QuON – a Quad-Tree Based Overlay Protocol for Distributed Virtual Worlds

Helge Backhaus and Stephan Krause
Institute for Telematics
Universität Karlsruhe (TH)
Zirkel 2, D–76128 Karlsruhe, Germany
Email: [backhaus, stkrause]@tm.uka.de

***Abstract*—Massively Multiplayer Online Games and Virtual Worlds are among the most popular applications on the Internet. As player numbers increase, the limits of the currently dominant client/server architecture are becoming obvious. In this paper we propose a new distributed event dissemination protocol for Virtual Worlds and MMOGs. This protocol is based upon the idea of mutual notification: All players send their game event messages directly to all neighboring players inside their Area of Interest. The connectedness of the system is ensured by binding neighbors. They are selected using quad-trees. We show by simulation that the proposed system achieves practical performance for Virtual Worlds and Massively Multiplayer Online Games.**

I. Introduction

In recent years, Massively Multiplayer Online Games (short: MMOGs) have become a popular genre among computer gamers. Additionally, virtual worlds like Second Life have created much interest in mainstream media. Traditionally, MMOGs and Virtual Worlds rely on central servers or server farms to disseminate game events among players. As player numbers increase, however, these approaches reach their limits. Not only is maintaining a huge server infrastructure a significant cost factor in providing an MMOG, the capacity of the server also limits the number of concurrent players in a game. Most MMOG developers today resort to techniques known as instancing. Instead of providing one shared world for all players, they provide several copies of the game world, each containing only a fraction of the players. Players in different instances of the game world can usually not interact.

P2P technologies have emerged as a possible way to solve the scalability problem of client/server based architectures. There is a growing interest in P2P protocols for MMOGs and Virtual Worlds in the research community. Several different new architectures have been proposed. In a previous work, we classified these so called *Distributed Virtual Environments* (*DVE*) and evaluated their performance [1]. Our research results suggested that mutual notification protocols offer very good performance in many scenarios. In these protocols, players exchange update messages directly with their neighbors without the need for supernodes or additional overlay routing.

However, we found that some existing mutual notification protocols could not provide full neighbor awareness [2]. Others had to use complex backup mechanisms to overcome this issue, or induced a high traffic overhead. In this paper we propose a new scalable architecture, that can provide near-full neighbor awareness without any of these drawbacks.

II. Related Work

In [3], E. Tanin et al. propose a *distributed quad-tree index* in peer-to-peer networks. In this approach, the underlying space is divided by a MX-CIF quad-tree. Objects in the space are associated with a region in this quad-tree, and stored at a peer that is responsible for that regions. Chord [4] is then used to find the responsible peers.

This approach can be used to store and retrieve spatial data with range queries. However, it is not well suited for interactive applications. All queries have to be routed through Chord, which may lead to intolerable delays. Also, the constant movement of players may induce a high load on the system.

COVER [5] was proposed by Morillo et al. In COVER, all players send movement updates to all their neighbors, that in turn forward the update to their "second-tier" neighbors. Players, whose AOI is not completely overlapped by other player's AOI rely on a supernode architecture for neighbor awareness: The playground is partitioned into a global quad-tree. Each region in the quad-tree contains up to a certain number of uncovered players and a supernode. All players send their movement update to their supernode, which in turn can inform uncovered neighbors about moving or new players. Regions will be split and merged dynamically dependent on the number of players inside the region.

While the authors of COVER provide an evaluation of their system in [5], they do not analyse the stability of their system in case of churn. Failing nodes may impair the neighbor awareness. Additionally, the evaluation only simulates a very short timespan. As merging of regions is not always possible, the global quad-tree can degenerate over time and lead to a highly inefficient supernode distribution. Combined with the fact that all movement updates have to be forwarded to all neighbors' neighbors, COVER can lead to a huge traffic overhead.

C. GaultierDickey et al. propose *N-tree* [6] for ordering and disseminating events in peer-to-peer games. N-trees are a generalization of quad-trees. The authors propose to partition the game space dynamically by a quad-tree. Events only relevant for one region in the quad-tree are delivered directly to all players in this region. Events whose scope exceeds the region will be forwarded over the quad-tree to neighboring regions.

With uniformly distributed players and only local events, this approach scales extremely well. However, this is not a realistic assumption. Most events, even movement, affect at least all players in the visual range. Additionally, players in DVEs tend to form crowds or groups, as interactions between players are central to DVEs. Groups of players that roam near a border of a region in the N-tree will cause a high load on the leaders of the regions and to delays in the event message propagation. Also, the frequent changing of regions that may occur in such a scenario may lead to inconsistencies in the tree.

Vast[7] (Voronoi-based adaptive scalable transfer) is an example of a mutual notification protocol. It uses Voronoi diagrams for neighbor discovery and classification. Every player builds a Voronoi diagram containing his own and his neighbors' positions. Voronoi diagrams seamlessly partition a region in non-overlapping subregions. Each region contains exactly one player. Neighbors are classified via the Voronoi diagram and the Area Of Interest (AOI), a circular region around each player. New neighbors are found with the help of the boundary neighbors, that are neighbors whose Voronoi region is intersected by a player's AOI. If a moving player *M* enters the AOI of a player *P*, a boundary neighbor of *P* will notice that event and notify *P*.

A churn and latency can lead to a partitioning of the overlay, Vast cannot provide full player awareness [2] without any additional measures. To improve the connectedness of the network, complex backup mechanisms have to be added [8].

All these protocols mentioned previously use a globally defined tesselation of the virtual world. However, due to message latency and churn players will in effect not allways have a consistent view on this tesselation. This can impede protocol performance.

To avoid this, our proposed protocol does not use a globally defined tesselation. Instead, the tesselations of all players are independent of each other.

III. Protocol Design

Distributed Virtual Environments and especially MMOGs often require soft real time constraints. In our approach, when two players need to exchange information, we want them to communicate via a direct connection. The advantage of this is that all messages exchanged between players are always transmitted in one virtual hop in the overlay network. Latency therefore only depends on the routing in the underlay network without the need of additional routing overhead on overlay level.

A. Interest management

While a fully meshed overlay network works well for a small number of nodes such a topology is not feasible for a larger number of participants. Therefore, we define the *Area of Interest* (*AOI*) of a player as a circular region around his position. A player should only be informed about events inside his AOI, as all other events are out of his scope of perception. Thus, a player connects to all other players inside his own AOI. Connecting to these neighbors, however, is not sufficient to guarantee the connectedness of the overlay network. Players sometimes need to keep connections outside their AOI to prevent a partitioning of the network. Therefore, players also remain connected to so-called *binding neighbors*, that may be located outside their AOI. Neighbor classification and management is done via a local point quad-tree[9], which all players keep independently of each other.

B. Point Quad-Tree

Using point quad-trees for neighbor classification has several advantages. Point quad-trees are easily constructed and well suited for range queries and k-nearest neighbor searches. In contrast to Voronoi diagrams, point quad-trees can be represented with a simple data structure. Construction of and searching within a quad-tree relies mostly on checking points against axis aligned boxes.

In QuON, each player constructs a quad-tree that consists of his own and all his known neighbors' positions. Figure 1 shows a quad-tree that results from a player with nine neighbors. First, the player's own position, marked by a black dot, is inserted into the quad-tree. After that, the positions of players *1* to *9* are inserted in the tree. Every time a point is inserted into the quad-tree, the region containing the point is split into four subregions at the points position. Thus, the gameworld is partitioned into non overlapping regions by the quad-tree, with each region containing one point at the most.

Point quad-trees are dependent on the order in which points are inserted into the tree. This does not matter for QuON, however, since the first split is always done through the players position. Searching is then only done in the four resulting subquadrants.

In that regard it is important to notice, that there is no globally valid quad-tree, which splits the whole game world into fixed regions. Instead, point quad-trees are only maintained locally by each player and used for range queries among known neighbors. The local quad-trees, however, do not necessarily relate to each other or the overall structure of the overlay network.

C. Neighbor Classification

In QuON each player maintains a list that contains all neighbors he knows. QuON differentiates between 3 types of neighbors:

Direct neighbors are all players within a certain player's AOI. They need to receive status updates from that certain player, in order to have a consistent view of the game world. Direct neighbor relationships are always bilateral.

Binding neighbors are needed to ensure the connectedness of the overlay network. They are used for discovering and maintaining connections to far-away groups or single neighbors.

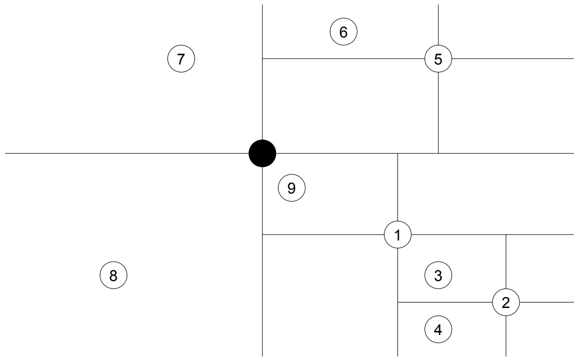


Fig. 1. Resulting point quad-tree for nine neighbors.

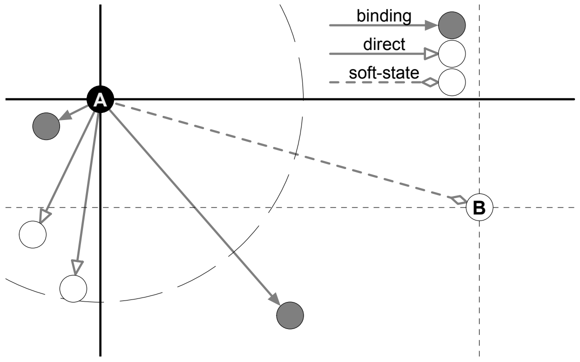


Fig. 2. QuON neighbor types.

Binding neighbors are found by performing a k-nearest neighbors search on a player's local point quad-tree. The first level of the quad-tree always splits the game world in four quadrants at the player's position. Each player tries to select one binding neighbor in each of this four quadrants in every classification process and can thus have up to four binding neighbors. If there is more than one neighbor in a quadrant, then the neighbor closest to the player is marked as a binding neighbor. Therefore, binding neighbors inside of a player's AOI are preferred. Binding neighbor relationships are not necessarily bilateral.

Soft-state neighbors are needed to make a binding relationship between two players bilateral. A player will send status updates to all soft-state neighbors. A soft-state neighbor relationship will be dropped if it is not refreshed regularly. Every time a player moves or receives a movement update from one of his neighbors, he has to rebuild his local quad-tree and reclassify his neighbors.

An example of the neighbor classification process is given in figure 2. Player *A* has three neighbors inside its AOI. One of those is a binding neighbor, the other two are direct neighbors.

Additionally, player *A* is a binding neighbor of player *B*, while *B* is not a binding or direct neighbor of *A*. Since *B* needs status updates from *A*, *B* informs *A* that he is a binding neighbor of *B*. *A* then adds *B* as a soft-state neighbor to his list of neighbors and sends status updates to *B*, as long as *B* regularly informs *A* about his binding neighbor status.

D. Neighbor Discovery

A player can discover new neighbors in two different ways. When a player receives a neighbor's movement update, he checks all his neighbors against the moving player's old and new AOI. The moving player is then informed about all neighbors which lie outside his old and inside his new AOI. The moving player then establishes a connection to his new direct neighbors.

New neighbors can also be found through a player's binding neighbors: Every time a player moves, he sends an message to each of his binding neighbors comprising of the positions of his other binding neighbors. When a player receives such a binding neighbor notification, he checks whether any of the new neighbors are closer to his position than the old binding neighbor in the respective quadrant. If this is the case, the old binding neighbor will be replaced by the new one. As this process is repeated in regular short intervals, a player will recursively find the closest binding neighbor in each quadrant.

This mechanism will guarantee that a player *P* will be informed about all players that enter his AOI: If his binding neighbor in the respective quadrant is not inside *P*'s AOI, a moving player *M* that enters *P*'s AOI must be closer to *P* than his old binding neighbor and will therefore become the new binding neighbor. If *P*'s binding neighbor in the respective quadrant is inside of *P*'s AOI and *M* is not a binding neighbor of *P*, *M* will have at least one neighbor that also has a neighbor relationship with *P* and can inform *P* about *M*'s presence.

This binding neighbor topology scales well for large numbers of players: As a player can have a maximum of four binding neighbors, he never sends more than four binding neighbor updates per update cycle to his binding neighbors. As each update message comprises a maximum of three binding neighbors, bandwidth consumption for binding neighbor exchange is independent of the total number of players within the overlay or the number of neighbors inside a player's AOI. Therefore, the bandwidth per player used for inter-group communication for any number of groups never exceeds a certain limit.

Figure 3 shows the overlay topology for 15 players, with a very small AOI, so that the structure resulting from the binding and soft-state neighbors is visible.

E. Backup Mechanisms

Assuming an ideal world, the above neighbor discovery would ensure network connectedness and full neighbor awareness. But due to message delay or failing overlay nodes inconsistencies or partitioning can occur. Therefore, QuON utilizes two backup mechanisms in order to detect and repair inconsistencies and prevent partitioning.

An AOI buffer is used to compensate for message latency and simultaneous neighbor movement. When classifying direct neighbors, a player multiplies AOI sizes by a certain factor. This factor is dependent on the maximum movement speed of the players within the game world. This prevents the loss of neighbors that are close to the edge of a players AOI.

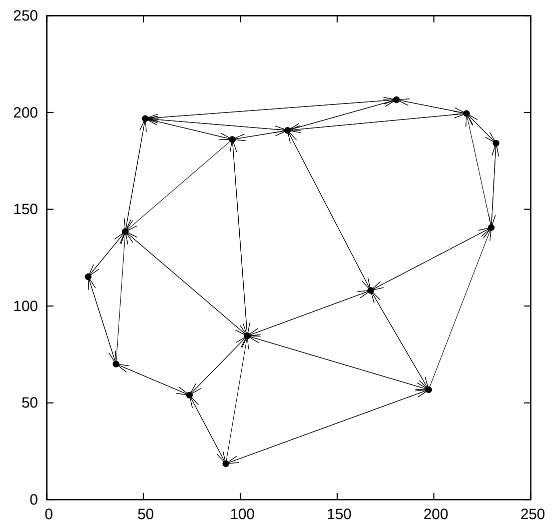


Fig. 3. QuON overlay structure for 15 players.

The second mechanism is a list of backup neighbors. If a player switches a binding neighbor and the old neighbor's position was outside his AOI, he stores the old binding neighbor in the list of backup neighbors. Only one neighbor per quadrant is stored, old entries are discarded. Thus, the list of backup neighbors never contains more then four entries.

The entries in this list are contacted in regular intervals. They answer with information about their binding neighbors. If groups of players are separated due to the failure of a binding neighbor, the information gained from contacting the entries in the list of backup neighbors can be used to regain connectivity. With a properly chosen interval to exchange backup neighbor information, partitioning of the overlay network can be effectively prevented.

F. Joining the Overlay

When a player *P* wants to join the overlay, he sends a join request to an arbitrary player that is already connected to the overlay[1]. The join request contains the player's initial position.

The accepting player compares this position with his own and the positions of all his neighbors. If he knows any player closer to *P*'s position, he will forward the join request to this player. This is repeated recursively, until the join request reaches the player closest to *P*'s position. This player responds with a join acknowledge message to *P*. The message contains a list of all his known neighbors. *P* then classifies all obtained neighbors and establishes connections to them.

Assuming an equal distribution of players in the gameworld, join overhead grows linearly with the distance beetwen the accepting player's position and the inital position of the joining player. Thus, a cache with a list of players that are located roughly in the area where players want to join respectively is useful for the joining procedure.

[1]We assume that such a player can be found by the established mechanisms for overlay bootstrapping.

G. Leaving the Overlay

When a player is about to leave the overlay, he sends a leave notification to all his neighbors. Attached to the leave notification is a list of all his known neighbors, so that the other players can update their list and reclassify their neighbors.

When a player leaves the overlay due to failure, other players detect the missing neighbor via a timeout mechanism. If the failing player was another player's binding neighbor, a replacement binding neighbor is found through the binding neighbors exchange or via the list of backup neighbors.

IV. Evaluation

To evaluate the performance of QuON, we run a series of simulations. We measured communicational overhead, network connectedness and neighbor awareness. As a reference, we repeated the same simulation with Vast. Vast was extended with the backup mechanisms proposed in [8]. For all simulations we used OverSim [10] as simulation environment.

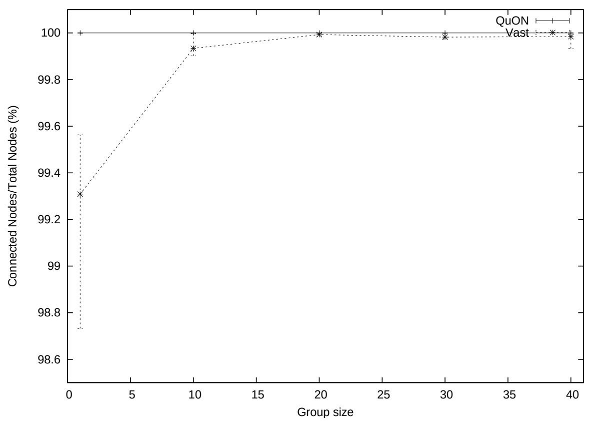
A. Simulation Parameters

The simulations were done using OverSim's *SimpleUnderlay*. In this underlay model, nodes are placed into a two-dimensional space. Message delay is calculated using the nodes' access net delay and the nodes' euclidean distance. This allows simple, yet very realistic delay calculations for Internet-like settings [11]. Node positions were chosen to fit the measurements from CAIDA's skitter project [12].

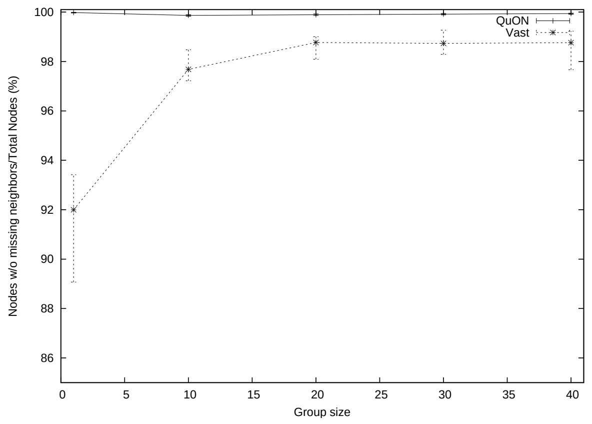
Node joins and leaves were initiated by OverSim's *ParetoChurn* churn generator. This churn generator simulates heavy tailed node session times, using a Pareto distribution for calculating a node's live and dead time [13]. Heavy tailed session times in MMOGs are supported by several measurement studies [14][15]. Our average mean session time was 100 minutes [15]. The churn generator was configured to create on average 500 live nodes. Each simulation run lasted two hours of simulated time. On average, 1400 player joins and 900 player leaves occurred per simulation.

For a realistic simulation of a MMOG we implemented a simple gaming application. This application is aware of a player's position on a virtual game field. The game field measures 1,000m*1,000m. With an average of 500 players, the player density is roughly equivalent to the most crowded zone in "World of Warcraft".

The AOI size of all players is set to 50 meters, i.e. players are aware of all neighbors inside a circular shaped area with a radius of 50m. The AOI Buffer Factor in QuON is set to 1.1. Thus, QuON will regard any player in a distance of up to 55m as a direct neighbor. Players move according to a random waypoint model. As players in MMOGs tend to play together in groups, we extended this movement model to a group based random waypoint. Here, players build groups of a configurable size. Groups agree on a common waypoint. As players in the same group stay close to each other, different group sizes result in different local player densities. This can have a big influence on a protocol's performance [1]. Thus,



(a) Connectedness



(b) Neighbor awareness

Fig. 4. Simulation Results

we repeated the simulations with group sizes varying from 1 to 40 players.

To avoid players walking in a straight line, a flocking algorithm [16] is applied to the moving players. The walking speed of all players is set to $5m/s$, which is roughly the running speed used in "World of Warcraft". As players move, the gaming application generates position updates which are then sent to the peer-to-peer protocol layer. This is done six times per second, which is a realistic average movement update frequency for MMOGs [15].

B. Results

Three metrics are important when judging the performance of a P2P protocol for Virtual Worlds and MMOGs:

- **Network connectedness and neighbor awareness:** In a Virtual World or MMOG it is vital that all players are aware of all their neighbors. This can only be achieved if the network stays fully connected.
- **Bandwidth:** In a P2P protocol all events are disseminated directly by the peers. Thus, peers are required to send a considerable amount of messages. However, the resulting bandwidth requirements must not exceed a player's resources.
- **Latency:** Many interactions in Virtual worlds and especially MMOGs are sensitive to message propagation delay. Studies showed that many players will not tolerate latencies above 500ms [17]. Many players will notice even smaller latencies [18].

1) Network connectedness and neighbor awareness: Network connectedness and neighbor awareness are the main criteria for a P2P protocol for MMOGs: If players are dropped off the network, or if they frequently miss some of their neighbors, they will stop using the protocol.

Figure 4a shows the percentage of players that are connected to at least one other player in the overlay network in relation to the group size. The graph for QuON shows a straight line at 100%: No player ever is disconnected from the QuON network, regardless of the group size.

The graph for Vast shows a negative correlation between group size and the fraction of connected players: With a group size of 1, roughly 99.3% of all players are connected to the network, i.e. about 0.7% have been dropped. At a group size of 10, more than 99.9% stay connected. With larger group sizes, this number fluctuates around 99.98%.

This correlation can be explained by Vast's backup mechanism that is responsible for avoiding lost players in case of churn: Vast identifies nodes that have less neighbors than the average as „critical". Those „critical nodes" will distribute backups of their list of neighbors to all their known neighbors. In case of a node failure, this redundancy can be used to repair the network. However, in case of almost equally distributed players, no nodes will deem themselves critical, and thus in effect disable the backup mechanism.

Figure 4b shows the neighbor awareness measured by the fraction of nodes without any missing neighbors. In QuON, neighbor awareness is almost unaffected by group size. It fluctuates around 99.9%, with a minimum of 99.86% and a maximum of 99.98%.

As expected, Vast's neighbor awareness graph is shaped similar to its network connectedness graph: Vast's neighbor awareness is influenced by the group size in a similar way. The neighbor awareness starts at roughly 92% for a group size of 1, and reaches a maximum of 98.8% for a group size of 20.

2) Bandwidth: As players usually have limited bandwidth capacities, an overlay protocol should minimize its traffic overhead. Figure 5 shows the maximum and average bandwidth QuON and Vast utilize in relation to the group size. The graph shows that traffic increases as group sizes grow. With larger group sizes, a player has on average a higher number of players inside its AOI and is such required to send his update messages to a higher number of other players. This causes higher traffic.

With a group size of 1, QuON used 8.2 kBytes/s on average. The bandwidth demand grew with the group size, reaching an average of 18.6 kBytes/s at a group size of 40. The maximum bandwidth demand observed in the simulation runs was 14.6

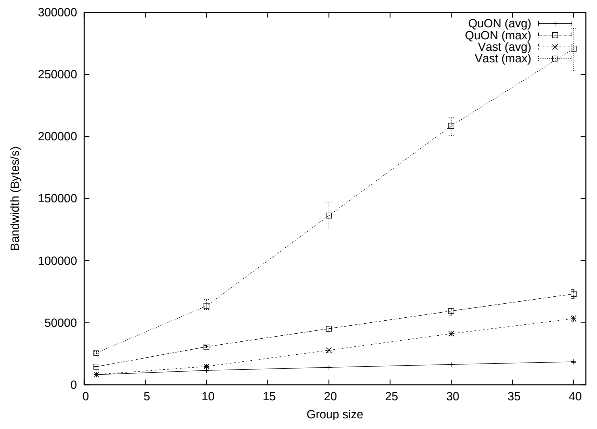


Fig. 5. Simulation Results – Bandwidth

kBytes/s for a group size of 1, growing to 73.2 kBytes/s for a group size of 40.

For a group size of 1, Vast's bandwidth demands were only slightly larger than QuON's: On average it needed 8.4 kBytes/s. However, as group sizes increased, Vast started to cause much higher traffic: With a group size of 40, Vast needed 53.3 kBytes/s on average. When looking at maximum traffic figures, the differences between Vast and QuON show even more clearly: Vast needed 25.8 kBytes/s for a group size of 1, and 270.7 kBytes/s for a group size of 40. This is likely to exceed the resources of most users.

3) Latency: As in QuON and Vast—as mentioned in sections II and III—all event messages are exchanged directly between the neighbors, no message will travel more than one hop. Therefore, the delay stayed nearly constant throughout all simulation scenarios for both protocols. In our setting, the resulting delay was around 90ms.

V. Conclusion

In this paper we have proposed a new overlay protocol for Virtual Worlds and MMOGs, denoted as QuON. The advantage of QuON as a mutual notification protocol is that all event notifications are exchanged directly between neighbors, thus minimizing the latency. No server or additional infrastructure is needed for the protocol.

The key feature of QuON is the intelligent selection of a fixed number of binding neighbors, which is done with the help of quad-trees. This design limits the traffic overhead while providing full network connectedness.

Evaluation results show that QuON is able to provide full network connectedness and almost full neighbor awareness in a simulation comprising of 500 players, regardless of group sizes. The required bandwidth did not exceed reasonable boundaries. As a result, we conclude that QuON provides a scalable peer-to-peer architecture for Virtual Worlds and MMOGs with a practical performance. As future work we are evaluating the gains of dynamic AOI sizes for QuON to avoid overloading in case of densely populated spots.

References

[1] S. Krause, "A Case for Mutual Notification: A Survey of P2P Protocols for Massively Multiplayer Online Games," in *Proceedings of NetGames 2008 Network and Systems Support for Games*, Worcester, Massachusetts, USA, Oct. 2008, pp. 28–33.

[2] H. Backhaus and S. Krause, "Voronoi-based adaptive scalable transfer revisited: gain and loss of a voronoi-based peer-to-peer approach for mmog," in *NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, 2007, pp. 49–54.

[3] C. GauthierDickey, V. Lo, and D. Zappala, "Using n-trees for scalable event ordering in peer-to-peer games," in *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, 2005, pp. 87–92.

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, 2001, pp. 149–160.

[5] P. Morillo, W. Moncho, J. M. Ordua, and J. Duato, "Providing full awareness to distributed virtual environments based on peer-to-peer architectures," in *Advances in Computer Graphics*, ser. Lecture Notes in Computer Science, vol. 4035. Springer Berlin / Heidelberg, 2006, pp. 336–347.

[6] E. Tanin, A. Harwood, and H. Samet, "Using a distributed quadtree index in peer-to-peer networks," *The VLDB Journal*, vol. 16, no. 2, pp. 165–178, 2007.

[7] S.-Y. Hu and G.-M. Liao, "Scalable peer-to-peer networked virtual environment," in *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, 2004, pp. 129–133.

[8] J.-R. Jiang, J.-S. Chiou, and S.-Y. Hu, "Enhancing neighborship consistency for peer-to-peer distributed virtual environments," in *ICDCSW '07: Proceedings of the 27th International Conference on Distributed Computing Systems Workshops*. Washington, DC, USA: IEEE Computer Society, 2007, p. 71.

[9] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, March 1974.

[10] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, May 2007, pp. 79–84.

[11] T. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 170–179 vol.1, 2002.

[12] A. Ma, "Caida : tools : measurement : skitter," http://www.caida.org/tools/measurement/skitter/, June 2008, accessed on 2008-06-08.

[13] Z. Yao, D. Leonard, X. Wang, and D. Loguinov, "Modeling heterogeneous user churn and local resilience of unstructured p2p networks," in *ICNP '06: Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 32–41.

[14] D. Pittman and C. GauthierDickey, "A measurement study of virtual populations in massively multiplayer online games," in *NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, 2007, pp. 25–30.

[15] K.-T. Chen, P. Huang, and C.-L. Lei, "Game traffic analysis: an mmorpg perspective," *Comput. Netw.*, vol. 50, no. 16, pp. 3002–3023, 2006.

[16] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[17] M. Claypool and K. Claypool, "Latency and player actions in online games," *Commun. ACM*, vol. 49, no. 11, pp. 40–45, 2006.

[18] J. Smed, T. Kaukoranta, and H. Hakonen, "Aspects of networking in multiplayer computer games," in *Proceedings of International Conference on Application and Development of Computer Games in the 21st Century*, L. W. Sing, W. H. Man, and W. Wai, Eds., Hong Kong SAR, China, Nov. 2001, pp. 74–81.