Scalable Localized Histogram Aggregation for P2P MMOGs

Master Thesis

Patrice Müller
patrice@student.ethz.ch

Prof. Roger Wattenhofer
Distributed Computing Group
ETH Zurich
Switzerland
wattenhofer@tik.ee.ethz.ch

Prof. Junehwa Song
Network Computing Laboratory
Division of Computer Science
KAIST
Republic of Korea
junesong@cs.kaist.ac.kr

July 8, 2005

Abstract

Overview maps are a highly demanded feature in Massively Multiplayer Online Games. Current server-based techniques will be replaced by distributed networks due to scalability and investment costs. In a P2P MMOG it is more complex to calculate a local histogram map for every player in the virtual environment than with a centralized approach. In this work a process is proposed which aggregates local information to a 2-dimensional histogram for all players. The aggregation performs on a dynamic Delaunay triangulation network with compass routing in a clustered virtual space. It is shown that the aggregation is consistent and the Delaunay triangulation can be maintained with the proposed dislocation limitation algorithm. Furthermore, it was mathematically proved that the localized histogram aggregation process is asymptotically average optimal in message complexity.

Contents

List of Figures

1 Introduction

At the early ages of computers they were mainly used for academic purposes to perform scientific calculations. As those universal calculators became fast, smaller and more affordable people started to use them more widely. Nowadays almost any business fully depends on PCs. Besides using personal computers only at work people also started to use them for their own enjoyment. Computer games developed at an early stage and improved their quality of animation and realism tremendously.

Computers started to be interconnected through the World Wide Web where they could exchange messages mutually. Multiplayer games emerged where more than one player interacts in the same virtual environment. Two or more players can interact with each other by exchanging useful information during the course of action among the programs connected. Here our notion of how these games are connected with each other is rather ambiguous. There are many different approaches on how these players are connected. In any case a certain predefined topology is necessary such that two or more players can communicate with each other efficiently. As we are dealing with games where online interaction is very important the efficiency aspect of communication is crucial. Once a user interaction is taken by one player which affects other players, those players need to be aware of this fact as soon as possible. Depending on the gender of game the meaning of *as soon as possible* can vary but to avoid out-of-date information the sooner is always the better.

In the following parts of this introduction we show the background of Massively Multiplayer Online Games *MMOGs* and the Peer to Peer *P2P* topology approach. Then we formulate our idea of aggregating information of players in a local area where the players are connected in a P2P topology. We also mention the idea of clustering the virtual space wherein the players move around and how aggregations can be reused among nodes. These ideas will then be further developed and focused on in the main part of this report. As a last part of this introduction several examples of possible future applications in P2P MMOGs are presented to give an idea about what could be done with the results of this research.

1.1 Background

Before we can start to describe our idea and solution approach we need to have a general understanding of massively multiplayer online games as well as the peer to peer characteristics. First we should understand in more detail what MMOGs are. In the following subsections a definition of MMOG is given. The current server-based technique with its limitations is shown and we do a simple cost analysis. Then a selected list of popular MMOGs with their number of subscribed users is presented.

In the second subsection we look at the peer to peer characteristics. We discuss the scalability of P2P systems and why this approach can be very useful. The required topology construction and maintenance is indicated and we would also like to put a viewpoint on the investment and maintenance costs for P2P systems versus server-based systems. At the end some widely used applications using a P2P approach are listed.

1.1.1 Massively multiplayer online games

1.1.1.1 Definition Until now we have used the term MMOG referring to Massively Multiplayer Online Games but no definition about what it is was given. The following quotation is a definition by the internet community.

> A massively multiplayer online game (MMOG) is a type of computer game that enables hundreds or thousands of players to simultaneously interact in a game world they are connected to via the Internet. Typically this kind of game is played in an online, multiplayer-only persistent world.[1]

There are three main points in this definition giving a detailed description

- *Hundreds or thousands of players:* Very many players need to be involved in the game concurrently. *Meridian 59*, which is considered to be the Neanderthal game of MMOGs, launched in 1996 and registered a maximum of 3000 players distributed on 12 servers.[2]
- *Over the internet:* To interact with the game environment all the communication necessary needs to be transmitted over the internet. In that sense every player has the same possibilities to connect to a dedicated server(s) or directly to other players.
- *Persistent world:* It is the virtual fantasy world used for online role-playing games. All events therein happen persistently. Even when some of the players are not online the world continues to live and changes can occur. When a player is online then it can influence and change the persistent world.

1.1.1.2 Current technique To our knowledge all current running implementations of MMOGs use a server-based approach. In such an environment all players act within one persistent world which is managed by a central server. Actions taken by players are sent by internet to the central server. There all events are gathered and the fantasy world is updated according to the games' objective. Afterwards the outcome is returned to all individual players where the local updates can be performed. At any time very numerous players may be involved in the game. This is exactly the idea of MMOGs but can cause a problem on the server side. In practice many central servers are interconnected in a cluster to provide the required computation speed and bandwidth for communication.

1.1.1.3 Limitations To have an idea about how many clients can be handled by one server we reference the keynote speech of Bill Gates at WinHEC 2005. During the presentation of the 64-bit software architecture he mentioned that today's MSN messenger[3] servers are capable of accepting 60'000 connections each. Instant messaging only forwards IP packages from the sender to the receiver. MMOGs have much higher requirements on their servers as events

[1] Definition of MMOG from *http://en.wikipedia.org/wiki/MMOG*.
[2] Historical information from *http://archive.gamespy.com/amdmmog*
[3] An instant messaging program of Microsoft Corporation.

are processed on the central servers and in general communication frequency is much higher than for instant messaging. It is easy to see that when a very high number of players want to play together that a big and more importantly an extremely expensive cluster of servers is required.

1.1.1.4 Cost analysis A company projecting a new MMOG is faced with high development costs for programming and high investments for the central cluster. Even after the launch of the developed game maintenance costs apply. This causes a cerain market entrance barrier to financially weak companies. To cover the initial investments and the maintenance costs companies most likely want their players to subscribe to a service fee. We will see in the peer to peer approach that these problems are much smaller or even disappear.

1.1.1.5 Popular MMOGs and its number of users As we already mentioned *Meridian 59* can be considered as the first MMOG. Thereafter various games were realized and caught on wide public especially in Korea, Taiwan and Japan. Some of the top names in MMOGs include *EverQuest, Lineage, Lineage II, Ultima Online, World of Warcraft.* Lineage reached a total number of subscribers over 1 million already at the beginning of 2000 and its climax was above 3 million subscribers in 2003. Nowadays more and more MMOGs push into the market but in May 2005 Lineage and Lineage II together still came up with almost 50% of market share of subscribers to MMOGs. For further information we would like to reference to [1, 2].

1.1.2 Peer to peer

1.1.2.1 No central server Whenever the term P2P is mentioned it refers to a virtual network connecting computers without the use of a central server. In this model every computer or also called node is both a client as well as a server. The nodes maintain their virtual network themselves. Links are updated as required by joining, leaving or possibly for moving nodes. As no central server is required the network can be scalable to the number of users. With more nodes joining the network the total computation power also increases linearly. In addition a P2P network does not suffer from a bandwidth bottleneck as every node is responsible for its own connection and practical P2P networks will only cause a small overhead to maintain the topology.

1.1.2.2 Topology control With the absence of a central server messages cannot be sent directly from the source to the target in general. Otherwise every node would require global view which is already unfeasible with a fairly small number of nodes. A logical topology among the nodes must be maintained which forms the virtual network. Over this network all nodes are interconnected by multiple hops. A suitable routing algorithm ensures a correct path from the source to the target for exchange messages. At any time individual nodes may leave the network or get disconnected. Therefore, a P2P topology must be able to perform correctly even with failing nodes and be able to recover up to a certain degree of failing nodes.

1.1.2.3 Cost analysis A peer to peer network needs to be decentralized and self-managed. This means that there are no or very minimal investment costs necessary for maintaining such a network. People using the network basically provide their own computation power and bandwidth. For P2P there are no such market entrance barriers as exist for current MMOGs. It is also possible for capitally weak companies or groups to provide some application as long as they come up with the implementation.

1.1.2.4 Popular application P2P protocols have been around since the beginning of the internet. The BGP protocol for IP address routing let the internet community grow so seamlessly as it is decentralized, scalable and fault tolerant. However, P2P applications for public use started many years later as music file sharing became popular. Until now file sharing has remained the most popular application of the P2P technology but much more can be realized in a distributed manner. Within the last few months the internet telephone application Skype [4] started to attract a very wide public. By using a proprietary P2P network it achieves free internet calls with very good sound quality. Until May 2005 it reached almost 3 million[4] concurrent online users.

1.2 Idea

The provided background gives a good overview of MMOGs and P2P networks. Today's server-based technique for online games is not scalable to the number of users and involves a market entrance barrier to low capital companies. These existing problems could be solved by taking MMOGs one step further in their development and implementing them on top of a peer to peer environment. We will further describe this idea in the first subsection. In the second subsection we describe the main idea of this work about localized histogram aggregations in a distributed virtual space.

1.2.1 MMOGs using P2P technology

We believe that the only way to keep up with the fast growing number of online gamers and the increasing bandwidth requirements due to more complex games can only be achieved by using a peer to peer approach in the future. Every game needs some sort of virtual space where individual players can move around. Since computers will be connected in a P2P network this virtual space is distributed and every player is located at one position therein. Players shall be able to move around as the game goes on and they shall also be able to communicate with their local neighborhood in the virtual space. For most MMOGs it will be sufficient to have a local view as the players can act where they are located. After moving to another location they will be capable of acting in that environment with other players as their neighbors. In subsection 4.1.6 we list the properties of a Delaunay triangulation and in section 7 we present an algorithm to maintain the triangulation among moving nodes. This will be the basis for our 2-dimensional virtual environment overlay network.

[4] Number registered from within a running Skype application.

1.2.2 Localized histogram aggregation in distributed virtual space

We have mentioned that players in a MMOG can interact with their neighborhood however there is one particular functionality that is required by almost any multi-user game. Players want to collect information from other players. Since those players interact with each other online they also want to know more details about other players. However in a MMOG there can be a huge amount of players located in a small area of interest. In this case detailed information about every individual player can result in an information overflow easily. Humans cannot interpret too much information at a time and during a game the players just want to know and see relevant information.

To overcome the problem of information overflow the general technique of aggregation can be applied. For a detailed description about aggregates and their properties we would like to refer to section 3. Now we further explain the idea of aggregates in the virtual space of MMOGs. In our model every player of the game is located at a particular position in the 2-dimensional virtual environment. Once a player would like to know more details about its local surrounding it would be convenient to have some localized aggregation mechanism at hand which can be applied to this distributed 2-dimensional virtual world. A player may be interested in a simple aggregation value for its local surrounding but it may also be interested in a more detailed analysis of its surrounding. To be able to receive a detailed overview we would like to focus our work on a localized histogram aggregation. With histogram data a player knows the aggregation values in different areas of its surrounding and thereby has a better notion as by a simple aggregation with one value.

In this paper we show how a localized histogram aggregation can be made scalable by clustering the virtual space. We will show in detail how the values in the clusters are aggregated in a static (section 5) as well as a dynamic (section 6) environment. Furthermore we will give a message complexity analysis of a collective aggregation where all the nodes in the whole virtual environment want to have the same aggregated information about their individual surrounding area.

1.3 Applications

The mentioned idea is a rather general case. In this section we would like to show three concrete applications to our aggregation idea. All three applications are of different genre which shows how widely our localized technique can be used. We can also see that different taxonomies of aggregates can be calculated as long as the desired function allows in network[5] aggregation [11].

1.3.1 Neighborhood map

In a MMOG or more precisely in a MMORPG[6] where players act individually in the virtual world they may require to see where other players are located in their surrounding. In a main screen the players will see their proximity where close-by neighbors are displayed one by one in the exact positions. Besides the

[5] In network aggregation is the technique developed for MANETs to reduce energy requirements and to make it scalable.

[6] Massively Multiplayer Online Role-Playing Game.

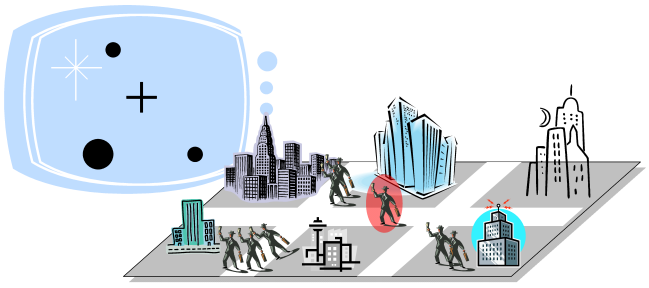


Figure 1: Neighborhood map of local virtual space

main screen a smaller window could show a much bigger surrounding where accumulations of players are displayed.

In Figure 1 we see an example of such an overview window on the upper left side. It gives the player a quick idea of where other players are located relatively to its current position. In the 2-dimensional virtual space the players are shown as business men in a city. The local player is the highlighted one located in the center of the city. Its view may be very limited due to obscuring buildings. With the overview window the game application can show each user where other players are located. As this additional information is to help the player it must be limited to the amount of information the player is interested in. Subareas of the overview window are aggregated separately and are shown as independent units. Our research and development was exactly focused on achieving this task. Once we have a localized histogram aggregation mechanism for MMOGs in a distributed virtual environment then we can easily construct a neighborhood map as described in this specific application. The aggregation function will simply add up the number of players within each area or cluster as we will introduce it in section 5.1.1. Once the aggregation values from all clusters are received then the application can display the neighborhood map with highly and sparse populated areas as in the example given.

1.3.2 Exceptional neighbors

For many game strategies the most dangerous or the best players are of great interest. It may be necessary for a player to flee from the most dangerous gamers or to attack the best players in hope of increasing one's ranking. Whenever a player wants to find exceptional players in its vicinity, all players in the area contribute to the final result by deciding whether they are among the exceptional players or not. To make this searching process scalable we need a mechanism other than centralized selection.

Finding the best player in the local surrounding is equivalent to a *max* aggregation for finding the highest score among the local nodes. Our localized histogram aggregation function finds the best players in local subareas. Thereafter a histogram map can be built by combining the partial result. At this point we are not interested in histogram data but in an overall (still local) aggregation value. As we will see in section 3.1 about taxonomy of aggregates, the *max* function belongs to the distributive aggregation functions. This means that the overall maximal value can be deduced easily from the partial maximal

values. Consequently, our localized histogram aggregation mechanism can also find information efficiently about exceptional neighbors in the virtual space of MMOGs.

1.3.3 Ranking system

A main goal for gamers to continue playing a MMOG is to increase their skills and top other players with their talent. However in a P2P approach where players only communicate with close-by neighbors it is more difficult to measure one's abilities in comparison with other players. For MMOGs that use a central server it is fairly easy to calculate and maintain a ranking among all players. To achieve a similar global ranking of the top most players in the P2P virtual environment a global $dmax$[7] aggregation would be required. Several techniques are already existent to solve such an aggregation as can be seen in the related work section 2.

We believe that it is not always necessary to have a complete global view to construct some ranking system and motivate players to improve their cleverness. If there are numerous players in the local aggregation area a statistical ranking system can be implemented. By aggregating simultaneously the average of x ($\overline{x}$) and the average of x^2 ($\overline{x^2}$) as described in the subsection 3.1.2 about algebraic aggregates, we get the variance as $\sigma^2 = \sqrt{\overline{x^2} - \overline{x}^2}$. Using the variance σ^2 and the average $\overline{x}$ a normal distribution is constructed. The sum of all x and the private value of x let us calculate the estimated rank in either absolute or relative figures. Our localized histogram aggregation technique can therefore also be used to realize an estimated but efficient ranking system without global view.

2 Related work

There exists already a good selection of work dealing with aggregation problems and P2P overlay networks. We are only interested in distributed aggregations for our P2P MMOG approach. In this section we give an overview over some techniques that can be applied in distributed envrionments. Besides existing work about aggregation algorithms an overview about overlay networks is given. For our localized histogram aggregation in a distributed virtual space the existing work about aggregations and about overlay networks is of interest. However, there is no related work specifically dealing with localized histogram aggregations.

2.1 Aggregation algorithms

Several techniques can be applied to aggregate information among distributed nodes. The techniques that can be applied depend strongly on the topology the nodes are located in. Wireless ad-hoc networks are not capable of building up the same topologies as can be done in a fully connected environment. We are not going to focus our work on aggregations in MANETs but the existing work can give us a good idea about what is important for efficient and scalable aggregations.

[7] The d highest values are aggregated.

A technique called *in-network* aggregation is the key to make the aggregation process scalable. Instead of collecting all the values of the nodes and calculating the aggregation value at once (which is equivalent to a centralized approach) many intermediate steps are taken. This mechanism is possible for distributive and algebraic aggregation functions as they don't require a global view. This in-network technique can be implemented in various ways. In the following subsections we list two very general techniques (tree based and epidemic) and two more specific implementations (hierarchical gossip and hypercube) for aggregating information.

2.1.1 Tree based

Many papers [6, 11, 12, 13, 23] are published that use some sort of tree to aggregate information either in a wireless ad-hoc environment or on a predefined overlay network. To simplify and abstract the aggregation process in wireless ad-hoc networks Madden et al. proposed a technique to retrieve aggregates with SQL queries. In [11, 12] they showed how in-network aggregation can be conducted in sensor networks. In the first phase the query is injected into the network by pushing it down any routing tree. In the second phase the values are aggregated by propagating the partial aggregation values from children to their parents. While pushing the query into the network TAG requires parents to include a maximum waiting time until they want to receive the results from their children. It was shown that this technique is efficient and results in an order of magnitude bandwidth reduction over centrally aggregating solutions. The weak point of this work is that it was not shown what happens with moving sensors. When the routing tree changes during the aggregation process, then the result may become meaningless without notice.

Building an aggregation tree on top of a DHT has been investigated by Ji Li et al. [6, 13]. The key point of their idea is to use a predefined parent function. Every node in the DHT can locate its parent by using that function applied to the node's location in the DHT. If the predefined function is well suited then it can assure to build a spanning tree of finite height. The tree is constructed using a *bottom-up* approach with the parent function and it is maintained using *soft-state*. Using soft-state means that the parent function chooses a parent without any previous information (such as its previous parent or children). A moving node[8] would persistently connect to the node which fulfills the parent function of the current location.

2.1.2 Epidemic dissemination

A very different approach to aggregations than what we might think of at the beginning is the use of an epidemic dissemination. Epidemic studies have a long history and date back to the nineteenth century. In [8] Eugster et al. showed in detail how information can be distributed exponentially in a P2P environment on the Internet or in ad-hoc networks. One node starts the *infection* and sends the information to a random subset of neighbors. Every node receiving a message forwards it to a random subset of its neighbors. At each node $O(log(n))$ neighbors are randomly chosen. After $O(log(n))$ rounds all nodes of the system

[8] DHT was not constructed for moving nodes but the players in our MMOG virtual environment are moving around.

are infected (did receive the information) with high probability. This epidemic spreading is robust and highly resilient to failing nodes. Several problems remain open or uncertain and are areas for future research.

Montresor et al. applied the epidemic dissemination technique to overlay networks for the purpose of aggregating global information. Every node repeatedly exchanges its current local value with its neighbors, updates its local value as required by the aggregation function with the value received from the neighbor and continues with other neighbors. This process of exchanging values with local neighbors is called *gossiping*. The basic idea of this process is to keep the aggregation value among all nodes constant but to decrease the variance of the values. In [7] it was shown that the variance converges exponentially. When the gossiping stops the estimated aggregation result is either the local value or a mathematical result thereof. The desired aggregation result is estimated by the local value or a mathematical function of it after the gossiping process stops. Every node in the overlay knows the aggregation result right away as every node has done the same converging gossip algorithm. This aggregation process is also very fault-tolerant and efficient for random topologies. It is however unclear how good the algorithm behaves on well structured overlay topologies.

2.1.3 Hierarchical gossiping aggregation

Global epidemic dissemination is quite fault-tolerant to failing nodes but can be time consuming to ensure a good completeness. It would be nice to have a fault-tolerant aggregation mechanism which includes the values of all nodes in the final aggregation result with a high probability. In [9] a hierarchical aggregation technique is proposed. All nodes are distributed in grid boxes of predefined constant size. The algorithm starts by aggregating these grid boxes independently using the gossiping technique where knowledge about known nodes is disseminated in the limited space of the individual boxes. When this phase has completed phase two starts by repeatedly combining grid boxes hierarchically in rounds. For this task also a randomly choosing gossiping is performed to avoid wrong results due to failing nodes. After every round all the nodes evaluate the partial aggregation value before they move to the next round for combining with other grid boxes. The algorithm terminates when all grid boxes have been combined together.

It could be shown that applying this algorithm guarantees a good completeness probability. This assures an accurate value as the global aggregation result. In addition to accuracy it is also poly-logarithmically sub-optimal in time and message complexity. The drawback of this method is that all members should know about each other at the beginning. This requirement can be relaxed as mentioned in the paper. However, more than completely local neighbor information in a virtual space is required as gossiping among independent grid boxes (for hierarchically combining) could only start very slowly otherwise.

2.1.4 Hypercube

In network aggregations can be calculated easily by using a hypercube topology. All nodes of the P2P network are located at an edge of a d-dimensional hypercube. Acting in rounds, every node increasingly moves through all dimensions of the hypercube and sends the aggregation value of the sub-cube up to

the current dimension to the neighbor in that dimension. This process allows a pipelined aggregation in $\Theta(\log n)$ rounds. In [10] it was shown how such a hypercube topology can be constructed and it was show that it is also resilient to dynamic adversarial churn. Placing and maintaining $\Theta(\log n)$ peers at every edge of the hypercube makes it very worst-case fault-tolerant. The dimension is dynamically adapted and nodes are moved around among neighbors to keep $\Theta(\log n)$ peers at every edge. The drawback of this approach for P2P MMOGs is that nodes cannot choose *where* they get connected to the network whereas in MMOGs players want to be locally connected to other players in their virtual surrounding.

2.2 Overlay networks

The domain of overlay networks is rather broad. The basic idea is that a virtual communication structure is logically laid over (or established on top of) a physical network. This virtual structure may allow specific functionalities which are not directly provided by the underlaying network. Distributed data storage and lookup, distributed computation, fault-tolerant routing and multicast are examples of such functions. With its enhanced capabilities it is creating a more 'intelligent' network as the physical network is on its own.

With the evolution of overlay networks many different topologies, routing mechanisms and applications have been proposed. In this section we give three different concepts of overlay networks in historical order of evolution. It is to be noted that for various applications different requirements exists. Therefore, older concepts are not necessarily worse for all applications and may be well suited for their particular purpose.

2.2.1 Supernodes

A first step to distribute calculations and bandwidth bottlenecks was achieved by a two-level P2P network. Instead of having one central server or a cluster of servers, the server functionalities are distributed among selected nodes of the network. The popular peer-to-peer file sharing program Kazaa by Shareman Networks [3] implements this technique.

Nodes with high available capacities for calculation and bandwidth are selected to become Supernodes within the network. Regular or client nodes connect to a couple of Supernodes which are their local servers. In the case of the file sharing application the Supernodes store a list of content available at their local clients. This allows clients to search for a file by asking their Supernodes in the same way as clients ask a server in the client-server based model. However those Supernodes only know their locally connected clients. As every client wants to be able to search for files globally the Supernodes need to interact with each other. Once it receives a search from a client it first searches within its local content list and then forwards the search to connected Supernodes. This process can be seen as the first results arrive very quickly and then more and more results arrive which must be from other Supernodes by traveling multiple hops.

The Kazaa protocol was introduced in March 2001 and still remains a very popular application with 3 million concurrent online users in average[9]. The Su-

[9] Historical and current information about Kazaa from *http://en.wikipedia.org/wiki/Kazaa.*

pernode approach seems to work rather well for file sharing which requires only a distributed data lookup service. Little bandwidth and calculation is required for searching which allows many peers to connect to one Supernode. In addition location awareness is unimportant as nodes are not restricted where they get connected to the network. For applications where frequent and complex calculations need to be performed only a few peers could connect to every Supernode. This leads to the idea of having a one-level P2P network where every node has equal rights and duties.

2.2.2 Distributed Hash Tables

All peers are represented by a hash number of predefined size as all nodes within the system should be clients and servers without any functional differences among the nodes. Most research about DHTs has focused on the use of a one-dimensional space shaped in a circle for continuity. A multi-dimensional DHT (CAN) has also been proposed but it has a fragmentation problem when several adjacent nodes leave the network. The main idea behind a hashed virtual space is that everything is hashed to a certain location therein and every peer (which is also hashed into the DHT) is responsible for a certain range. Most likely the range is defined as the space from one node's hash value to the next node's value in the DHT.

The only thing that is supported by such a distributed hash table is a simple *lookup(key)* function which returns the network location of the peer currently responsible for that key. In [5] two different techniques to find the network address to a given key are explained. These techniques were developed for the concepts of several DHTs. It was shown that these techniques fulfill interesting properties such as global searching in $O(\log n)$, robustness and scalability. As 'good' hash functions randomly distribute original objects uniformly into the DHT, there is no correlation between the content of the object and the hashed location. Current DHTs don't cover the properties required for location-dependent applications where the proximity of objects is important. Neighboring objects in the real world may not be able to find each other in the hash table anymore.

2.2.3 Virtual environment overlays

The latest research areas in overlay networks have been semantic overlays. For these networks objects are specifically mapped to a virtual world to suit the needs of a particular application. Many applications require a local awareness within the virtual space or in other words a virtual environment that allows location based services. Such applications include MMOGs but also P2P auctions and other local services. Many papers [14, 16, 17, 18, 20, 21] have been published that deal with virtual environment overlays. Some specify it very detailed by focusing on MMOGs and others mention it just briefly.

An important approach to location based services was done with a Geographical Hash Table GHT [14] in sensornets. Objects are hashed into the two-dimensional world of sensors. Hashes are unlikely to fall into a sensor's location however there is always one unique sensor responsible for any location. The node closest to the hash value is responsible and that node can be found by applying the proposed GPSR algorithm. The idea is similar to the Supernode concept by dividing the geographical plane into areas with one sensor.

The papers [20, 17] focused on P2P architectures for MMOGs. The former is designed on top of a Pastry DHT. This assures fast lookups and global routings among nodes in the game but is fully location unaware. On top of Pastry a Scribe infrastructure is proposed which allows application level multicasts in membership groups. The later paper divides the virtual space into zones with one zone owner each. This owner has similar functionalities as a central server but it is a regular node in the game which resembles again the Supernode concept. It was shown that this architecture is favorable up to 500 players (which we do not consider to be a true MMOG due to its small number of users). Both techniques group players in memberships or zones and therefore are not fully location aware due to cutting the virtual space into areas.

The well known mathematical structures Delaunay triangulation (section 4.1.6) and Voronoi diagram[10] are used in [18] and [16]. The first proposes *GeoPeer: A location-aware peer-to-peer system* uses a Delaunay triangulation as its overlay network. The system is not dynamic as nodes cannot move around in the overlay but geographically scoped queries are supported. There is exactly one node responsible for any location in the geographical world and long range contacts are discussed for efficient global connections also in unbalanced areas. In the second paper a Voronoi-based P2P networked virtual environment is constructed and maintained. Every node connects to all nodes whose Voronoi area intersects its variable-sized area of interest. This approach allows nodes to move around but it was not specified how much they can move in order to keep a consistent Voronoi diagram. For a big area of interest very many connections may be required and it is not clear how large areas could be aggregated efficiently.

SOLIPSIS is another approach for creating a virtual world as an overlay for a large number of users. Every node connects to a number of local nodes. The algorithm assures local awareness and global connectivity. The nodes can move around freely and even teleportation is described among an unlimited number of entities. The radius of area of interest is variable and can be adapted to the local density such that every node's physical capacity is sufficient for maintaining all connections. As mentioned in [16] SOLIPSIS has a discovery problem of approaching nodes and therefore has a problem for guaranteeing consistency. Besides this problem it is not evident how local (in a larger region than the awareness area) information could be aggregated efficiently.

3 Aggregation

The process of aggregation refers to the general process of collecting data from different sources but of the same gender and combining those data samples into a single representation. This technique is already very well known from database systems and started to pull attention for MANETs in recent years [9, 11, 12]. Especially in sensor networks the measured value of a single sensor may not be very meaningful due to local variances. In many cases an aggregated value of several sensors is more significant as the calculated value represents the true value with a higher probability. In addition to a higher precision the aggregation also simplifies the data available and makes further use of it simpler. Particularly humans would be overwhelmed very easily with too much data. For

[10] The Voronoi diagram is the *dual* to the Delaunay triangulation. It partitions the space into areas of closeness to every Delaunay point.

them a rough overview of analyzing aggregates fulfills all their needs in most cases.

In this section we would like to show what differences exist between diverse aggregates. Collecting all data values first at a central place and then calculating the aggregation value is trivial. Once we start to optimize by decentralizing the whole process several interesting topic arise. First we will see that there exists a taxonomy classifying aggregates by equal behavior about how data can be collected. In the second part differences in the area of interest are presented.

3.1 Taxonomy of aggregates

In practical cases mathematical functions that can be calculated with computers are used to receive a simplifying overview over data values. Due to the nature of mathematics these functions can be divided unambiguously into 5 classes. Some of them favor in network aggregations and some require more data load towards the aggregating node.

3.1.1 Distributive

The class of distributive aggregates is the most favorable for in network aggregation. It allows partial data to be aggregated and combined with other partial aggregates or new data values. There is no limitation on how often the mathematical function is applied as long as every data value is only added once in its original form. The partial states are exactly the aggregation values of the respective partitions.

$$\text{partial state} = \text{aggregation of partition data}$$

This fundamental property of distributive aggregates can also be expressed with the mathematical property

$$f(N) = f(\{f(N_1), \cdots, f(N_i)\}), \quad N = N_1 \cup \cdots \cup N_i$$

Functions such as *min*, *max*, *sum* and *count* belong to this class. One can verify easily that these functions have the required property. Therefore we know that these basic and widely used functions can be calculated inside the network without any further difficulty.

3.1.2 Algebraic

Algebraic aggregates are a little bit more complex and do not allow in network aggregation out of the band. For these functions it is not longer true that the partial states are just the aggregation of the partial data. It is necessary to store some more information within the partial state such that it can be further used for more global aggregates. By definition of algebraic functions the amount of information necessary to store for partial states is of constant size.

$$\text{partial state} = \text{constant size}$$

The functions *average* and *geometric average* belong to this class. The partial state of the average function contains the partial sum and the number of

values which is the count function. A partial state containing these two values can be combined with other partial states. When everything necessary is aggregated the final average value can be evaluated by a simple division.

$$\begin{aligned} f(N_1) &= < sum_1, count_1 > \\ f(N_2) &= < sum_2, count_1 > \\ &\vdots \\ f(N_i) &= < sum_i, count_i > \\ f(N) &= < sum_1 + \cdots + sum_i, count_1 + \cdots + count_i > \\ &= < sum_N, count_N >, \quad N = N_1 \cup \cdots \cup N_i \\ avg(N) &= \frac{sum_N}{count_N} \end{aligned}$$

To calculate the geometric average the same procedure can be applied. The sum only needs to be replaced by the product of the values and at the end k-th root has to be taken where k is the number of values aggregated.

3.1.3 Unique

In the class of unique aggregates in network aggregation gets even more difficult and requires more data for partial states. For unique functions a constant size is not sufficient as intermediate results. Partial results require a size that is proportional to the number of distinct values as those partial states cannot be combined further without the loss of required information.

partial state = proportional size to distinct values in partition

The fact that partial states require more than constant size an in network calculation can become unfeasible due to scalability. Having more distinct values will eventually require more data to be sent among nodes and will be a bottleneck for a large value space.

The function *count distinct* belongs to the unique aggregations. Without keeping track of all distinct values it is not possible to calculate the number of distinct values. Therefore the size of partial states is of size equivalent to the number of unique values.

3.1.4 Content-sensitive

In certain cases the size required for partial states also depends on the values to be aggregated but is not simply proportional to the number of distinct values. The size can be proportional to any property of the data in the partition. It may also be that the size depends on some statistical property of the content provided.

partial state = proportional size to some proptery of data (statistical)

For this class it may be very difficult to predict how much data is required for intermediate results. Depending on the content given it cannot be decided a priori whether a certain function can be aggregated in network or whether the

partial states would require unfeasible size. The same bottleneck as for unique functions is also true on content-sensitive functions.

One out of many functions in the content-sensitive class is the *constant-width histogram*. The size of partial states obviously depends on the data and an a priori estimate of the width of the histogram cannot be given.

3.1.5 Holistic

With certain functions no in network aggregation can be preformed. This means that the data collected at nodes cannot be combined until everything is collected and then gets evaluated completely. Holistic aggregations are therefore adverse to optimizations. The partial states cannot do any combination of data. The size just grows by the number of values.

$$\begin{aligned} \text{partial state} &= \text{size of respective partial data} \\ &= \text{not containing any partial aggregation} \end{aligned}$$

As all data needs to be sent to the aggregating node in its original form, in network aggregation increases the message size dramatically. It can be chosen to perform either in network aggregation with increasing message size or straight forward aggregation with increasing number of messages. Therefore the functions contained within the holistic class are the most problematic functions and scalability cannot be achieved for large groups.

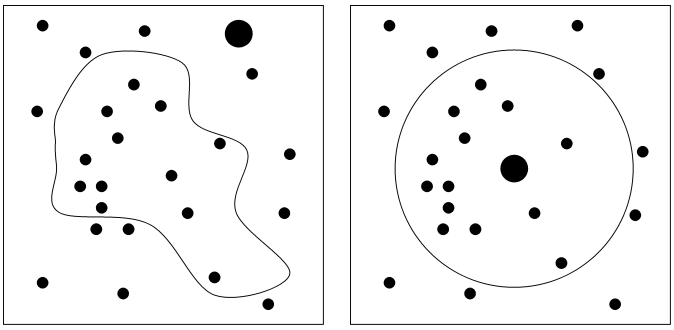
The function *median* belongs to the holistic class. To evaluate the median value of a group of values no intermediate partial results can be calculated. All values are required for the final evaluation. We can see that even a trivial function may cause problems to aggregate considering scalability issues.

3.2 Area of interest

The area of interest must be decided for every aggregation before its execution. Only the values of the nodes within the specified area will be included in the final aggregation. With a database point of view all aggregations are global and get restricted by some condition clause. However in a distributed environment the aggregation cannot be calculated as straight forward as can be done for a centrally stored database. Messages need to be exchanged to perform the same task on distributed nodes. The number of messages that can be exchanged is limited by a certain bandwidth and should not be exploited unnecessarily. Restricting the area where messages are exchanged to the area of interest is the very first approch to reduce the total number of messages sent for aggregations. Following are the three different types of aggregation areas described.

3.2.1 Global

The first and most simple aggregation is to include all the nodes into the calculation. There is no restriction on the area by any sense. Therefore the area of interest is global. In a distributed network the aggregation can be evaluated in many different ways as long as all nodes are taken into consideration and no node promotes its value multiple times.



(a) Regional area of interest (b) Localized area of interest

Figure 2: Areas of interest

3.2.2 Regional

The regional aggregation introduces a certain restriction on the number of nodes that are included in the evaluation process. The aggregation doesn't represent a global view anymore. In a distributed virtual world, where every node is located at some specific geographical location, aggregating only within a geographical area can be useful. The area must be predefined before the aggregation is started and the area must consist of exactly one n-space in the n-dimensional Euclidean space. In the practical case of a 2-dimensional virtual space the area is a bound and connected face. For many applications a circle with a radius r or a square with a diameter d around a central node makes most sense. Besides being reasonable the circular and square areas can be defined by only two values (circle and radius). Even though the regional area is not limited to a circle or square, more complex geometrical shapes require more information to be defined and therefore can be impractical to setup in a distributed fashion.

The aggregating node(s) do not need to lie within the area of interest but are not prohibited to do so either. This means that a node may want to know some information about an area far away from itself. In Figure 2 we can see an example of a rather complex regional area of interest with the aggregating node lying outside of it.

3.2.3 Localized

When the area surrounding the aggregating node is of interest then it refers to a localized aggregation. The node performing the aggregation lies within the area of interest and more restrictively it is center of the area of interest in some sense. For simplicity the area can be thought of as a circle with the center located at the aggregating node and a certain radius r. In Figure 2 this simple area of interest is illustrated. The area however doesn't need to be circular and can be of virtually any form as long as the aggregating node is its center. For practical applications only simple geometrical shapes such as circles, ellipses, squares, rectangles or simple symmetrical polygons seams to be of importance.

In the case where the whole or part of the virtual environment needs to be aggregated with several localized aggregations the shape of the area of interest receives another restriction. Besides being local it also needs to cover a certain

area without overlapping. Circles and ellipses provably cannot fulfill this restriction. The simple solution to this problem is by using square areas of interest lying next to each other. We will use this approach later on for solving our localized histogram aggregation.

4 Geometric graphs and networks

Long before computers existed geometric graphs were examined. Nowadays, many problems in computer science can be solved by using mathematically defined structures. In this section we would like to give an overview of geometrical graphs and methods that can be of interest for a virtual world overlay. First we present various planar graphs as well defined structures which may be used as two-dimensional computer networks. In the second part we list several techniques used for routing messages in geometric graphs. Some of those techniques are also applied to unstructured networks such as wireless ad-hoc networks.

4.1 Planar graphs

There are countless possibilities on how to connect points in a plane. A subset of all variations forms the group of planar graphs. Their special property by definition is that no two edges intersect each other in the plane. This property simplifies many problems (such as routing and dynamic node discovery) in distributed networks. Every planar graph has a well defined property such that for a given set of points the edges connecting those points are set unambiguously. In Figure 3 we can see the six common planar graphs which we describe in more detail in the following subsections.

4.1.1 Closest Pair (CP)

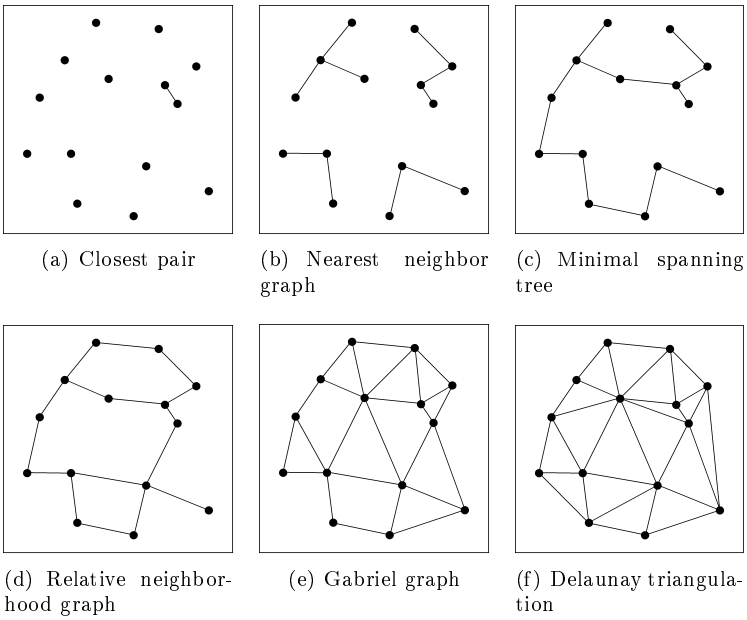
As we can see in Figure 3 in the first graph there is only one edge connecting two nodes. Having the set of vertices $\{v_1, v_2, \ldots, v_N\}$ the edge $e = v_i v_j = \min(d(v_i, v_j)) \mid i \neq j$ is selected. It is the edge with the shortest distance between any two vertices. Here $d(v_i, v_j)$ is the Euclidean distance from vertex v_i to v_j. A closest pair graph leaves most vertices without any connected edge. Obviously it is not connected and therefore of little use for communication networks.

4.1.2 Nearest Neighbor Graph (NNG)

If we want our set of vertices to have at least one connected edge per vertex then a nearest neighbor graph has to be constructed. It selects for every vertex $v_i \in \{v_1, v_2, \ldots, v_N\}$ the edge $e_i = \min(d(v_i v_j)) \mid i \neq j$. Consequently, the number of edges in the graph is $|E| \leq N$. The graph is not necessarily connected and most likely it is partitioned into smaller subgraphs in a tree structure as can be seen in Figure 3.

4.1.3 Minimal Spanning Tree (MST)

A spanning tree connects every vertex in a graph with any other vertex with exactly one path of consecutive edges. The minimal spanning tree is a special



(a) Closest pair (b) Nearest neighbor graph (c) Minimal spanning tree

(d) Relative neighborhood graph (e) Gabriel graph (f) Delaunay triangulation

Figure 3: Planar graphs

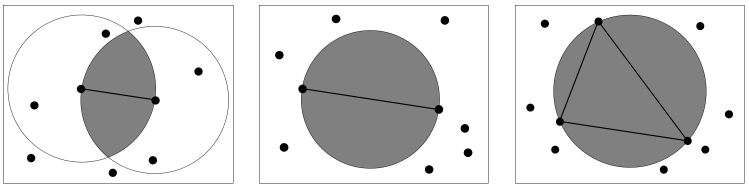
case of a spanning tree. It is the tree that has the smallest sum of edge lengths. The following two conditions are met by any minimal spanning tree

- $\forall v_i v_j \exists \{e_1, \ldots, e_k\} \mid \begin{cases} e_i \in E \\ e_1 = (v_i v_.) \wedge e_k = (v_. v_j) \\ e_i = (v_. v_x) \rightarrow e_{i+1} = (v_x v_.) \end{cases}$
- $MST = \min(\sum_{e \in \{e_1, \ldots, e_k\}} d(e))$.

This is the graph with the shortest possible global connectivity which makes it interesting for various applications. In computer networks where the number of hops is more important than the distance traveled between two nodes, a MST may not be an optimal solution.

4.1.4 Relative Neighborhood Graph (RNG)

Another graph which has potentially more edges than a tree is the relative neighborhood graph. Its idea is to connect two vertices that are 'relatively close' to each other. By definition two points p_i and p_j are 'relatively close' when $d(pi, pj) \leq \max[d(p_i, p_k), d(p_j, p_k)] \forall K = 1, \ldots, n,\ k \neq i, j$. This local property is illustrated in Figure 4. An edge between two vertices v_i and v_j exists if and only if the intersection of the two local discs with radius $r = d(v_i, v_j)$ doesn't contain any other vertex. More details about the RNG and how it can be constructed centrally can be found in [29]. In Figure 3 we see that the graph is well connected but doesn't form well defined shapes. This may make it more difficult to route messages in communication networks of moving peers.



(a) Relative neighborhood property (b) Gabriel graph property (c) Delaunay triangulation property

Figure 4: Local properties of planar graphs

4.1.5 Gabriel Graph (GG)

The Gabriel graph is also a globally connected graph with a simple local property. For every edge $e = (v_i v_j)$ it requires that the $disc(v_i, v_j)$ does not contain any other nodes [30]. This local property is illustrated in Figure 4. The complete graph is rather well connected as we can see in the set of vertices in Figure 3. Many triangles are formed but there can be exceptions which prevent the Gabriel graph from being a true triangulation. However this graph is of big interest as the indicated local property can be checked very easily.

4.1.6 Delaunay Triangulation (DT)

We will use the Delaunay triangulation later in this paper and therefore we give a closer look at this graph's properties here. As the name already suggests it constructs a triangulation with special properties. With a set of vertices there are potentially many triangulations possible. In general the Delaunay triangulation defines a unique triangulation out of all possibilities [25, 26]. There are degenerated cases we won't look at in more detail here but we will come back to them later on when we deal with a dynamic Delaunay maintenance.

4.1.6.1 Local empty-circle The very fundamental property of every Delaunay triangulation is the local empty-circle property shown in Figure 4. For every triangle $\angle v_1 v_2 v_3$ where all edges are selected $\{v_1v_2, v_2v_3, v_3v_1\}\partial E$ there exists no other vertex inside the circum circle of that triangle.

- $\nexists v_i \forall \angle v_1 v_2 v_3 \mid \begin{cases} \{v_1v_2, v_2v_3, v_3v_1\}\partial E \\ i \neq 1, 2, 3 \\ d(v_i, cc(v_1, v_2, v_3)) < d(v_1, cc(v_1, v_2, v_3)) \end{cases}$ [11]

This property assures that there is no vertex inside any triangle of the Delaunay triangulation. Furthermore, there is a certain empty space outside the triangle next to its edges.

4.1.6.2 Max-min angle For any two adjacent triangles having a convex hull (forming a diamond shape) two inner edges are possible. The Delaunay triangulation selects one of those two possibilities unambiguously. When the quadrilateral is divided into two triangles with one inner edge then six inner

[11] cc is the circum center of three given vertices.

angles exists. The smallest angles of both inner edges are compared with each other. The edge with the larger minimum angle is selected for the Delaunay triangulation. In the example in Figure 5 the inner edge e_1 is selected as it forms the *max-min angle*.

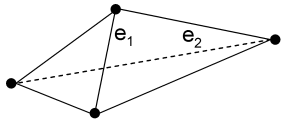


Figure 5: Max-min angle property of Delaunay triangulation

4.1.6.3 Locally optimal Every inner edge that fulfills the local empty-circle property or the max-min angle property is called to be locally optimal. In fact these two properties are equivalent for all inner edges. A triangulation is a Delaunay triangulation iff all inner edges are locally optimal.

4.1.6.4 Global connectivity For a communication network it is very important that a graph is globally connected. As we will se in section 4.1.7 the DT is a superset of the MST. As the minimum spanning tree is globally connected by definition we know that the Delaunay triangulation also must be connected.

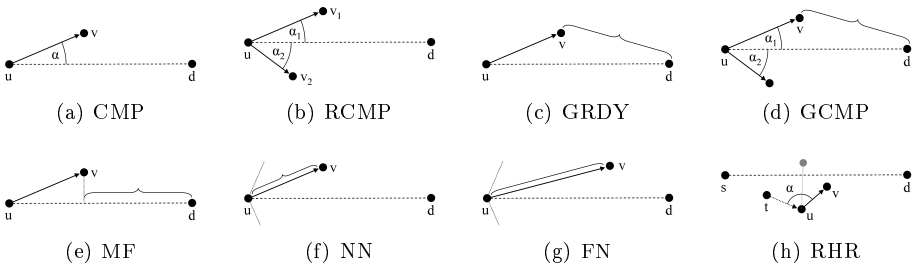
4.1.6.5 No edge intersections The Delaunay triangulation is part of the planar graphs which assures us that there cannot be any crossing connections between vertices. This property is particularly important for routing issues and delivery guarantees. We will use this knowledge later for routing on edges of triangles dividing the plane into lots.

4.1.7 Hierarchical relation among planar graphs

Up to here we have listed several planar graphs and their properties. In the presented order we could realize that latter graphs always have more edges than former graphs. Figure 3 also supports this theory and it even indicates that former graphs are subgraphs of later graphs. This intuition was proved in [29] and the following relation among the presented planar graphs holds

$$CP(V) \subseteq NNG(V) \subseteq MST(V) \subseteq RNG(V) \subseteq GG(V) \subseteq DT(V). \quad (1)$$

Some relations are quite trivial to see and others need a more detailed analysis. The relation among the first three graphs is obvious and could already be seen in the above explanations. The relative neighborhood graph is located between the minimal spanning tree and the Gabriel graph. In [29] it was shown that $MST(V) \subseteq RNG(V) \subseteq DT(V)$. To show that the Gabriel graph lays in between the RNG and the DT a simple geometrical analysis is sufficient. By looking at the local-empty properties of the RNG and the GG (see Figure 4) we can see that the RNG has a larger local empty area. In addition RNG's empty area encloses GG's empty area which clearly makes the RNG a subset of the GG. The relation between the GG and the DT can be shown in a similar



(a) CMP (b) RCMP (c) GRDY (d) GCMP

(e) MF (f) NN (g) FN (h) RHR

Figure 6: Localized routing techniques

way. Delaunay's empty area of a whole triangle is always fully covered by the corresponding circles for a Gabriel graph. Therefore all relations in 1 are shown.

4.2 Routing in geometric networks

In communication networks a connecting graph is not sufficient for efficient and guaranteed message exchanges. To send a message from one peer to another a predefined routing mechanism must be used if flooding needs to be avoided. Many general routing techniques have been studied in geometric graphs as well as wireless networks. In this section we present a selection of eight routing mechanisms and give information about their definitions and delivery guarantees for routing in triangulations. All eight localized routing techniques are illustrated in Figure 6. We only consider localized routing algorithms [22, 28, 30] since our P2P virtual world will be distributed and a global view of the graph is unavailable.

4.2.1 Compass (CMP)

A very simple mechanism for routing messages is the compass routing. Every node u needs to select one directly connected node v in the network as its successor to whom it will forward messages from a source s to a destination d. Compass routing selects the node v with the smallest angle in direction to the destination as its successor. This selection can be mathematically described as follows

$$\min_{v \in n(u)} (\measuredangle vud)$$

In the above formula $n(u)$ indicates all local neighbors of u. It was shown that the compass routing algorithm cannot be defeated by any regular triangulation [24]. This means that it is guaranteed that a message will be routed correctly from a source to a destination in any case. Compass routing also guarantees correct routing in a Delaunay triangulation as the following relationship among triangulations exists.

$$\mathrm{DT} \subseteq \mathrm{RT} \subseteq \mathrm{T} \subseteq \mathrm{CS} \mid \begin{array}{lcl} \mathrm{DT} & = & \text{Delaunaytriangulation} \\ \mathrm{RT} & = & \text{Regular triangulation} \\ \mathrm{T} & = & \text{Triangulation} \\ \mathrm{CS} & = & \text{Convex subdivision} \end{array} \tag{2}$$

The algorithm always succeeds in routing but it cannot guarantee a *c-competitive* Euclidean routing path in a Delaunay triangulation. This means that the 'travel distance' (the sum of the lengths of all edges utilized) from source s to destination d is not necessarily limited by $c\,\|sd\|$ for any constant c. However a *c-competitive* path exists on a DT and it can be found with a parallel Voronoi routing algorithm [24]. In wired communication networks the number of hops may be of much bigger interest though as it is the main factor for routing delays.

4.2.2 Random Compass (RCMP)

A probabilistic selection among two candidates as the successor is performed with the random compass routing algorithm. In a first step it selects the two local neighbors with the smallest angle clockwise and counter clockwise from the direction to the destination. In a second step it randomly selects one of those two nodes (each with probability $p = 0.5$) as its successor.

$$rand\left(\min_{v_1}\left(\overrightarrow{\measuredangle} v_1ud\right), \min_{v_2}\left(\overrightarrow{\measuredangle} duv_2\right)\right)$$

We introduce the notation $\overrightarrow{\measuredangle} xyz$, which indicates that the angle is measured at y in clockwise orientation between $\overrightarrow{yx}$ and $\overrightarrow{yz}$. The routing guarantee for a randomized algorithm can be give if the probability for routing a message from the source to the destination is not zero. In [24] it was proved that the randomized compass routing is more powerful than the standard compass routing. It was also shown that it guarantees routing in any triangulation. However, the routing path from a source to the destination is not deterministic which makes in network aggregation complicated for dynamic peers.

4.2.3 Greedy (GRDY)

The greedy routing method is also a very intuitive mechanism as it tries to approach the destination as close as possible with the selected successor. A node u forwards a message to the node v which is closest node to the destination among all neighbors of u and u itself.

$$\min_{v}\left(\|vd\|\right)$$

It is easy to see that the greedy algorithm reduces the distance to the destination with every single step. Eventually it will reach the destination or get trapped at a distant node. The algorithm guarantees a correct delivery in a Delaunay triangulation. However, in supergroups of the Delaunay triangulation (see the relationship of triangulations in 2) the delivery cannot be guaranteed. Later we will also see that the greedy algorithm is not sufficient for our special routing requirement in a Delaunay triangulation.

4.2.4 Greedy Compass (GCMP)

Greedy compass routing combines the two ideas of selecting the smallest angle and distance to the destination. First the two nodes with the smallest angle clockwise and counter clockwise towards the destination are nominated. Then

the node with the shorter Euclidean distance to the destination is selected among the two nominees.

$$\min_{\{v_1,v_2\}} (\|v_i d\|) \mid \begin{cases} v_1 = \min_v \left(\overrightarrow{\measuredangle} vud\right) \\ v_2 = \min_v \left(\overrightarrow{\measuredangle} duv\right) \end{cases}$$

This method is more reliable than both its two individual routing techniques as it assures correct routing in any triangulation [30]. This property can be particularly interesting as it guarantees correct routing in triangulations without using probabilities as in the randomized compass routing. A message being routed with greedy compass always reaches the destination.

4.2.5 Most Forwarding (MF)

This technique seems to be interesting as the projected distance to the destination decreases with every step on the straight line from s to d. A successor v of a nodes u is selected such that it minimizes the distance to the destination on sd. All local neighbors are first projected perpendicular onto sd and then the peer v is selected.

$$\min_v (\|\tilde{v}d\|) \mid \tilde{v} = \overrightarrow{uv} \cdot \overrightarrow{ud}$$

It is easy to construct a graph in which this routing technique will drift away from the straight line sd. Therefore it will not reach the destination. Also in the special case of a Delaunay triangulation the delivery cannot be guaranteed which makes it impractical in such a graph.

4.2.6 Nearest Neighbor (NN)

The nearest neighbor routing protocol selects the closest peer to the current node within an angle α to the destination.

$$\min_v (\|uv\|) \mid \measuredangle vud \leq \alpha$$

Also this method can be defeated easily by a specially constructed graph. For wireless networks this approach may make sense though as shorter connections require less transmission energy.

4.2.7 Farthest Neighbor (FN)

The farthest neighbor routing protocol does the exact opposite of the nearest neighbor routing by selecting the node farthest way within a given cone.

$$\max_v (\|uv\|) \mid \measuredangle vud \leq \alpha$$

This method does not route properly either on any graph and can be defeated by Delaunay triangulations as well. But it can be very efficient focusing on the number of hops in dense wireless environments where its result can be similar to the one of the greedy routing protocol.

4.2.8 Right Hand Rule (RHR)

A different approach to routing than the above rules is the right hand rule routing. This routing mechanism ignores all the connections intersecting the straight line from the source s to the destination d. On this subgraph the message is routed counter clockwise on the face containing s and d on its circum and the straight line sd in its interior. To achieve this face routing mechanism [15] a node u must know from which node t a message came from and where the source s and destination d are. Then at node u the node v with the smallest angle measured clockwise from the direction to $u's$ predecessor t is selected. The condition of intersecting the straight line sd much be always checked and if the selected node is intersecting then another local neighbor with a larger angle needs to be selected. This selection process can be described mathematically as

$$\min_v \left(\overrightarrow{\measuredangle} tuv \right) \mid \begin{cases} t \prec u \\ \overrightarrow{sd} \shortparallel \overrightarrow{uv} \end{cases}$$

In the above formula the symbol $\prec$ indicates that the peer t is the preceding peer of u (the message arrived from t at u) on a message's path from s to d. The symbol $\shortparallel$ indicates that the two vectors $\overrightarrow{sd}$ and $\overrightarrow{uv}$ do not intersect each other. The right hand rule routing is very powerful as it can route correctly in any planar graph. The disadvantage of this method is that it requires the location of the source for every intermediate routing decision. It is not clear how an in network aggregation could be performed with this requirement on a dynamic network.

5 Static localized histogram aggregation

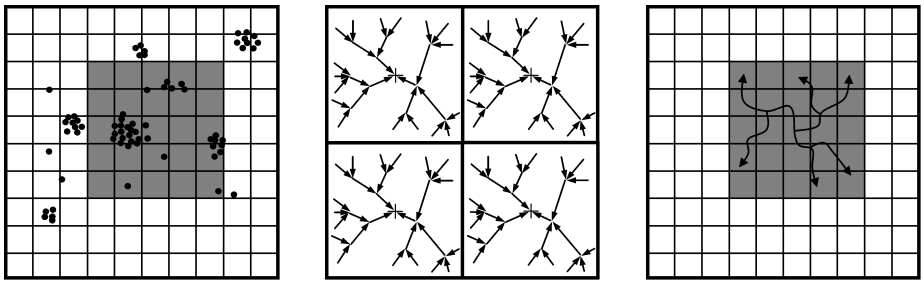
We divide the description of our localized histogram aggregation method into two sections. In a first and simpler step we present the process in a static environment where the nodes (or players in a MMOG) cannot move around. With the overview the reader can see the whole process of collecting, aggregating and distributing the values in our aggregation. The aggregating part is central to our method and needs a much closer look at routing on top of the chosen Delaunay triangulation. In a second step in section 6 we present solutions to make our solution applicable to dynamic players.

5.1 Overview

Our design makes it possible that every player in the MMOG can know details about its local neighborhood in the virtual world without overloading the P2P overlay network. The whole process is split into three parts. In the following subsections we describe these parts in detail.

5.1.1 Clustered virtual space

The whole virtual environment is partitioned into clusters. We cluster the environment with a 2-dimensional grid of square cells. This allows us to divide the whole virtual world without having any overlapping areas and still covering the entire space. This grid is of predefined constant size depending on the implementation and the gender of the MMOG. The size and location of the grid



(a) Clusted virtual space (b) Independant cluster aggregation (c) Broadcast aggregation

Figure 7: Aggregation process overview

fields is known by every node at startup of the MMOG and cannot be changed once the network is operating. It can be seen as a static part of the virtual environment. If an aggregation function should deliver a very fine histogram map then small grid fields are required. If a rough histogram map is sufficient than much fewer grid fields are necessary in the virtual world. However, all grid fields have the same side length which makes it possible to identify the enclosing grid field given any location in the virtual space.

For the histogram aggregation process every cluster is regarded as a unit which cannot be divided. When an aggregation is performed a cluster can either be included entirely in the result or left out completely. In Figure 7 the clustered virtual space can be seen with the locations of players in the game. We can also see that the local aggregated area is not the exact local environment of every node but the environment of all its surrounding clusters. We limit the precision of the aggregation function to the size of the static clusters. This allows us to aggregate every cluster once and reuse the same result numerously for every node that is interested in that cluster. All the nodes within the same cluster will therefore receive the same aggregation result even if their locations are not exactly the same. We believe that this approach is reasonable for minimizing message exchanges (as we will see in section 8) since the nodes within the same cluster are close together in the virtual world and therefore the view of their distant surrounding is very similar.

5.1.2 Independent cluster aggregation

After the virtual world is divided into clusters those grid fields are aggregated separately. As the clusters cannot be split for any aggregation process all the nodes in its area need to be aggregated to one result. In the distributed environment of a P2P MMOG this process needs to be well coordinated and requires most of our effort in this paper. The value of every node in a cluster should be included exactly once in the final aggregation result. In addition the process of exchanging messages should require as little messages as possible to minimize the message complexity of the overall aggregation.

We propose to use a tree-based aggregation in every cluster. This allows us to perform in network aggregations wherever tree branches join at a node. The

nodes at the boarder of the cluster (those that have no child node in the same cluster) initiate the aggregation by sending their value to a parent node in the tree. The tree also needs to have a root somewhere in the cluster. We designed the tree such that it has its root at the center of every cluster (see Figure 7). In order to establish this tree we choose the geometrically well defined Delaunay triangulation as overlay. It allows us to construct the distributed tree, route messages to the center and assure in network aggregation even with dynamic[12] nodes.

5.1.3 Broadcast aggregation

Once a cluster has aggregated its value it needs to be sent to the nodes who want to know it. This can be done by broadcasting the value over the same tree as for the aggregation but extended to the whole area of interest. This procedure can be seen in the third picture of Figure 7. This mechanism is particularly useful when all the players want to have the same aggregation of the same area of interest at the same time. The overview map of the virtual space is such an application as all the players want to see the number of players in different areas updated in certain intervals. The area of interest can be fixed in the MMOG to show a predefined area in a subwindow on every player's screen.

When we analyze the in network aggregation in a dynamic environment the possible topology changes in the Delaunay triangulation convince us that broadcasting informs all the nodes. As this broadcasting uses the Delaunay triangulation it may require many hops to distribute the information to distant nodes in dense areas. It seems to be possible to establish long range contacts (LRCs) as described in [18] to minimize the distribution delay. For our focus on the message complexity of the aggregation problem the proposed broadcasting allows us to make the overall process scalable. Therefore and due to time constraints we do not focus in more detail on the broadcasting part.

5.2 Routing

To assure that our independent clusters are aggregated correctly we need to define how the messages are aggregated using in network aggregation. As we already mentioned we propose to aggregate the value of each cell towards its center. First we show how a spanning tree is constructed on top of the Delaunay triangulation which is required for an in network aggregation. Then we show how a value is routed from a node to a static node on the Delaunay triangulation and last we extend this routing to a geometric location which is the main interest for our overall aggregation technique.

5.2.1 Spanning tree on Delaunay triangulation

If we can build up a spanning tree on the Delaunay overlay network then the in network aggregation can be performed using this tree. Wherever branches of the tree join a partial aggregation is calculated and forwarded to the next parent in the tree. In this way the links towards the root of the tree are used only once

[12] The dynamic needs to be limited to maintain the Delaunay triangulation which can be done with the DeDiLi algorithm of section 7.4.

compared to a straight forward aggregation where every value is routed to the root individually.

The following requirements are necessary for our tree to support a distributed aggregation:

1. One unique parent exists for every node in the network
2. One unique path exists from every node to the root
3. Local geometric information determines the tree
4. Children and parents can identify themselves mutually

Requirements 1 and 2 assure us that a spanning tree is constructed. Requirement 3 allows the tree to be constructed and maintained with local neighborhood information. The requirement 4 lets a child forward its value to its parent node which on the other hand waits on all its children (that are well known due to this requirement) before performing the in network aggregation and forwarding the result.

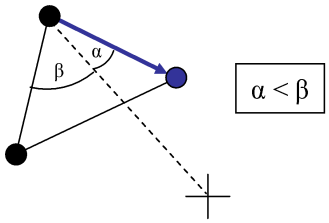
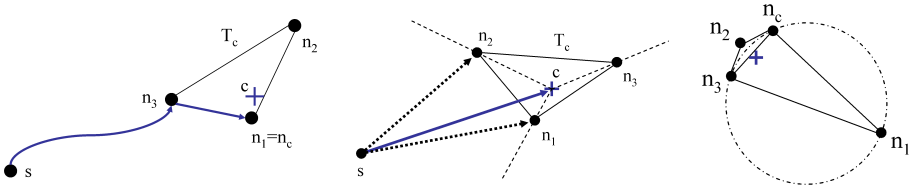


Figure 8: Spanning tree on Delaunay triangulation

In every *intersecting* triangle in the Delaunay triangulation a parent-child link can be identified. Our definition of an *intersecting* triangle is a triangle that contains one node whose direct line to the center intersects with the edge of the other two nodes. Such an intersecting triangle can be seen in Figure 8. The parent-child identification is done with compass routing. As we can see in Figure 8 every node in the network chooses its parent by selecting the neighbor in its intersecting triangle with the smaller angle to the root [19]. It is easy to verify that every node intersects exactly one neighboring triangle towards the root fulfilling requirement 1. As compass routing is correct on the Delaunay triangulation for every node, a spanning tree is built up rooted at the destination (requirement 2). We described how a parent is identified. In a similar way children can be identified by analyzing the neighbors' locations of a node. Therefore also requirements 3 and 4 are fulfilled giving us a locally constructed spanning tree on a Delaunay triangulation rooted at one of its nodes.

5.2.2 Routing to static node

As we have seen in the previous section a spanning tree is built up locally using the compass routing rule. To perform the overall aggregation every node identifies its parent and all its children. If a node has no children, no children in the same cluster or has received all partial aggregates from all children then



(a) Overall routing process (b) Compass routing decision (c) Failing geometric greedy routing on DT

Figure 9: Geometric routing on Delaunay triangulation

it can send its partial result to its parent. Executing this process at every node eventually aggregates all nodes of a cluster at the root of the spanning tree. Since a spanning tree connects all nodes of a cluster and compass routing guarantees delivery on a Delaunay triangulation, all values are properly included in the final result.

5.2.3 Routing to geometric location

Until now the messages are routed and aggregated towards a root node. Our aggregation process however should work in a dynamic environment of a MMOG. Therefore, we proposed to take the center of ever cluster as a fixed root. As there is no node at the exact location of the center in general one node must take responsibility as the root. For aggregation purposes the node 'closest' to the center is chosen. The closest node in the Delaunay triangulation is the one which is closest among the three nodes enclosing the center with a triangle.

The complete routing to a geometrical location can be split in two parts. First, a message is routed from a source to one of the three enclosing nodes of the center c. Second, the message is sent to the central node n_c. We will further specify the details in the following subsections and show that the routing is correct. The first graph in Figure 9 shows the idea of routing from a source s to the central node n_c. This will give us the possibility of using geometric routing for our aggregation process in a dynamic environment (see section 6).

5.2.3.1 Routing to a node enclosing the destination As the first step of routing a message to the central node n_c we want the message to route correctly to one of the three nodes enclosing the center. On the way to the center in network aggregation is performed wherever routing routes join at one node. This routing process is done in the same way as routing to a static node with compass routing. Nodes far away from the central triangle T_c build up the same spanning tree as if one of the enclosing nodes $\{n1, n2, n3\}$ were the root of the tree. Nodes closer to the center do not need to have the same parent for all of the three central nodes. However we show now that one of the enclosing nodes is always reached from any node sending a message with compass routing over the Delaunay triangulation.

In the second graph of Figure 9 the compass routing decision is illustrated. We can see that the direction from any node to the center is always in between the directions to two nodes (n_1, n_2) of T_c. As the message is forwarded using

compass routing the following node (parent) is also in between the parents of routing to n_1 and n_2. In addition cycling is not possible when routing to a node using compass routing on a Delaunay triangulation [24]. Therefore it is also not possible to cycle around T_c or cycling around a fictive node at the center would be possible. The two properties of 'in between direction' and 'non-cycling' make the triangle T_c behave like a single node.

Conclusion Compass routing to a geometric location in a Delaunay triangulation always arrives at one of the enclosing nodes of the central triangle.

5.2.3.2 Routing from an enclosing node to the closest node Once the cluster has been aggregated on the defined spanning tree then the partial results are located at the three enclosing nodes. To get the final result little is left to be done. Those three nodes need to aggregate their partial result to the overall cluster result which then will be broadcasted. The three nodes n_1, n_2, n_3 are connected with each other and can therefore mutually identify the node closest to the center n_c. The partial results are sent to n_c who is the current root of that cluster. Doing so completes the aggregation of a cluster in a static environment with the center of the cluster as the root.

5.2.3.3 Non-universal delivery guarantee In the previous two subsections we have shown and proved that the compass routing algorithm applied to a geometric routing on top of a Delaunay triangulation is correct. We also mentioned that the enclosing triangle behaves like a single node. However, it cannot be universally assumed that when a routing to a node is correct that the same routing technique is also correct routing to a geometric location. We show a counter example with the greedy routing to prevent future mislead delivery assumption. The third graph in Figure 9 shows clearly that node n_3 is closest to the center but is not part of the enclosing triangle. Therefore, a message arriving at node n_3 should be forwarded to the central node n_c. This however is not possible with the greedy routing algorithm as $n_3$is the closest node to the center in the whole cluster.

Conclusion The delivery guarantee for routing on a Delaunay triangulation is *not* universally equivalent for node and geometric routing.

5.2.4 Routing distance

It was shown in [24] that the Euclidean routing distance in a Delaunay triangulation using compass routing is not upper bound by a constant factor to the Euclidean distance between source and destination. The routing distance in the virtual environment on our spanning tree is of small interest. In general the virtual distance among nodes is in no correlation with the distance among the actual nodes. It is much more relevant how many hops a message requires to be routed from the source to the destination. This gives a much better idea on how far and how long a message has to 'travel' on the Delaunay triangulation.

First we show that circling around the destination in the Delaunay triangulation is not possible. For the sake of contradiction let us assume the circling exists. Such a routing path is illustrated in Figure 10. The gray shaded area

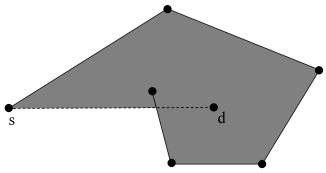


Figure 10: No circling guarantee

inside the spiral cannot contain any nodes connected to a spiral node. Otherwise that node would have been chosen for routing due to a smaller angle to d. However, all nodes in the shaded area need to be connected (with multiple hops) somewhere to the spiral. This is necessary as all nodes are connected with each other in a correct Delaunay triangulation and all nodes $n \notin CH$ are connected with edges spread at most 180 degrees (to fulfill the triangulation requirement). We conclude that the destination node d is in contradiction with the circling compass routing path. Therefore, circling around the destination is not possible.

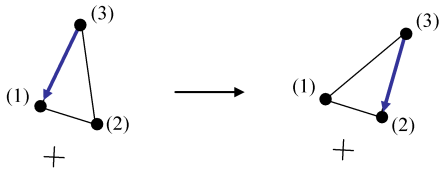
It is easy to see that the previous property can be extended such that circling around any point in the DT is not possible either as cycling was show to be impossible. With this property we know that compass routing uses a linear routing path from the source to the destination. The whole 2-dimensional virtual environment is covered with $n-2$ triangles, where n is the total number of nodes in the environment. On a linear path across the 2-dimensional environment $\sqrt{n}$ triangles are passed in average. Every triangle is passed with one routing step.

Conclusion The average number of hops on a Delaunay triangulation with compass routing is $\sqrt{n}$ for the diameter of the network with n nodes.

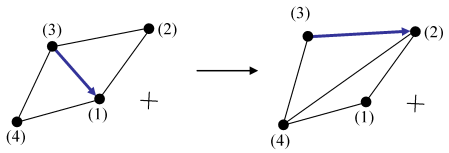
6 Dynamic localized histogram aggregation

In this section we focus on the aggregation process in a dynamic environment. The difference to the previous section is that the nodes in the Delaunay triangulation are not located at a static position anymore. For any realistic MMOG players need to be able to change their positions in the virtual world. We have seen how to aggregate correctly on a spanning tree using the in network technique. The spanning tree used is no longer static in the dynamic environment. As the tree is build up locally it can always be constructed properly. However, among moving nodes the parent function (compass routing) can change a node's parent depending on that node's neighbors' locations. This involves no problem for routing individual messages to the center, but for our aggregation process the in network aggregation becomes more complicated.

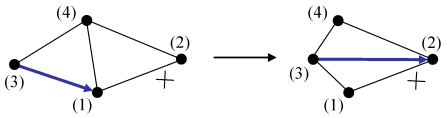
In this section we show all the parent changes that can occur and we present our solution to resolve all those changes with the Waiting Rules. We show that these rules are effective to aggregate properly in network and do not block the overall aggregation process. First we need to define what we understand of a *dynamic* Delaunay triangulation. It is obvious that nodes cannot move arbitrarily or the triangulation cannot be maintained locally. From now on we are using the following assumption on *dynamic*.



(a) Another parent



(b) Parent-child link dropped



(c) Parent-neighbor link dropped

Figure 11: Parent changes on flips in DT

Assumption Only FLIPS of inner edges can occur in our *dynamic* Delaunay triangulation between topology updates.

A FLIP occurs when two adjacent triangles change their inner edge to the one which was not in the Delaunay triangulation before. In the example of Figure 5 the edge e_1 is deleted and the edge e_2 is added to the DT. In section 7 we will show an algorithm to assure that only flips occur in a Delaunay triangulation network. This will ensure our assumption which we take as granted for now.

6.1 Parent changes

In this subsection all possible structural changes under the assumption are analyzed. This is necessary in the dynamic environment as the in network aggregation requires a controlled aggregation from the nodes towards the root. When a child in the spanning tree sends its partial aggregation to its parent, the child assumes that the parent is still waiting on that child's value. If the nodes move and the child receives a different parent (always chosen with compass routing) which has already forwarded its partial aggregation value then there is a problem for the in network aggregation technique. Following the three possible changes involving a parent-child connection are shown.

6.1.1 Another parent

The parent of a node can change without any topology change in the Delaunay triangulation as a child selects its parent solely by the smallest angle to the center. We can see this happening in the first graph of Figure 11. First the

node 1 is the parent of node 3. After a slight movement of the node 3 it receives node 2 as its parent node. In this case the parent has changed for node 3 and the DT topology was unchanged.

6.1.2 Parent-child link dropped

When a flip occurs and the parent-child link is dropped it is obvious that the parent of the child is changed. The new parent will be a neighbor of the previous parent. In Figure 11 we see that node 2 is the new parent of node 3. This is due to the fact that the edge (n_1, n_2) intersects the direct line from the child node 3 to the center. Here the parent of the node 3 changes to a node which was already a neighbor of the child before the movement.

6.1.3 Parent-neighbor link dropped

In the case when a parent-neighbor link is flipped among the adjacent quadrilateral a new edge gets connected to the child. If this new edge has the smallest angle to the center then the child's opponent node on this edge will be the new parent. This means that the child node 3 did not have any knowledge of the new parent node 2 before the flip as is illustrated in the last graph of Figure 11. However, the new parent was and still is a local neighbor of the previous parent node 1.

6.2 Waiting Rules

Parent changes as discuss in the previous subsection are a problem for the in network aggregation. If it were possible to assure that all nodes changing parents would select new parents that have not yet sent their partial aggregation value then the parent changes are no problem for the in network aggregation. It is not possible to select a new parent depending on its status of having sent the value or not. This is due to the fact that compass routing is used which assures a correct geometrical spanning tree at all times in the dynamic environment. The second possibility is to let nodes wait on forwarding their partial aggregation value. All potential parents should wait on forwarding their message until no further node can become its child anymore.

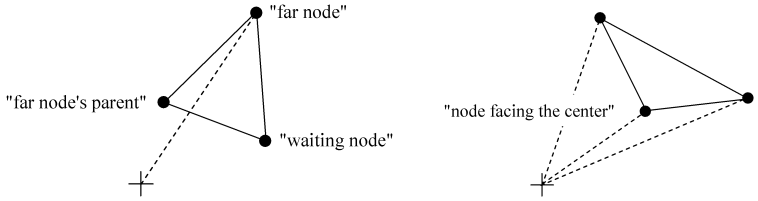
6.2.1 Functionality of rules

We propose to use three Waiting Rules (WR1, WR2, and WR3) that assure that all the parent changes are 'covered'. By 'covered' we mean that any new potential parent cannot send its partial aggregation towards the center until the potential has vanished. Following are the three Waiting Rules:

1. Parents wait on children
2. Intersecting triangle: only the *far node* can send its value
3. Non-intersecting triangle: *node facing the center* must wait on neighbors

For an explanation of an intersecting triangle go to section 8. In Figure 12 an example of an intersecting and a non-intersecting triangle is given. In addition the terms used in the Waiting rules are illustrated in the same Figure. The

far node is the node in an intersecting triangle that intersects an edge of the triangle by its straight line to the center. The *node facing the center* is the node in a non-intersecting triangle that is not part of the convex hull of the triangle's nodes and the center. It is the node of that triangle facing the center.



(a) Intersecting triangle (b) Non-intersecting triangle

Figure 12: Triangle types

Now we want to show that the proposed Waiting Rules cover all parent changes and assure a correct in network aggregation. This can be assured by showing that the aggregation is performed from the leaves to the root in the spanning tree and all parent-child changes do not influence the correctness of the aggregation. At any point in time an aggregation value is sent to its *current* parent.

WR1 prevents the current parent from sending its partial aggregation as long as there is a child that has not forwarded its partial result yet. This rule assures that the aggregation is performed on the spanning tree from the leave nodes (those without any children in the same cluster) to the triangle enclosing the cluster.

WR2 assures that *another parent* and *parent-child link dropped* is resolved. In both case we can see in Figure 11 that the node 3 changes its parent from 1 to 2. The WR2 does not allow the node 2 to send its partial aggregation value until the node 3 has sent its result to the current parent. In the event of a parent change the node 3 can still send its value to the new parent 2 and the in network aggregation process can continue correctly.

WR3 in conjunction with WR2 assures that also the *parent-neighbor link dropped* is resolved. In the case of two intersecting triangles as in Figure 11 the WR2 prevents the new parent 2 from sending its value before 3 has sent its value. This is due to the fact that WR2 makes node 4 wait which recursively forces node 2 to wait as well. In the case where the new parent 2 is the node facing the center in a non-intersecting triangle then WR3 is required. It assures that the node 2 also cannot send its value until node 1 and 4 have sent theirs. Therefore, any new parent of a node after a parent-neighbor link drop is still waiting on a potential new child.

The three Waiting Rules guarantee a proper spanning tree aggregation. All possible parent changes caused by structural and link changes (under the assumption of dynamic given) do not obscure the aggregation process.

Conclusion The 3 Waiting Rules assure correct in network aggregation in a dynamic Delaunay triangulation with only flipping edges.

6.2.2 Implementation of rules

The mentioned Waiting Rules should not only work properly but should also be implemented somehow. The three rules make nodes wait until a certain condition about their neighbors is fulfilled. The first Waiting Rule is easily implemented as parents can identify all their children. Then a parent just needs to wait until it has received the values from all children. A child node c explicitly informs its parent p that c is not blocking p anymore by sending $c's$ partial aggregation value to p. The second Waiting Rule needs additional communication among the nodes to be implemented. The *waiting node* in an intersecting triangle as seen in Figure 12 can identify itself as it has knowledge of the neighboring nodes in the Delaunay triangulation. Therefore, it can wait but needs to be notified once the *far node* has sent its value. This can be achieved if the *far node* sends its partial aggregation to its parent and the *waiting node*. The third Waiting Rule also requires a notification to the *node facing the center*. A node forwarding its value therefore needs to send the value to neighboring nodes of non-intersecting triangles.

In a general Delaunay triangulation there are few edges connected to every node. In average every node has exactly 6 connections which make the additional message exchange for the Waiting Rules a constant factor in average. Consequently, the proposed Waiting Rules are not only technically functional but can also be implemented practically.

6.2.3 Non-blocking aggregation

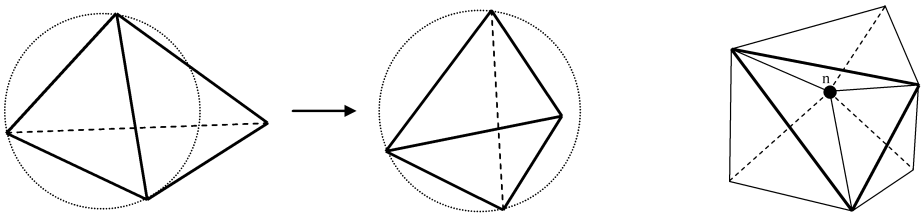
Waiting Rules allow a correct in network aggregation and now we show that the proposed Waiting Rules also do not block the overall aggregation at any point in time. We show this with the number of nodes that can be blocked within one triangle and the nodes that are obscured from adjacent triangles.

The WR1 blocks a maximum of 1 node in an intersecting triangle. WR2 also blocks 1 node in maximum. Having 2 blocking nodes there is always 1 node in an intersecting triangle that is not blocked by the triangle itself. In a non-intersecting triangle WR3 may block 1 node. The WR1 may also block the *node facing the center* by its two neighbors. But this is just the same node as is blocked by WR3 already. Therefore, there can only be 1 blocked node in maximum within a non-intersecting triangle.

It is easy to verify that adjacent triangles cannot block the node that is not blocked by the triangle itself with the Waiting Rules 2 and 3. However the 'unblocked' node can be blocked with the WR1 from an outside triangle. But the WR1 only guarantees that the in network aggregation is performed on the current spanning tree. The leaves of the tree cannot be obscured from outside triangles anymore and initiate the aggregation. Then the spanning tree shrinks and finally converges to the central triangle where the final result is evaluated. Hereby the three Waiting Rules are sufficient and necessary for a correct in network aggregation.

7 Dynamic Delaunay triangulation maintenance

So far our overall aggregation process performs correctly on a static and dynamic virtual environment using a Delaunay triangulation as overlay network. To allow



(a) Flip of quadrilateral

(b) Maximum non-overlapping flips

Figure 13: Flip properties of Delaunay triangles

the in network aggregation stay consistent we had to impose an assumption on the dynamic of the nodes in the Delaunay triangulation. In this section we show how the assumption of having only flips in adjacent triangles can be enforced. To do so the requirements for a dynamic aggregation focusing on flips is analyzed. Then the possible degenerated cases are presented which make the DT change its structure. After showing that any violation in the DT can be detected locally the DeDiLi algorithm is introduced. We will see that this algorithm allows the nodes to be dynamic and still assures the flip assumption.

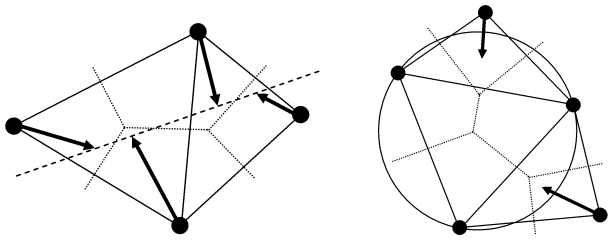
7.1 Requirements for dynamic aggregation

In section 6 we proposed three Waiting Rules that allowed us to aggregate correctly as long as only inner edges of quadrilaterals[13] can be exchanged. In this subsection we analyze more precisely what it means for individual nodes and triangles to have a flip in the overall Delaunay triangulation. First we show what happens locally when a quadrilateral gets flipped. Afterwards the maximal number of changes in adjacent triangles to a node is indicated.

7.1.1 Flips only

Let us remember the empty-circle property of the Delaunay triangulation from subsection 4. Any triangle cannot contain any other node within its circumcircle. This rule is particularly interesting for adjacent triangles. Since two adjacent triangles always have one common inner edge there exist two different circumcircles. Both circles cover partly the area of the adjacent triangle but they do not contain the third node of the adjacent triangle. In a dynamic environment all nodes should be able to move around by changing their location in the virtual space. The empty-circle property must hold at any point in time. The circles continuously change their locations and size according to their triangle's nodes. When the nodes of a quadrilateral move into a position where the circumcircles enclose the adjacent node then the inner edge of those two adjacent nodes needs to be flipped. This adjustment of the Delaunay triangulation can be seen in the first graph of Figure 13. It can be verified that performing

[13] Two adjacent triangles having a convex hull. It is also said that the two triangles are in diamond shape.



(a) Collinearity (b) Cocircularity

Figure 14: Degenerated cases in DT

this flip of the inner edge preserves the empty-circle property of the DT as the new edge with its circumcircles cannot cover the adjacent node anymore.

7.1.2 No overlapping flips

We have seen that quadrilaterals are allowed to switch their inner edge as nodes move around. Now we consider the whole Delaunay triangulation with all its triangles. Many quadrilaterals are overlapping by sharing a common triangle. Two overlapping quadrilaterals that move their nodes such that both want to flip their edges would result in an unpredictable outcome of the new connecting inner edges. It is trivial to see that the proposed Waiting Rules cannot cover all possibilities of such overlapping flips. Therefore, it must be prevented.

Any two overlapping quadrilaterals contain exactly one node that is connected to every other node of those quadrilaterals. It must be this nodes responsibility to limit the movement of the involved nodes such that a maximum of one quadrilateral can flip its inner edge. In the second graph of Figure 13 node n is taking that responsibility for its adjacent nodes. In the graph the absolute maximum number of allowed flips in node $n's$ local environment can be seen.

7.2 Degenerated cases

In [25, 26] it was shown that only (i, j)-transitions $(i \geq 2, j \leq d, d = dimension)$ of adjacent $(d+1)$-tuples can occur in $DT(S')$ $(S' = S \cup \infty)$ except for degenerated cases. In our case where $d = 2$ only $(2, 2)$-transitions $(= flips)$ occur. This is exactly what we wanted. Now we consider the degenerated cases that need to be avoided in our Delaunay triangulation overlay network.

7.2.1 Collinearity

There exists a degenerated case when more than $d + 1$ nodes are coplanar [26]. In the 2-dimensional case the Delaunay triangulation is defined by the empty-circle property. The degenerated case happens when two adjacent triangles with its 4 nodes become aligned on a straight line (see Figure 14). In the context of moving nodes the triangles will be defined again once the nodes have passed the collinearity. However, it can be seen that major changes in connecting edges to

surrounding nodes may become necessary. This can involve much more complex topology changes than flips and must be strictly prevented.

A first approach to keep the dislocation of a node local and lowering the possibility of passing collinearity is to use the local Voronoi area. We define the local Voronoi area of a node such that every node can identify it with its neighbor information.

Definition The *local Voronoi area* of a node n is the combination of the areas in every triangle closest to the node n.

In Figure 14 the local Voronoi areas can be seen. In every adjacent triangle of a node the local Voronoi area is given individually by the edges or the triangle and their perpendicular bisectors. This method prevents nodes from moving too far and still leaves the possibility to move in every direction. However, it cannot avoid all occurrences of collinearity among moving nodes as is shown in the first graph of Figure 14. Therefore, the Delaunay dislocation limitation algorithm in section 7.4 also has to prevent collinearity.

7.2.2 Cocircularity

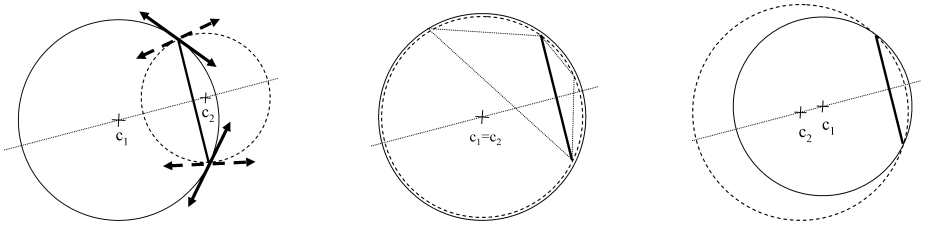
For our analysis the Delaunay triangulation is in general position[14] at the beginning. Nodes moving around in a dynamic environment can violate the empty-circles of other triangles. Not every violation needs to be avoided as they are necessary for flips to occur. However, there exists a degenerated case when more than $d + 2$ points lie in a hypersphere in the d-dimensional space [26]. In the 2-dimensional virtual world of a MMOG the degenerated case happens when more than 4 points become cocircular.

A distributed network does not support continuous updates among nodes. Therefore, we cannot consider the event of 4 or more nodes becoming cocircular. There need to be discrete location updates and verifications for the degenerated case. We transform the continuous requirement into a discrete requirement. In the discrete context no more than one node can move into any empty-circle between topology updates. In the second graph of Figure 14 we can see two adjacent nodes moving into the same empty-circle within their local Voronoi area. Such an event must be prevented before the topology gets updated. If we can show that an empty-circle violation can be detected locally then the responsible node of overlapping quadrilaterals (as mentioned in subsection 7.1.2) can be the solution to this problem.

7.3 Local Delaunay violation detection

In this subsection we show that any empty-circle violation can be detected by analyzing the movements of adjacent nodes. By adjacent nodes to a triangle we mean the three nodes that are double connected to the triangle. Those nodes form the adjacent triangles. In the following subsections we will see the properties of adjacent empty-circles and their impact on the detection possibility. Then we proof recursively that violations can always be identified locally. In addition it can be shown that the time of encounter with the empty-circle can be evaluated efficiently as long as all nodes move on a linear path with constant speed.

[14] The empty-circle property including its boundary is fulfilled.



(a) Adjacent empty circles (b) Identical empty circles (c) Impossible Delaunay circles

Figure 15: Adjacent empty-circles

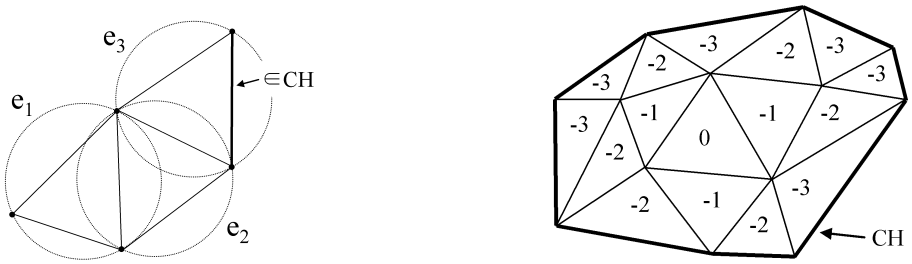
7.3.1 Adjacent empty-circle properties

Every triangle in the Delaunay triangulation has its own empty-circle. This circumcircle overlaps the triangle in three partitions next to every edge. The violation of the empty-circle can occur in any of those three partitions. For the simplicity of the analysis we focus on only one partition without the loss of generality. In the first graph of Figure 15 such a partition with its edge is shown. The two adjacent empty-circles sharing this common edge are illustrated. Both centers of the circles lie on the perpendicular bisector of the common edge. It is also necessary that the centers' mutual location corresponds to the side on which the third nodes are located in respect to that edge. The center c_1 is on the left of c_2 and c_1*'s* circle also contains the adjacent triangle node to the left of the common edge and vice versa for c_2 on the right. Due to this mutual location of the centers on the perpendicular bisector the partition of c_1 on the right of the edge is fully covered by c_2's circle. Covered means that the mentioned partition is protected against violating nodes by the adjacent empty-circle when the nodes move continuously. Only the third node on c_2's circle can move into c_1's partition without destroying other empty-circles beforehand.

In the case when the two centers reside at the same location then the 'two' circumcircles are identical (see Figure 15). Since our Delaunay is assumed to start always in a general position the empty-circles cannot coincide. The third graph in Figure 15 shows the circumcircles when their centers are in inversed mutual locations on the perpendicular bisector. This situation cannot exist for any Delaunay triangulation as c_1's circle covers the c_2's partition on the side of the common edge where it should have its third adjacent node.

7.3.2 Local detection of empty-circle violations

In this subsection we show recursively that an empty-circle cannot be violated firsthand by any node other than its adjacent nodes. For the sake of *contradiction* let us assume that a non-adjacent node can enter the empty-circle. The previous subsection gives the property that an adjacent empty-circle must be destroyed *beforehand*. In our example of Figure 16 this means that the empty-circle e_2 must be violated before e_1 can be violated in e_2's partition. The same is true for e_3 as it must be destroyed prior to e_2. Recursively adjacent empty-circles need to be violated before the local empty-circle can be violated by a node other than the three adjacent nodes. At some point the recursion path arrives



(a) Adjacent circles towards CH

(b) Timely violation requirement

Figure 16: Recursive timely violation requirement

at the convex hull of the Delaunay triangulation where no further empty-circle exists.

In fact the whole Delaunay triangulation can be put in a timely violation requirement among triangles for every starting triangle. Adjacent triangles always need to have an earlier timestamp which decreases towards the bounding convex hull (see Figure 16). For every edge of the convex hull no node lies on the half-plane opposite the triangulation network. Therefore, there exists no node that could violate the empty-circle containing the edge of the convex hull in that edge's partition. This is in *contradiction* to the required timely relation where the empty-circle on the convex hull should be violated *first*.

Conclusion It is sufficient to check the three adjacent nodes to any triangle in the Delaunay triangulation for detecting its empty-circle violation.

7.4 DeDiLi: Delaunay Dislocation Limitation algorithm

With the knowledge that Delaunay violations of moving nodes can be detected locally, we can implement an algorithm that limits the dislocation of every node such that the DT can be maintained. Our approach is to prevent the degenerated cases and overlapping flips from occurring by limiting the movements of nodes *before* they change their position. The following two functions CCW and $INCIRCLE$ allow us to check nodes on collinearity and cocircularity respectively. We will need these in our DeDiLi algorithm.

$$CCW\,(x,y,z) = \det \begin{bmatrix} x_x & x_y & 1 \\ y_x & y_y & 1 \\ z_x & z_y & 1 \end{bmatrix} \tag{3}$$

$$CCW\,(x,y,z) \begin{cases} > 0 \rightarrow \text{z lies on left of } \overrightarrow{xy} \\ < 0 \rightarrow x,y,z \text{ are collinear} \\ = 0 \rightarrow \text{z lies on right of } \overrightarrow{xy} \end{cases} \tag{4}$$

$$INCIRCLE\,(x,y,z,d) = \det \begin{bmatrix} x_x & x_y & x_x^2 + x_y^2 & 1 \\ y_X & y_y & y_x^2 + y_y^2 & 1 \\ z_x & z_y & z_x^2 + z_y^2 & 1 \\ d_x & d_y & d_x^2 + d_y^2 & 1 \end{bmatrix} \tag{5}$$

Algorithm 1 DeDiLi: Delaunay Dislocation Limitation

choose $\tilde{n}$ (=desired new location);
send $\tilde{n}$ to $neighbors\,(n)$; receive $\tilde{n}\,[1..N]$ from $neighbors\,(n)$;
sort $\tilde{n}\,[1..N]$ clockwise according to $n\,[1..N]$;
$t := 1$;
$t\,[1..N] := 1$;
$flipped\,[1..N] := 0$;
$rand\,[1..N] := rand\,(\{1, 2, \ldots, N\})$;
for $i := 1$ **to** N
  $j := rand\,[i]$;
  $\tilde{t} := \min_{t\in[0,1]}\,(CCW\,(n_{j-1}\,(t)\,, n\,(t)\,, n_j\,(t)) > 0)$;
  $\tilde{t} := \min_{t\in[0,1]}\,\left(\tilde{t}, CCW\,(n_j\,(t)\,, n\,(t)\,, n_{j+1}\,(t)) > 0\right)$;
  if $(flipped\,[j-1] == 1 \| flipped\,[j+1] == 1)$ **then**
    $\tilde{t} := \min_{t\in[0,1]}\,\left(\tilde{t}, INCIRCLE\,(n_{j-1}\,(t)\,, n\,(t)\,, n_j\,(t)\,, n_{j+1}\,(t)) < 0\right)$;
  else if $(INCIRCLE\,(n_{j-1}\,(t)\,, n\,(t)\,, n_j\,(t)\,, n_{j+1}\,(t)) \geq 0)$
    $flipped\,[j] := 1$;
  $t := \min\left(t, \tilde{t}\right)$;
  $t\,[j-1] := \min\left(t\,[j-1]\,, \tilde{t}\right)$;
  $t\,[j] := \min\left(t\,[j]\,, \tilde{t}\right)$;
  $t\,[j+1] := \min\left(t\,[j+1]\,, \tilde{t}\right)$;
for $i := 1$ to N
  send $t\,[j]$ to neighbor j;
  receive $t\,[j]$ from neighbor j;
$t := \min\,(t, t\,[1..N])$;
dislocate to $n\,(t)$;

Definitions:
$n\,(t) := n + (\tilde{n} - n)\,t$;
$n_j\,(t) := n\,[j] + (\tilde{n}\,[j] - n\,[j])\,t$;

$$INCIRCLE\,(x, y, z, d) \begin{cases} > 0 \rightarrow \text{d inside } circle\,(x, y, z) \\ < 0 \rightarrow \text{d outside } circle\,(x, y, z) \\ = 0 \rightarrow \text{d on } circle\,(x, y, z) \end{cases} \tag{6}$$

We want the nodes to negotiate with their neighbors about how far they can move. For this process every node needs to select a *desired* new location $\tilde{n}$ to where it would like to move until the next topology update. We make the following assumption on how the nodes move between update.

Assumption Nodes move *linearly* with *constant speed* between topology updates.

This means that a node will be at position $n\,(t) = n + (\tilde{n} - n)\,t$ at time t where n is the old position and $\tilde{n}$ is the desired new position before the next topology update. Using linear motions of nodes $n\,(t)$ as the inputs in the above equations 3 and 5 gives us polynomials of degree 3 and 4. Evaluating the smallest root ($t \in [0, 1]$) of the polynomial returns the time of the occurrence of a first collinearity or empty-circle violation. This root can be found efficiently due to the low degree of the polynomials. If no root exists in $t \in [0, 1]$ then the new

desired locations of the examined nodes do not require any topology updates in the Delaunay triangulation. Those nodes don't need to be limited by themselves and $t = 1$ can be chosen. However, if a root exists collinearity must be prevented and edge flips (equivalent to an *INCIRCLE* violation) must be avoided if an overlapping quadrilateral was already allowed to flip. A maximum allowed dislocation can be found by setting the new maximum possible t of all involved nodes just before the root that must be prevented. Choosing the smallest dislocation t all neighbors agree on prevents degenerated cases from happening and the whole Delaunay triangulation can be maintained (see subsection 7.5.1 for a more detailed maintenance requirement). The DeDiLi pseudo code in Algorithm 1 on the preceding page presents the mentioned idea more formally.

7.5 Algorithm analysis

The given DeDiLi algorithm proposes a distributed algorithm that allows a Delaunay triangulation to maintain its topology among dynamic nodes. The nodes cannot freely choose their new location as they get limited in the distance they can move in a desired direction. In this subsection we show that the requirements for maintaining the Delaunay triangulation are fulfilled and we give a simple worst case of a flipping edge.

7.5.1 Requirements fulfillment

From subsection 7.1 we know that to maintain a dynamic Delaunay triangulation only flips of adjacent triangles are allowed and that no overlapping flips shall occur. In addition the degenerated cases from subsection 7.2 cases must be prevented. The DeDiLi algorithm is executed at every node which means that for every quadrilateral there are two nodes that are connected to all four nodes. The algorithm randomly processes all its neighbors (line 8 in Algorithm 1). Performing the *INCIRLCE* function on line 13 detects the first occurrence of an empty-circle violation if an adjacent neighbor was allowed to flip already (checked in line 12). Therefore, adjacent neighbors cannot both be flipped. This assures that only flips occur as no overlapping quadrilaterals can both change their inner edge. It is also the requirement necessary to prevent the degenerated case of cocircularity (see Figure 14).

The second degenerated case (collinearity) is prevented with the lines 10 and 11. There the maximum dislocation of the adjacent triangles to the checked node is strictly limited such that node collinearity among three nodes exists. Since we do not allow collinearity among three nodes to occur it also cannot happen among four and more nodes.

The algorithm considers the limitations of all neighbors (line 20 through 23). It decides on the maximum dislocation that is allowed by all neighbors (line 23). Doing so assures that only flips as desired can occur in the Delaunay topology between location updates.

7.5.2 Worst case analysis of edge flip

The DeDiLi algorithm prevents nodes from moving too far such that overlapping quadrilaterals cannot flip at the same time. A worst case occurs when a

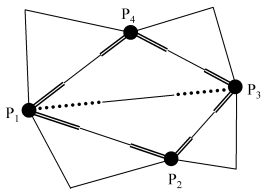


Figure 17: Worst case quadrilateral flip

quadrilateral wants to flip and all its overlapping quadrilaterals also want to change their inner edge. Such a situation is illustrated in Figure 17. The edge P_1P_3 wants to be exchanged by the edge P_2P_4 and all other inner edges should change as well for the desired new locations. We want to show that the edge P_1P_3 can still have a chance to flip and the DeDiLi algorithm does not deadlock in such a situation.

The DeDiLi algorithm randomly checks quadrilaterals around every node. In the worst case all quadrilaterals want to flip. The quadrilateral that is checked first is allowed to flip and the overlapping ones will be prevented. In the example given the nodes P_1 and P_3 are the key points that the flip of the dotted line can occur. Both of them need to check the dotted edge before they check the two adjacent double edges. This happens with a probability of $\Pr = 1/3$. The nodes $P_2$and P_4 cannot prevent the flip from happening directly and will block the double edge after they receive the maximum dislocations from P_1 and P_3. Therefore, the deadlock is prevented with a probability$\Pr = 1/3 * 1/3 = 1/9$.

8 Analysis of message complexity

In this section we are analyzing the number of messages needed to send among nodes to perform a localized histogram aggregation. First, the message complexity in two simple aggregation techniques is evaluated. Then our clustered algorithm is analyzed. The analyses are based on a *collective* aggregation among the nodes. This is the type of aggregation that is best supported by our clustered aggregation procedure and it is also the aggregation that is of very high interest for future MMOGs.

Definition A *collective* aggregation uses the same *aggregation function* and the same *size of area of interest* at the same *time* for all node.

The neighborhood map mentioned in the application examples of subsection 1.3 uses exactly such a collective aggregation. Every player in the game wants to see its own neighborhood map. For this application a simple *count* is used as the aggregation function. All players want their neighborhood map updated frequently which means that all of them are interested in the aggregation result whenever it is available. In other words all players 'consume' the aggregation at the same time. The size of area of interest can be predefined by the implementation of a MMOG. The neighborhood map therefore is an application of high interest requiring a *collective* aggregation.

8.1 Straight forward aggregation

For this first analysis of how many messages need to be sent for a collective aggregation the most pragmatic way (straight forward) of aggregating locally is used. The value of every node within the radius gets routed to the aggregating node individually. We are interested in the final result that shows the number of messages sent per node in average.

Definition In a *straight forward aggregation* all values within the area of interest are individually routed to the aggregating node.

Following we define some variables which we are using in our mathematical evaluation.

- n = number of nodes in the whole virtual environment
- r = radius of aggregation
- $4r^2$ = aggregation area (square with side length $2r$)

We also make some assumptions on the virtual environment.

- Environment area = 1 ($\rightarrow$ node density = 1)
- Node locations are uniformly distributed

First we calculate the number of nodes within one node's aggregation area $N = 4nr^2$ which is the aggregation area times the node density. Now the average number of hops to send a message from a source to the target is needed. We use the conclusion from subsection 5.2.4 which tells that $\sqrt{n}$ hops are required in average to cross the diameter of an uniformly distributed environment with n nodes. The analysis used the compass routing on a Delaunay triangulation but the $\sqrt{n}$ seems to be a reasonable average bound for any virtual environment with *only* local connections. We want to know how many hops are necessary for a Euclidean distance e which is

$$h(e) = \sqrt{n}e \tag{7}$$

The infinitesimal square area at a distance e from the aggregating node is

$$\begin{aligned} a(e) &= 4(r+dr)^2 - 4r^2 \\ &= 4r^2 + 8r\,dr + 4dr^2 - 4r^2 \\ &= 8r\,dr + 4dr^2 \\ &\approx 8r\,dr \end{aligned} \tag{8}$$

It follows that there are approximately $n(e) \approx 8nr\,dr$ nodes in the infinitesimal area at distance e. Using this result and equation 7 we can integrate over the distance from the aggregating node to the radius of the aggregation area.

This gives us the average number of messages sent in the environment for the aggregation value at one node.

$$\begin{aligned} m &= \int_0^r \sqrt{n}r \cdot 8nr\, dr \\ &= 8n^{3/2} \int_0^r r^2\, dr \\ &= \frac{8n^{3/2}r^3}{3} \\ &= O\left(n^{3/2}r^3\right) \end{aligned} \tag{9}$$

All nodes in the environment require the same amount in average. Therefore, the above result also represents the average number of messages every single node in the environment needs to send for a collective localized aggregation.

8.2 In network aggregation

Instead of routing values individually partial aggregates can be calculated on the way to the aggregating node. This in network aggregation can be applied to distributive and algebraic aggregation functions. In this analysis we do the in network aggregation wherever it is possible at nodes of joining routes. Any spanning tree rooted at every node extending to the radius of the aggregation area is used. This allows only one aggregation value (no histogram information) for the whole range which deteriorates the notion about the node's local environment. The same definitions and assumptions as in the previous subsection are used.

Since every node is linked to a parent node in the spanning it only needs to send 1 message for every aggregating node. No messages need to be forwarded since the nodes wait until they can send the partial aggregation value of their subtree. Consequently, the number of messages sent for the aggregation of one node is equivalent to the number of nodes within its area of interest. It also represents the number of messages a nodes needs to send in average for a collective environment aggregation with radius r

$$\begin{aligned} m &= 4nr^2 \\ &= O\left(nr^2\right) \end{aligned} \tag{10}$$

8.3 Clustered aggregation

In this subsection we analyze our proposed localized histogram aggregation technique. The virtual environment is clustered in independent square cells. Every cell aggregates on its own with the in network technique on a spanning tree. Then the result is broadcasted to the radius of the area of interest on the extended spanning tree. At this point it is important that a *collective* aggregation is performed. Only if all nodes have the same universal radius r then the broadcasting to r can assure delivery. Otherwise the required broadcasting radius is unknown to the cells. There might be nodes far away with a large area of interest that would not receive the required broadcast of a smaller radius.

For the mathematical analysis we need two additional definitions

- f = number of clusters in the virtual environment
- g = number of clusters in the universally sized area of interest

As in the previous subsection every node sends exactly 1 message for the in network aggregation process. Since there are no overlapping spanning trees due to clustering the total number of messages for the aggregation process is is also 1. We need to consider the additional messages required for the Waiting Rules of the dynamic aggregation. In subsection 6.2.2 we proposed to send a value to the parent and to the neighbors who need to know when the message was sent. In average a Delaunay triangulation has 6 neighbors at each node. We take another 6 messages per node into our average calculation which is definitely an upper bound for the average case. We do not include the messages required to maintain the Delaunay triangulation as we consider the topology maintenance apart from the aggregation process. After the aggregation of every cluster those results are broadcasted which involves again 1 message per node on the spanning tree of every cluster. The total number of messages m_c required for broadcasting one cluster equivalent to the average number of nodes in its radius.

$$m_c = 4nr^2$$

The total number of messages M in the whole virtual environment is composed of the aggregation, Waiting Rule and broadcasting messages

$$M = n + 6n + f \cdot 4nr^2$$

This leads to an upper bound on the average number of messages every single node needs to send for a collective aggregation. The property $g = 4fr^2$ that the number of cluster in the area of interest is in relation with the total number of cluster and the radius is used in the evaluation.

$$\begin{aligned} m &= 4fr^2 + 7 \\ &= g + 7 \\ &= O(g) \end{aligned} \tag{11}$$

If a node wants to build up a clustered histogram it needs to receive at least as many messages as there are clusters in its area of interest. When every node in the environment wants such a histogram then the number of outgoing and incoming messages is the equal in average. This leads to the lower bound of sending messages per node for a clustered histogram aggregation.

$$m = \Omega(g)$$

Conclusion The localized histogram aggregation is asymptotically average optimal in message complexity.

8.4 Comparing algorithms

Considering the asymptotical average number of messages (equations 9 and 10) required per node, we see that the in network algorithm performs better than the

straight forward aggregation. On the other hand straight forward aggregation can handle any aggregation function whereas with in network only distributive (min, max, sum, prod) and algebraic (avg, geometric avg) functions are supported. In addition the in network algorithm does not support a histogram aggregation which results in less detailed information for the aggregating node.

Our clustered algorithm is the only one which does not contain n in its message complexity. This means that it is the only one among the three that is fully scalable to the number of users in the virtual environment. The clustered aggregation however comes with an offset in precision as the aggregation cells are predefined and the exact aggregation range cannot be calculated. It allows like the in network algorithm the distributive and algebraic functions. These 'simple' functions are of high importance and cover all the example applications given in subsection 1.3. We can also see that the number of messages in dense areas does not increase as it depends solely on the clusters.

In a sample virtual environment with $n = 10^6$ players and an aggregation radius $r = 0.05$ with $f = 10^4$ clusters there are $g = 100$ clusters inside the area of interest. With these 'reasonable' assumptions for a future MMOG we show the numerical message complexity for the above three aggregation techniques.

1. $m \approx 333'333$ for straight forward aggregation
2. $m \approx 10'000$ for in network aggregation
3. $m \approx 107$ for clustered aggregation

The above numerical analysis shows a decrease of 2 magnitudes in message complexity in favor of the newly proposed localized histogram aggregation technique. The message complexity stays at a low degree for massively players whereas the complexity of the first two algorithms is completely out of bound.

9 Conclusion

In the presented work there are several achievements that count towards our contribution. First we identified the need for a localized histogram aggregation for future MMOGs based on a P2P topology. Without the existence of related work about distributed histogram aggregations we proposed a clustered virtual space to reduce the message complexity for collective aggregates. The overview of the whole aggregation process was designed independent from the underlaying topology using the idea of aggregating in network to the center within each cluster. It allows the calculation of distributive and algebraic aggregation functions by exchanging messages of constant size in the network.

The Delaunay triangulation was chosen as the overlay topology. It supports the distributed construction of a spanning tree (required for in network aggregation) with compass routing and mutual parent-child identification. We showed that the three introduced Waiting Rules assure a correct aggregation among dynamic nodes in the virtual world. The dynamic of the nodes had to be limited such that only flipping edges in the Delaunay triangulation can occur. However, we believe that this assumption is acceptable as all possible movements can be performed with subsequent flips.

We could prove that Delaunay triangulation violations can be detected locally. Using this knowledge the Delaunay Dislocation Limitation (DeDiLi) algorithm was created to assure only non-overlapping flips. Therefore, we are able to maintain the Delaunay triangulation as required for the correct dynamic aggregation. The dynamic can be limited very much but our worst case analysis shows that a flipping edge always has a certain chance to do so.

The average message complexity at every node of a collective aggregation was evaluated to the lower bound of a localized histogram aggregation. Therefore, an asymptotical average optimal message complexity could be shown. We believe that our process is a first approach to a localized histogram aggregation for P2P MMOGs that is scalable to the number of players in the game.

10 Future work

The purpose of this work was to find a scalable aggregation technique. The message complexity was analyzed and found to be optimal in average. Applications in a P2P MMOG such as the neighborhood map want their data to be updated frequently. Since every player would like to see the *current* local overview of players the aggregation data should be up to date. Our broadcasting step requires $\sqrt{n}r$ hops in average to reach all players in the area of interest. This can cause long delays for clusters at the border of the reception area. Long range contacts (LRCs) as proposed in [18] could be a future improvement to the *aggregation speed*.

In our analysis we showed how an aggregation can be performed on *reliable* peers. However, in any P2P overlay network nodes can join, leave or fail. The overall network should be able to remain in a consistent state and recover from the topology changes. The solution presented in [27] to construct a Delaunay overlay network incrementally might be used for a local and fast reconstruction of the neighborhood of a failing node. Together with a consistent aggregation a *fault-tolerant* overlay network could be constructed.

The proposed Delaunay Dislocation Limitation algorithm (subsection 7.4) assures that the Delaunay triangulation can be maintained with only flips. However, it limits the movements of nodes between topology updates. In future work the *dynamic* should be analyzed and possibly improved with a more advanced limitation algorithm.

Acknowledgment

I would like to express my very warm thank you to Professor Roger Wattenhofer and Professor Junehwa Song. Their open-minded and international thinking gave me the exceptional opportunity to write my master thesis in South Korea in collaboration between ETH Zurich (Swiss Federal Institute of Technology Zurich) and KAIST (Korean Advanced Institute of Science and Technology). A great thank you also goes to Dr. Jaesun Han, Sungwon Choe, Sunghwan Ihm, Hyonik Lee and Jinwon Lee who helped in many meetings and discussions to find the right tracks. Many other members of Professor Song's Network Computing Laboratory gave useful inputs in discussions and shall be mentioned herein.

A very special thank you goes to my parents who supported me during all my studies. I owe them the accomplished international experiences that gave me an extremely valuable insight into various cultures on this planet. My friends from the Swiss lifesaving swim team SLRG helped me recharge my brain after hours of studying. Thankful for everyone's support I come to the completion of my *Master of Science ETH in Computer Science*.

References

[1] Gamespy.com, *http://archive.gamespy.com/amdmmog.*

[2] MMOGCHART.COM, *http://www.mmogchart.com.*

[3] Kazaa, *http://www.kazaa.com.*

[4] Skype - Free Internet telephony that just works, *http://www.skype.com.*

[5] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking Up Data in P2P Systems. *Communications of the ACM*, 46(2), pages 43-48, February 2003.

[6] Ji Li and Dah-Yoh Lim. A Robust Aggregation Tree on Distributed Hash Tables. In *Proc. MIT Student Oxygen Workshop (SOW)*, Ashland, MA, USA, September 2004.

[7] Alberto Montresor, Márk Jelasity, and Ozalp Babaoglu. Robust Aggregation Protocols for Large-Scale Overlay Networks. In *Proc. of the 2004 International Conference on Dependable Systems and Networks (DSN)*, pages 19-28, Florence, Italy, June 2004. IEEE Computer Society.

[8] P. Th. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. From Epidemics to Distributed Computing. *IEEE Computer*, 37(5), pages 60-67, May 2004.

[9] I. Gupta, R. van Renesse, and K. P. Birman. Scalable Fault-tolerant Aggregation in Large Process Groups. In *Proc. of the 2001 International Conference on Dependable Systems and Networks (DSN)*, pages 433-442, Goteborg, Sweden, July 2001. IEEE Computer Society.

[10] Fabian Kuhn, Stefan Schmid, and Roger Wattenhofer. A Self-Repairing Peer-to-Peer System Resilient to Dynamic Adversarial Churn. In *Proc. of the 4th Inter-national Workshop on Peer-To-Peer Systems (IPTPS)*, Ithaca, New York, USA, February 2005.

[11] Samuel Madden, Michael J. Franklin, Joseph Hellerstein, and Wie Hong. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *Proc. of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, USA, December 2002.

[12] Samuel Madden, Rober Szewczyk, Michael J. Franklin, and David Culler. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. In *Proc. of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 49-58, Callicoon, New York, USA, June 2002. IEEE Computer Society.

[13] Ji Li, Karen Sollins, and Dah-Yoh Lim. Implementing Aggregation and Broadcast over Distributed Hash Tables. To appear in *ACM Computer Communication Review*, 2005.

[14] Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, Ramesh Govindan, Li Yin, and Fang Yu. Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table. *Mobile Networks and Applications*, 8(4), pages 427-442, August 2003.

[15] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. In *Proc. of 3rd ACM Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL)*, pages 48-55, Seattle, USA, August 1999.

[16] Shun-Yun Hu and Guan-Ming Liao. Scalable Peer-to-Peer Networked Virtual Environment. In *Proc. of ACM SIGCOMM 2004 workshops on NetGames '04: Network and system support for games*, pages 129-133, Portland, Oregon, USA, August 2004.

[17] Takuji Iimura, Hiroaki Hazeyama, and Youki Kadobayashi. Zoned Federation of Game Servers: a Peer-to-peer Approach to Scalable Multi-player Online Games. In *Proc. of ACM SIGCOMM 2004 workshops on NetGames '04: Network and system support for games*, pages 116-120, Portland, Oregon, USA, August 2004.

[18] Filipe Araújo and Luís Rodrigues. GeoPeer: A Location-Aware Peer-to-Peer System. In *Proc. of 3rd IEEE International Symposium on Network Computing and Applications (NCA)*, pages 39-46, Cambridge, MA, USA, August 2004. IEEE Computer Society.

[19] Jörg Liebeherr and Michael Nahas. Application-layer Multicast with Delaunay Triangulations. In *Proc. of IEEE Globecom 2001*, San Antonio, Texas, USA, November 2001.

[20] Björn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins. Peer-to-Peer Support for Massively Multiplayer Games. In *Proc. of the twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, pages 96-107, Hong Kong, China, March 2004.

[21] Joaquín Keller and Gwendal Simon. SOLIPSIS: A Massively Multi-Participant Virtual World. In *Proc. of International Conference on Parallel and Distributed Techniques and Applications (PDPTA 2003)*, CSREA Press 2003, Vol. 1, pages 262-268 ,Las Vegas, Nevada, USA, June 2003.

[22] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass Routing on Geometric Networks. In *Proc. of 11th Canadian Conference on Computational Geometry*, pages 51-54, Vancouver, BC, Canada, August 1999.

[23] Richard Gold. Self-Organizing Route Aggregation for Active Ad-Hoc Networks. In *Proc. of 4th IEEE Conference on Open Architectures and Network Programming (OPENARCH 2001)*, Anchorage, Alaska, April 2001.

[24] Prosenjit Bose and Pat Morin. Online Routing in Triangulations. In *Proc. of the 10th International Symposium on Algorithms and Computation (ISAAC 1999)*, pages 113-122, Chennai, India, December 1999.

[25] Thomas Roos. Voronoi diagrams over dynamic scenes. *Discrete Applied Mathematics*, 43(3), pages 243-259, June 1993.

[26] Gerhard Albers, Leonidas J. Guibas, Joseph S. B. Mitchell, and Thomas Roos. Voronoi Diagrams of Moving Points. *International Journal of Computational Geometry and Applications*, 8(3), pages 365-380, 1998.

[27] Masaaki Ohnishi, Ryo Nishide, and Shinichi Ueshima. Incremental Construction of Delaunay Overlaid Network for Virtual Collaborative Space. In *Proc. of the Third International Conference on Creating, Connecting and Collaborating through Computing (C5'05)*, pages 75-82, Kyoto, Japan, January 2005.

[28] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed Construction of a Planar Spanner and Routing for Ad Hoc Wireless Networks. In *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2002)*, New York, USA, June, 2002.

[29] Godfried T. Toussaint. The Relative Neighborhood Graph of a Finite Planar Set. *Pattern Recognition*, Vol. 12, pages 261-268, 1980.

[30] Xiang-Yang Li and Yu Wang. Quality Guaranteed Localized Routing for Wireless Ad Hoc Networks. In *Proc. of the International Workshop on Mobile and Wireless Networks (MWN 2003)*, Providence, Rhode Island, USA, May 2003.