

# Collaborative Visualization: A Review and Taxonomy

Ian J. Grimstead, David W. Walker and Nick J. Avis  
School of Computer Science, Cardiff University,  
Queen's Buildings, 5 The Parade, Roath,  
Cardiff CF24 3AA, United Kingdom  
{I.J.Grimstead|D.W.Walker|N.J.Avis}@cs.cardiff.ac.uk

## Abstract

We present a brief review of 42 collaborative visualization systems, grouped into four application areas: collaborative problem-solving environments, virtual reality environments, multi-player online games and multi-user enabling of single user applications.

The systems are then compared by five attributes: number of simultaneous users, user access control, communication architecture, type of transmitted data and user synchronisation. We review the characteristic properties of each application area, overall trends in characteristics and recommend improvements for future systems.

The taxonomy of visualization and accompanying bibliography are available on-line via the RAVE project pages [21], with on-line references hyper-linked where available.

## 1 Introduction

Collaboration has always been utilised in computing, tracing its history back to early demos by Douglas C. Engelbart [16] in 1968. From this early text-based collaboration, graphics become available in the mainstream (such as the release of the Apple Macintosh in 1984, featuring a window based GUI with mouse). Graphical collaboration was then possible, such as Stefik's [48] WYSIWIS (What You See Is What I See) system in 1986, with its collaborative chalkboard. Real-time graphical interaction has since expanded and diverged into many areas, rather than basic teleconferencing support.

We wish to investigate how collaborative visualization has diverged, searching for any unique ideas in one application area that would be of benefit to another. With this in mind, we present a brief review of 42 collaborative visualization systems published over the last eight years. We then compare the systems and their underlying application areas,

searching for trends and differences between them. We follow this with possible improvements for future systems and finish with our conclusions.

## 2 Review of Collaborative Visualization

We briefly review computer based collaborative visualization by five application areas: collaborative problem solving environments (PSEs), virtual reality environments (VREs), multi-player online games (MPOGs) and multi-user enabling of single user applications (MUEs). Five attributes have been used for system comparison, selected to be common to all application areas, enabling cross-application comparison. The attributes are: number of simultaneous users, user access control, communication architecture, type of transmitted data and the synchronisation between users. Note that type of transmitted data uses "scene graph" to imply that a shared scene graph is maintained (a common set of geometry), whereas "raw data" implies bare updates to fields are sent (ranging from positional updates to surface modifications). The systems and their attributes are presented in Table 1, where any estimated attributes are marked in *italic* font. Each application area is now analysed in a separate section.

### 2.1 Collaborative Problem Solving Environments

Visual problem-solving environments (PSEs) are often modular visualization environments, but with the addition of computational steering and loose coupling between the data source and the local modification of the data for visualization. Namely, parameters can be adjusted whilst the simulation is still in progress, without having to wait hours or even days for a complete simulation run before performing analysis.

cAVS [30], COVISA [56], MANICORAL [15] and Sieve [29] are all component-based workflows, extending existing modular visualization environments such as

<i>Visualization System</i>	<i>Number of Users</i>	<i>Access Control</i>	<i>Communication Architecture</i>	<i>Transmitted Data</i>	<i>User Synchronization</i>
<b>Problem Solving Environments</b>					
cAVS [30]	128	None	Single server	Raw data	Loose
COVISA [53]	10	<i>Unspecified</i>	Single server	Raw data	Loose
COVISE [33]	10	Per session	Single server	Scene graph	<i>Loose</i>
CUMULVS [32]	20	Per object	Single server	Raw data	Loose
ICENI [19]	10	Per object	Single server	Frame buffer	<i>Loose</i>
MANICORAL [15]	10	None	Single server	Raw data	Loose
Sieve [29]	100	<i>Unspecified</i>	Peer-to-peer	Raw data	Loose
<b>Multi-User VR Environments</b>					
Avango [51]	100	None	Multiple servers	Scene graph	Loose
Community Place [35]	700	None	Multiple servers	Scene graph	Loose
DEVA3 [40]	100	None	Multiple servers	Scene graph	Asynchronous
DIVE [17]	1,000	None	Peer-to-peer	Scene graph	Loose
ERCIS [6]	11	None	Single server	Raw data	Loose
MUVEES [10]	60	None	Multiple servers	Scene graph	Loose
<b>Collaborative VR Environments</b>					
COVEN [37]	100	Unknown	<i>Peer-to-peer</i>	Scene graph	Loose
CSpray [39]	10	<i>Per object</i>	<i>Peer-to-peer</i>	<i>Raw data</i>	Asynchronous
CVD [34]	10	<i>Unspecified</i>	<i>Peer-to-peer</i>	<i>Scene graph</i>	<i>Loose</i>
DIS [1]	1,000	N/A	Peer-to-peer	Raw data	Loose
DIVISION Mockup [41]	10	<i>Unspecified</i>	<i>Peer-to-peer</i>	Scene graph	<i>Loose</i>
EQUIP [20]	10	<i>Unspecified</i>	Single server	Scene graph	<i>Unspecified</i>
HLA [2]	1,000	N/A	Peer-to-peer	Raw data	Loose
Nomad [55]	100	<i>Unspecified</i>	Multiple servers	Scene graph	Loose
Octopus [23]	100	<i>Unspecified</i>	<i>Single server</i>	<i>Unspecified</i>	<i>Loose</i>
PaRADE [43]	1000	Per object	<i>Peer-to-peer</i>	<i>Raw data</i>	<i>Loose</i>
RAVE [22]	100	Per object	Multiple servers	Scene graph	Loose
SCAPE [25]	10	<i>Unspecified</i>	Single server	N/A	N/A
StudierStube [45]	10	Per object	Peer-to-peer	Scene graph	Loose
Virtual Worlds 5 [28]	1,000	<i>Unspecified</i>	Multiple servers	Scene graph	<i>Loose</i>
<b>Multi-Player Online Games</b>					
Age of Empires [8]	8	Per Object	Peer-to-peer	Raw data	Loose
Butterfly.net [9]	20,000	Per Object	Multiple servers	<i>Unspecified</i>	Loose
Half-Life [7]	100	Per Object	Single server	Raw data	Loose
Net-Z [14]	100	Per Object	Multiple servers	Raw data	Loose
TRIBES [18]	128	Per Object	Single server	Raw data	Loose
<b>Multi-User Enabling of Single-User Applications</b>					
Access Grid [11]	10	<i>Unspecified</i>	Multiple server	Frame buffer	<i>Unspecified</i>
AG Remote Ren.[31]	100	<i>Unspecified</i>	Multiple server	Frame buffer	Loose
NetMeeting [12]	10	Per session	<i>Single Server</i>	<i>Scene graph</i>	Loose
Sametime [27]	1,000	<i>Per session</i>	Multiple Servers	<i>Scene graph</i>	Loose
SGAB [57]	10	Per object	Peer-to-peer	Scene graph	Loose
SharedX [24]	10	Per session	Single server	Scene graph	Loose
VizServer 3.0 [46]	10	Per session	Single server	Frame buffer	Lockstep
VizServer 3.1 [47]	30	Per session	Single server	Frame buffer	Lockstep
VNC [42]	8	None	Single server	Frame buffer	Lockstep
XXM [5]	10	Per session	Single server	Scene graph	Loose

**Table 1. Comparison of Collaborative Visualization Environments**

AVS/Express [50] or NAG Explorer [54]. COVISE [49] is a component-based workflow in its own right.

ICENI [19] and CUMULVS [32] are both middleware libraries, instrumenting existing code. ICENI is a Jini-based middleware used in the RealityGrid [36] project, where clients connect to a single server and steer the simulation, where the output is rendered remotely and the rendered frame buffer sent back to the client, rather than local rendering. Custom AVS/Express modules have been created with CUMULVS [32], extending AVS to be collaborative.

PSEs are often based on single server models, possibly because systems are tightly integrated into the dataflow map, which is designed for a single machine [3], leading to a centralisation of control. This leads to a limit to the scalability of the system, hence an upper bound of 10's of users rather than 100's. Remote machines may be used to process part of the environment, but the data is often still routed through a single, central server for the above reason.

PSEs scale better than most applications, as users are accustomed to waiting for simulations to produce new results. Hence a delay of up to a second (due to server load, etc.) would be considered acceptable,

Systems assume that direct inter-user communication is achievable through other means, such as a third party video conference system or a basic telephone. As PSEs are used for work between colleagues, a level of trust is assumed, which is relied upon by several systems (cAVS [30], MAN-ICORAL [15]) rather than include access controls.

## 2.2 Virtual Reality Environments

Virtual Reality environments (VREs) have often been used to support multiple users, ranging from few users in a high fidelity fully immersive environment (such as CAVES [13]) to massive multi-user systems with minimal immersion. Some systems are targeted at simply enabling multiple users to co-exist in a shared space, whilst others are extended to support collaboration; this is highlighted in Table 1 where VREs are presented as “Multi-User VR Environments” or “Collaborative VR Environments”.

VREs may support large numbers of users (100+), so are designed to be scalable through various techniques, which we now discuss. Sending update messages for objects within the immediate proximity of the user (Community Place [35], COVEN [37], DIVE [17], PaRADE [43]) or immediately updating the local world whilst lazily transmitting and receiving updates to the network (DEVA3 [40]) both reduce network traffic whilst maintaining high interactivity. Note that DEVA3 is sufficiently loosely coupled to be marked as “asynchronous” in the table. These techniques result in the local user not having a full, up-to-date copy of the world, but is sufficiently close as to not introduce visible anomalies. CSpray [39] is fully asynchronous,

enabling users to review recorded interaction sessions, but is a specialised data visualization tool rather than a generic VRE. To reduce rendering and maintenance overhead, “impostors” can replace complex geometry with prerendered 2D textures created on a remote server (MUVEES [10]).

Of the VREs supporting low numbers of users (10 or less), two of them are fully immersive systems (CVD [34], SCAPE [25]). The limit on the number of users could be a design decision due to the small number of users with immersive equipment, as there is little advantage in supporting hundreds of users when your userbase is in the 10's.

Multi-user and collaborative variants of VREs have very similar characteristics; collaborative VREs appear to make more use of peer to peer networks, possibly to distribute the additional load required when objects can be access locked and “owned” by users.

Load balancing is also possible with multiple servers or peer-to-peer, where object maintenance is distributed or replicated amongst machines (DEVA3 [40], Nomad [55], Octopus [23], RAVE [22], StudierStube [45]).

With large numbers of supported users, resilience is important; this is supported via replication of data or servers (Community Place [35], DIVE [17], RAVE [22]). The peer-to-peer model uses special peers record a copy of all interactions and stream them to disc, or objects are supported by redundant peers in case of failure. This provides a resilient infrastructure, but is open to abuse if a local user modifies their peer to transmit fake information.

Several of the systems presented are actually middleware rather than complete environments, such as Avango [51], the Distributed Interactive Simulation (DIS) standard [1], High-Level Architecture (HLA) [2], Octopus [23] and StudierStube [45]. These systems can then be used to create an actual application.

Large-user systems often include basic inter-user communication support, such as textual chat (COVEN [37]) or video conferencing (DIVE [17], Virtual Worlds [28]).

## 2.3 Multi-Player Online Games

Multi-player onlinegames (MPOGs) are similar to multi-user VREs, supporting many users through distributed architectures, although MPOGs will assume that each client has sufficient resources to receive raw data rather than pre-prepared sections of scene graph. Strategies to cope with low bandwidth and high latency include just sending player control information (Age of Empires [8]), interpolating between already received updates to ensure smooth movement (Half-Life [7], TRIBES [18]), allocating network bandwidth on an importance basis to game objects (TRIBES [18]) and sending messages to affect only objects local to a player (Butterfly.net [9]).

Users often pay to play MPOGs, unlike VREs which

are usually “in-house” systems used for teaching or exploration. If users pay, then a system must be secure (to prevent financial loss and to prevent players from cheating). To maintain security, a central, trusted point of authority is often used (Half-Life [7]), all under the control of the MPOG supplier. With simpler, stand-alone MPOGs where state is not persistent between games (Age of Empires [8]) or players do not pay to play, peer-to-peer networks are viable.

Butterfly.net [9] and Net-Z [14] are infrastructures to support multiplayer games, supporting load (object) distribution and resilience as VREs.

MPOGs usually include some form of inter-user communication, as players are often geographically dispersed without any other form of communication (as players do not know each other outside the game). Text-based chat is often included, with audio telephony becoming popular now that broadband is widely available, enabling the use of additional bandwidth for communication outside of the game mechanism.

## 2.4 Multi-User Enabling of Single User Applications

Multi-user enabling applications (MUEs) extend existing applications to support multiple simultaneous users without modification, but this comes at a price—the application must be run on a single machine, which will disable or interrupt its local user.

As pre-existing applications are being enabled, no knowledge can be assumed about the data to be transferred over the network, so the local frame buffer (or its description) are transmitted. The transmission of frame buffers from a single machine locks the users to a single display and hence interaction rate. This restricts the number of users supported as the single machine will rapidly become overwhelmed with network traffic. An exception to this is video conferencing or teaching, where a single machine is broadcasting its frame buffer and not receiving input from the remote machines (Sametime [27]). Hence the broadcaster is sending to a single logical machine, but is actually being received by many machines, from 10 to 100+.

MUEs mostly support the basic windowing environment (namely, text, static images and window decoration), unlike OpenGL VizServer 3.1 [47] and Scene-Graph As Bus (SGAB) [57] which support graphical environments either as a bitmap or as the original polygons respectively.

Security is also an issue with MUEs, as this can enable a remote user direct access to a machine—not just the intended shared application. Several systems only transmit the shared application, locking out the rest of the machine (NetMeeting [12], VizServer [47]). Simpler systems however open the entire machine to remote access (such as VNC [42]).

MUEs are usually targeted at the workplace, they often contain support for enhanced user communication outside of the shared application, such as video conferencing (AccessGrid [11], NetMeeting [12], Sametime [27]).

AccessGrid [11] is different in that it is actually a wide-area video conferencing system, into which existing programs may be “hooked” by supplying their output as a live video feed. An example is presented in Karonis [31], where geometry is divided as tiles amongst several machines (using Chromium [26]) and encoded as a video stream, to be reassembled at the AccessGrid client.

## 3 Comparison of Systems and Application Areas

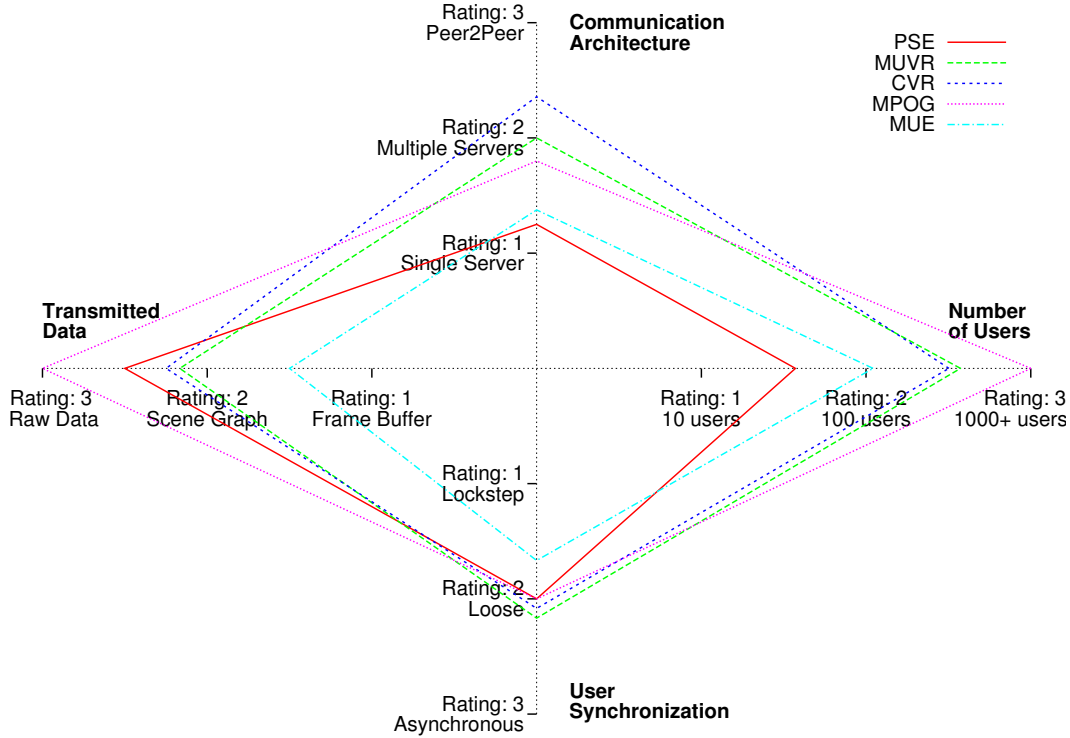
In this section we will compare across application areas, looking for common system characteristics.

### 3.1 Initial Comparison of Application Areas

From examining Table 1, it can be seen that user access control is often undefined in the literature, making full analysis of this aspect difficult. The remaining attributes are either well defined or possible to estimate, so we will initially concentrate on these. We have assigned non-numeric attributes a rating in the range 1..3, with the greater the number indicating increased scalability (such as peer-to-peer assigned “3”, multiple servers assigned “2” and central server assigned “1”). We can then group the systems by application area and average their attributes to form a polar plot (either averaging numerical value or their numerical rating), as presented in Figure 1. The least scalable system is represented by a small quadrilateral (a diamond) with all values rated “1”, namely a single server, 10 simultaneous users, lockstep synchronisation and transmitting a complete frame buffer.

MUEs appear to be least scalable from the diagram, as they form the smallest quadrilateral, and average a rating of  $\sim 1.5$  for all attributes, except number of users—this is skewed by Sametime supporting 1,000 users. Excluding Sametime, this result is expected, as MUEs are usually tied to a single machine (where the single user application is hosted), which forms a bottleneck for the system. MUEs do not scale to multiple machines, except in a broadcast mode where users can only view the application.

The usage of asynchronous updates enables a system to support more users in return for a reduced response time. However, this does not appear to be reflected in the comparison chart. This can be explained in that maintaining asynchronous updates in a real-time environment is more complex (and hence resource consuming) than using network traffic reduction techniques with loosely synchronised users. Such traffic reduction techniques enable systems to



**Figure 1. Comparison of Averaged System Attributes for each Application Area**

scale further, especially when combined with load balancing techniques across peers or servers. These techniques are prevalent amongst MVEs and MPOGS, which support the largest number of simultaneous users.

The trade-off between sending minimal information over the network against increased compute at the client does not appear to impede the scalability of a system. It was considered that sending full scene graph updates would bottleneck on network capacity rather than sending raw data (object updates such as “drive car”, rather than directly transmitting the geometry). This appears to be incorrect, as raw data is used by many systems that only support 10’s of users, whilst scene graph data supports 100’s of users. This may be due to erroneous estimation of transmitted data, or alleviated through network traffic reduction schemes, such as world partitioning and bandwidth allocation.

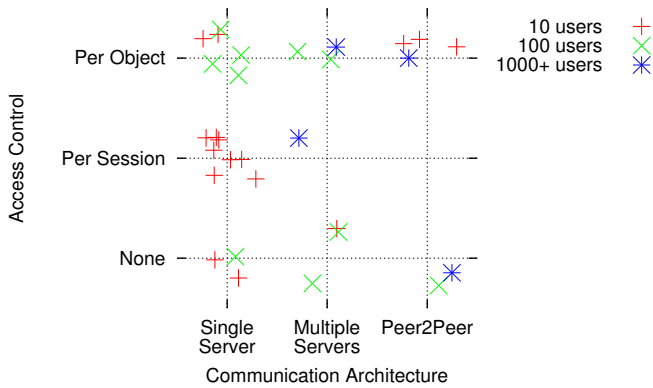
Multiple servers are the most popular choice for scalable systems, rather than peer-to-peer. Multiple servers often incur additional maintenance overheads (as users may not be permitted to host their own servers, instead relying on the application provider), also the network will bottleneck at the servers. With gaming, the use of multiple servers is required to prevent players cheating. The server’s network bandwidth is significantly higher than the player’s, as players use their home connections (limited to current broadband

speeds at roughly 1Mbit/sec, or even analogue modems at 56Kbit/sec) compared to a leased line (such as 100Mbit/sec or 1Gbit/sec). Distribution of servers over geographic location further distributes the network traffic, rather than bottlenecking on a single incoming line. Hence servers can cope with the network traffic and enable a more reliable service, as they are under the direct administrative control of the service provider.

### 3.2 Further Attribute Analysis

Given that user synchronisation is fairly standard, we will now contrast the number of simultaneous users against communication architecture and type of transmitted data. 20 systems are presented in Figure 2, where the axis represent access control and communication architecture, whilst point type indicates the number of users supported by the system. As many points are estimates, we have introduced a slight random scatter to assist legibility—this ensures that multiple points do not occlude each other, whilst remaining close to their original data point (the scatter was  $\pm 0.3$  the width or height of an attribute’s column or row).

This highlights that most  $\sim 10$  user systems support a single server with per-session locking; per-session locking is relatively simple with a single server, as this requires a single point of authority and a single security check. Large



**Figure 2. Comparison of Individual System Attributes**

multi-user systems (>100 users) tend to either have no security or per-object locking; otherwise, a single user would lock the entire world to alter something that would only have an immediate effect on a low percentage of simultaneous users. Combining per-object locking with world partitioning schemes would actually produce less traffic than a single per-session lock without world partitioning, as messages would not be broadcast to all users.

### 3.3 Advances in Technology over Time

The numbers of users supported by application areas is compared against date of publication in Figure 3. There appears to be a constant stream of small-sized systems (catering for up to 10 users), whilst higher-end systems supporting > 100 users are more sporadic; this could be explained by their added complexity, and possibly a smaller user base (and hence smaller demand).

With the capacity of networks and machines increasing over time, there is no obvious pattern of increasing number of supported users. New systems are appearing, but supporting the same order of magnitude of numbers of users. This may again be due to a smaller user base, but appears that we are re-inventing the same size of environment, producing variations rather than quantum leaps in support.

The new push for Grid technology (where access to compute and storage resources is likened to that of electricity on a power grid) is opening up new areas, such as Butterfly.net [9] which purports to support 20,000 users. This area produces new opportunities for deploying multi-user systems across heterogeneous platforms on demand (such as RAVE [22]), enabling use of a wider range of resources and hence more scope for load balancing. Systems can be deployed as simple archives into a container such as Jakarta Tomcat [4], or via emerging Web/Grid standards such as the WSRF

[38].

Overall, it appears that there is no major trend towards more simultaneous users or any particular application area. If anything, the same ideas are being reused in later systems in different combinations, rather than any major paradigm shift. For instance, Community Place [35] introduced the idea of “locales” to restrict network traffic to objects close to a user; this was in turn taken up by DIVE [17] and then COVEN [37]. Butterfly.net [9] uses a world partitioning system to achieve a similar effect.

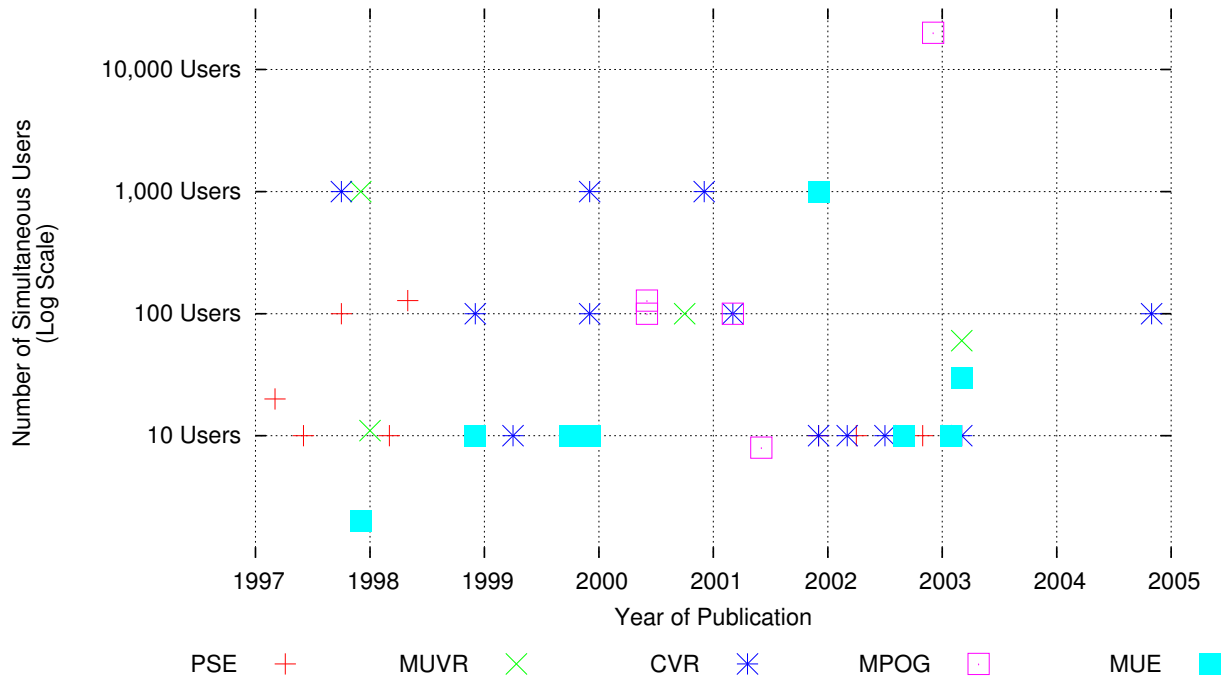
Systems tend to be created to take advantage of new operating systems or underlying hardware, with Butterfly.net [9] being a good example. The concept of load balancing and world partitioning has been used before, but now this is combined with Grid technology (namely Globus [44]) to enable the automated selection and usage of servers through supplied middleware. This permits the designers of Butterfly.net to concentrate on game design rather than server maintenance.

## 4 Improvements to Existing Systems

The arrival of Grid technology need to be observed carefully, as this permits the detection of idling machines and the deployment of servers on demand. This enables a pool of resources to be shared amongst many services—be it MVEs or PSEs, or any other application. Grid services are at present based on SOAP [52] for message passing—this is an XML document encoding parameters for a remote function call. As XML is extremely verbose, this is not a scalable approach as it will quickly consume network bandwidth and compute (for marshalling/demmarshalling messages). Grid/Web services can be used for initial handshaking of service creation and registration (such as RAVE [22]), but alternatives (such as binary XML or raw TCP/IP transfer) are required for large payloads or repeated small messages or streams.

To support a large number of users, existing systems may need to be redesigned to cope, such as recoding to work as a peer-to-peer network, migrating work between users, or discovering new resources. Such concepts are needed at the core of a piece of software, and are difficult to retrospectively engineer. Some systems already support workload migration, but there is a danger of continuously reinventing the same mechanism, such as reimplementing it using Grid services—we are just rewrapping existing ideas rather than moving forward.

The integration of video/audio/textual conferencing into collaborative systems (as DIVE [17], Virtual Worlds [28]) enables the allocation of bandwidth between the system and conferencing element. At present, if the video conferencing is a separate application, this will “fight” for network bandwidth with the collaborative system, especially on a low ca-



**Figure 3. Year of Publication against Number of Simultaneous Users**

capacity network (such as a home broadband connection).

Large systems also require resilience; several systems reviewed included the idea of redundant servers or peers, which were only used when the original supplier of a service has failed. An example is multiple servers/peers writing the session to disc; in the event of a catastrophic failure, it is likely that at least one machine has recorded a full copy of the system before the failure, and can hence recreate the state pre-failure. Ideally, if servers/peers are geographically distributed, the likelihood of simultaneous catastrophic failure is minimised—but the machines are then in different administrative domains, so maintenance is more complex.

For systems that do not require tight synchronisation, asynchronous behaviour would enable the support of many more users than is possible with the same resources (as users are no longer waiting for updates from each other). CSpray [39] and DEVA3 [40] are the only real-time systems we reviewed that support asynchronous access, but it do not reveal how many simultaneous users they support. We can only assume (as mentioned in the previous section) that maintaining asynchronous updates in a real-time environment is more complex (and hence resource consuming) than use network traffic reduction techniques with loosely synchronised users.

An alternative solution would be to attempt hybrid solutions for scalability. For instance, a group of users could run peer-to-peer amongst themselves with loose synchronisation if they are located in a neighbouring part of the world,

with any updates reflected from the users asynchronously to a remote server. The remote server is then part of a multiple, distributed server network that maintains a persistent copy of the world which is reflected to other users (forming other localised peer-to-peer subnets) and servers on demand. The server acts as a more powerful peer that can recreate the world after a failure, and possibly check for violations (cheating) by users/players. Actual object maintenance can then be distributed amongst players, whilst a persistent copy is maintained at the server (receiving asynchronous updates from users). This would reduce network traffic, whilst retaining the advantages of persistency and a single point of authority albeit at an increased latency, unless techniques such as pre-emptive ownership acquisition are used (as in PaRADE [43]).

Finally, it has to be noted that all systems have difficulty when interacting with each other; there does not appear to be a common communication protocol supported. DIS [1] and HLA [2] are protocols published as IEEE standards, but the very nature of IEEE certification reduces take-up due to the expense of obtaining a copy of the standard. Various infrastructures exist (such as Avango [51], Octopus [23] and StudierStube [45]), but these do not interact with each other. Ideally, a standard protocol is required to permit systems to interact with one another, enabling legacy systems to be used with newer systems (such as old datasets, etc.). This also permit the testing of new environments and serving technologies alongside existing applications, or the ex-

tending of existing system by adding new facilities through additional servers.

## 5 Conclusion

We have briefly reviewed many collaborative visualization systems, grouping them by application area. We found the scalability of virtual environments and online games to be far greater than that of collaborative problem-solving environments or multi-user enabling systems, noting the techniques used.

We conclude that systems must consider if they will need to support many simultaneous users; various systems cannot scale due to limitations in their architecture, but could scale if designed to make maximum use of distributed resources and network facilities.

Finally, the taxonomy of visualization (and accompanying bibliography) are available on-line via the RAVE project pages [21], with on-line references hyper-linked where available.

## References

- [1] IEEE Standard for Distributed Interactive Simulation-Application Protocols. IEEE Standard 1278, 1995.
- [2] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA). IEEE Standard 1516, 2000.
- [3] J. Ahrens, C. Law, W. Schroeder, K. Martin, and M. Papka. A Parallel Approach for Efficiently Visualising Extremely Large, Time-Varying Datasets. Technical Report LAUR-001630, Los Alamos National Laboratory, 2000.
- [4] Apache. Apache Jakarta Project—Tomcat. <http://jakarta.apache.org/tomcat/index.html>.
- [5] J. Bazik. Sharing X Applications with XMX. Tutorial, Department of Computer Science, Brown University, October 1999.
- [6] E. Berglund and H. Eriksson. Distributed Interactive Simulation for Group-Distance Exercises on the Web. In P. Fishwick, D. Hill, and R. Smith, editors, *The International Conference on Web-Based Modelling and Simulation*, volume 30, pages 91–95. The Society for Computer Simulation International, January 1998.
- [7] Y. W. Bernier. Latency Compensating Methods in Client/Server In-Game Protocol Design and Optimization. In *Proceedings of the 15th Games Developers Conference*, March 2001.
- [8] P. Bettner and M. Terrano. 1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond. In *Proceedings of the Game Developers Conference*, June 2001.
- [9] I. . Butterfly.net. Butterfly.net: Powering Next-Generation Gaming with Computing On-Demand. Case study, 2003.
- [10] J. X. Chen, Y. Yang, and B. Loftin. MUVEES: a PC-based Multi-User Virtual Environment for Learning. In *Proceedings of the IEEE International Symposium on Virtual Reality 2003 (VR '03)*, pages 163–170. IEEE Computer Society, March 2003.
- [11] L. Childers, T. Disz, R. Olson, M. E. Papka, R. Stevens, and T. Udeshi. Access Grid: Immersive Group-to-Group Collaborative Visualization. In *Proceedings of the 4th International Immersive Projection Technology Workshop*, 2000.
- [12] M. Corporation. Microsoft Windows netMeeting 3. Readme file, 1999.
- [13] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. *Computer Graphics (SIGGRAPH 93 Proceedings)*, 27:135–142, August 1993.
- [14] C. Dionne, M. Lavoie, and K. Trueman. NetZ - Multiplayer Architecture for Online Games. In *Proceedings of the Game Developers Conference*, June 2000.
- [15] D. A. Duce, J. R. Gallop, I. J. Johnson, K. Robinson, C. D. Seelig, and C. S. Cooper. Distributed Cooperative Visualization - The MANICORAL Approach. In *Proceedings of Eurographics UK Conference*, pages 69–85, March 1998.
- [16] D. C. E. et al. 90-minute live public demonstration of the online system NLS. <http://sloan.stanford.edu/MouseSite/1968Demo.html>, December 1968.
- [17] E. Frécon and M. Stenius. DIVE: A Scaleable Network Architecture for Distributed Virtual Environments. *Distributed Systems Engineering Journal (DSEJ)*, (5):91–100, 1998. Special Issue on Distributed Virtual Environments.
- [18] M. Frohnmayer and T. Gift. The TRIBES Engine Networking Model. In *Proceedings of the Game Developers Conference*, June 2000.
- [19] N. Furmento, W. Lee, A. Mayer, S. Newhouse, and J. Darlington. ICENI: An Open Grid Service Architecture Implemented with Jini. In *Proceedings of SuperComputing 2002*, November 2002.
- [20] C. Greenhalgh. Equip - extensible platform for distributed collaboration. In *Second Workshop on Advanced Collaboration Environments, held in conjunction with the Eleventh IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, July 2002.
- [21] I. J. Grimstead. RAVE - Resource Aware Visualization Environment. <http://users.cs.cf.ac.uk/I.J.Grimstead/RAVE/index.html>, May 2005.
- [22] I. J. Grimstead, N. J. Avis, and D. W. Walker. Automatic Distribution of Rendering Workloads in a Grid Enabled Collaborative Visualization Environment. In *Proceedings of SC2004: SuperComputing 2004*, November 2004.
- [23] P. Hartling, C. Just, and C. Cruz-Neira. Distributed Virtual Reality Using Octopus. In *Proceedings of IEEE International Symposium on Virtual Reality (VR'01)*, pages 53–62. IEEE Computer Society, March 2001.
- [24] Hewlett-Packard. *HPUX SharedX*. Online manual pages.
- [25] H. Hua, L. D. Brown, C. Gao, and N. Ahuja. A new collaborative infrastructure: Scape. In *Proceedings of the IEEE International Symposium on Virtual Reality 2003 (VR '03)*, pages 171–179. IEEE Computer Society, March 2003.
- [26] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. Kirchner, and J. T. Klosowski. Chromium: A Stream Processing Framework for Interactive Graphics on Clusters. In *ACM SIGGRAPH*, pages 693–702, 2002.
- [27] IBM. IBM Lotus Sametime 3. Specification sheet, IBM Corporation, 2002.

- [28] B. Interactive. Virtual Worlds Platform 5. Product specification, 2001.
- [29] P. L. Isenhour, J. B. Begole, W. S. Heagy, and C. A. Shafer. Sieve - A Java-Based Collaborative Visualization Environment. In *Proceedings of IEEE Visualization '97 Late Breaking Hot Topics*, pages 13–16, October 1997.
- [30] G. Johnson. Collaborative Visualization 101. *ACM SIGGRAPH Computer Graphics*, 32(2):8–11, May 1998.
- [31] N. T. Karonis, M. E. Papka, J. Binns, J. Bresnahan, J. A. Insley, D. Jones, and J. M. Link. High-Resolution Remote Rendering of Large Datasets in a Collaborative Environment. Preprint ANL/MCS-P1030-0203, February 2003.
- [32] J. A. Kohl, P. M. Papadopoulos, and G. A. G. II. CUMULVS: Collaborative Infrastructure for Developing Distributed Simulations. In *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing*, March 1997.
- [33] R. Lang, U. Lang, H. Nebel, D. Rainer, D. Rantza, A. Wierse, and U. Wössner. COVISE version 4.1.1a. User's Manual (draft), University of Stuttgart Computer Centre, June 1997.
- [34] C. M. Lascara, G. H. Wheless, D. Cox, R. Patterson, S. Levy, A. Johnson, J. Leigh, and A. Kapoor. Teleimmersive virtual environments for collaborative knowledge discovery. In *Proceedings of the Advanced Simulation Technologies Conference '99*, pages 416–423, April 1999.
- [35] R. Lea, Y. Honda, and K. Matsuda. Virtual Society: Collaboration in 3D spaces on the internet. *Computer Supported Cooperative Work*, (6):227–250, 1997.
- [36] LeSC. RealityGrid. Project flyer, London e-Science Centre, September 2002.
- [37] V. Normand, C. Babski, S. Benford, A. Bullock, S. Carion, N. Farcet, E. Frécon, N. Kuijpers, N. Magnenat-Thalmann, S. Raupp-Musse, T. Rodden, M. Slater, G. Smith, A. Steed, D. Thalmann, J. Tromp, M. Usch, G. V. Liempd, J. Harvey, and N. Kladias. The COVEN project - Exploring Applicative, Technical, and Usage Dimensions of Collaborative Virtual Environments. *Presence—Teleoperators and Virtual Environments*, 8(2):218–236, 1999.
- [38] O. Open. Web Services Resource Framework (WSRF). [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsrf](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf), March 2005.
- [39] A. Pang and C. Wittenbrink. Collaborative 3D Visualization with CSpray. *IEEE Computer Graphics and Applications*, 17(2):32–41, March-April 1997.
- [40] S. Pettifer, J. Cook, J. Marsh, and A. West. Deva3: Architecture for a Large-Scale Virtual Reality System. In *Proceedings of ACM Symposium in Virtual Reality Software and Technology*, pages 33–40. ACM Press, October 2000.
- [41] PTC. DIVISION Mockup. Data sheet, Parametric Technology Corporation, 2002.
- [42] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, January/February 1998.
- [43] D. J. Roberts and P. M. Sharkey. Minimising the Latency Induced by Consistency Control Within a Large Scale Multi-User Distributed Virtual Reality System. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, volume 5, pages 4492–4497. IEEE Computer Society, October 1997.
- [44] T. Sandholm and J. Gawor. Globus Toolkit 3 Core—A Grid Service Container Framework. Globus Toolkit 3 Core White Paper, July 2003.
- [45] D. Schmalstieg and G. Hesina. Distributed Applications for Collaborative Augmented Reality. In *Proceedings of IEEE Virtual Reality Conference 2002 (VR '02)*, 24-28 March 2002, Orlando, Florida, USA, pages 59–66. IEEE Computer Society, March 2002.
- [46] SGI. SGI OpenGL VizServer 3.0. Data sheet, SGI, September 2002.
- [47] SGI. SGI OpenGL VizServer 3.1. Data sheet, SGI, March 2003.
- [48] M. Stefik, D. G. Bobrow, S. Lanning, D. Tatar, and G. Foster. WYSIWIS revised: early experiences with multi-user interfaces. In *CSCW '86: Proceedings of the 1986 ACM conference on Computer-supported cooperative work*, pages 276–290, New York, NY, USA, 1986. ACM Press.
- [49] H. P. C. C. Stuttgart. COVISE Features. <http://www.hlrs.de/organization/vis/covise/features/>, HLRS, September 2000.
- [50] A. V. Systems. AVS/Express Product Sheet. Product sheet, 2001.
- [51] H. Tramberend. Avango: A Distributed Virtual Reality Framework. In *Proceedings of AFRIGRAPH 2001, 1st International Conference on Computer Graphics, Virtual Reality and Visualization in Africa*. ACM, November 2001.
- [52] W3C. Simple Object Access Protocol (SOAP)—W3C Recommendation Version 1.2—Primer. <http://www.w3.org/TR/soap12-part0/>, June 2003.
- [53] M. Walkley, J. Wood, and K. Brodlie. A Distributed Cooperative Problem Solving Environment. In P. M. A. Sloot, C. J. K. Tan, J. Dongarra, and A. G. Hoekstra, editors, *Computational Science - ICCS 2002, International Conference, Amsterdam, The Netherlands, April 21-24, 2002. Proceedings, Part I*, volume 2329 of *Lecture Notes in Computer Science*, pages 853–861. Springer, 2002.
- [54] J. Walton. Data Visualisation with IRIS Explorer - What's New? Technical Report IETR/8 (NP3070), NAG Ltd., Oxford, UK, 1996.
- [55] S. Wilson, H. M. Sayers, W. Myles, and M. D. J. McNeill. Nomad: An Architecture to Support the Development of Collaborative Virtual Environment Application. In *Proceedings of Eurographics UK 2000*, pages 151–158, 2000.
- [56] J. Wood, H. Wright, and K. Brodlie. Collaborative Visualization. In *Proceedings of IEEE Visualization 1997*, pages 253–269, 549, 19-24 October 1997.
- [57] B. Zeleznik, L. Holden, M. Capps, H. Abrams, and T. Miller. Scene-Graph-As-Bus: Collaboration between Heterogeneous Stand-alone 3-D Graphical Applications. *Computer Graphics Forum*, 19(3), 2000. Proceedings of Eurographics 2000.