Applicability of Group Communication for Increased Scalability in MMOGs

Knut-Helge Vik
University of Oslo
knuthelv@ifi.uio.no

Carsten Griwodz
University of Oslo
griff@ifi.uio.no

Pål Halvorsen
University of Oslo
paalh@ifi.uio.no

ABSTRACT

Massive multiplayer online games (MMOGs) are today the driving factor for the development of distributed interactive applications, and they are increasing in size and complexity. Even a small MMOG supports thousands of players, the biggest support hundreds of thousands of concurrent players. Since they are typically built as strict client-server systems, they suffer from the inherent scalability problem of the architecture. Computing power and bandwidth limitations close to the server limit the possible number of players. Also, the latency of communication between players through the server will be higher than using direct communication. In the paper, we address these issues and investigate improvement options.

A typical MMOG consists of a virtual world with a concept of time and space that is similar to the real world. In it, players are represented by avatars. Only subsets of these avatars interact with each other at any given time. This allows us to divide them into groups, and communication among group members becomes a multi-party communication problem. Thus, to reduce resource consumption, we compare the performance of several algorithms for group communication with the current central server approach. We use overlay multicast as the means of providing group communication, and research algorithms for creating shortest path trees, spanning trees, delay-bounded spanning trees and, more specific, applying Steiner tree heuristics.

Our experimental results indicate that different approaches are useful to reduce resource consumption while achieving a good perceived quality under varying conditions, such as frequent changes in group membership and the demand for low latency.

1. INTRODUCTION

Massive multiplayer online games (MMOGs) are today the driving factor for the development of distributed interactive applications. Game companies are maintaining centralized systems for their games (single server or clustered) in spite of scalability problems. They do it because of the advantages of this architecture. Compared to distributed architectures, they are easier to manage, the game state remains consistent, cheating is easier to detect and prevent, and clients remain anonymous from each other while being known to the game company.

However, most current MMOGs do not optimize event distribution. There are no proxy servers, and clients are not interconnected with each other at all. The server collects and processes game events generated by clients, and uses unicast communication for the distribution of game state updates to all the clients. The overhead of handling all communication among clients through the server can result in long delays and a large amount of consumed networking resources. The traffic concentration at the server is also vulnerable to network performance problems. The game traffic must compete with all other traffic in the network, and it is usually implemented using standard protocols. While resource-wasting already today, such a system will have severe scaling issues when the number of clients and amount of data (e.g., streaming audio and video in future games) increase. We are therefore looking for resource-reducing options while maintaining centralized control wherever it is relevant. Improving the gaming experience in MMOGs is tightly linked to reducing latency, and also making better use of available resources in general.

In a typical MMOG, clients move avatars (in-game characters) around in a virtual game world. The physical location of a client is independent of the virtual location of the avatar(s) it controls. Consequently, the virtual location in the MMOG is the most important factor to determine whether clients are interacting. Client interaction occurs when avatars (in the virtual world) influence each others' gameplay, i.e., by triggering game events such as shooting, talking etc. Game events happening virtually far away from where a client is controlling an avatar are usually irrelevant for its gameplay. And, to address this problem, it is standard practice to define areas of interest (AoI) in the virtual world. Figure 1 illustrates virtual and actual location of clients in an MMOG. Each AoI region contain several avatars controlled by clients. Clients sharing an AoI region receive the game events that happen within that particular virtual region, i.e., avatar position update.

We can derive that clients located in the same AoI will benefit from communicating with each other. Furthermore, group management for AoI regions may save resources like CPU cycles and bandwidth for a typical MMOG.

While alleviating server load by increased group communi-

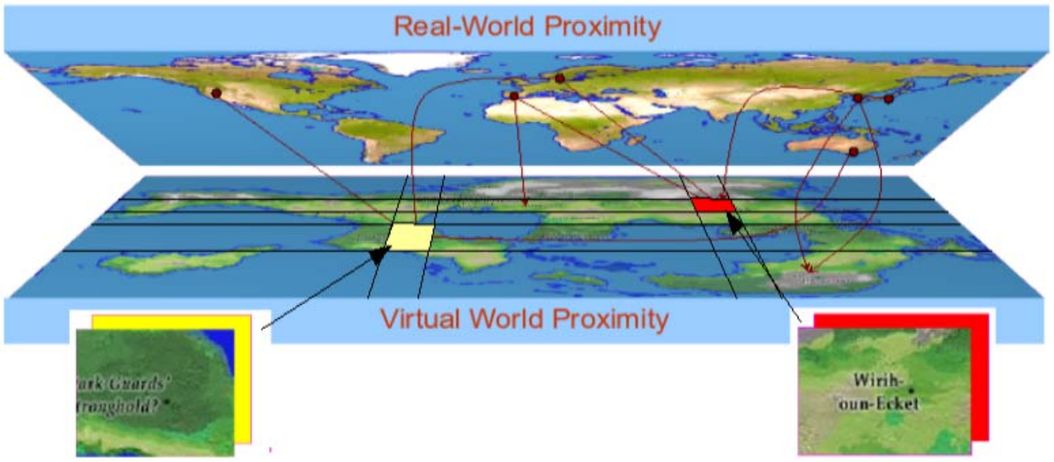

Figure 1: Location in the real world, and area of interest in the virtual world.

cation would be helpful, even more resources would be saved if physical distance could be taken into account in group maintenance. To better understand the relation of physical and virtual proximity, we have therefore taken a closer look at an anonymized tcpdump file of a game trace. This trace, kindly provided by Funcom, contains all anonymized and encrypted packets from one of a few hundred game regions of their popular MMORPG *Anarchy Online* [14]. Only avatars that are located on the same game field may share an AoI. In the one hour trace, we found connections to 175 concurrent clients. We considered that clients who shared a network path to the server would be physically close to each other. To determine whether clients were likely to share a path, we examined the probability that the RTTs of their messages share a probability distribution[1]. The result is shown in figure 2. It visualizes the results of a Wilcoxon test of connection pairs that checks whether clients' RTT values stem from the same RTT base value set. Every dot in the graph shows two connections that share common RTTs with a likelihood of 80% or higher. We observed that many clients shared a network path to the server, and concluded from this that it is worthwhile to examine local distribution and local sub-distribution.

The observations regarding AoI group management and the Wilcoxon test led our focus to one efficient means to improve both aspects, namely multicast. And in this paper, we research application level multicast for current centralized game server architecture as a first step towards a more efficient game system architecture. The rest of the paper is organized as follows: Section 2 looks at game traffic and motivates our use of application layer multicast. In section 3, we introduce our experiments using reduced graphs in relation with tree algorithms and present results. Section 4 discusses and highlights the practical values of the statistics. Finally, we conclude and give directions for further work in section 5.

[1]The first thought was loss correlation but the number of packets per stream is so thin that packet loss correlation is invisible

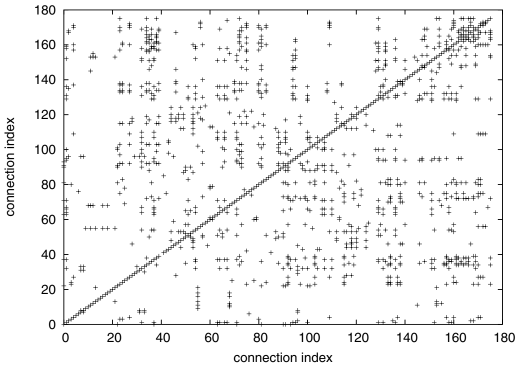

Figure 2: Path sharing matrix.

2. RELATED WORK

Most of the initial network-related research on multiplayer games was aimed at understanding the characteristics of game traffic [26, 8, 3, 29, 6, 11, 7, 4]. The research includes insights on latency and bandwidth issues. In games where players control every action of their avatars, latency becomes detectable at about 100 ms and makes game play impossible at approximately 200 ms [26], whereas, real-time strategy games can tolerate higher latencies [8] and have no particular latency or bandwidth requirements [3, 29]. The bandwidth requirements between a game server and every individual client is generally very low [16]. Furthermore, the traffic is bursty and shows a correlation of inter-arrival times on the minute scale within individual streams [7].

Concrete solutions for improving the scalability of games are less frequent. Several architectures based on proxy servers [2, 24] or a peer-to-peer (P2P) architecture [18, 20] have been proposed. Yamamoto presented distributed event delivery to peers [37], and admission control has been proposed presented for a simple, wireless game [5].

P2P architectures are very successful, but, using a pure P2P model would make it very hard to keep a consistent game state. P2P would distribute the game state among peers and have no central server. Hierarchical architectures using proxies are an alternative that may be more suitable for MMOGs. They include central servers and a set of distributed proxy servers [34].

Nevertheless, independently of the choice of distributed architecture, communication among players should be handled efficiently. We are therefore investigating group communication mechanisms that are appropriate for game traffic. However, we are not specifically investigating event distribution for P2P or central architecture, but rather comparing different multicast distribution tree organizations and relating them to possible game architectures.

2.1 Group communication

In this paper, we look at alternatives to a centralized communication architecture, because centralized systems are used for MMOGs today. As mentioned, AoIs in MMOGs define groups of clients that benefit from communicating directly with each other. We understand this as a group communication problem. One efficient means to provide group communication is multicast. Ideally, IP Multicast could be used. However, group maintenance would require a large number of addresses, and as it is not widely available anyway, we focus on application-layer group communication. There is not much work in this area of research that is specific for games. One of the few examples include Vogel et al. [35] who propose a priority function combined with Dijkstra's shortest path tree (SPT) and minimum spanning tree (MST) algorithms, that can build trees with more direct paths, or longer paths depending on the input priority. Their graphs use priorities to achieve low latency, but they do not consider a maximum end-to-end latency. Neither are minimal tree cost, tree generation time or dynamic group management covered. Unfortunately, all these are very important to online games.

On the other hand, there is a considerable body of work on multicast group maintenance, both at the network and application layer. Several studies have been performed aiming for efficient overlay tree construction and maintenance. For example, Yoid [13] provides an architecture for both space- and time-based multicast. NICE [1] arranges group members into a hierarchy of layers and proposes arrangement and data forwarding schemes. In Narada [12], end systems self-organize into an overlay structure using a fully distributed protocol. Further, end systems attempt to optimize the efficiency of the overlay by adapting to network dynamics and by considering application level performance. ALMI [27] is an application level group communication middleware, tailored toward support of multicast groups of relative small size with many to many semantics. These proposed mechanisms are starting points, but none of these approaches support constraints such as the latency constraints demanded by games. Furthermore, none of them has an understanding of maintaining subsets of a larger set of nodes. An approach that looks at the maintenance of subgroups within a larger set of overlay or P2P nodes is PartyPeer [23]. This system creates subgroups by forming overlay multicast groups as subtrees of a tree that covers the entire set. However, the approach taken can lead to degrading performance since subgroups are always created as subtrees of a single tree for the entire application.

2.2 Graph algorithms

The related work on multicast group maintenance include many different graph algorithms. In our research, we apply basic tree building and graph algorithms to group communication in MMOGs. In particular, we focus on basic tree algorithms solving the problems of finding the shortest path tree (SPT), minimum spanning tree (MST) and Steiner minimal tree in networks (SMT) [15, 19].

An SMT is a cost-minimized tree connecting a subset of vertices within a larger weighted graph, and SMTs can add vertices that are not a part of the group (receivers), i.e., this makes it usefull for MMOG multicast. These extra vertices are called steiner points, and they are used to minimize the cost of the tree. An SMT with no steiner points is equal to an MST.

These algorithms can build trees with diverse properties, and MMOGs have situations that call for different algorithms. For example, the same MMOG may have first-person fighting scenes with frequent group membership changes, regional updates (weather, economy) with a slow update frequency, and global chat channels with membership changes only when players enter and leave the game. It is also common that one MMOG needs different communication styles for different events, such that all-to-all, many-to-many and one-to-many styles are required within the same application. Thus, several trees for different subsets of players with different optimization goals are often desirable in an MMOG.

2.2.1 *Steiner Minimal Trees (in Networks)*

The physical distance between clients within an AoI is potentially very large. It is beneficial to use an SMT because it can add steiner points (usually well connected nodes) to minimize the total cost of the tree. Those steiner points could simply be chosen from the set of all active clients whenever a recomputation of the multicast tree becomes necessary. This would, however, require a rather complex evaluation function that takes more properties of a client into account than just the latency that is introduced by the links that connect it to other clients. To mind come available host resources, bandwidth (in particular uplink bandwidth), location, and stability of the client, which would be dominated mainly by the average duration of a gaming session of that client. Instead of using a complex evaluation function, the MMOG provider could pre-select potential steiner points based on those properties, reduce the size of the problem, and simplify the cost function. One way of doing this is for the provider to establish strategically placed nodes that are well connected and offer only these as potential steiner points in SMT algorithms. Such nodes could be proxy servers that are placed in hosting centers which are frequently well-connected as well. Another way is to select clients that have beneficial properties as steiner points.

No matter what approach to steiner point selection is chosen, the latency-critical flows in MMOGs that are related to movement and action are very thin (they consume very little bandwidth) and the groups are short-lived, such that graph algorithms may not choose any steiner points because of the latency penalty. On the other hand, in an MMOG that includes audio or video streaming, such nodes may fit well.

2.2.2 *Tree Algorithms*

In our experiments, we used nine different tree algorithms (Table 1), including five SMT algorithms. We apply the

SMT problem in an undirected network $G = (V, E, c)$, where V is the number of vertices ($|V| = n$), E is the number of edges ($|E| = m$), $c : E \rightarrow R$ is the edge cost function, $Z \subseteq V$ is a subset with $|Z| = p$. The objective is to find a minimum cost connected subnetwork $G_Z = (V_Z, E_Z, c)$ of G with $Z \subseteq V_Z$. The SMT heuristics we have implemented are:

- Delay Constrained Shortest Path Tree (DCSP) [10], has a time complexity of $O(n^2)$. It is based on Dijkstra's shortest path algorithm. A constrained minimum cost tree and a shortest delay path tree are created in parallel. They are combined to find a delay-constrained shortest path tree.

- Shortest Path Heuristic (SPH) [33], has a time complexity of $O(pn^2)$. It is based on Prims's MST algorithm [15]. The tree is built from a source, and for each iteration, SPH connects the next Z-vertex through the minimum cost path to the tree. Waxman [36] showed that SPH is only performing 5% worse than the optimum SMT on average.

- Delay Constrained Shortest Path Heuristic (DCSPH), is based on SPH, only it uses a delay constraint in addition to cost. It creates a delay constrained tree using the same approach as DCSP.

- Distance Network Heursitic (DNH) [22], has a worst case time complexity of $O(pn^2)$. DNH starts by building a distance network graph using only Z nodes. It then replaces the individual edges in the distance network graph with the original paths in the input graph. Finally, a MST is run from the source to find the DNH SMT.

- Average Distance Heuristic (ADH) [28], has a worst-case time complexity of $O(n^3)$. ADH starts with a forest of trees, each containing only one vertex, and iteratively connecting them using the minimum cost path until there is one tree spanning all Z-vertices.

In addition, we tested Minimum Diameter Degree Bounded Spanning Tree (MDDBST) [31]. It is a heuristic of the Minimum Diameter Degree Limited (MDDL) problem. It has a time complexity of $O(n^3)$, and is based on Prim's MST algorithm. It creates a tree of minimum diameter and uses a degree bound to limit the number of out-edges from a node. Further, we used Prim's Minimum Spanning Tree (MST) and Dijkstra's Shortest Path Tree (SPT) [15]. Moreover, to test centralized event distribution we defined Server-SPT (SSPT), an SPT where the server is always the root node.

3. EXPERIMENTS

To test and evaluate different apporaches, we implemented an application layer multicast simulator, SimALM. It simulates group communication in an MMOG using a centralized server architecture. In the following subsections, we introduce SimALM, describe our test setup and present results.

3.1 The SimALM Simulator

SimALM uses a fully meshed (complete) shortest path graph (SPG) to simulate group communication. The SPG is an application layer graph, where all the nodes are clients.

Algorithm	*Span. tree*	*Optimization goal*	*Constr.*	Time compl
DCSP	no	src/dst hop-cnt	dist	$O(n^2)$
SPH	yes	sum of dist	–	$O(pn^2)$
DCSPH	yes	hop-cnt	dist	$O(pn^2)$
DNH	yes	sum of dist	–	$O(pn^2)$
ADH	yes	sum of dist	–	$O(n^3)$
MDDBST	yes	min-diam. dist	out-deg	$O(n^3)$
MST	yes	sum of dist	–	$O(n^2)$
SPT	no	src/dst dist	–	$O(n^2)$
SSPT	no	server/dst dist	–	$O(n^2)$

Table 1: Algorithms implemented in SimALM .

The underlying router topology is invisible from the point of view of the SPG.

It has been argued that the topology generator BRITE [25] generates Internet-like topologies [21, 17], and we have used it to generate router networks. For our experiments, we used flat undirected Waxman topologies [36] with 200 routers, where the router placement was random in a plane of size 1000×1000 units. The placement in the plane is used by BRITE to calculate the edge weights between the routers (maximum possible edge weight is 16.77 deci units). We assume that every router has one client connected to it.

Next, the BRITE network graph is translated into an undirected fully meshed SPG on the application layer, $G = (V, E, c)$, where V the set of clients in the MMOG, E is the set of undirected shortest path edges connecting the clients, and the edge cost function $c : E \rightarrow R$ that is the sum of the edge weights on the shortest paths from the BRITE network layer graph.

The nodes in the SPG are now clients, and the edges are shortest paths connecting each client pair. These shortest paths between clients include a varying number of routers, that are invisible from the application layer. Each edge in the SPG has two cost values associated to it. One is the edge weight distance, the other is the network layer hop count. They are calculated from the BRITE router network graph, by summing the edge weights on the shortest paths, and the resulting hop count. Remember, each router in the BRITE network graph was transformed to a client in the SPG. Figure 3 shows how a BRITE network graph is converted to an SPG. Furthermore, relaying packets through intermediate nodes contributes to the end-to-end delay, thus, a relay penalty is added for each intermediate router on the shortest path.

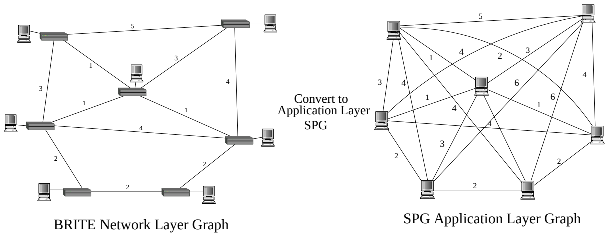
BRITE Network Layer Graph
Convert to Application Layer SPG
SPG Application Layer Graph

Figure 3: A BRITE (router) network layer graph is converted to an application layer SPG.

From the SPG, we choose the best connected client to act as the MMOG server. The remaining are clients controlling an avatar in the MMOG. These avatars roam the virtual world, where they enter and leave AoIs, and they are simulated using a random walk model on a torus. The torus is divided into $|V|$ squares, where each square represents an AoI. At the start of a simulation, each AoI contain one avatar. When an avatar changes AoI, the client sends enter and leave group requests to the central server. The central server maintains one distribution tree for each AoI, such that when it receives a group request for an AoI, it updates the group information, creates a new tree and distributes it to the new members of the AoI. For every enter and leave request that is received by the central server, tree statistics are calculated for the group tree that is created. The central server stores the state for all the groups in the MMOG. Figure 4 illustrates how SimALM works.

In our simulations, the number of clients is fixed, but an MMOG has a varying number of clients playing at any given time. However, we still have unpredictable dynamics in group membership. For now, we assume that member dynamics are handled and that the server always has the latest member view. In addition, we assume that sufficiently good latency estimation for all clients is available to the server. When we drop either of these conditions in future work, we must reconsider this simplification. Furthermore, we are aware that building a new tree for each request is an overhead if there are frequent AoI changes and/or low latency requirements[2]. However, the main purpose of our experiments is to investigate how suitable each algorithm is for different group sizes and game-dependent conditions. We evaluate graph and tree algorithms to determine how well they can be applied in games under different conditions.

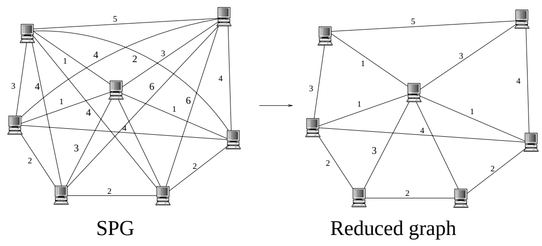

Figure 5: SPG is reduced to a graph with fewer edges.

The speed of the tree algorithms is highly influenced by the input graph. As previously explained, an application layer graph is fully meshed. This is an advantage in terms of connectedness, but the number of edges in a fully meshed graph is a permutation of pairs of vertices, i.e., a graph with 10 vertices has 45 edges, a graph with 45 vertices has 990 edges. Thus, tree algorithms will become considerably slower when the number of vertices increases. Reducing the number of edges in the input graph will improve the computation time, and SimALM supports a set of strategies for creating subgraphs. They are introduced in section 3.4. Figure 5 shows an example of reducing an input graph.

The distribution trees are created using the tree algorithms listed in table 1. They are introduced in section 2.2.2. DCSP and DCSPH both produce trees even if delay constrained paths cannot be found.

[2]Our ongoing work address this issue

3.2 Performance Metrics

We used several metrics to compare the variety of tree algorithms. In our investigation, we mainly consider these four metrics:

- *Total cost* is the sum of all the edge weights in a tree.
- *Maximum pair-wise distance* is the maximum distance in terms of edge weights between any two nodes in the tree.
- *Execution time* is the time in milliseconds (ms) that the computation of the tree takes.
- *Out-degree* is the number of direct neighbors in the resulting tree.

The *total cost* of an AoI distribution tree is especially important considering the current centralized server approach that the vast majority of MMOGs use. They apply an SSPT where all the clients are leaves and the server is the source. The resource usage on the server is very high, which is why the MMOG provider must scale up the central server cluster when the number of players increase. Thus, as the total cost can give an indication of the required resource consumption, we aim for a mechanism minimizing the *total cost* .

Another important aspect in group communication is the *maximum pair-wise distance*. It is particularly important when the event distribution is many-to-many within a multicast tree, as often in MMOGs. The current situation with a centralized server does have drawbacks when it comes to *maximum pair-wise distance*. The location of the server in relation to the location of the clients, has a significant effect. I.e., a central server cluster in North-America, gives the lowest *maximum pair-wise distance* for clients located in North-America, but higher for clients in, for example, Europe. A distribution tree with vastly different *maximum pair-wise distance* may give unfair playing conditions. Keeping the *maximum pair-wise distance* as small as possible should be a goal.

The *execution time* of an algorithm provides an estimate of its performance when the group size increases, but also how it behaves when the group membership changes frequently. The time it takes to create a new AoI distribution tree, must be a function of the expected dynamics in an AoI, i.e, if avatars join/leave rapidly, a distribution tree must be configured fast. In our experiments we investigate how the tree algorithms are usable. The *execution time* affects the game play if there is fast group changes.

Until now, *out-degree* has not been a concern in MMOGs because flows are small. But, more resource-intensive audio-video streams are features that MMOGs will soon support. A client-server architecture would have scalability problems. In a distributed approach where also clients forward streams, their *out-degree* needs to be limited, because each link consume limited client resources.

In our experiments, the DCSP and DCSPH algorithms used a delay constraint of 5, and MDDBST a degree constraint of 4 throughout the simulations. Due to time constraints, we did not vary these parameters. Moreover, every tree algorithm was tested for each network, and every experiment was run 10 times on Pentium 4 CPUs. Finally, the graph and tree algorithms are implemented using the Boost graph library [32].

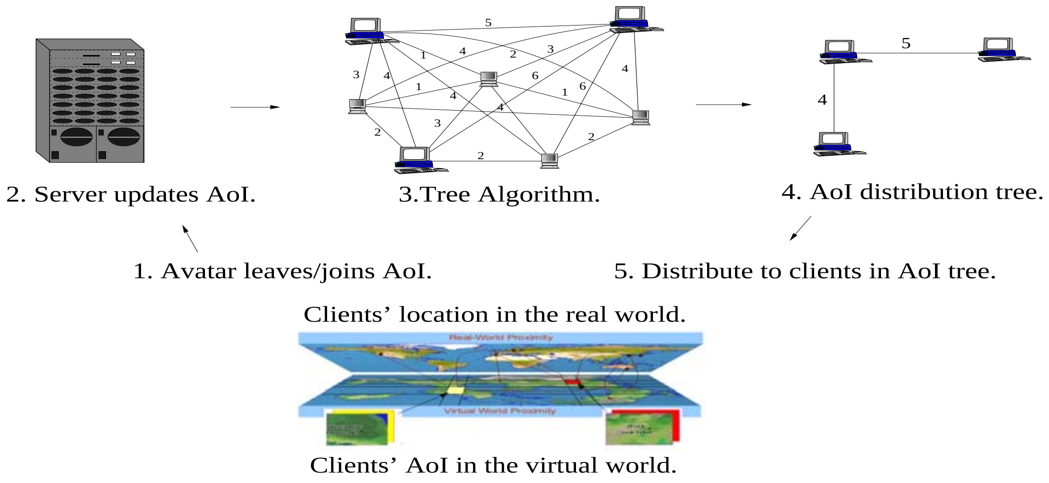
2. Server updates AoI.
3.Tree Algorithm.
4. AoI distribution tree.
1. Avatar leaves/joins AoI.
5. Distribute to clients in AoI tree.
Clients' location in the real world.
Real-World Proximity
Virtual World Proximity
Clients' AoI in the virtual world.

Figure 4: SimALM - 1) An avatar changes AoI and the client sends leave/join requests to a central server, 2) the central server updates the AoI group, 3 and 4) the server creates new AoI-trees, 5) the new trees are distributed to clients in AoI tree.

In the next sections, the results from the experiments we performed are presented. First, we performed experiments using the entire SPG as input to the tree algorithms for each group request (section 3.3). Second, we used graph reduction algorithms to reduce the number of vertices and edges in the input graph (section 3.4).

3.3 Complete Graph

The next results are from experiments that used the full SPG as input to the tree algorithms. Every input graph had 200 clients, and was fully meshed. This is a massive overhead, but provides a benchmark when we later test reduced graphs. The experiment parameters are listed in table 2.

3.3.1 Results with complete graphs

Figure 6 presents the average *total cost* for the AoI groups. It is computed by adding all the edge weights in the trees. We observe that the spanning tree algorithms (MST, SPH, DNH and ADH) produce considerably cheaper trees than the shortest path algorithms SPT, SSPT, and DCSP. DCSPH is a spanning tree algorithm based on SPH. Along with DCSP, it optimizes for network layer hops (calculated from the BRITE graph) within the delay constraint. Each edge has an associated network layer hop (cost), and edge distance (weight). such that the *total cost* is increased. MDDBST produces the minimum diameter tree, within the degree constraint. If MDDBST had no degree constraint (infinity), it would produce a tree where some clients have a much higher out-degree, and the tree would resemble a shortest path tree.

Figure 7 show snapshots of some actual group trees from the simulations. The figures illustrate how results can differ so much. Figure 7(a) (SSPT) visualizes what a group tree using the current centralized architecture looks like. The server is at the center, and all group members are leaves. This increases the *total cost* of a tree, since it is only the shortest path from the source that is considered. Figure 7(b) shows DCSP, which is based on Dijkstra's shortest path algorithm, and the example tree shows this influence by connecting a large number of group members to the center node.

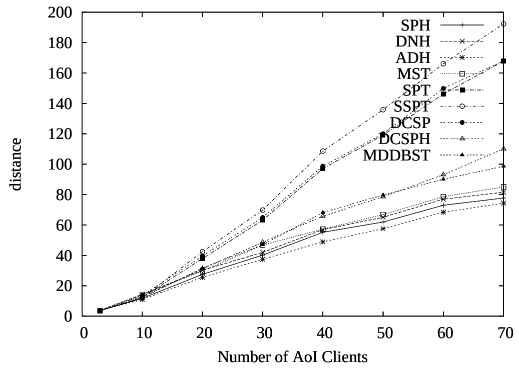

Figure 6: SPG: Average *total cost* of the AoI distribution trees.

Figure 7(c), DCSPH, is a spanning tree algorithm with a delay constraint of 5, and we observe that the tree is fairly compact, although it is a spanning tree algorithm. Figures 7(d) and 7(e) are based on the spanning tree algorithms SPH and ADH, respectively. They are longer and do not have a clear center, because they optimize for the sum of edge weights. This fact makes the resulting trees likely to have longer *maximum pair-wise distances* within the tree, thus affecting many pairwise connections.

Figure 8 plots the average *maximum pair-wise distance* between leaf clients in AoI distribution trees. As expected, spanning tree algorithms contain node-pairs with a higher average *maximum pair-wise distance* than the shortest path tree algorithms. Spanning tree algorithms optimize for *total cost*, whereas, shortest path algorithms optimize for source-destination distance. Thus, shortest path algorithms produce a more stable (even) *maximum pair-wise distance*. In spanning trees there will be node-pairs with a much higher distance. Among the SMT algorithms we observe that ADH has the lowest *maximum pair-wise distance*. Furthermore, the delay constrained algorithms DCSP and DCSPH can be

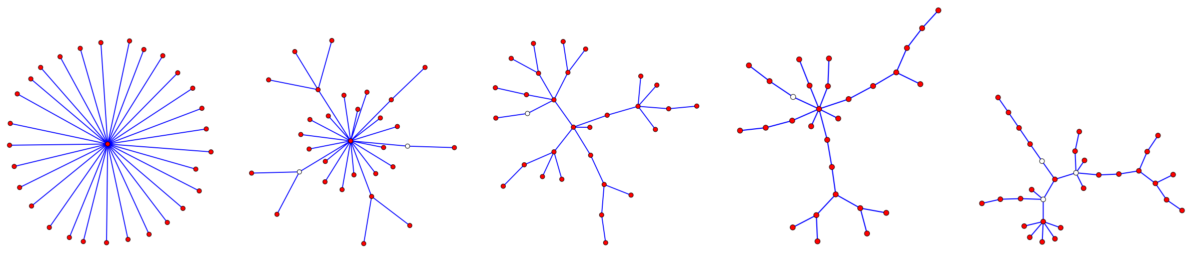
(a) SSPT (b) DCSP (c) DCSPH (d) SPH (e) ADH

Figure 7: AoI group-tree snapshots. Group members are dark, well connected clients are light.

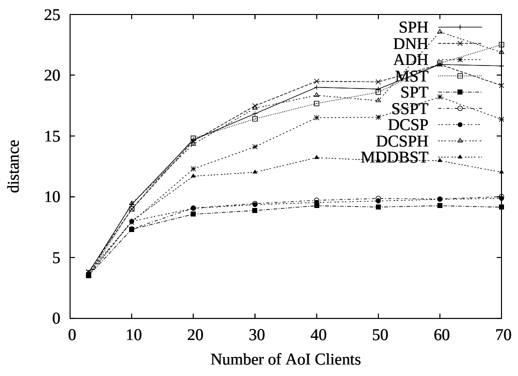

Figure 8: SPG: Average *maximum pair-wise distance* between leaf clients in the AoI distribution trees.

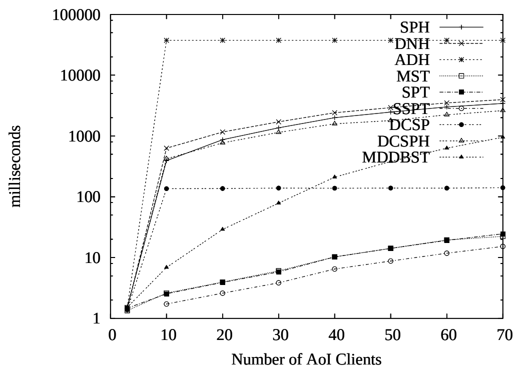

Figure 9: SPG: Average *execution time* (ms) of the AoI distribution trees.

tuned according to the delay constraint, such that with a loose (high) delay constraint, the resulting tree will have a higher *maximum pair-wise distance*. We would expect DCSPH to have a lower *maximum pair-wise distance*. But, it is a spanning tree algorithm that optimizes for hop count and ignores edge delays. This combined with a loose delay constraint results in higher pairwise distances.

The *execution time* is measured in milliseconds, and is how long a tree algorithms requires to create a new AoI group tree. Figure 9 shows the average *execution time* of the AoI distribution trees. We observe that ADH is by far the slowest algorithm. It takes on average 37 seconds to create a new distribution tree. Two new trees are created for each group change (leave/join) operation, hence we can deduct that ADH is not suitable for dynamic AoI group management. The reason ADH is so expensive lies in the algorithm details. It calls Floyd-Warshall [15], an $O(n^3)$ algorithm, and has very expensive table update routines. This makes the execution time of ADH very dependent on the input graph. And, in the case of SPG the input graph size is fixed. However, ADH is an alternative if the groups are static and/or not time-critical.

The fastest SMT algorithm is DCSP, because it consists mainly of two calls to Dijkstra's shortest path algorithm. The remaining SMT algorithms (SPH, DNH and DCSPH) have the same complexity (table 1) and close to the same execution times.

From the results shown in figure 10, we see that SPT, SSPT and DCSP have an average *out-degree* that is much higher than the other algorithms. The *out-degree* of delay constrained algorithms is expected to increase as the delay constraints are loosened, and we can see this tendency in DCSP and DCSPH. However, DCSP is based on Dijkstra's SPT, and thus, has a higher maximum out-degree. The remaining spanning tree algorithms (SPH, DNH, ADH and MST) all have lower *out-degree* on average. The *out-degree* of MDDBST is always four because it was the degree constraint.

3.4 Reduced Graph

The previous experiments used SPG as input to the tree algorithms. Here, we introduce subgraph algorithms that reduces the number of edges and vertices, i.e., with the AoI members as a minimal set of vertices. The reduced graph is then used as input to the tree algorithms. Such a reduction has effects on the output tree quality and especially the algorithm *execution time*.

An SPG holds all the MMOG clients. However, the number of vertices can be reduced by allowing only AoI members in a reduced graph. Moreover, when we apply SMT algorithms, we allow a set of well-connected clients to enter the reduced graph. We will refer to them as well-connected clients, but they may just as well be proxy nodes from the MMOG provider [34]. However, in our experiments, all

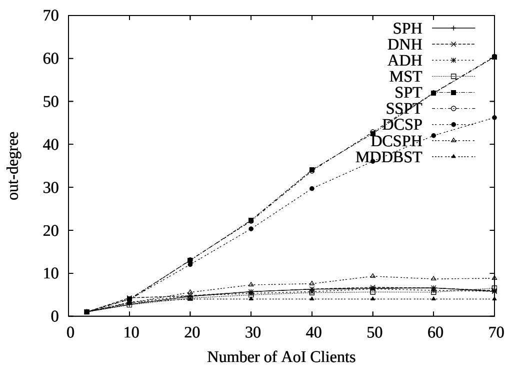

Figure 10: SPG: Average *out-degree* of the AoI distribution trees.

nodes are also clients controlling an avatar in the MMOG.

By applying these simple techniques, the number of vertices in the reduced graph is bounded by the number of members in the AoI (Z) and the number of well-connected clients we choose to include. The result is the vertex subset V_Z of G, where $V_Z - Z$ is the set of well-connected clients. The number of well-connected clients for each input graph can, for example, be a server-set parameter. In our experiments, the total number of well-connected clients is fixed to 30 (in the entire network) and the number of non members for each input graph is $|Z| * 0.25$. The selection criteria is based on the relative location of the clients.

The fully meshed SPG has a permutation of pairs of vertices. In our experiments, SPG includes $|V| = 200$ and $|E| = 19,900$. However, the resulting tree will contain only $|E_Z| = |V_Z| - 1$. We use the graph pruning algorithm k-best-links (kBL) [38], to reduce the number of edges in the input graph. It works as follows: 1) For each member vertex, include the k best out-edges that are not yet included, 2) For each well connected non-member vertex, include an edge to each member vertex. Step 2 was added to the original kBL to ensure connectedness of the reduced graph. kBL produces a graph with $|E| = k*|Z| + |nonmembers| * |Z|$. Suppose we have 50 vertices, where 40 are member clients, and 10 are well connected non-member clients. A fully meshed graph has 1225 edges, while kBL, with $k = 2$, produces a graph with $2 * 40 + 10 * 40 = 480$ edges. After applying kBL we are left with the subset $S(Z, G)$ of G, where $S(Z, G)$ is the subset of edges of G that contain a node in Z. The final subgraph is then $G_Z = (V_Z, S(Z, G), c)$.

The well connected clients are, in our experiments, contained in a fixed size pool. They are available to the server at the start of the simulation, and are selected to be in the pool based on the relative location in the SPG. A well connected client that is included in an AoI distribution tree is most likely beneficial to keep longer in the tree. Such that, the well connected clients already in a distribution tree are kept in the input graph when the next group change occur. We test both keeping well connected clients across AoI reconfigurations, and selecting new ones for each time.

The above observations form the foundation of the experiments using reduced graphs as input to tree algorithms. Next, we list the subgraph creation algorithms that we used in our experiments:

Description	*Parameter*
Number of nodes in SPG	$n = 200$
Delay constraint (DCSP and DCSPH)	5
Degree constraint (MDDBST)	4
k-Best Links (kBL)	$k = 2$
Well connected nodes, pool size (fixed)	30
Non-members in (if appl.) reduced graph	$\|Z\| * 0.25$
Test computers	P4 CPU

Table 2: The configuration of the experiments.

- **CM**, the Complete Member (CM) graph, $(Z, S(Z, G), c)$, is a fully meshed graph with only members from an AoI.
- **RR-CM**, a CM graph, pluss s non-members selected in a round-robin manner from a pool of well connected clients. The set s is new for each tree creation.
- **F-CM**, same as RR-CM, only now the set s of non-members include, non-members from the previous instance of the tree, and from a FIFO queue.
- **RR-kBL**, is an RR-CM with reduced number of edges, using kBL.
- **F-kBL**, is an F-CM with reduced number of edges, using kBL.

The experiment parameters are listed in table 2.

3.4.1 Results with reduced graphs

We observed in section 3.3.1 that using the complete SPG as input to a tree algorithm is a massive overhead in terms of *execution time*. Next, we use reduced graphs that are created from the SPG. It is important that a reduced graph does not decrease the quality of the resulting tree, in terms of the performance metrics (section 3.2).

Figure 11 shows the average *total cost* of trees with AoI size between 40-50, using SPG, F-CM and F-kBL. As expected, SPG does have the lowest *total cost* . However, the reduced graph approaches are surprisingly close, i.e., reducing the input graph has little influence on the total cost. DCSP and DCSPH optimize for hop-count and not distance, thus a clear tendency cannot be extracted. However, the motivation for using reduced graph algorithms in relation to tree algorithms is clearly there. Furthermore, CM is the most basic reduced graph algorithm, because it only includes the group members as vertices. When CM is used, it is only useful to apply the non-SMT algorithms, since SMT reduces to an MST problem when $p = n$ ($|Z| = |N|$). The non-SMT algorithms are faster, but they do not support adding potential well connected clients to the tree. Figure 13 shows the *total cost* improvement with (F-CM) and without (CM) well connected clients. As we observe the *total cost* improvement using well connected clients is not particularly high.

Table 3 shows the average *execution time* of AoI distribution trees with 40-50 members. We observe that, among the SMT algorithms, DCSP is much faster. This is expected, when looking at the complexity of the algorithms. The *execution time* drops considerably using the reduced graphs CM, F-CM and RR-CM, compared to SPG. ADH, now only uses half a second, while SPH, DNH and DCSPH uses within

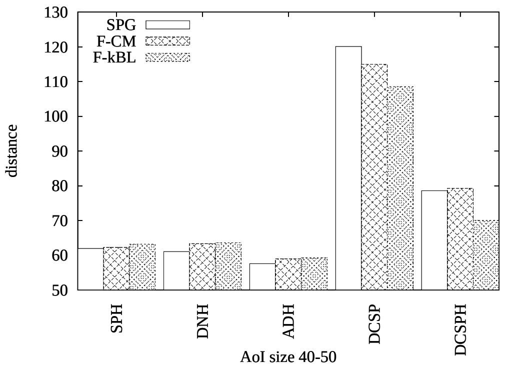

Figure 11: Average *total cost* variation of applying the heuristics to SPG, F-CM and F-kBL. AoI size 40-50.

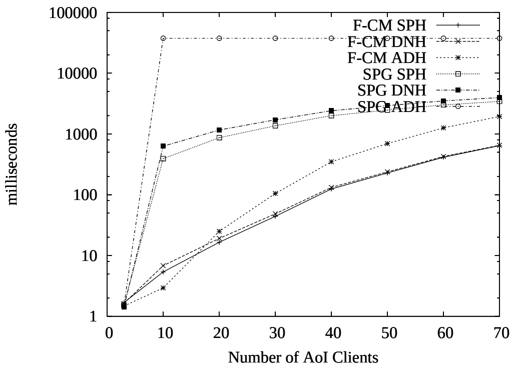

Figure 12: Average *execution time* (ms) improvement using reduced graph F-CM compared to SPG.

200 milliseconds. Figure 12 illustrates the *execution time* between SPG and F-CM as the number of AoI members increase. Also, we notice that the SMT algorithms increase the *execution time* slightly, when well-connected clients are added to the input graph. When the edge pruning algorithm kBL is applied in F-kBL and RR-kBL, we observe that the *execution time* drops to 100 ms, for SPH, DNH and DCSPH, while ADH still uses about half a second. Figure 15 illustrates the execution time variation of applying the heuristics to F-kBL instead of F-CM.

As was explained previously, RR-CM does not keep well connected clients across tree reconfiguration while F-CM does. This makes the the distribution trees more stable

Test	*SPH*	*DNH*	*ADH*	*DCSP*	*DCSPH*
SPG	2600	3250	37510	140	2720
CM	140	140	300	10	160
RR-CM	170	180	530	10	190
F-CM	170	190	520	20	180
RR-kBL	100	110	470	10	100
F-kBL	80	100	480	10	90

Table 3: Average *execution time* in milliseconds for AoI distribution trees of size 40-50.

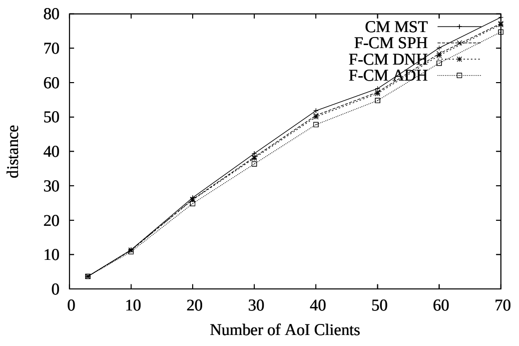

Figure 13: Average *total cost* improvement with (F-CM) and without well connected clients (CM).

across several group requests. As shown in figure 14, it has a positive effect on the average tree cost, but the improvement is only slight. When we apply F-kBL and RR-kBL we see from figure 14 that the *total cost* does not suffer much. This is, again, a clear indication that the *total cost* of the trees is not heavily influenced by kBL. DCSP and DCSPH optimize for hop-count with a delay constraint and actually have a lower *total cost.*

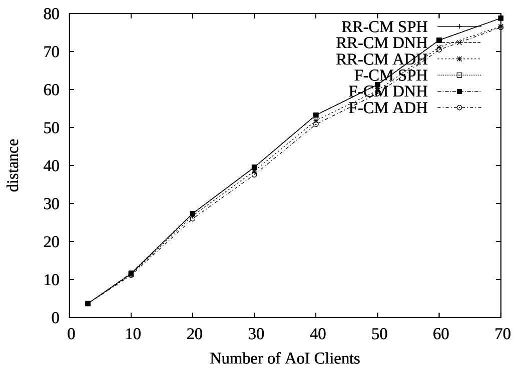

Figure 14: Average *total cost* improvement when chosen well connected clients are kept across group events.

Figure 16 compares the *maximum pair-wise distance* between FF-CM and F-kBL. We observe that there is no noticeable effect on the *maximum pair-wise distance* when the edge pruning strategy (kBL) is applied.

Table 4 shows that the number of well connected clients in the AoI distribution tree increase when they are kept across tree reconfigurations (RR-CM vs F-CM). In addition, a noticeable increase is detected when kBL is used. Previously, we observed that the *total cost* suffered only slightly when kBL was used. Remember, kBL connects every well connected client to each AoI member, and the effect is that the well connected clients are more likely to be included in the AoI distribution tree.

Finally, we observe from figure 17 that there is no significant effect on the *out-degree* when we compare F-CM, F-kBL and SPG, except for DCSP which benefit from kBL.

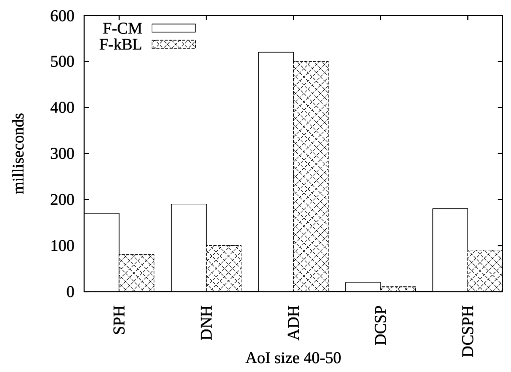

Figure 15: Average *execution time* (ms) variation of applying the heuristics to F-kBL instead of F-CM. AoI size 40-50.

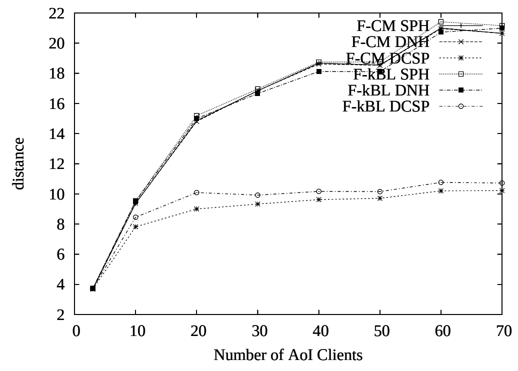

Figure 16: Average *maximum pair-wise distance* variation of F-CM and F-kBL.

4. DISCUSSION

We believe that an optimized centralized architecture will be the next natural step in the evolution of MMOGs. Although such an approach involves the clients to a fairly small degree, clients will need an application layer multicast implementation that handles packet forwarding. To look at such a scenario, we have performed simulations that compare a set of graph algorithms for their efficiency of maintaining multicast groups in an MMOG scenario. The experiments simulate group communication with a centralized approach to group management. All computations are performed by a central game server that creates a new multicast tree for each AoI involved in a join/leave operation.

From the results, we observed that using a large fully meshed graph as input to tree algorithms increases the *execution time* substantially. In particular, ADH suffered when it used the full SPG as input. A reduced graph approach did not decrease the quality of the distribution tree, in any particular way. The *out-degree* and *maximum pair-wise distance* stayed about the same, and the *total cost* only slightly increased, but the *execution time* dropped considerably. We saw the same effect when the pruning strategy kBL was applied, and the *execution time* still dropped. Moreover, the number of well connected clients increased in the AoI distribution tree. Of the tree algorithms, we saw that the SMT algorithms SPH and DNH performed well, and had a tolerable *execution time* of about 100 ms when kBL was used. DCSP, DCSPH and MDDBST needs to be tested with varying constraints before a proper evaluation can be made.

Test	*Stat*	*SPH*	*DNH*	*ADH*	*DCSP*	*DCSPH*
RR-CM	Ave	0.62	0.76	2.98	2.14	0.87
	Med	1.0	1.0	2.0	3.0	0.0
F-CM	Ave	1.33	1.6	5.17	3.39	1.63
	Med	2.0	3.0	7.0	4.0	2.0
RR-kBL	Ave	1.92	2.14	3.55	4.83	3.42
	Med	3.0	3.0	3.0	4.0	3.0
F-kBL	Ave	2.84	3.35	5.66	5.87	4.63
	Med	4.0	5.0	5.0	4.0	7.0

Table 4: Number of non-members (well-connected clients) in AoI distribution trees of size 40-50.

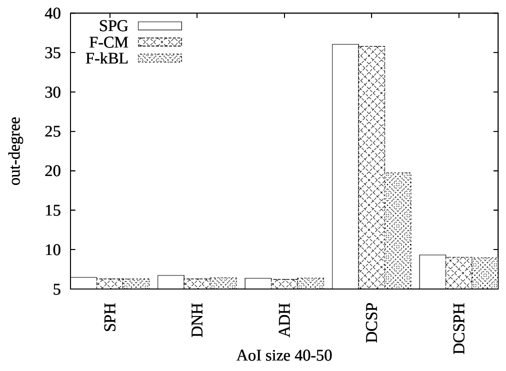

Figure 17: Average maximum *out-degree* using SPG, F-CM and F-kBL.

Thus, the results show us that the various event streams in MMOGs, with different requirements and characteristics, could use distribution trees from most of the tested algorithms because of their varying properties. For example, some algorithms produce low cost trees, but are complex and thus time consuming to execute (like ADH), i.e., such an algorithm could be used for static groups like a multiparty team play audio conference. On the other hand, an algorithm with lower *execution time* (like DCSP) is required for rapidly changing AoIs, with the tradeoff of higher *total cost*.

For the tests presented in this paper, we have generated a new tree for each change in the AoI. However, in most cases, this is not neccessary, and our ongoing research include algorithms to add and remove clients dynamically to AoI distribution trees, without a total reconfiguration. In addition, we are looking at subgraph reconstruction, where only parts of the tree is reconfigured. The reconfiguration can be done by applying the graph and tree algorithms presented in this paper.

For the group management, we have chosen to use graph algorithms instead of overlay multicast approaches known from the literature. The latter are frequently distributed and may seem more appropriate for the distributed systems problem that we want to solve. However, MMOGs are centralized today, such that we have not considered it a major limitation. Additionally, we do see the advantage of starting with well-known heuristics from graph theory that have known performance properties, even though they were usually not intended for "degraded graphs" such as the fully meshed graph that we have available at the application layer. The basic algorithms have provided us with implementation options for cost optimized trees and delay constrained trees, both of which are separately applicable to the various communication needs of MMOGs.

Our experiments used reduced graphs as input to tree algorithms. The reduced graphs included group members and (optionally) additional well-connected clients, that were chosen based on the relative location to the group members. We propose that a central server in an MMOG has a pool of proxy servers or selected clients it can use in the group management. From this, the central server can reduce the load by applying group communication if necessary. An important consideration for an MMOG provider is his need to prevent cheating and provide fair gaming. Using only proxy servers makes it easier to meet these requirements. On the other hand, enabling arbitrary clients as potential relay nodes saves cost in terms of setting up and maintaining dedicated hosts. The MMOG provider saves money, but it requires a trust relation between the client and the MMOG provider to prevent cheating. Another consideration in favor of the proxy solution is that it is relatively straight-forward to install proxy servers in well-connected hosting centers, while well-connected clients must be selected dynamically.

A study of user patterns relating to MMOGs [6] showed that a large number of users will try out an MMOG once it is released. The number of players decreases after the initial boom, and flattens out. Moreover, it was shown that the physical location of active players is heavily influenced by day and night-time. The study clearly shows that it is possible to predict what the user patterns and consequently, what the long-term traffic patterns look like in MMOGs.

5. CONCLUSIONS AND FUTURE WORK

Today's online game servers are typically built as strict client-server systems, i.e., they suffer from the inherent scalability problem of the architecture. Computing power and bandwidth limitations close to the server limit the number of players, and the users might experience huge latencies [16]. We have therefore evaluated several application layer multicast mechanisms in the central server scenario to look at options for possible improvements.

The results from the experiments provide guidelines for the future work in our project. Among the algorithms we tested, we found that the delay constrained algorithms DCSP and DCSPH give basic trade-offs between a latency balanced tree and low hop count. The SMT algorithm ADH produced trees with a low total cost, and a lower maximum pair-wise latency than the other spanning tree algorithms (SPH, DNH, DCSPH and MST). However, ADH is $O(n^3)$ and we have seen that it is not suitable if we want a distribution tree fast. SPH, DNH and DCSHP reduced the execution time to an acceptable level and kept the tree quality when kBL was applied. DNH can be further optimized using techniques highlighted by Poggi and Werneck [9]. The overall results makes us more motivated for studying SMT algorithms. Inventing one universal algorithm is hard because our experimental results indicate that the algorithms are useful for different event streams to reduce resource consumption while achieving a good perceived quality under varying conditions, such as frequent changes in group membership and the demand for low latency.

To improve our current system, we are currently looking at dynamic algorithms that do not require full tree reconfiguration each time we have a change in the AoI. With an increased use of audio-video streaming in MMOGs, the out-degree of individual nodes in the trees also becomes a concern. We will therefore examine the Minimum-diameter, degree-limited spanning tree problem that addresses out-degree limitations [30], and related algorithms. We tested MDDBST, but due to time constraints, further evaluation is required.

Finally, we have focused on optimizing a centralized architecture, where the server can apply group communication to reduce the server load. The centralized group communication that we simulate is located on the main game server, i.e., there is still room for considerable improvements using distributed architectures. Future work will consist of investigating distributed architectures like proxy technology and P2P. Combining group communication and partial game state distribution has many challenges that we shall look into. For instance, we plan to study the group steiner tree problem [19] in relation with proxy technology and P2P.

6. REFERENCES

[1] S. Banerjee and B. Bhattacharjee. Analysis of the NICE application layer multicast protocol. Technical Report UMIACSTR 2002-60 and CS-TR 4380, Department of Computer Science, University of Maryland, College Park, June 2002.

[2] D. Bauer, S. Rooney, and P. Scotton. Network infrastructure for massively distributed games. In *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, pages 36–43, Braunschweig, Germany, Apr. 2002.

[3] P. Bettner and M. Terrano. 1500 archers on a 28.8: Network programming in Age of Empires and beyond. In *Game Developers Conference*, San Jose, CA, USA, 2001.

[4] M. S. Borella. Source models of network game traffic. *Elsevier Computer Communications*, 23(4):403–410, Feb. 2000.

[5] M. Busse, B. Lamparter, M. Mauve, and W. Effelsberg. Lightweight QoS-support for networked mobile gaming. In *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, pages 85–92, Portland, OR, USA, 2004.

[6] C. Chambers, Wu-chang Feng, S. Sahu, and D. Saha. Measurement-based characterization of a collection of on-line games. In *Proceedings of the USENIX Internet Measurement Conference (IMC)*, pages 1–14, Berkeley, CA, USA, 2005.

[7] K.-T. Chen, P. Huang, C.-Y. Huang, and C.-L. Lei. Games traffic analysis: An MMORPG perspective. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 19–24, Stevenson, WA, USA, 2005. ACM Press.

[8] M. Claypool. The effect of latency on user performance in real-time strategy games. *Elsevier Computer Networks*, 49(1):52–70, Sept. 2005.

[9] M. P. de Aragão and R. F. F. Werneck. On the implementation of MST-based heuristics for the Steiner problem in graphs. In *ALENEX*, pages 1–15, 2002.

[10] G. Feng, Tak-Shing, and P. Yum. Efficient multicast routing with delay constraints. *Int. J. Commun. Sys.*, 12:181–195, January 1999.

[11] W.-c. Feng, F. Chang, W.-c. Feng, and J. Walpole. Provisioning on-line games: a traffic analysis of a busy Counter-strike server. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, pages 151–156, Marseille, France, 2002.

[12] G. Fox and S. Pallickara. The Narada event brokering system: Overview and extensions. In *PDPTA '02: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 353–359. CSREA Press, 2002.

[13] P. Francis, S. Ratnasamy, R. Govindan, and C. Alaettinoglu. Yoid project.

[14] Funcom. Anarchy online. http://www.anarchy-online.com/, Feb. 2006.

[15] M. Goodrich and R. Tamassia. *Algorithm Design: Foundations, Analysis and Internet Examples*. John Wiley and Sons, 2002.

[16] C. Griwodz and P. Halvorsen. The fun of using TCP for an MMORPG. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Newport, RI, USA, May 2006. ACM Press.

[17] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz. On realistic network topologies for simulation. In *International workshop on models, methods and tools for reproducible network research*, pages 28–32, Karlsruhe, Germany, 2003.

[18] S.-Y. Hu and G.-M. Liao. Scalable peer-to-peer networked virtual environment. In *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, pages 129–133, Portland, OR, USA.

[19] F. Hwang, D. Richards, and P. Winter. The Steiner tree problem. *Elsevier Annals of Discrete Mathematics*, 53:203–282, 1992.

[20] T. Iimura, H. Hazeyama, and Y. Kadobayashi. Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, pages 116–120, Portland, OR, USA, 2004.

[21] C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator. Technical Report CSE-TR-433-00, Dept. of EECS, University of Michigan, Ann Arbor, Sept. 2000.

[22] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Inf.*, 15:141–145, 1981.

[23] L. S. Liu and R. Zimmermann. Immersive peer-to-peer audio streaming platform for massive online games. In *Proceedings of the IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME)*, Las Vegas, NV, USA, 2006.

[24] M. Mauve, S. Fischer, and J. Widmer. A generic proxy system for networked computer games. In *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, pages 25–28, Braunschweig, Germany, Apr. 2002.

[25] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: Universal topology generation from a user's perspective. Technical Report 2001-003, 1 2001.

[26] L. Pantel and L. Wolf. On the impact of delay on real-time multiplayer games. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 23–29, Miami Beach, FL, USA, 2002.

[27] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, pages 49–60, 2001.

[28] V. Rayward-Smith and A. Clare. The computation of nearly minimal Steiner trees in graphs. *International Journal of Mathematical Education in Science and Technology*, 14(1):8pp, 1983.

[29] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu. The effect of latency on user performance in Warcraft III. In *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, pages 3–14, Redwood City, CA, USA, 2003.

[30] S. Shi and J. Turner. Routing in overlay multicast networks. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, New York, NY, USA, 2002.

[31] S. Y. Shi, J. Turner, and M. Waldvogel. Dimensioning server access bandwidth and multicast routing in overlay networks. In *Prceedings of NOSSDAV 2001*, pages 83–92, June 2001.

[32] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, 2002.

[33] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japonica 24*, 24(6):573–577.

[34] K.-H. Vik. Game state and event distribution using proxy technology and application layer multicast. In *Proceedings of the ACM International Multimedia Conference (ACM MM)*, pages 1041–1042, Singapore, 2005. ACM Press.

[35] J. Vogel, J. Widmer, D. Farin, M. Mauve, and W. Effelsberg. Priority-based distribution trees for application-level multicast. In *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, pages 148–157, Redwood City, CA, USA, 2003.

[36] B. M. Waxman. Routing of Multipoint Connections. *IEEE Journal of Selected Areas in Communications*, 6, December 1988.

[37] S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito. A distributed event delivery method with load balancing for mmorpg. In *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, pages 1–8, Hawthorne, NY, USA, 2005.

[38] A. Young, C. Jiang, M. Zheng, A. Krishnamurthy, L. Peterson, and R. Wang. Overlay mesh construction using interleaved spanning trees, 2004.