemented and can be exploited using the top-left menus. More explanations can be found on our Web site (www.eurecom.fr/~abel/cybernet/)

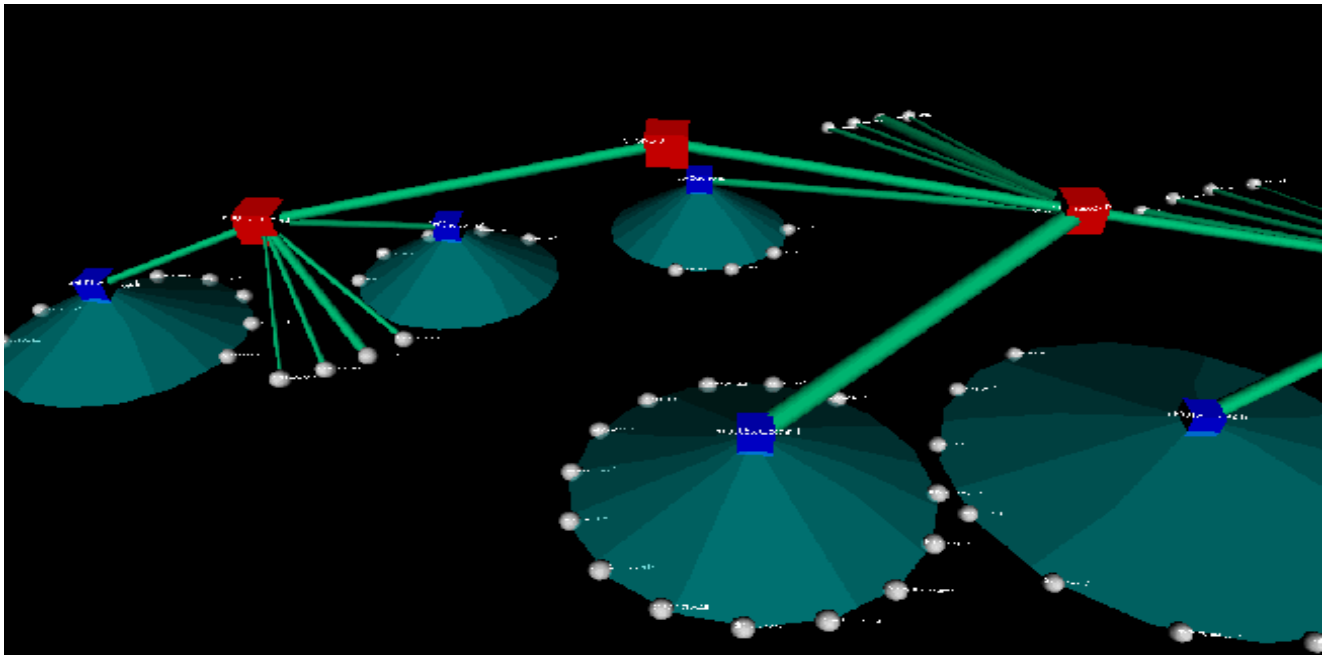


Figure 8 : Topological view of a subpart of the Eurecom Institute network using a cone tree visualization model.The width of the links is dependent on network traffic and the shading is related to the packet loss for that link. The transparency of a cone is dependent on the collision rate of the hub it is associated with.

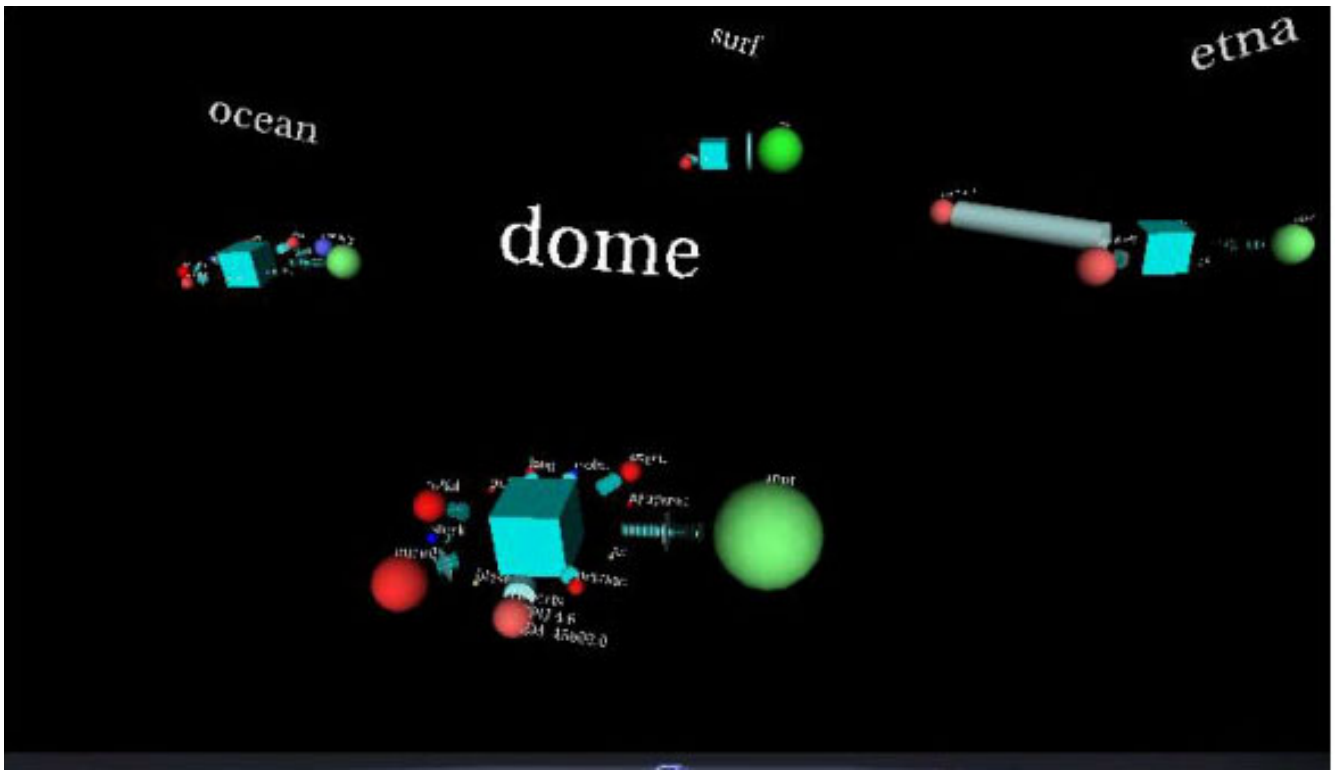


Figure 9 : This picture shows several workstations with its users around it using a solar system metaphor. Each user is a sphere. The sphere's color corresponds to the unix group of the user,the size to the memory the user is using and the color's intensity to the CPUTime. Between the users (sphere) and the workstation (cube), user's processes are displayed as cylinders. The cylinder's visualsparameters follow the same guideline as sphere in order to keep coherency (size→memory,color→cpu)

5. IMPLEMENTATION

The CyberNet system is designed so as the user just need a VRML-enabled WWW browser to visualize information. The External Authoring Interface of VRML is used to create and update VRML worlds. The CyberNet plateform is written in JAVA and all the communication are based on CORBA distributed object technology. Our approach for designing the repository is based on the combined use of a Tuplespace middleware[1] together with the observer design pattern[2]. Tuplespaces provide communication, synchronization and notification primitives for building distributed system data repositories. The most advanced tuplespaces[3,4] support template queries and event notification. An entity is registered as a tuple, and a relation is implemented as a template, which is transmitted to the tuplespace as both a query and a registration to the event notifier. In the current implementation, we are using Tspaces[4].

Aside this dynamic environment, we have developed the same framework in a static environment (i.e. the raw data are not distributed and do not evolve in time) in order to do some fast experiments. This system is implemented in Perl and generates a VRML file with embedded interactions.

At present time,some of the metaphors (building, cities, landscape) have been implemented using the static framework but we plan to integrate them in the dynamic framework. A cone tree and a solar system have been implemented in the dynamic environment.

6. CONCLUSION & PERSPECTIVE

Managing networks using virtual environments requires being able to process huge amounts of dynamic data, build and update 3D interactive worlds in real-time. Our framework addresses the problem of collecting and structuring raw data through a distributed objects architecture, and a clear structuring of the collected data in order to construct a service that contains all the information to be visualized. We have also presented how we are able to dynamically build 3D worlds using metaphors that are composed of graphical components. Our metaphors also contain specific navigation mechanisms that help the user to explore the virtual environment and other interaction mechanisms to help him manage the visualization.

Preliminary feedback from users showed that 3D metaphoric visualization seems promising. It eases problem detection and understanding. However, we are still looking for new visualizations that make the most of the opportunities of virtual environment. Another current research interest is to finish the automatic mapping of a service onto a metaphor. New interaction mecanisms also need to be studied. In particular, we want to provide ways to interact with the real world (e.g. kill a process on a workstation).

We feel that an added value of our system is that it provides a generic framework for collecting, structuring and automatically presenting large volumes of dynamic information in 3D interactive worlds. Strictly speaking, the only part of the system that has to be adapted is the collecting layer. However, the global system's architecture remains valid, should we apply it to Web administration or the Stock Exchange, for instance.

ACKNOWLEDGMENTS

The CyberNet project is supported by France Telecom and the Eurecom Institute.

REFERENCES

1. D. Gelernter and A. Bernstein. "Distributed Communication via a Global Buffer". *Proc. ACM Symp. on Principles of Distributed Computing*, pages 10–18, 1982.
2. E. Gamma et al.*Design patterns : elements of reusable object-oriented software*. Addison Wesley, Reading 1996.
3. J. Waldo et al. "Javaspace specification - 1.0". Technical report, Sun Microsystems, March 1998.
4. P. Wyckoff, S. McLaughry, T. Lehman, and D. Ford. "T spaces". *IBM Systems Journal*, 37(3):454–474, 1998.

5. Sougata Mukherjea, James D. Foley, and Scott Hudson. “Visualizing complex hypermedia networks through multiple hierarchical views”. *Proceedings of ACM CHI’95 Conference on Human Factors in Computing Systems, volume 1 of Papers: Creating Visualizations*, pages 331–337, 1995.
6. S. Card, J. Mackinlay, B. Shneiderman, eds. *Readings in Information Visualization: Using Vision to Think* . Morgan Kaufmann, San Mateo, CA, 1998.
7. Shumin Zhai, William Buxton, and Paul Milgram. “The partial-occlusion effect: utilizing semitransparency in 3D human-computer interaction”. *ACM Transactions on Computer-Human Interaction*, 3(3):254–284, September 1996
8. Mei C. Chuah and Stephen G. Eick. “Information rich glyphs for software management data”. IEEE Computer Graphics & Applications, 18(4), July – August 1998. ISSN 0272-1716.
9. Laurence A. Crutcher, Aurel A. Lazar, Steven K. Feiner, and Michelle X. Zhou. “Managing networks through a virtual world”. IEEE parallel and distributed technology: systems and applications, 3(2):4–13, Summer 1995.
10. George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. “Cone trees: Animated 3D visualizations of hierarchical information”, *Proc. ACM Conf. Human Factors in Computing System* , Scott P. Robertson, Gary M. Olson, and Judith S. Olson, editors, pages 189–194. ACM Press, 1991
11. W. Wright. “Research report: Information animation applications in the capital markets”. *In Proc. Information Visualization '95*, pages 19--26, October 1995.
12. Jock D. Mackinlay. “Automating the Design of Graphical Presentations of Relational Information”. ACM Transactions on Graphics, 5(2):110-141, April 1986.
13. Randall M. Rohrer and Edward Swing. “Web-based information visualization”. IEEE Computer Graphics and Applications, 17(4):52–59, July/August 1998.
14. M. Chen, S. Joy Mountford, and Abigail Sellen. “A study in interactive 3-D rotation using 2-D control devices” *Proceedings of SIGGRAPH ’88* ,22(4):121-129, August 1988.
15. J.D. Mackinlay, Card S.K and G.G. Robertson, “Rapid controlled movement through a virtual 3D workspace”. *Proceedings of SIGGRAPH’90*, pp. 171 – 176, 1990.
16. B. Schneiderman, “Dynamic queries for visual information seeking", IEEE software,11(6),70-77,1994

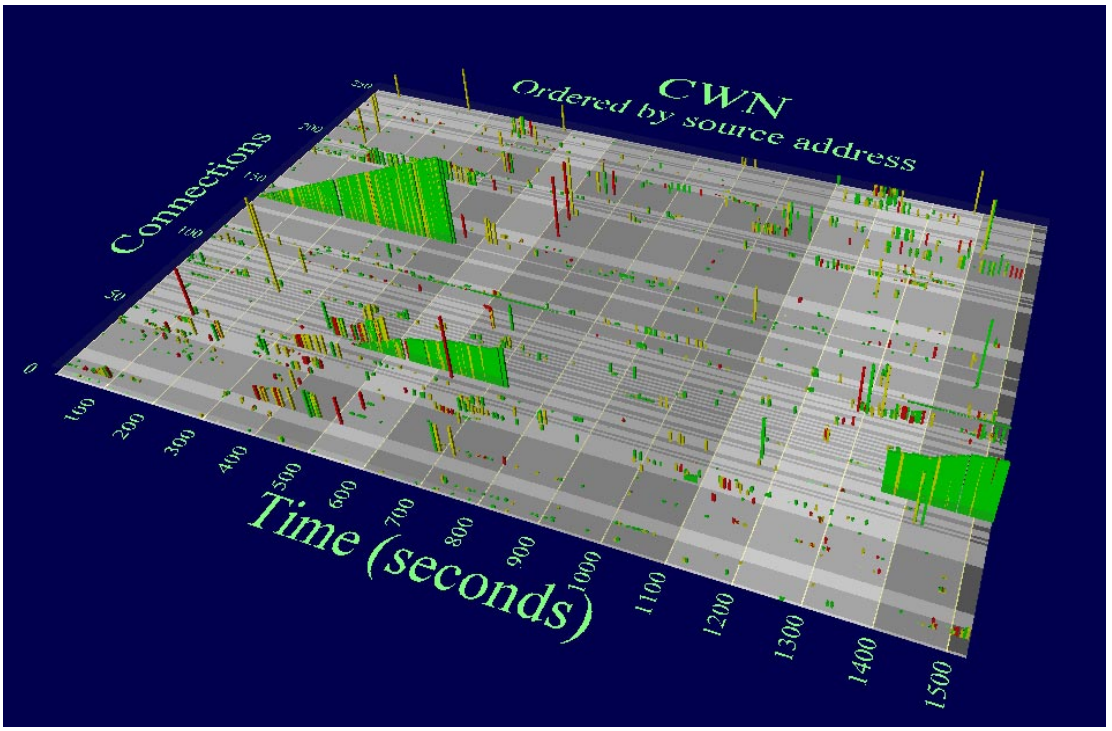


Figure 10: This static landscape displays data captured by a network sniffer. Data has been structured by our system and thus several views have been generated easily. This view shows the evolution of the size of the TCP windows for each connection. An interaction tool (not shown here) helps the user to filter data that are below a threshold. More information can be found on our Web site.