TKN Telecommunication Networks Group

Technical University Berlin

Telecommunication Networks Group

Scalability of a Distributed Virtual Environment Based on a Selected Structured Peer-To-Peer Architecture

Jiehua Chen, Sven Grottke,
Jan Sablatnig, Ruedi Seiler, Adam Wolisz

{chen, sfs, jon, seiler}@math.tu-berlin.de, wolisz@tkn.tu-berlin.de

Berlin, September 2009

TKN Technical Report TKN-10-003

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

We investigate the scalability of distributed virtual environments (DVEs), which are based on a structured peer-to-peer (P2P) overlay. We are interested in the average network load and routing latency and how they depend on the number of hosts in the DVE. To this end, we study a prototypical DVE consisting of a simple game scenario and a P2P architecture based on Pastry and Scribe as proposed by Knutsson et al. in their important work *"Peer-To-Peer Support for Massively Multiplayer Games"*[3]. Both our theoretical analysis and simulation results show that the network load as well as the routing latency grow $\mathcal{O}(log(N))$, where N is the number of hosts (c.f. figure 6.9 and 6.11).

Our results are in partial contradiction to those given by Knutsson et al. As shown in Fig. 6.9, their experimental results of average message rate are constant for 1000 and 4000 hosts, while ours scale with $\mathcal{O}(log(N))$. We suppose this is due to Knutsson et al. only measuring the subscriber message rate. This is confirmed by our own theoretical analysis and measurement of the subscriber message rate, which display the same $\mathcal{O}(1)$ behavior as seen in Fig. 6.9.

Contents

Chapter 1

Introduction

Distributed virtual environments (DVEs) are simulated systems in which human participants come together in a virtual world and interact with each other. A typical example of this is online games, such as World of Warcraft or Second Life. Each participant in the DVE is associated with a host. The hosts are connected through the Internet. Two important aspects of DVEs are both the consistency of the world views of the participating hosts and the scalability. The focus of this work is the scalability in terms of network load and the routing latency. Since each host contributes its bandwidth in a peer-to-peer (P2P) system, such systems have a better balancing and robustness compared to client-server systems. P2P systems are therefore a good candidate to support distributed virtual environments with thousands of participants. A much cited work on the scalability of P2P-based distributed virtual environments comes from Knutsson et al. In their work "peer-to-Peer Support for Massively Multiplayer Games" of 2004 [3], they proposed a simple P2P approach from existing components, the Pastry [5] overlay and the Scribe [1] application-level multicast together. They have also simulated their self-designed prototype game scenario SimMud and measured the distribution of the message rate, i.e. the number of messages received per second by a host, as well as the routing latency of messages for simulations with 1000 and 4000 participants[1]. In their evaluation, they have expressed that

1. the distribution of the message rate in 1000 and 4000 participants are "very similar" and distributed virtual environments with a growing number of participants are therefore achievable, and

2. the average number of routing hops[2] scales with $log(N)$, where N is the number of participants, because the Pastry routing algorithm determines the routing of a message and ensures that a message is typically delivered within $log(N)$ hops.

We notice three points about Knutsson et al.'s results and evaluation in particular:

1. Knutsson et al. measured only two points of measurement. This is not sufficient to analyze the scaling behavior completely and draw a conclusion about it.

2. In network overlay based on Pastry and Scribe the average message rate should behave similar to the routing latency. One would therefore expect that the message rate scaled

[1] They also measured other aspects, is to not addressed in this work.
[2] Unfortunately the text does not make clear, which kind of routing they meant

logarithmically with the number of participants and not as Knutsson et al. said, have quite similar behavior.

3. Knutsson et al.'s measured average routing latency of multicast messages decreases with the number of participants.

To clear up the ambiguities and contradictions, this thesis has the following objectives:

- The results of Knutsson et al. are to be verified.
- The scalability is to be verified experimentally and analytically.

For this, we reconstructed the experimental setup of Knutsson et al.[3], made more and more extensive measurements and at the end drew a quantitative as well as qualitative comparison with Knutsson et al.'s work.

The most important result is that both the average message rate and the average routing latency of messages scale logarithmetically with the number of participants. There are some discrepancies between our work and the one from Knutsson et al., e.g. the average message rate from Knutsson et al. is similar with different number of participants. These discrepancies can largely be resolved by a reinterpretation of Knutsson et al.'s measurements.

[3]Specifically, the Scribe protocol and the prototypical game scenario SimMud are reconstructed

Chapter 2

Basic Concepts

We explain here some terms that are used in the thesis in more detail.

2.1 Distributed Virtual Environments (DVEs)

Definition 1 (Virtual World (VW)):
A virtual world is a simulated world in which human participants, each represented by his own avatar, can interact with each other. It consists of a set of objects, such as balls or doors, which can be characterized by their state, e.g. position, color, open / closed.

Definition 2 (Distributed Virtual Environment (DVE)):
A DVE is a system of multiple hosts (computers) that communicate over the Internet and together simulate a virtual world.

Definition 3 (Area of Interest (AOI) [4]):
A participant of the DVE may have a limited ability to perceive as a human, and his avatar moves only with a limited speed in the virtual world. Therefore, we define the AOI of an avatar as the field of the virtual world in which the states of the objects it contains are relevant for the participant of this avatar. Typically, the AOI of an avatar is derived from its position in the virtual world.

Definition 4 (Partial Replication [7]):
Hosts in a DVE can replicate the virtual world totally or partially. Partial replication allows at least one host that does not replicate the entire virtual world.

2.2 Pastry: P2P Routing Overlay

Pastry is a structured P2P overlay consisting of a set of hosts on the Internet. Each host is assigned a unique ID. Pastry delivers a message with a given key to the host with the ID numerically closest to this key ID. The average routing latency of a message is $\lceil log_{2^b} N \rceil$ hops, where N is the number of hosts in the Pastry network, and b is a configuration parameter with typical value b = 4. Below you will find a description of the relevant terms and concepts of Pastry. A detailed description of Pastry can be found in [5].

Definition 5 (Pastry Host & ID):
Any host that joins the Pastry network is called a Pastry host and receives a unique k-bit ID[1] called Pastry ID. The Pastry ID is randomly generated. All IDs are equally probable and are between 0 and $2^k - 1$.

Definition 6 (Pastry Host State (Routing Table & Leaf Set)):
Each Pastry host maintains a routing table of $(2^b - 1) \cdot \lceil log_{2^b} N \rceil$ entries and one leafset with ℓ entries, where ℓ is a configuration parameter with typical value of 32. Each entry maps the Pastry ID of a host to its IP address. Pastry host states are used to route messages.

Definition 7 (Pastry Routing Path):
The Pastry routing of a message with the key $KeyID$ is a sequence of nodes (h_0, h_1) $\ldots (h_{n-1}, h_n)$, which describes the routing of this message through Pastry from a sender h_0 to a receiver h_n. h_i is the host who receives the message by the Pastry routing algorithm from host h_{i-1}. The receiver h_n is the host with a Pastry ID that is numerically closest to $KeyID$.

Definition 8 (Pastry Routing Hops):
The number of routing hops via Pastry is the length of the Pastry routing path. A path of length K means, including the sender and receiver, a total number of $K + 1$ hosts.

A Pastry host can use the following routing operations:

route(msg, keyID): A Pastry host may send a message *msg* with $keyID$ using operation *route(msg, keyID)* to a host whose Pastry ID is numerically closest to $keyID$.

send(msg, IP-Addr): A Pastry host can send a message *msg* directly to a Pastry host whose IP address is *IP-Addr*.

2.3 Scribe: Tree Based Application Level Multicast

Scribe is an application level multicast protocol based on Pastry. The original description of Scribe can be found in [1].

Definition 9 (Scribe Host):
Each host that uses the Scribe protocol is known as a Scribe host. Each Scribe host is also a Pastry host.

Definition 10 (Scribe Group and Group ID):
A Scribe group or a multicast group is a set of hosts. Each group is assigned a unique k-bit group ID[2]. All group IDs are mapped to the same ID space like Pastry IDs. In the following the terms Scribe group and multicast group are used interchangeably.

[1] In this work, we use a 32-bit ID to keep the CPU consumption as low as possible, thus enabling simulations with more than 4000 hosts.

[2] In this work we use 32 bits. See also Note 1 on page 6.

Definition 11 (Coordinator of a Scribe Group):
The coordinator of a group is the Scribe host whose Pastry ID is numerically closet to the group ID of this Scribe group.

Definition 12 (Scribe Tree):
A Scribe group is implemented as a multicast tree, which consist of the members of the group and maybe some other hosts. The root of the tree is the coordinator of the associated Scribe group. The tree is formed as the union of the Pastry routing paths of all group members to the root. In the following the terms Scribe tree and multicast tree can be used interchangeably.

Definition 13 (Subscribers of a Scribe Group):
Members of a group are also called subscribers of this group. Since a Scribe group is realized as a Scribe multicast tree, which is formed by an algorithm similar to the "Reverse Path Forwarding" [2], the leaves of the tree are always subscribers. But subscribers are not necessarily always leaves.

Definition 14 (Forwarders of a Scribe Group):
Forwarders of a Scribe group are those hosts of the associated Scribe tree who are not the subscribers of this group[3].

Definition 15 (Scribe Host State (Children List & Parent Host)):
Each Scribe host keeps a list of children hosts and a reference to a parent host for each Scribe group, in which he serves as a subscriber or a forwarder. The Scribe host state is used to realize the tree based application level multicast.

Each Scribe host can use various Scribe functionalities. With *create(groupID)* it may create a new multicast group with group ID *groupID*. With *join(groupID)* a host can join an existing multicast group with group ID *groupID* or leave it by *join(groupID)* again. *multicast(groupID, msg)* offers a host the ability to multicast a message *msg* to all members of an existing multicast group with group ID *groupID*. This works as follows: a multicast message is first sent to the root node (a.k.a. the coordinator) of the corresponding Scribe multicast tree. From there, the message is forwarded to all members within the tree. The description of the realization of these functionalities in this work can be found in appendix B.3 on page 50.

Definition 16 (Scribe Message Types):
A Scribe message with type

SCRIBE_CREATE creates a multicast group.

SCRIBE_JOIN joins an existing multicast group.

SCRIBE_MULTICAST can be sent to all members of the corresponding multicast group.

SCRIBE_LEAVE leaves a multicast group.

[3] In [1] each host of a Scribe tree is defined as the forwarder of this tree.

SCRIBE_MULTICAST_REPLY informs the subscriber to a multicast group of the IP address of the coordinator.

Chapter 3

Starting point of this work: the DVE of Knutsson et al. 2004

Knutsson et al. propose a P2P-based approach to realize distributed virtual environments where thousands of participants can interact with each other. The architecture of their approach is described in more detail in 3.1. To evaluate this approach they have designed a simple game scenario called SimMud (Section 3.2), which they could simulate up to 4000 participants. Their simulation results about scalability are shown in Section 3.3. In Knutsson et al.'s work some details remain unclear, which need to be determined for our own experiments (C.f. chapter 4 on page 18).

Prior to the discussion of Knutsson et al.'s actual approach we explain first some terms more precisely. They come mainly from [3].

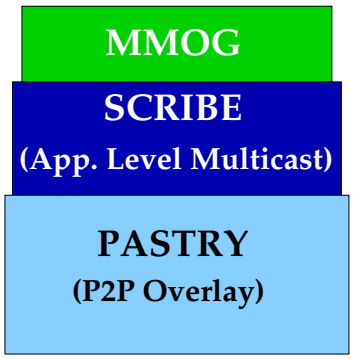
Definition 17 (Massively Multiplayer Online Game (MMOG)):
An MMOG is a special kind of DVE with **thousands** of human participants participating in a virtual **game world**.

Definition 18 (Mutable Objects & Avatars in Virtual Worlds):
Two important components of a virtual world are mutable objects and avatars. The difference between the two is that each avatar is controlled directly by a human participant through his host. Avatars can change their current positions in the virtual world, interact with other avatars, and access mutable objects.

3.1 Knutsson et al.'s P2P Architecuture

The article "*Peer-To-Peer Support for Massively Multiplayer Games*" proposed a P2P architecture is built on top of Pastry [5]. In order to efficiently send messages to a group of hosts it uses also Scribe [1], an application level multicast protocol. This approach can support different application scenarios, such as online games. The properties of the network latency between two hosts and message loss rate are given (C.f. figure 3.1).

Interest management [4] is applied. The whole virtual world is divided into fixed disjoint regions. Each avatar or any mutable object remains at each time in exactly one region. Avatars can change their own regions, mutable objects can't. Each participant is interested in all avatars and mutable objects that reside in the same region as his avatar. His host



[8]

Fig. 3.1: Knutsson et al.'s P2P-Architektur

replicates exactly these avatars and objects. In figure 3.2, for example, 3 regions of a virtual world are displayed. Host α replicate the blue region, namely the avatars A, E and F and apples (mutable objects) G, H, as well as their movements and actions, for his avatar A is currently located in this region. An avatar can only interact with avatars and mutable objects in the same region as itself. In this example, this means that avatar A of host α can move in the blue region, try to change the state of apples G and H, and interact with avatar E or F.

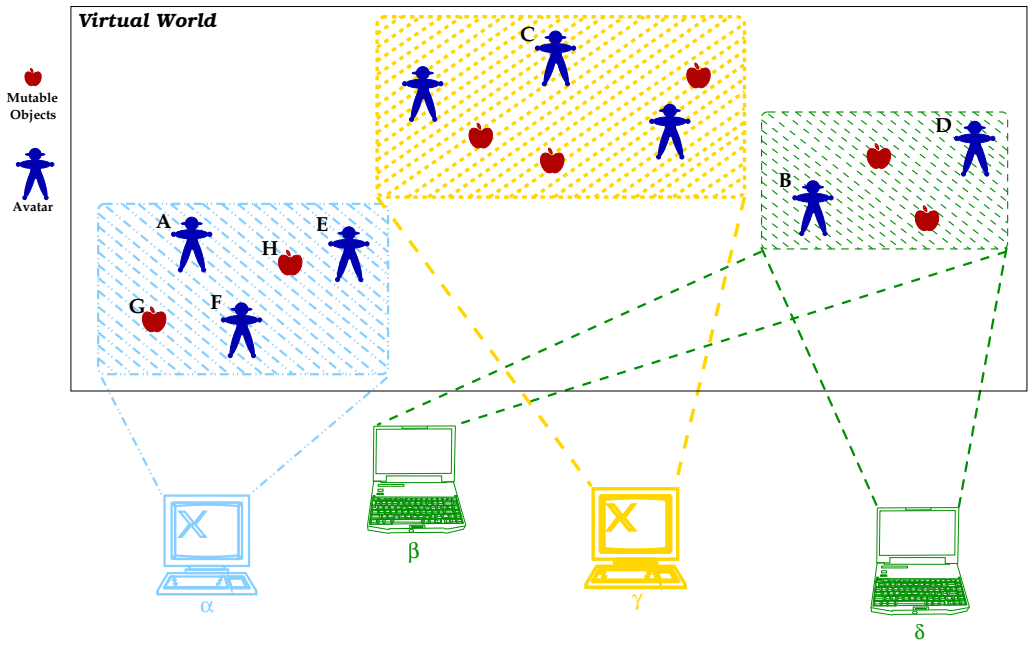


Fig. 3.2: Knutsson et al.'s system model

We discuss our system model in two aspects: the aspect of the virtual world and the aspect of its realization, c.f. table 3.1:

Each region is also assigned a unique group ID and implemented in the real world as a multicast tree. All hosts whose avatars are currently in this region route using Pastry towards

Virtual World	Realization
N avatars	N hosts each running an AI script
G regions	G scribe multicast trees
Avatar changes position	Regular position updates
Avatar changes region	Scribe multicast group change
Avatar object interaction (in the same region)	Client coordinator protocol
Inter-avatar interaction (in the same Region)	Direct communication

Tab. 3.1: A possible MMOG and its realization

the group ID of this region. All these routing paths form a mutlicast tree. The root of this tree is the host, whose Pastry ID is numerically closest to the group ID. This root coordinates the observed region, e.g. resolves conflicting updates. In the following if the terms *coordinator or subscribers or forwarders of a region* is spoken, then the *coordinator or the subscriber and the associated forwarder Scribe group* are meant. Pastry IDs for hosts and regions will be selected randomly and are equally distributed. It may happen that a host is the coordinator for several regions.

Definition 19 (Subscriber Messages):
Messages that a host received are called this host's *subscriber messages*, if they are about the region which this host's avatar is also residing in. They include all multicast messages within the associated Scribe multicast group as well as messages that are routed via Pastry or sent directly to this host.

3.1.1 Avatar Activities

Movements of an avatar within a region are informed to the members of the Scribe group of this region by multicasting the avatar's current position periodically to all members of the Scribe group.

To access mutable objects, the host sends a message including the old and the new state of this object to the coordinator. It determines whether the request is permitted by comparing the old state in the message with its own local copy. If the states match, then it updates its local copy with the new state in the message and multicasts the update to the subscribers to the group through Scribe[1]. Any successful change means therefore each subscriber and the coordinator of the Scribe group receives an object update message.

Inter-avatar interaction in a region is realized through direct communication. Knutsson et al. provide no details on how this communication is exactly implemented.

Each region change of an avatar induces a Scribe multicast group change of this avatar's host in the real world. How this is done exactly is unfortunately not specified in the original work. In particular, we do not know whether the host who wants to change its Scribe group, would tell his old group that it leaves.

[1]This is one of the variants of the realization of the avatar object interaction that Knutsson et al. have discussed in their paper. Although they did not specify their exact implementation, this seems to be easy to implement and is therefore used in this thesis.

3.2 SimMud: A Prototyped Scenario

To test their architecture, Knutsson et al. have designed SimMud. SimMud is an abstraction of a simple online game and covers the activities of an avatar as described in above section. SimMud is simulated on a single computer. One can then measure the network load and message routing latency, which are relevant to the scalability investigation of a DVE.

The food objects are the only mutable objects in SimMud. It is not known how many food objects per region Knutsson et al. have implemented. Each avatar eats, on average, every 20 seconds.

Each avatar interacts ("fights"), on average, every 20 seconds with another avatar in its own region. Knutsson et al. have not indicated how the opponent is selected and how this communication is realized exactly.

Each avatar changes on average every 40 seconds its region, ie. each Scribe host changes his Scribe group on average every 40 seconds.

Bemerkung 20:
In this scenario, it must be decided when an avatar change his current region, eats or fights. Unfortunately, Knutsson et al. haven't specified how these decisions are made, but how often these actions take place only on average.

3.2.1 Analysis of Avatar Activities

Here we discuss how the activities of avatars in SimMud and their realizations in Knutsson et al.'s architecture effect the network load. Unfortunately some details relevant to the analysis are not described in Knutsson et al.'s paper and must therefore be supplemented (see section 4.3.2 on page 20).

Position update Every host multicast every 150 ms the current position of his avatar to all members of its group (i.e. approximately 6.7 messages per host per second). For a region with 10 avatars, this means that each subscriber, the coordinator and each forwarder[2] of the multicast group receives about 66.7 position messages per second. Forwarders or coordinators of a multicast group are usually subscribers of other groups. Therefore, it is expected that each of these hosts receives $66.7 + 66.7 \approx 133.3$ position messages per second.

Food access In a region of 10 avatars, each subscriber (and the coordinator and the forwarders) on average receives $10 \times \frac{1}{20} = 0.5$ food messages per second.

Avatar fighting There is not clear how the fighting is exactly done in Knutsson et al.'s SimMud. In the original work it was merely said that avatars fight on average every 20 seconds. If one assumes that the selection of an opponent is random and the attacker sends his opponent 1 exactly one message, then each host receives an average of $1/20 = 0.05$ fighting messages per second.

[2] See definitions 11, 13 and 14 on page 7.

Region change Each simulated avatar changes his region on average every 40 seconds. Knutsson et al. did not describe how a region change is realized exactly. In any case, the host of an avatar that changes his region receives a message with a description of the new region from the coordinator, ie. the states of the food objects and avatars residing in this region. A host receives therefore on average $\frac{1}{40} = 0.025$ such region information messages per second.

Based on this assessment, it is expected that the position updates dominate the network download per host with over 99% of the messages, which is also the case in Knutsson et al.'s work. See also table 3.2 on page 15.

3.3 Experiments for Scaling Behaviors

3.3.1 Experimental Setup

In investigating the scalability of their P2P approach Knutsson et al. have simulated the scenario SimMud, and using the following experimental setup. Some unclear points are discussed in section 4.3.1 on page 20.

- A network topology is generated randomly, in which the delay between any two hosts is between 3 - 100 ms. Unfortunately in the original work they have not specified no random assignment for the network delay.
- In the original paper it is not specified whether messages can be lost.
- In the experiments considered, there are no hosts joining in or leaving the the system during the simulation.
- The simulation time is 300 seconds.
- Each simulation is performed 5 times each with a different random seed and the datas are averaged at the end.
- In the experiment considered the region density, ie the average number of avatars in a region, is constant.

Definition 21 (d: Region or Group Density):
Let N be the number of avatars or hosts, G the number of regions (Scribe groups) of a system, then $d := N/G$ is the region or group density of this system.

Knutsson et al. used version 1.1 of FreePastry, an open source version of Pastry and Scribe, for their simulation. A number of FreePastry releases is available on the homepage[3].

[3]See also `http://www.freepastry.org/FreePastry/` .

3.3.2 Results

Using the experimental setup described in Section 3.3.1 Knutsson et al. have measured for 1000 and 4000 simulated avatars, each with 100 and 400 regions, the average number of messages received per second per host and the number of routing hops per unicast message and multicast per message respectively. Unfortunately, they did not indicate which messages are unicast or multicast messages in their paper.

The figures 3.3, 3.4 and 3.5 are extracted from the article [3]. They show the distribution of received message rate (# messages / s) and the distribution of the number of unicast routing hops as well as the distribution of the number of multicast routing hops from simulations with 1000 hosts and 100 regions and 4000 host and 400 regions respectively. The average region density d in both cases is constant (10 avatars / region).

The following text are their original conclusions:

> ... for an average of 10 players per region, the results for 1000 players and 4000 players are quite similar. We can thus conclude that as long as average region density is kept constant, increases in player population can be handled.

They commented about the routing latency:

> The message delay largely depends on the underlying overlay routing. Pastry routing will typically route a message within $log(N)$ hops, where N is the total number of nodes. This means that routing times will increase with population size, but only very slowly.

(See Chapter VII, Section C in [KLXH04].)

Now we investigate their three results in more details.

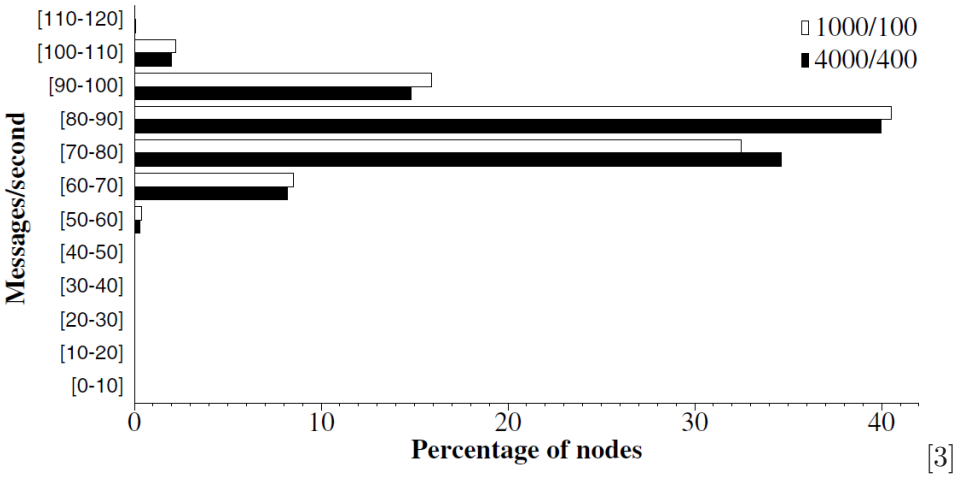


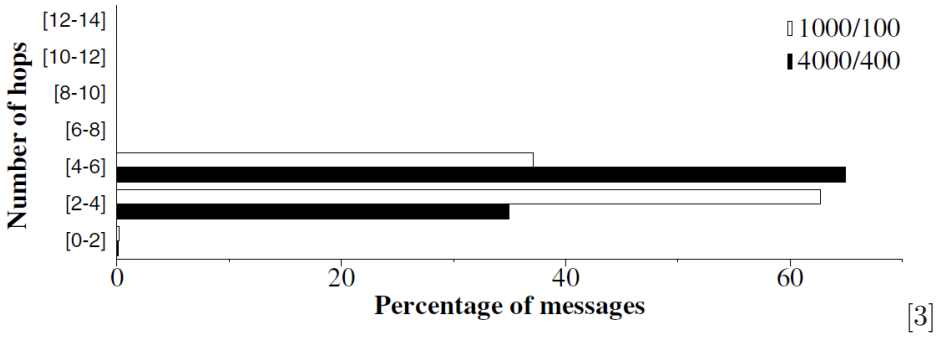
Fig. 3.3: Knutsson et al.'s distribution of message rate

Figure 3.3 shows that in both cases more than 70% of the hosts receive between 70 and 90 messages per second. Not more than 1% of all hosts receive the maximum message rate from 110 to 120 messages per second. The average number of messages received per second

per host, position updates received per second per hosts or mutable object updates received per second per host for simulations of 1000 hosts and 4000 hosts are recorded in table 3.2. As you can see here, about 99% of messages received are position update messages. Besides the fact that two points of measurements are not sufficient to observe the scaling behavior of the network load, the average message rate from Knutsson et al. are almost the same except for the decimal places.

#hosts/#groups	1000/100	4000/400
avg. # received msgs/s/h	82.17	82.12
avg. # received position/s/h	80.33	80.18
avg. # received objects/s/h	0.84	0.95

Tab. 3.2: Knutsson et al.'s different message rates (#/s/host)



[3]

Fig. 3.4: Knutsson et al.'s distribution of # of routing hops of **unicast** messages

Figure 3.4 shows the distribution of the number of routing hops of unicast messages. Practically all unicast messages are delivered within 6 hops. Since there is no record of the average number of routing hops of unicast messages shown in the original paper, we estimate it from the distribution of the figure. It results in a value of 3.7 hops per unicast message for the simulation with 1000 avatars and 4.3 hops per unicast message for the simulation with 4000 avatars.

Figure 3.5 shows the distribution of the number of routing hops of multicast messages. Similar to the case of unicast messages are most of the multicast messages delivered within 6 hops. There is also a small part (1%) of multicast messages that are routed in more than 18 hops. This is how Knutsson et al. have described, caused by the use Scribe of the FreePastry implementation. Similar to unicast, we estimated the average number of routing hops of all multicast messages from figure 3.5 because the original paper does not specify this value. It results in a value of 1.5 Hops per multicast message for the simulation with 1000 avatars and 1.4 Hops per multicast message for the simulation with 4000 avatars.

3.4 Discussion of Knutsson et al.'s Results

This section discusses some inconsistencies and unclear points of Knutsson et al.'s paper. A comparison between Knutsson et al.'s and our own simulation results are to be found in

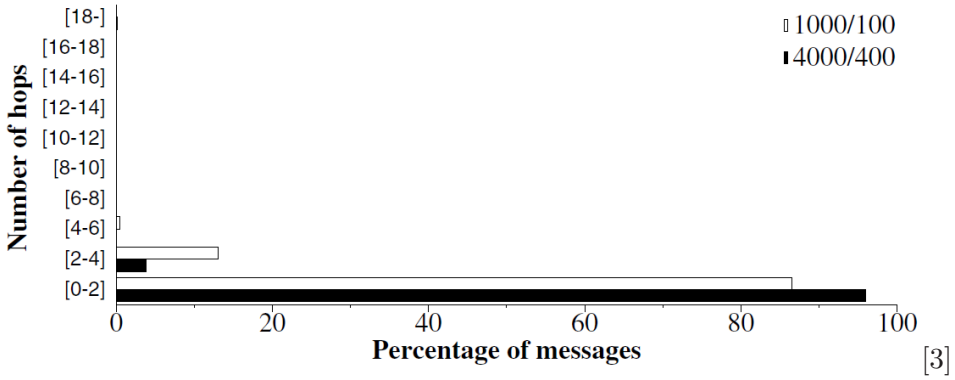


Fig. 3.5: Knutsson et al.'s distribution of # of routing hops of **multicast messages**

chapter 7, starting at page 42.

The results in Knutsson et al.'s paper on the one hand agrees partly not with our own expectations, which were formulated in section 3.2.1. On the other hand Knutsson et al. concluded a logarithmic increase in the message routing latency with the number of hosts without a specification of how the routing latency of a message is measured.

In the following these two points are elaborated. We propose one possible cause of the discrepancy between our expectation and the simulation results of Knutsson et al.. in section 3.4.1. In section 3.4.2 we formulate some possible approaches, how the message routing latency can be measured. They are necessary condition to interpret and verify Knutsson et al.'s results.

3.4.1 About the Network load

There is a discrepancy between our expected distribution of message rate received by a host and Knutsson et al.'s measured distribution of message rate received by a host (figure 3.3). For an average group density of 10 hosts per group we expect that approximately 10% of hosts are coordinators. A coordinator should receive at least 133.3 messages per second if he coordinates a group of 10 subscribers. Due the average density of 10 hosts / group there should also be some hosts with message rate of at lease 133.3 messages per second. Knutsson et al. but have measured that no more than 1% of the hosts receive the maximum message rate of 110-120 messages / s, which implies no host in the simulation receives at least 133.3 messages / s. This indicates that Knutsson et al. have not considered all types of messages in the measurement of message rate distribution. In particular, messages that sent to a coordinator would not have been shown in figure 3.3.

3.4.2 About the routing latency

Knutsson et al. have measured the routing latency of unicast messages and multicast messages respectively. Unfortunately, definition of neither unicast nor multicast messages is given in their paper, so we can only make some conjecture based on the figures given. Moreover, it is not clear how the routing latency of multicast messages is measured.

1. **Conjecture about the unicast messages**: Only messages that are routed via Pastry are taken into account but not the messages sent directly between two hosts.

   We can give an indirect argument here: Assume that the messages sent directly are also taken into account. A message sent directly costs only 1 hop. As shown in figure 3.4, there are less than 1% of messages delivered within 1 hop. The vast number of position updates multicasted via Scribe are firstly sent by a subscriber to its coordinator directly[4]

   Of the remaining messages there are on average 2 fighting messages ($\approx 0.7\%$), 2 mutable object messages and 1 message relevant to region change in a times pan of 40 seconds, where fighting messages are sent directly. Even if region change messages and mutable object messages would always be routed through Pastry, it is considerably more than 1% of the messages (100% - 0.7% - 0.7% = 98.6%) sent directly. If messages sent directly are considered as unicast messages, then the distribution in figure 3.3 should have shown more than 1% messages delivered within 1 hop.

2. **Presumption of multicast messages**: There are two ways to measure the number of routing hops of multicast messages.

   1st variant The average number of routing hops of a multicast message can be measured starting from the original sender-subscriber to the whole subscribers of the Scribe group.

   2nd variant The average number of routing hops of a multicast message can also be measured starting from the coordinator to the whole subscribers of its Scribe group.

   Unfortunately, Knutsson et al. have not specified in their paper, how they have measured the number of routing hops of multicast messages.

   In any case, one would expect that the distributions of the number of routing hops of unicast messages and multicast messages are similar, since the unicast messages are routed through Pastry and multicast messages through Scribe, and the multicast of Scribe follows the reverse routing path of Pastry. This expectation can unfortunately not be confirmed by Knutsson et al.'s measurements, see figures 3.4 and 3.5. In addition, Knutsson et al. have only measured for only two different numbers of hosts (1000 and 4000). The average number routing hops of multicast messages for 4000 hosts is slightly lower than the one for 4000 hosts, which can not **confirm** their presumed logarithmetic behavior of routing latency (See Chapter VII, Section C in [3].).

[4] See in section 2.3, how Scribe multicast works.

Chapter 4

Architecture & Model

4.1 P2P Architecture

As mentioned earlier, we use Knutsson et al.'s proposed architecture and the game scenario SimMud to investigate the scalability of the P2P based distributed virtual environments. We use our simulator, **Adam** [6], and implement on top of it the Pastry and Scribe routing protocols. Then we can simulate our implemented SimMud scenario.

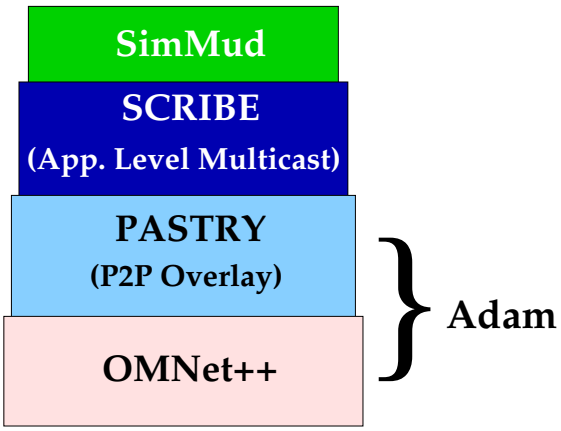


Fig. 4.1: Our architecture

4.2 Model

We use Knutsson et al.'s system model and experimental setup as defined in chapter 3. Some design details, that are not described in their work, but important for the experiments and for the understanding of their experimental results, need our special observation. And this will be found in section 4.3.

In a system with N participants, the virtual world of SimMud so will be divided in G fixed, disjoint regions, the average region density is constant ($= N/G$). The number of food objects (mutable objects) in each region is equal to N/G.

Figure 4.2 shows a SimMud game with 12 avatars, and 6 regions, and a subset of hosts involved replicating part scene of the virtual world. The average number of avatars in a region

is 2, and in every region there are 2 apples as the only food objects. Host α replicates the left upper region, because his avatar A is currently in this region. It has also a repository of the right upper region because it is coordinator for that region. If avatar C of host γ wants to eat apple D, γ must make a request on host α. If avatar B now enters the right upper region, its host β notifies the coordinator of the middle lower region of his leave and the coordinator α of his join. α then sends β the current state including the avatars and the apples of the right upper region. The hosts of other avatars in the same region is also informed of B's arrival. If avatar B wants to fight with the other avatar in the same region, then he simply sends a fighting message to the host of the opponent.

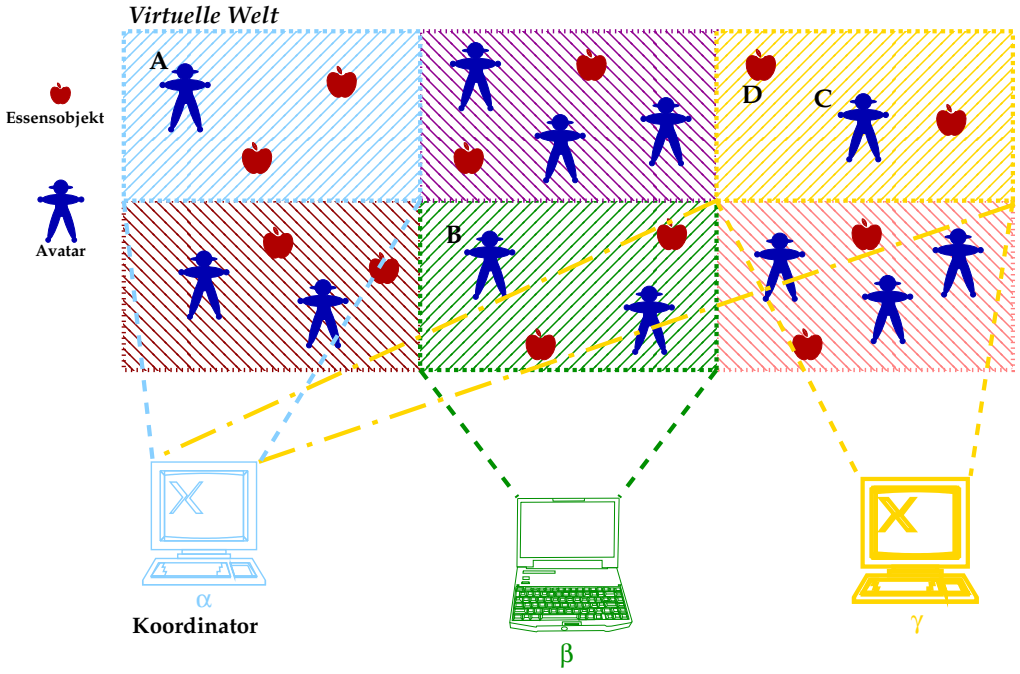


Fig. 4.2: A possible SimMud illustration

Definition 22 (a tick of the simulation):
Avatars can only change their behavior at fixed times. The constant interval between two consecutive times is known as one tick of the simulation.

4.3 Experimental Setup

The SimMud scenario will be simulated on the simulator **Adam**. The messages received by a host and the number of routing hops of a message will be measured. We define in the following the entire experimental setup and clarified some unclear details of the original paper [3].

4.3.1 General Experimental Setup

1. The network topology is randomly created such that the network delay between any two hosts is between 3 to 100 ms with a uniform distribution.
2. There is no message loss.
3. During the game no hosts fall out of or additionally get into the system.
4. A tick is 150 ms.
5. Each simulation lasts 300 s.
6. Each experiment is simulated 5 times.

4.3.2 Construction of SimMud

1. Each simulated host sends every 150 ms, ie once per tick, an update message of its avatar's current position (a.k.a position update message) to its Scribe Group.
2. If the region change is on, an avatar changes in each tick its each region with probability $0.15/40 = 0.00375$, ie on average every 40 seconds. If an avatar changes its region, its host informs the multicast groups of both the old and the new region. The coordinator of the multicast group of the old region informs then other members of the leave. And the coordinator of the multicast group of the new region sends the joiner the region information, ie the states of the avatars and the food objects in the region, and informs the new group's subscribers of the coming of the new joiner.
3. If the food access is turned on, each host tries in each tick with probability $0.15/20 = 0.0075$ (on average every 20 seconds) to access a random food object that is also in the same region as its avatar.
4. When fighting is on, each host selects at every tick with probability $\frac{0.15}{20} = 0.0075$ (on average every 20 seconds) a random avatar in the same region as its own avatar. Each avatar in this region is selected with equal probability. Since Knutsson et al. did not specify exactly how the communication takes place during a fight, we let the host, who wants to fight, sending the selected avatar's host a direct fighting message.
5. To measure the latency of multicast messages, we measure the number of hops starting from the coordinator, ie the second variant of the proposed possible counting approaches in item 2 of subsection 2 is used. If for example the length of the path from the root of a Scribe tree to a member of the Scribe group is l, then this multicast message needs l hops from the coordinator to the member (subscriber).
6. A unicast message is defined as the message that is routed through Pastry[1]. For the measurement of the number of routing hops of unicast messages, following messages are considered:

[1] See the Pastry API in Section 2.2 on page 5.

(a) For the first 500 ms after each region change by a host, messages that should be multicasted through Scribe also use the Pastry routing protocol. Each of these multicast messages causes therefore an additional unicast message.

(b) If a host wants to join a Scribe group, it routes a SCRIBE_JOIN message through Pastry. This kind of message is regarded as a unicast message.

4.3.3 Received Message Types

We distinguish the following types of messages that can receive a host (Figure 4.3):

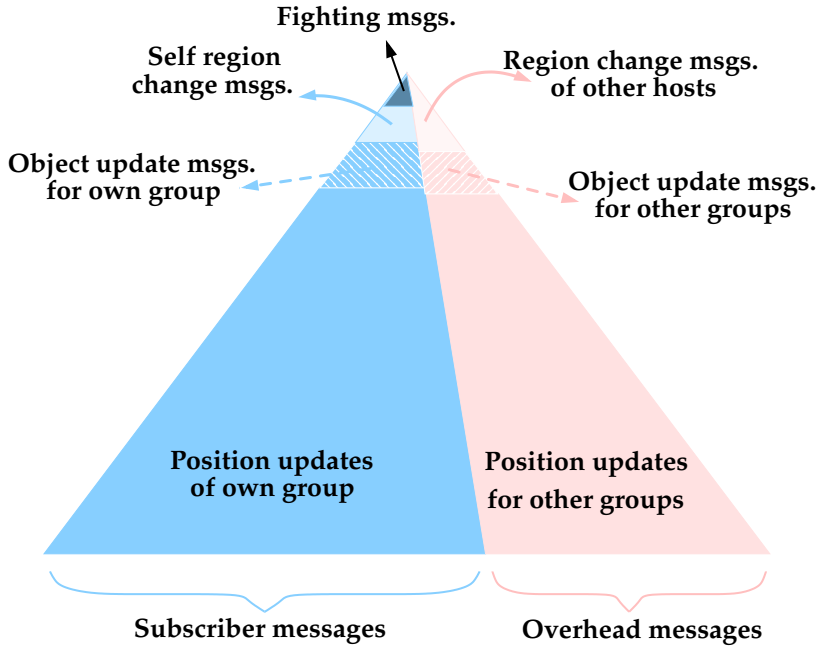


Fig. 4.3: Received Message Types

Messages Relevant to a Host as a Subscriber (Figure 4.3 left)

1. Position update messages of own region.
2. Object update message of own region.
3. Messages that are relevant to region change of avatars in the same region as my avatar: e.g. region information or message that saying an avatar is leaving.
4. Direct fighting messages.

Messages for a host as a forwarder or coordinator (Figure 4.3 right)

1. Position update messages of other regions.

2. Object update messages of other regions.

3. Messages that are relevant to region change of avatars in other regions as my avatar.

In the following some important measurements are defined (Table 4.1). Be the simulation time T. Only the received network load is taken into account.

Subscriber message rate:	$\frac{\lvert\text{Subscriber messages}\rvert}{T}$
Subscriber position message rate:	$\frac{\lvert\text{Subscriber messages}\cap\text{Position messages}\rvert}{T}$
Total message rate:	$\frac{\lvert\text{Total messages}\rvert}{T}$
Total position message rate:	$\frac{\lvert\text{Total position messages}\rvert}{T}$

Tab. 4.1: Def: Message rate of different message types

In particular the following relation counts:

Subscriber position messages	$\subset$	Subscriber messages
$\lvert\cap$		$\lvert\cap$
Total position messages	$\subset$	Total messages

Chapter 5

Theoretical Analysis

As we know from the analysis of the network download in section 3.2.1 on page 12, the position update messages have brought up the most of the network download traffic. Each multicast group has a different number of members and even more this number can still change if the avatars are allowed to change their regions. The number of subscriber position messages [1] that a host receives depends on the number of subscribers of the multicast group that this host is also subscribing. We discuss in this chapter how many position messages that a host can receives. To convert this to the position message rate, you must take into account the tick time of the simulation. We turn off the food access and the fighting of an avatar to better concentrate on the position messages, the most of the network download traffic.

In section 5.2, we describe the theoretical model for the simplified case, where region changes, food access, and avatar fighting are turned off and then analyze the distribution of the number of subscriber position messages by a host and the distribution of the number of position messages that a host totally receives. Section 5.3 analyzes the two distributions for the case with region changes. Before we start with the analysis, we look into an example (section 5.1) to see how the distribution of region size and the distribution of the number of subscriber position messages would be.

5.1 An Example

Assume a game with 6 avatars and 3 regions, ie with 6 hosts and 3 multicast groups. Each host chooses a region to let his avatar joining in, with each region having the same probability (=1/3) to be chosen. One possible case (see figure 5.1) would be, 5 avatars are joining in a region, 0 avatar in another region, and 1 avatar in the third region. The probability of this case equals $\frac{6!}{5!0!1!} \cdot (\frac{1}{3})^6 \approx 0.82\%$, because there are a total of $\frac{6!}{5!0!1!}$ possibilities (see figure 5.2) to group the avatars so that 5 of them in the 1st, 0 in the 2nd, and 1 in the 3rd region, and each of these cases has the same probability $(\frac{1}{3})^6 \approx 0.14\%$.

Since all regions are equal, so the probability of a region having totally 5 avatars equals the probability of one of the 3 regions having 5 avatars and one of the remaining 2 regions having 1 avatar:

[1] Subscriber position messages are subscriber messages and about the positions of avatars. See also def:19 for the definition of subscriber messages on page 11.

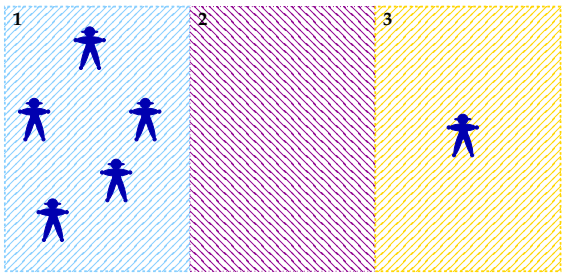


Fig. 5.1: One simple SimMud game with 6 avatars and 3 regions

$$\begin{aligned} h(k=5) &= P(\text{The 1st region has 5 avatars}) \\ &= \frac{6!}{5!0!1!} \cdot (\frac{1}{3})^6 + \frac{6!}{5!1!0!} \cdot (\frac{1}{3})^6 \\ &= 12 \cdot (\frac{1}{3})^6 \\ &\approx 1.6\% \end{aligned} \tag{1}$$

Each region can have 5 avatars with $h(5) \approx 1.6\%$ probability. Since there are 3 regions, on average there will be $3 \cdot 1.6\% = 0.048$ regions each having 5 avatars. This follows $5 \cdot 0.048 = 0.24$ avatars, on average, each of which is in a 5-er region. There are a total of 6 avatars in this example, so the probability that a given avatar in a 5-group is equal $\frac{0.24}{6} = 4\%$. A host receives as many subscriber position messages as the number of avatars of the region, which his avatar also resides in. So the probability that a host receives 5 subscriber position messages equals the probability of an avatar being in a 5-er region, which is in this case equal to 4%.

5.2 Probabilistic model of a host receiving subscriber position messages

Below we consider a situation, which can be derived from the well-known urn model. In a system of totally N avatars and G regions, there are equivalently N hosts. At the beginning each avatar chooses anyone of the G regions to join in, so the avg. region density is $d := \frac{N}{G}$ avatars per region. If there are k avatars in a region, then the host of each of these k avatars receives also k subscriber position messages of this region. In this model, we only investigate the position messages received by a host.

There are a total of G^N ways to divide the N avatars in the G regions. The probability that an avatar selects a particular region is equal $\frac{1}{G} =: p$.

Let k_i with $i \in \{1...G\}$ the number of avatars having chosen the region i. Then we call $\omega := (k_1, ..., k_G)$ with $\sum_{i=1}^{G} k_i = N$ a possible selection event [2]. This probability equals

$$p(\omega) = p(k_1, ..., k_G) = \frac{N!}{k_1! \cdots k_G!} \cdot \prod_{i=1}^{G} p^{k_i} = \frac{N!}{k_1! \cdots k_G!} \cdot p^{\sum_{i=1}^{G} k_i} = \frac{N!}{k_1! \cdots k_G!} \cdot p^N \tag{2}$$

[2] C.f. *multinomial distribution.*

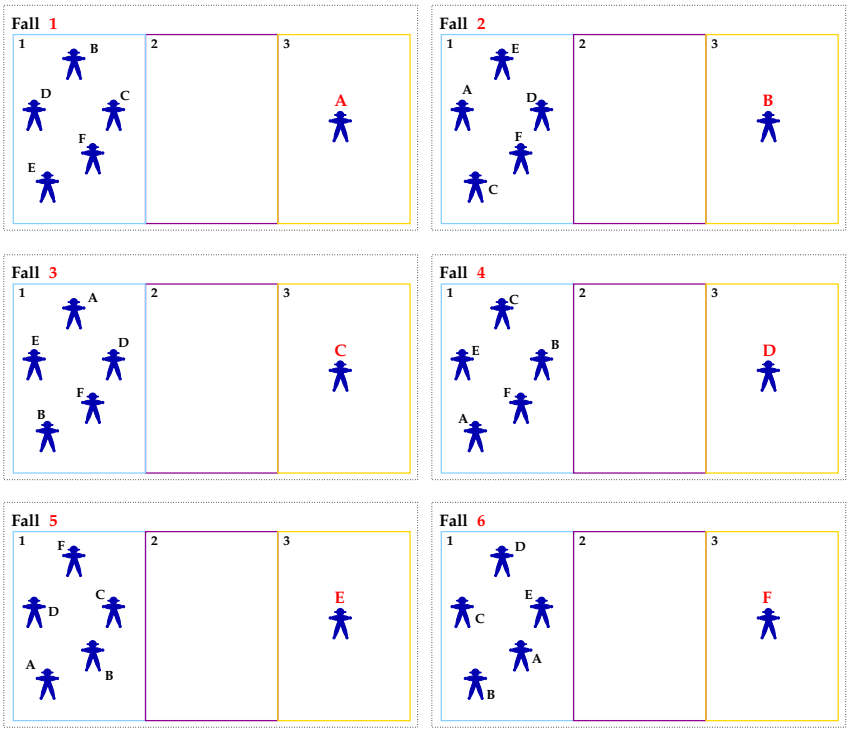


Fig. 5.2: All possibilities for the case of figure 5.1

Definition 23 (Probability of a region having k avatars):
Since all groups are equal, w.l.o.g, the probability $h(k)$ of the 1 region containing k avatars will be investigated.

$$
\begin{aligned}
h(k) &:= \sum_{\sum_{i=2}^{G} k_i = N-k} p(k, k_2, \ldots, k_G) \\
&\overset{(2)}{=} \sum_{\sum_{i=2}^{G} k_i = N-k} \frac{N!}{k! \cdot k_2! \cdots k_G!} \cdot p^N \\
&= p^k \frac{N!}{k! \cdot (N-k)!} \sum_{\sum_{i=2}^{G} k_i = N-k} \frac{(N-k)!}{k_2! \cdots k_G!} \cdot p^{N-k} \\
&\overset{\text{Multinomial theorem}}{=} p^k \cdot \binom{N}{k} \cdot \underbrace{(p + \ldots + p)}_{(G-1)\times}{}^{N-k} \\
&\overset{p=\frac{1}{G}}{=} p^k \cdot (1-p)^{(N-k)} \cdot \binom{N}{k} \qquad (3)
\end{aligned}
$$

Each region contains with probability $h(k)$ exactly k avatars. Then the average number of regions each having k avatars is equal to $G \cdot h(k)$:

$$H(k) := G \cdot h(k) \overset{(3)}{=} p^{(k-1)} \cdot (1-p)^{(N-k)} \cdot \binom{N}{k} \qquad (4)$$

5.2.1 Probability of the number of subscriber position messages that a host receives

If each of these $H(k)$ regions has k hosts, then $k \cdot H(k)$ avatars of totally N avatars are in a k-er region, The probability that an avatar is in such a region equals $\frac{k \cdot H(k)}{N}$, which is also the probability $k(k)$ that a host receives k subscriber position messages.

$$s(k) := \frac{k \cdot H(k)}{N} \overset{(4)}{=} \frac{k}{N} \cdot p^{(k-1)} \cdot (1-p)^{(N-k)} \cdot \binom{N}{k} = p^{k-1} \cdot (1-p)^{(N-k)} \cdot \binom{N-1}{k-1} \quad (5)$$

The distribution of the number of subscriber position message received by a host can be plotted for a system with 6 hosts (=avatars) and 3 regions and shown in figure 5.3:

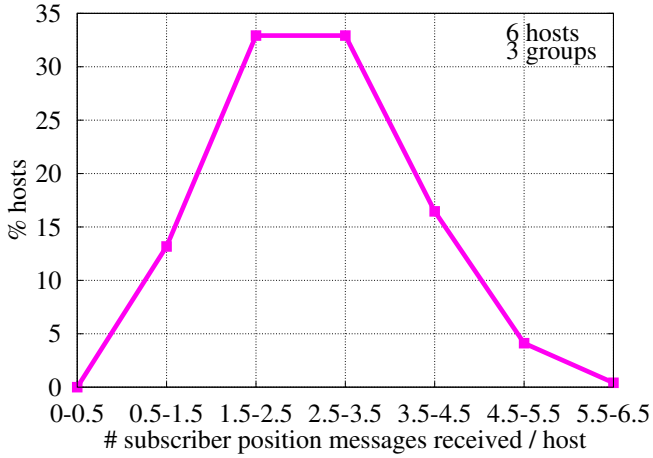


Fig. 5.3: Ex: distribution of # subscriber position messages received by a host

The expectation of the number of subscriber position messages received per host is:

$$\begin{aligned}
E &= \sum_{k=0}^{N} k \cdot s(k) \\
&\overset{(5)}{=} \sum_{k=0}^{N} k \cdot p^{k-1} \cdot (1-p)^{(N-k)} \cdot \binom{N-1}{k-1} \\
&= \sum_{k=1}^{N} (k-1+1) \cdot p^{k-1} \cdot (1-p)^{(N-k)} \cdot \binom{N-1}{k-1} \\
&= \sum_{k=1}^{N} (k-1) \cdot p^{k-1} \cdot (1-p)^{(N-k)} \cdot \binom{N-1}{k-1} \\
&\quad + \underbrace{\sum_{k=1}^{N} p^{k-1} \cdot (1-p)^{(N-k)} \cdot \binom{N-1}{k-1}}_{=(p+1-p)^{(N-1)}=1 \text{ (Binomial theorem)}}
\end{aligned}$$

$$
\begin{aligned}
&= p \cdot (N-1) \cdot \underbrace{\sum_{k=2}^{N} (p^{k-2} \cdot (1-p)^{N-k} \cdot \binom{N-2}{k-2})}_{=(p+1-p)^{(N-2)}=1 \text{ (Binomial theorem)}} + 1 \\
&= p \cdot (N-1) + 1 \\
&\overset{p=\frac{1}{G},\, d=\frac{N}{G}}{=} d - \frac{d}{N} + 1 \qquad (6)
\end{aligned}
$$

With increasing number of N and constant region density d the expectation of number of subscriber position messages received per host converges to $d + 1$.

5.2.2 Probability of the number of position messages received by a host

Since a host can apart from being a subscriber for a particular multicast group also be a forwarder or a coordinator for another multicast group, it receives additional position messages that don't come from its own subscribing group. This means if an avatar is in a x-er region, and his host is forwarder for a multicast group with a total of y subscribers, then this host receives totally $x + y$ position messages. It depends on the underlying Scribe and Pastry algorithm, if a host is a subscriber or a coordinator or a forwarder for a multicast group. If a host is a forwarder or a coordinator for just one additional Scribe group, then we expect that it receives on average twice as many position messages as a host, who is neither forwarder nor coordinator for any other multicast groups. Should a host be involved in several other multicast groups, the number of position messages received by it multiply accordingly. If the distribution of the number of additional multicast groups that a host is not subscribing but forwarder or coordinator for, is known, then you can convolute it with the theoretical analyzed distribution of the number of subscriber position messages. Under the assumption that the number of position messages received by a host does not correlates with the number of additional groups that a host in involved in [3], the resulting convoluted distribution should correspond to the distribution of the number of position messages.

Supposed that the distribution of the number of additional multicast groups that a host is additionally involving in is given in figure 5.4. Then 39% of the total hosts are not involving in any other groups, 1% of the total hosts in 1 additional group and the remaining 60% of the total hosts in 2 additional groups. If we convolute it with the theoretically analyzed distribution of the number messages (figure 5.3), then we obtain a probabilistic model for the distribution of position messages received by a host. This convolution is shown in figure 5.5.

One could certainly analyze theoretically what the probability it is that a host is involved in a total of $0, 1, \ldots, G$ additional groups. This theoretical analysis is part of the work of Pastry. In this thesis, the network load of the P2P-based distributed virtual worlds should be analyzed depending on the number of participating hosts and not

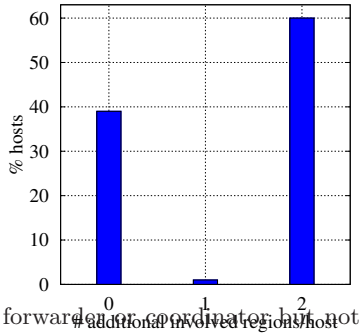
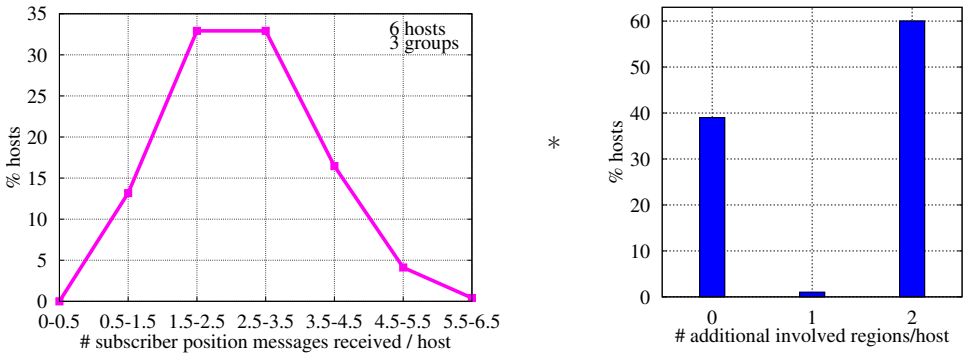


Fig. 5.4: Ex: distribution of # additional involved groups

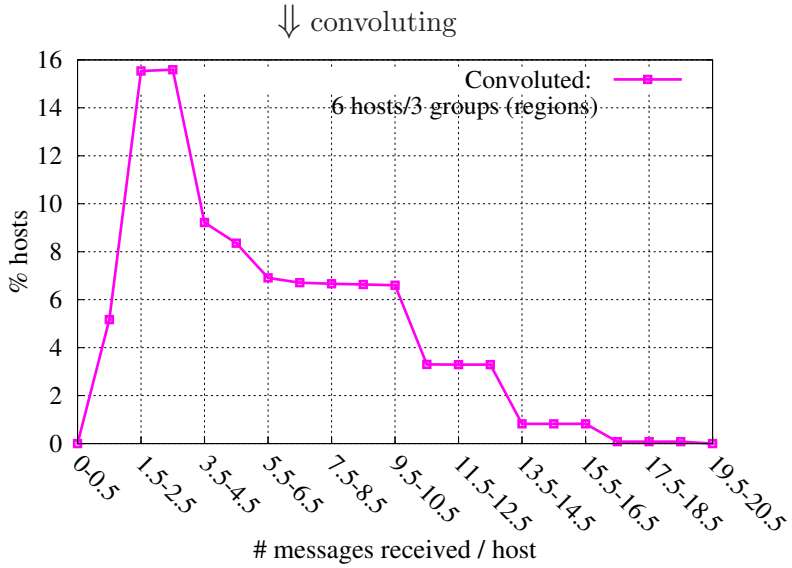
[3] A host is additionally involved in a multicast group, if it is forwarder or coordinator but not subscriber for this group.

the internals of the overlays used. So we measure the number of multicast groups that a host is additionally involved in. This allows us to figure out the distribution of the number of additional involved groups by a host and convolute it with the distribution from the theoretical model in section 5.2.1 on page 26.



(a) Distribution of # subscriber position msgs

(b) Distribution of # add. involved groups



(c) Theoretically convoluted distribution of # position msgs

Fig. 5.5: Ex: convolution of theoretically analyzed distribution of # msgs & distribution of # add. involved groups

5.3 Probabilistic model under region changes

Avatars in our simplified probabilistic model detailed in subsection 5.2 don't ever change their regions. Now we suppose a more compact situation, where each avatar can change his

region some times. If an avatar moves from region i with $\hat{k_i}$ members to region j with $\hat{k_j}$ members (after the join), his host receives firstly $\hat{k_i}$ A-messages and then $\hat{k_j}$ A-messages. We know from subsection 5.2.1 that a host receives k subscriber position messages for the case without region change with probability $s(k)$. Then the probability that it firstly receives $\hat{k_i}$ and then $\hat{k_j}$ subscriber position messages is thus equal to $s(\hat{k_i} \cdot \hat{k_j})$.

A Subscriber receives $\hat{k}_i$ position messages probability $s(\hat{k}_i)$ for all $i \in \{1...Y\}$ while he remains in a group. Then the probability $(s_Y(\hat{k}))$, that it with Y - 1 region change overall receives $\hat{k}$ position messages equal to the probability that it consecutively receives $\hat{k}_1$, ..., $\hat{k}_Y$ position messages with $\hat{k}_1 + \cdots + \hat{k}_Y = \hat{k}$. I.e.

$$s_Y(\hat{k}) = \sum_{\sum_i^Y \hat{k}_i = \hat{k}} s(\hat{k}_1) \cdots s(\hat{k}_Y) \tag{7}$$

5.3.1 One Region Change

If region changes are uncorrelated, the probability can be easily calculated by convolution. We investigate firstly the case of a region change. It is noteworthy that these convolutions can be compute explicitly:

$$\begin{aligned}
s_2(\hat{k}) &:= \sum_{\hat{k}_1+\hat{k}_2=\hat{k}} s(\hat{k}_1) \cdot s(\hat{k}_2) \\
&\overset{(5)}{=} \sum_{\hat{k}_1+\hat{k}_2=\hat{k}} p^{\hat{k}_1-1} \cdot (1-p)^{(N-\hat{k}_1)} \cdot \binom{N-1}{\hat{k}_1-1} \cdot p^{\hat{k}_2-1} \cdot (1-p)^{(N-\hat{k}_2)} \cdot \binom{N-1}{\hat{k}_2-1} \\
&= \sum_{\hat{k}_1+\hat{k}_2=\hat{k}} p^{\hat{k}_1+\hat{k}_2-2} \cdot (1-p)^{2N-(\hat{k}_1+\hat{k}_2)} \cdot \binom{N-1}{\hat{k}_1-1} \cdot \binom{N-1}{\hat{k}_2-1} \\
&= \sum_{\hat{k}_1=1}^{\hat{k}-1} p^{\hat{k}-2} \cdot (1-p)^{2N-\hat{k}} \cdot \binom{N-1}{\hat{k}_1-1} \cdot \binom{N-1}{\hat{k}-1-\hat{k}_1} \\
&= p^{\hat{k}-2} \cdot (1-p)^{2N-\hat{k}} \cdot \sum_{\hat{k}_1=1}^{\hat{k}-1} \binom{N-1}{\hat{k}_1-1} \cdot \binom{N-1}{\hat{k}-1-\hat{k}_1} \\
&\overset{\hat{k}_1' := \hat{k}_1 - 1}{=} p^{\hat{k}-2} \cdot (1-p)^{2N-\hat{k}} \cdot \sum_{\hat{k}_1'=0}^{\hat{k}-2} \binom{N-1}{\hat{k}_1'} \cdot \binom{N-1}{\hat{k}-2-\hat{k}_1'}
\end{aligned} \tag{8}$$

From the Vandermonde identity, we know that

$$\binom{m+n}{r} = \sum_{k=0}^{r} \binom{m}{k} \cdot \binom{n}{r-k} \tag{9}$$

The equation 9 can be inserted into the formula 8 $(s_2(\hat{k}))$ for the probability that a host receives a total of k position messages if its avatar has changed his region 1 time. It yields:

$$s_2(\hat{k}) \overset{(9)}{=} p^{\hat{k}-2} \cdot (1-p)^{2N-\hat{k}} \cdot \binom{N-1+N-1}{\hat{k}-2}$$

$$= \quad p^{\hat{k}-2} \cdot (1-p)^{2N-\hat{k}} \cdot \binom{2 \cdot N - 2}{\hat{k}-2} \tag{10}$$

5.3.2 Y-1 Region Changes

The probability that a host receives k subscriber position messages if its avatar has changed his region Y-1 times leads to a multiple convolution.

Satz 24 (Probability of a host receiving $\hat{k}$ subscriber position messages under Y-1 region changes):

$$s_Y(\hat{k}) = p^{\hat{k}-Y} \cdot (1-p)^{Y \cdot N - \hat{k}} \cdot \binom{Y \cdot N - Y}{\hat{k}-Y} \tag{11}$$

We can prove the formula (11) by induction, the proof is to be found in appendix 25 on page 48.

The expectation of the number of subscriber position messages received per host in this case:

$$\begin{aligned}
E' &= \sum_{\hat{k}=0}^{Y \cdot N} \hat{k} \cdot s_Y(\hat{k}) \\
&\overset{(11)}{=} \sum_{\hat{k}=0}^{YN} \hat{k} \cdot p^{\hat{k}-Y} \cdot (1-p)^{Y \cdot N - \hat{k}} \cdot \binom{Y \cdot N - Y}{\hat{k}-Y} \\
&= \sum_{\hat{k}=Y}^{Y \cdot N} (Y + \hat{k} - Y) \cdot p^{\hat{k}-Y} \cdot (1-p)^{Y \cdot N - \hat{k}} \cdot \binom{Y \cdot N - Y}{\hat{k}-Y} \\
&= Y \cdot \underbrace{\sum_{\hat{k}=Y}^{Y \cdot N} \cdot p^{\hat{k}-Y} \cdot (1-p)^{Y \cdot N - \hat{k}} \cdot \binom{Y \cdot N - Y}{\hat{k}-Y}}_{=1 \text{ (Binomial theorem)}} \\
&\quad + \sum_{\hat{k}=Y}^{Y \cdot N} (\hat{k} - Y) \cdot p^{\hat{k}-Y} \cdot (1-p)^{Y \cdot N - \hat{k}} \cdot \binom{Y \cdot N - Y}{\hat{k}-Y} \\
&= Y + (Y \cdot N - Y) \cdot p \cdot \underbrace{\sum_{\hat{k}=Y+1}^{YN-Y} p^{\hat{k}-Y-1} \cdot (1-p)^{Y \cdot N - \hat{k}} \cdot \binom{Y \cdot N - Y - 1}{\hat{k}-Y-1}}_{=1 \text{ (Binomial theorem)}} \\
&= Y + (Y \cdot N - Y) \cdot p \\
&\overset{p=\frac{1}{G}=\frac{d}{N}}{=} Y \cdot (d - \frac{d}{N} + 1)
\end{aligned} \tag{12}$$

With increasing number of hosts N and constant region density d this expectation converges to $Y \cdot (d+1)$, which is the product of the number regions one avatar having been in

and the expectation value in the case without region change[4].

[4]Compare to the equation 5 on page 26.

Chapter 6

Results

In the previous chapter we have analyzed theoretically, how the distribution of the number of subscriber position messages received by a host for the cases without and with region changes look like respectively. This theoretical distribution was made for a tick. To make the results comparable with our measurement, this theoretical distribution is rescaled with the factor 1 Tick/0.15 s. In this chapter, the experimental results are presented. We compare them firstly with the results from our theoretical analysis (section 6.1) and then present the scaling behavior of message rate as well as the average number of routing latency of unicast and multicast messages respectively (Section 6.2 and 6.3). The definitions for the different types of message rates can be found in section 4.1 on page 22.

6.1 Comparison of the theoretical and experimental results

6.1.1 Case 1: with only position updates turned on

The measurements used here are from the experiment with 1000 hosts and 100 groups, which is simulated 11 times each with a different random seed. Each simulated host sends every 150 ms a position message of his avatar to his Scribe multicast group. Group changes, food accesses, and fightings are turned off. All other parameters were described in chapter 4.3 on page 19.

Subscriber position message rate distribution The theoretically analyzed distribution of the subscriber position message rate from section 5.2.1 and the result of the measurement for 1000 simulated hosts and 100 Scribe multicast groups are shown in figure 6.1[1]. As you can see here, the theoretical model is within 95% confidence interval of the experimental result. The expectation of the average number of subscriber position messages per host per second is from our theoretical model equals $(10 - 10/1000 + 1)/0.15 = 73.27$ subscriber position messages / s / host and the measured average is 73.30 subscriber position messages / s / host.

Distribution of total position message rate A host can play alongside the role of a subscriber additionally the coordinator or a forwarder for another group. Therefore, the

[1] As mentioned in the introduction to this chapter, the theoretical curve is rescaled by a factor of 1 Tick/0.15 s.

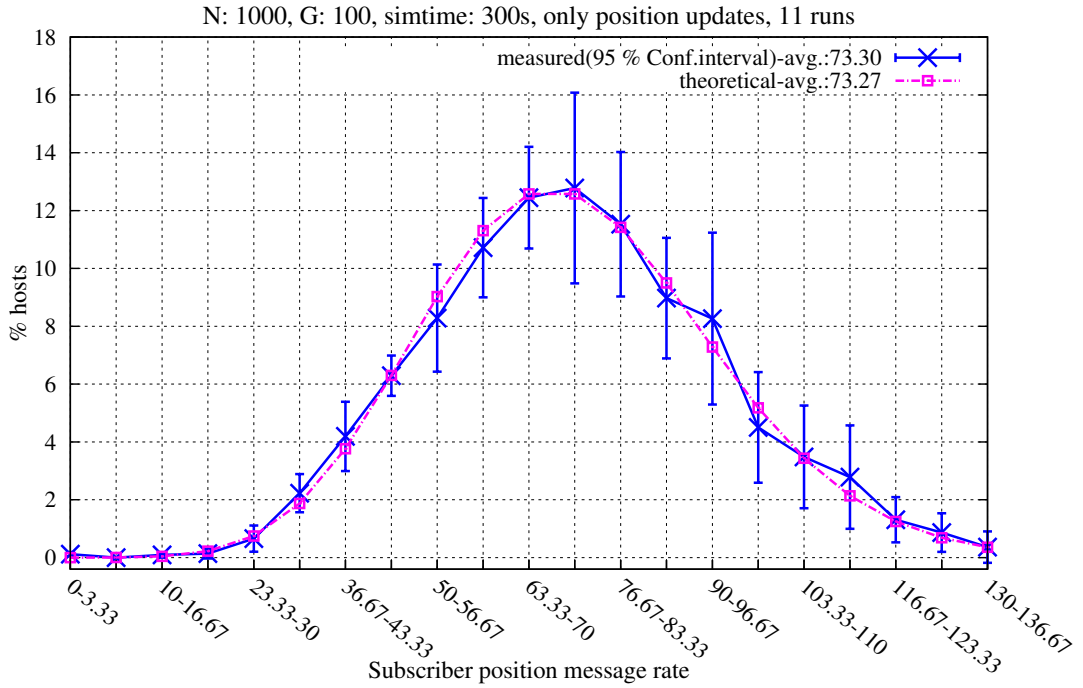


Fig. 6.1: Distribution of subscriber position message rate

measurement of the subscriber position messages is not sufficient to tell the real the network download completely. Section 5.2.2 on page 27 presents a theoretical model for the distribution of the total position message rate by a host. It is based on two criteria:

1. The subscriber position message rate distribution does not correlate with the distribution of the number of additional involved groups by a host.
2. The distribution of the number of additional involved groups by a host comes from the measurement.

Figure 6.3 shows the distribution of the number of additional involved groups by a host in a simulation of 1000 hosts and 100 groups, i.e. with what probability a host is forwarder or coordinator but not subscriber for a group. As you can see, 42% of the 1000 hosts are involved in no additional groups, so they are only subscribers for the groups of their own regions. But there is also a very small percentage of hosts (0.05%), which are involved in 8 other groups.

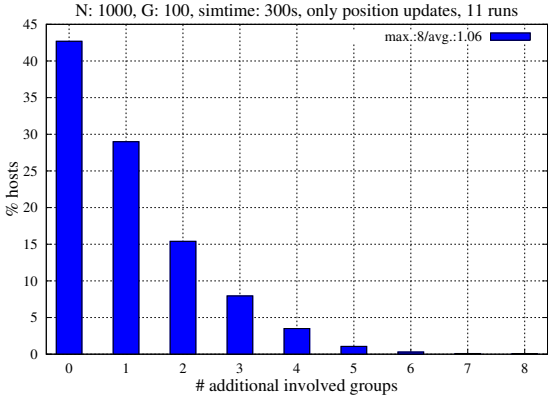


Fig. 6.2: Distribution of the number of additional involved groups by a host

In figure 6.3, we see both the distribution analyzed theoretically, which is calculated by the convolution of the position message rate distribution analyzed theoretically (the purple curve in figure 6.1) and the distribution of number of additional involved groups

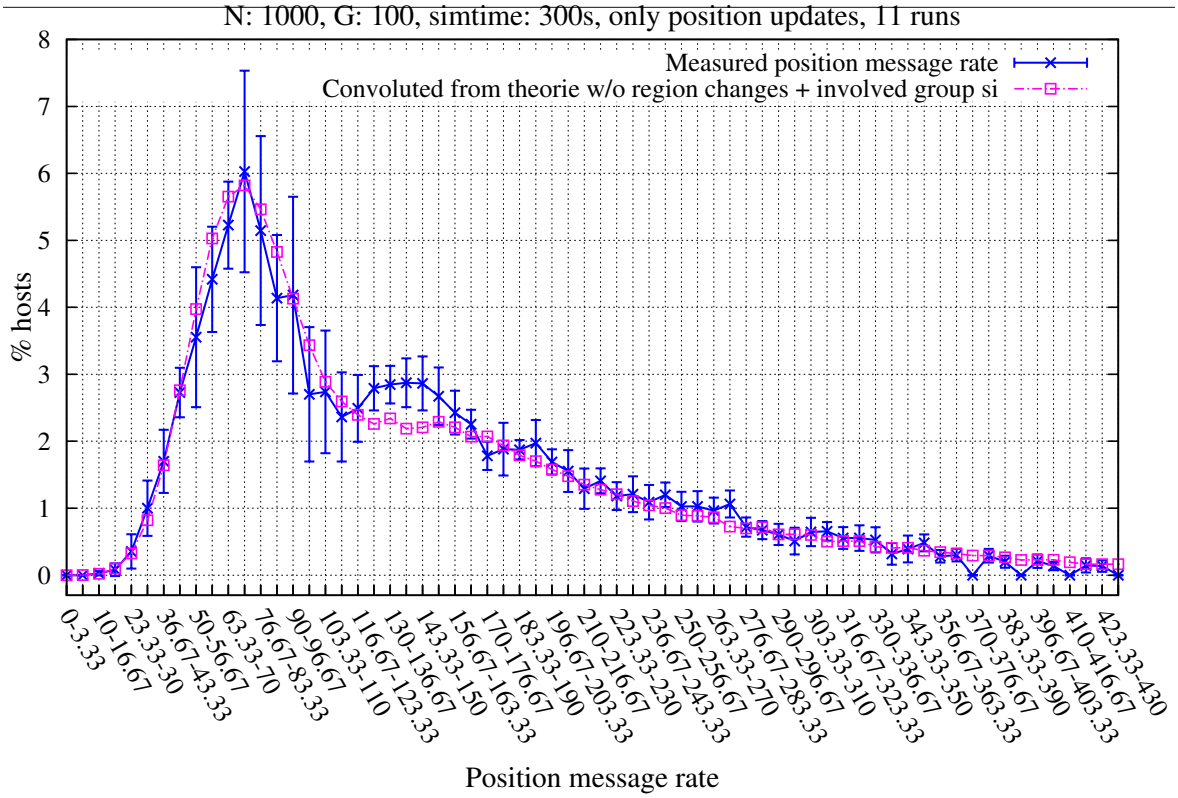


Fig. 6.3: Distribution of subscriber position message rate

by a host (figure 6.2), and the measured distribution of position message rate. The experimental and theoretical results agree very well. Only the range between 123.33 and 150, where the measured curve has a local maximum, shows a small percent of hosts of the theoretical distribution beyond the confidence interval. The reason for this could be that the number of subscriber position message rate by a host is not completely independent of the number of additional involved groups by this host.

To make it clear that the position updates contribute the most network download traffic, we show in figure 6.4 the distributions of measured position message rate (see the blue curve in figure 6.3) as well as the distribution of the total message rate. A host receives in the case without region changes in addition to position messages also the initial SCRIBE_JOINmessage and 1 message about the region information from its coordinator. As you can see here, these additional messages have no big effect on the whole network download, furthermore the average position message rate is 148.70 messages/s/host, while the average message rate is only 0.01 messages/s/host higher (148.71 messages/s/host). So we can say, the distribution of the position message rate is a very good estimation for the distribution of the total message rate. The measured distribution for the case of region changes plus position updates and the case of all avatar activities turned on can be found in figure 6.6 on page 37.

6.1.2 Case 2: Positions updates + region changes

The measurements used here are from the experiment with 1000 hosts and 100 groups, which is simulated 5 times each with a different random seed. Each simulated host sends every 150 ms a position update message of his avatar to the Scribe multicast group and changes on average every 40 s its group. All other parameters were described in chapter 4.3 on page 19.

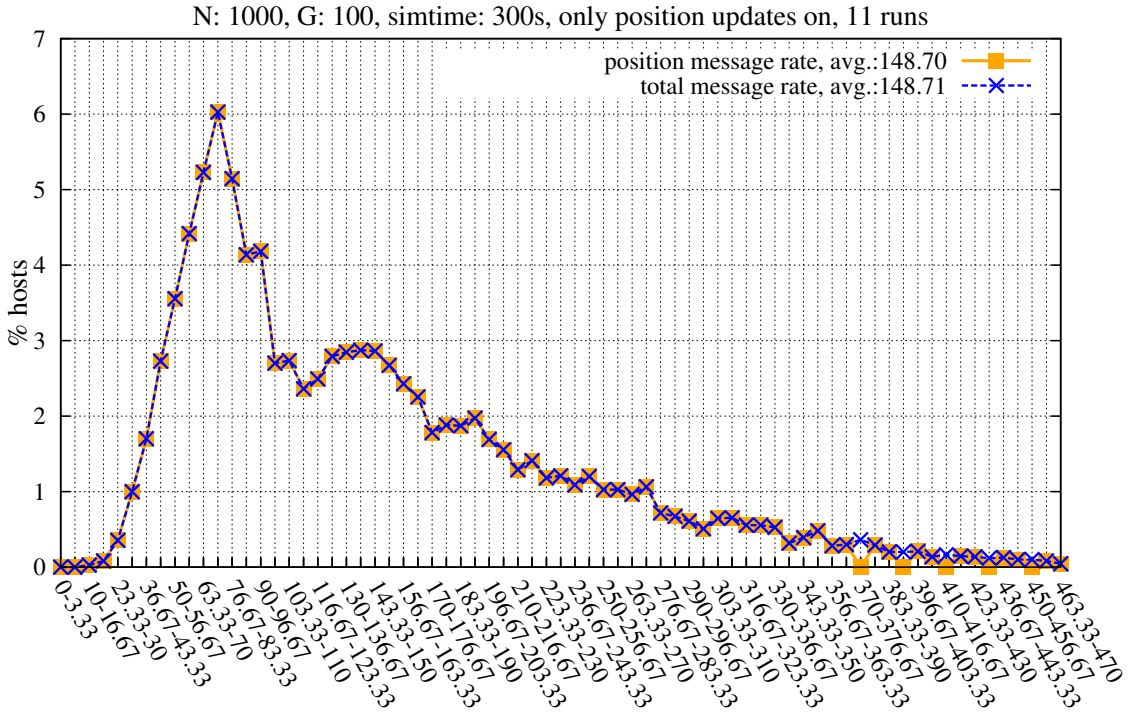


Fig. 6.4: Position message rate versus total message rate

Distribution of subscriber position message rate The subscriber position message rate of a host is calculated, in that we divide the number of subscriber position messages of this host by the simulation time. This rate shall be measured on all hosts. A distribution of the subscriber position message rate can be made in this way. As the food access and fighting between hosts are turned off, this case corresponds to an experiment of $Y - 1$ times region changes, ie the theoretical model from section 5.2.1 on page 26.

Figure 6.5 compares the measured result with the theoretical analyzed distribution of subscriber position message rate from formula 11 with 7.5 regions changes[2] on page 30. Here, too, the measured result corresponds to the expectation from the theoretical model.

The more frequently each host changes his group, the higher the percentage of hosts that receive the expectation of subscriber position message rate, ie 73.27 subscriber position messages/s (no group change: 12.38% versus with 7.5 group changes: 35.96%). This behavior reflects the law of large numbers from the probability theory.

Region changes + position updates vs. all avatar activities turned on The food accesses or fighting between avatars don't bring many network load traffic, since the frequency of each of these two activities is very low (each on average every 20 s), in contrast to the frequency of position updates (every 150 ms). We estimate, that each host should receive $\frac{1}{20} \times 10 = 0.5$ food messages/s and $\frac{1}{20}$ fighting messages/s, if food accesses and fighting are turned on.

The figure 6.6 about the distribution of total message rate confirms our expectation: The orange curve comes from our experiment for the case of region changes (on average every

[2] In a 300-second simulation, an avatar changes his region on average $\frac{300}{40} = 7.5$ times.

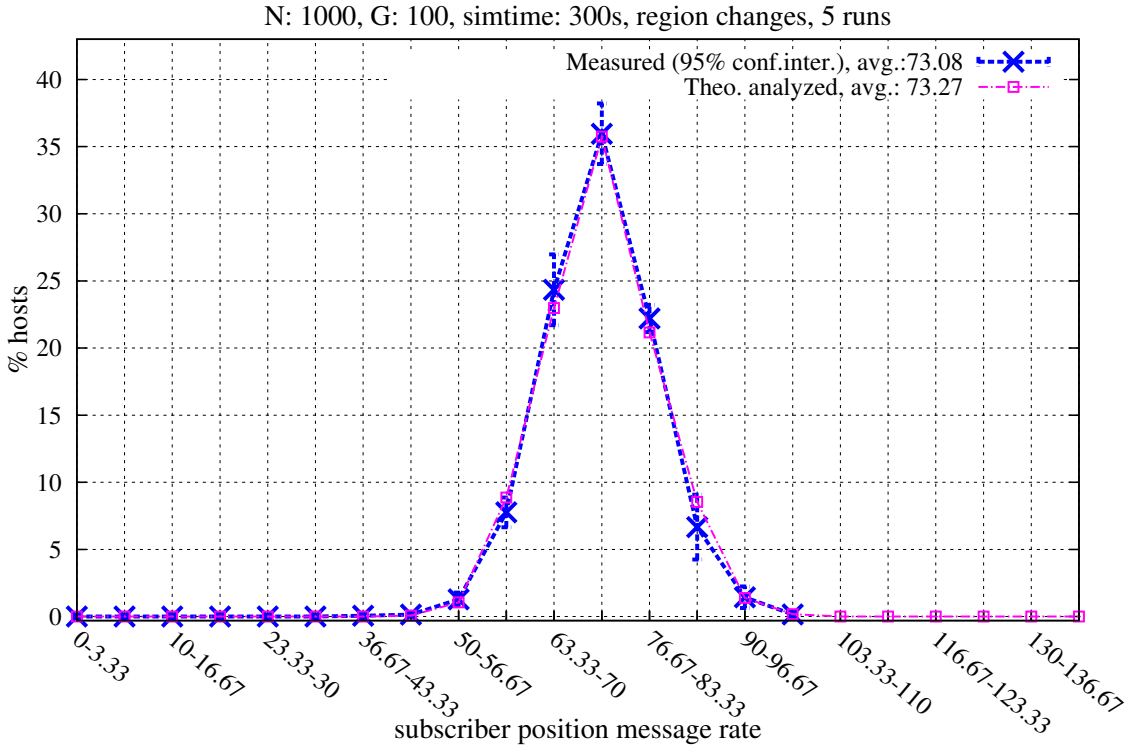


Fig. 6.5: Distribution of the subscriber position message rate for the case of region changes on

40 s) plus position updates are turned on, while the blue shows the experimental result for the normal SimMud game, ie. all avatar activities are turned on. The two curves are within their opposite confidence interval. The average total message rate is 149.50/s/host, if only region changes and position updates are turned on. For the normal SimMud game, this value is 2 messages/s/host higher (151.5 messages/s/host).

6.2 Scaling behavior of the network download

This section discusses the scaling behavior of two sizes, ie the subscriber position message rate and the total message rate[3].

6.2.1 About the subscriber message rate

Most of the subscriber messages are subscriber position messages. So we can predict from the theoretical analysis of the case with region changes (or group changes) turned on (See section 5.3 on page 28), how the average subscriber message rate scale with the number of hosts. As we already know from the formula 12 on page 30, that the average subscriber message rate depends only on the number of hosts and the average group density[4] and

[3] See table 4.1 on page 22.

[4] You have to divide it by the time interval of two consecutive position updates (0.15 s) to get the average subscriber message rate.

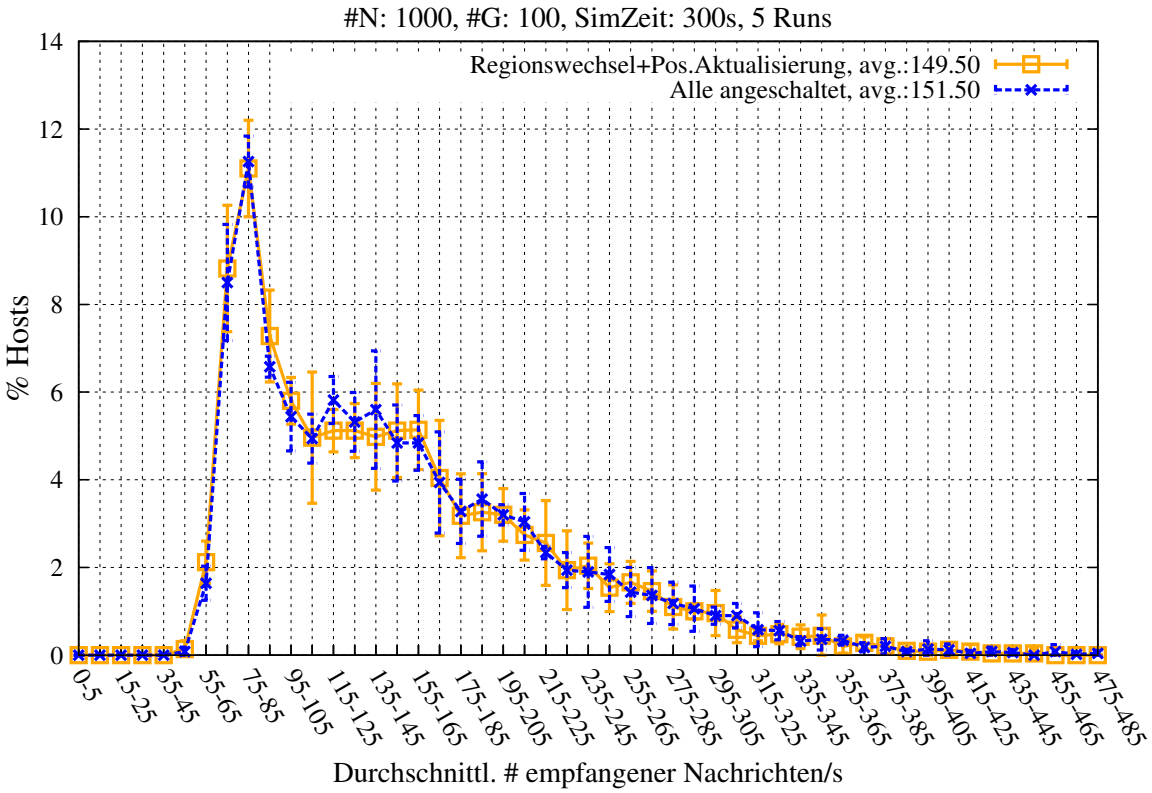


Fig. 6.6: Message rate distribution: only region change vs. all activities on

converges even for large number of participants and constant avatar density to $\frac{11}{0.15\ s} \approx 73.3$ [Subscribernachrichten / s / Host][5].

Figure 6.5 shows the distribution of the subscriber message rate [6] for a simulation with 1000 hosts and 100 groups. This distribution can be derived to calculate the mean and the standard deviation of the subscriber message rate. This process can be repeated for different number of hosts. The results are recorded in the blue curve of figure 6.7[7]. This curve shows an approximately constant behavior of number of subscriber message rate per host for simulations with different numbers of hosts. The expectations[8] from the theoretical analysis are also recorded in figure 6.7 (See the pink curve). The measured average subscriber message rate is about 1 subscriber message / s / host higher than the theoretically analyzed one. This follows from the fact that the subscriber messages contain addition to the subscriber position messages also the messages relevant for region changes, such as AVATAR_LEAVE or the region information messages, avatar fighting messages and object update messages. But as

[5]This is actually the limit for the average subscriber position message rate, but because the position messages account for over 99% of subscriber messages, one can take this to the limit of the average subscriber message rate.

[6]The subscriber messages received by a host contains some more additional messages than the subscriber position messages. See the blue curve in figure 7.2 on page 43. For the calculation in figure 6.7, the received subscriber messages are considered.

[7]The red curve comes from the measurement of Knutsson et al. and will be investigated in detail in the next chapter, section 7.2.2 on page 45.

[8]See section 5.3.2 on page 30 and note 4 on page 36.

we can find out, these additional messages take only a very small part of the whole subscriber messages received by a host.

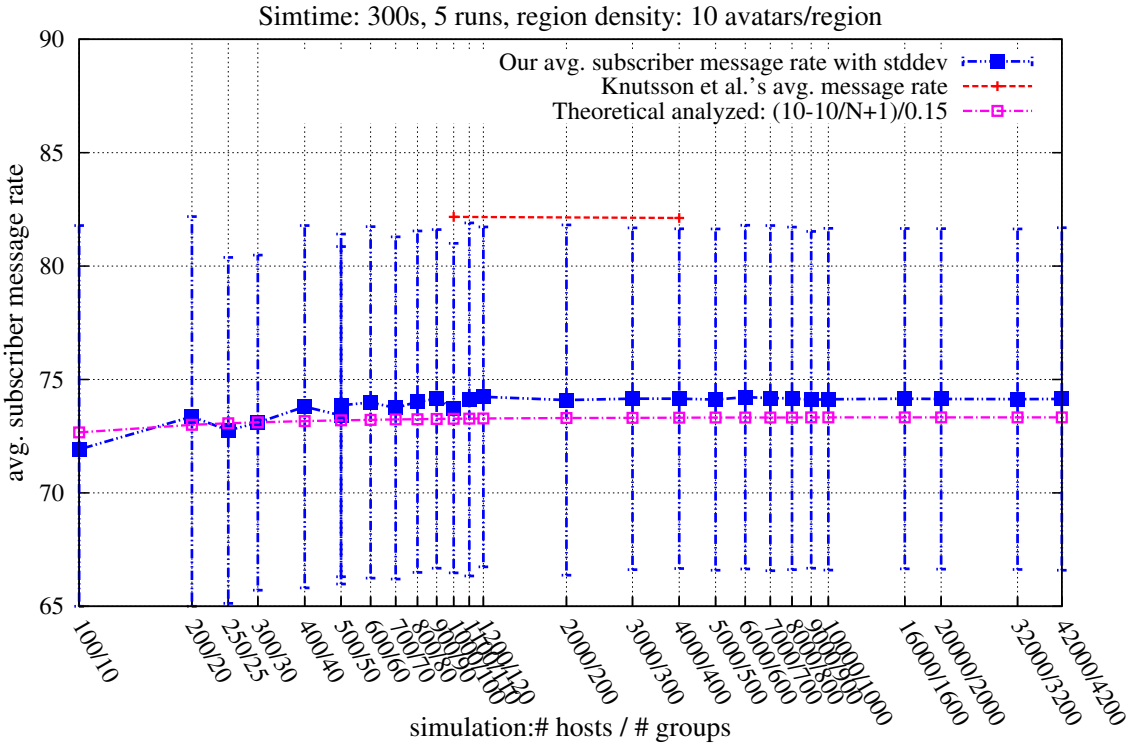


Fig. 6.7: Scaling of avg. subscriber message rate with the number of hosts

6.2.2 On the total message rate

We expect that in a system with N hosts, the average number of messages received per host per second scales with $log(N)$. This is because the Scribe [1] and Pastry [5] are used to route messages. Both protocols require an average $\mathcal{O}(log(N))$ hops to deliver a message from the sender to the receiver and each such hop causes a new message.

First, we take a look into figure 6.8 showing the distributions of total message rate of the simulation with 250, 1,000, 16,000 or 42,000 hosts, and each with 25, 100, 1600 or 4200 groups (average group density of 10 hosts per group) as shown: Simulations with more hosts have a flatter peak and a longer tail.

The blue curve in figure 6.6 on page 37 shows the distribution of the total message rate for 1000 hosts. This distribution can be derived to calculate the mean and one standard deviation of the total message rate. This process can be repeated for simulations with different number of hosts. The results are recorded in the blue curve of figure 6.9 with a semi-logarithmic scale[9]. It is clearly evident that this size scales logarithmically with the number of hosts.

[9]The red curve will be described in detail in next chapter, section 7.2.2 on page 45.

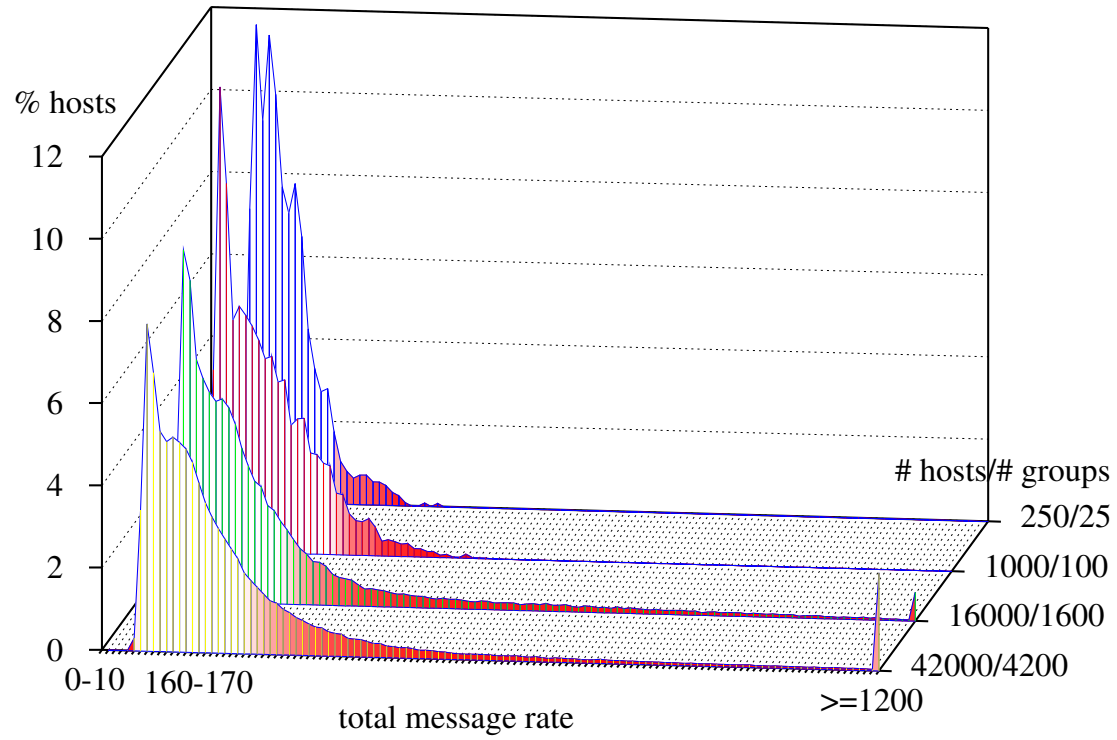


Fig. 6.8: message rate distributions for different number of hosts

6.3 Scaling Behavior of Routing Latency

How many hops are required to deliver a message depends on the routing algorithm. Pastry routing algorithm guarantees that the number of routing hops of a message is on average $log(N)$ hops [5].

Again the number of routing hops of unicast & multicast messages will be measured respectively for the simulation of 1000 participants and 100 groups[10]. Figure 6.10 shows the distribution of measured number of routing hops of unicast & multicast messages respectively. Unicast messages are routed through Pastry and multicast messages over Scribe. In addition, follow Scribe multicast message the reverse path of pastry routing. Therefore, the distributions of the routing latency of the two types of messages do not show any difference. As is evident from the figure, the measurement confirmed that presumption.

It is possible to calculate the mean and the standard deviation of the routing latency of both the unicast and multicast messages from the respective distribution in figure 6.10. These values are shown for different number of hosts and at constant average group density of 10 hosts per group in figure 6.11. The yellow curve in 6.11[11] with a semi-logarithmic scale shows the average number of routing hops and its standard deviation for unicast case. This value scales logarithmically with the number of hosts. The blue curve in the same figure, which is practically identical with the yellow curve shows the average number of routing hops and its

[10]The definitions of unicast and multicast messages are described in Section 4.3.2, Item 5 and Item 6 on page 20.

[11]The green and the red curve will be inspected in the next chapter, section 7.2.2 on page 45.

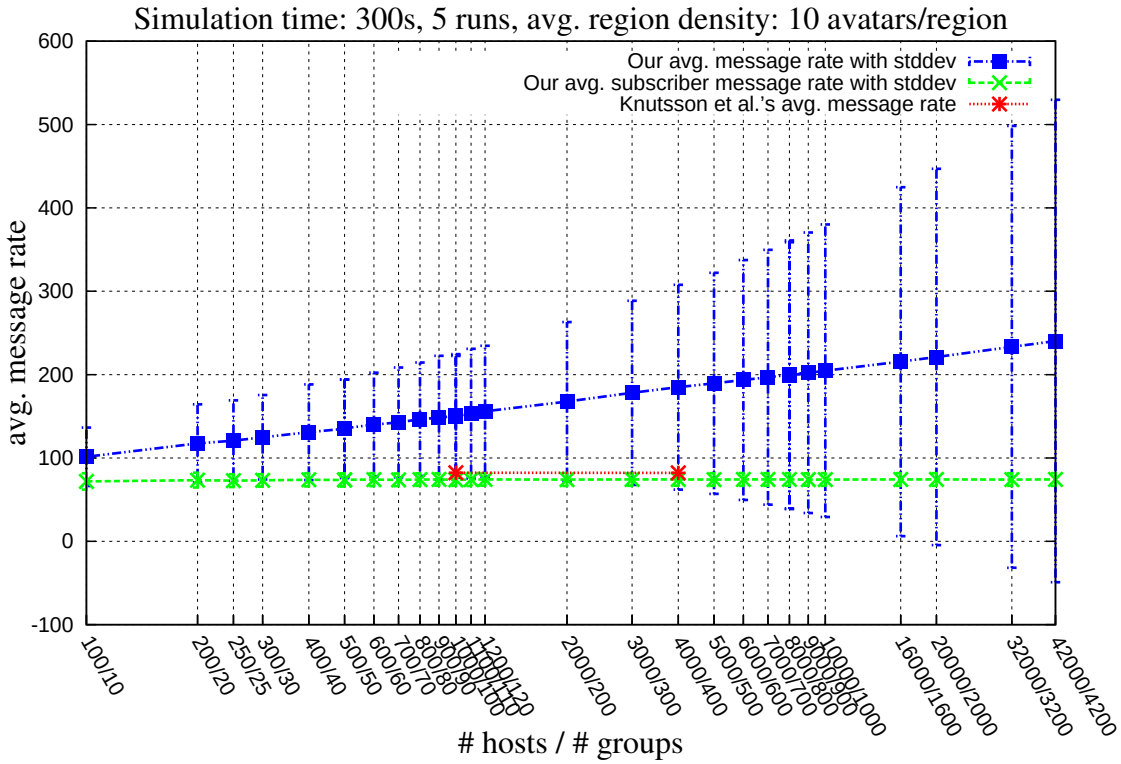


Fig. 6.9: Scaling of avg. total message rate with the number of hosts

standard deviation for multicast messages. As you can see here, the two measurements scale logarithmically with log (N), which is also the expectation of Knutsson et al.

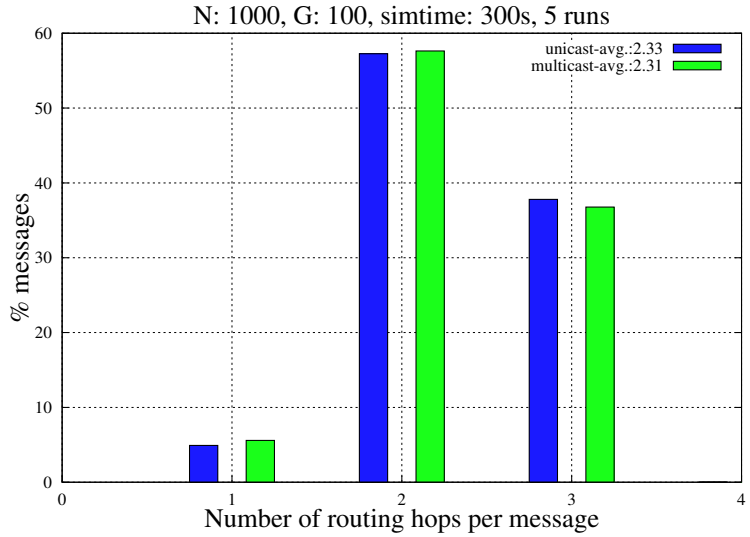


Fig. 6.10: Distribution of routing latency of unicast- & multicast messages respectively

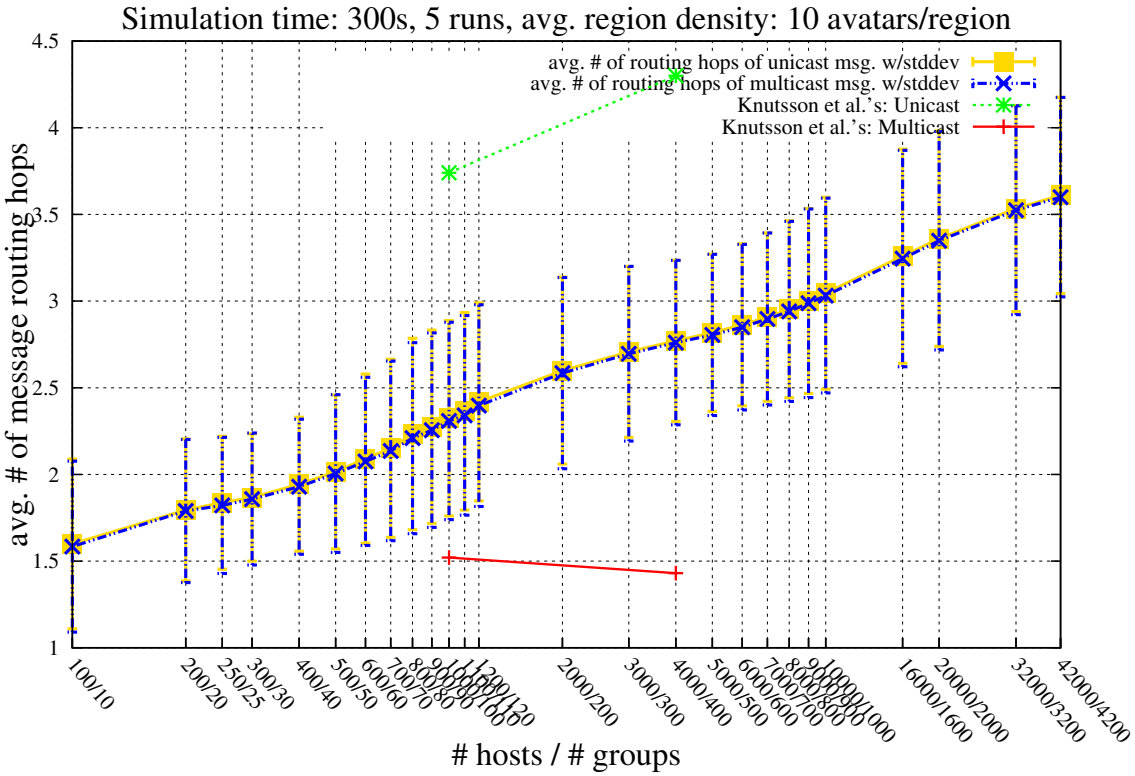


Fig. 6.11: Scaling behavior of the avg. number routing hops per unicast message and per multicast message respectively

Chapter 7

Comparison to Knutsson et al.'s Work

In this chapter, we compare our results (see chapter 6) with those of Knutsson et al.'s. This includes not only the differences of the measurements already made by Knutsson et al. but also the extended work that we go well beyond.

7.1 Compared to Knutsson et al.'s measurements

7.1.1 About the network load

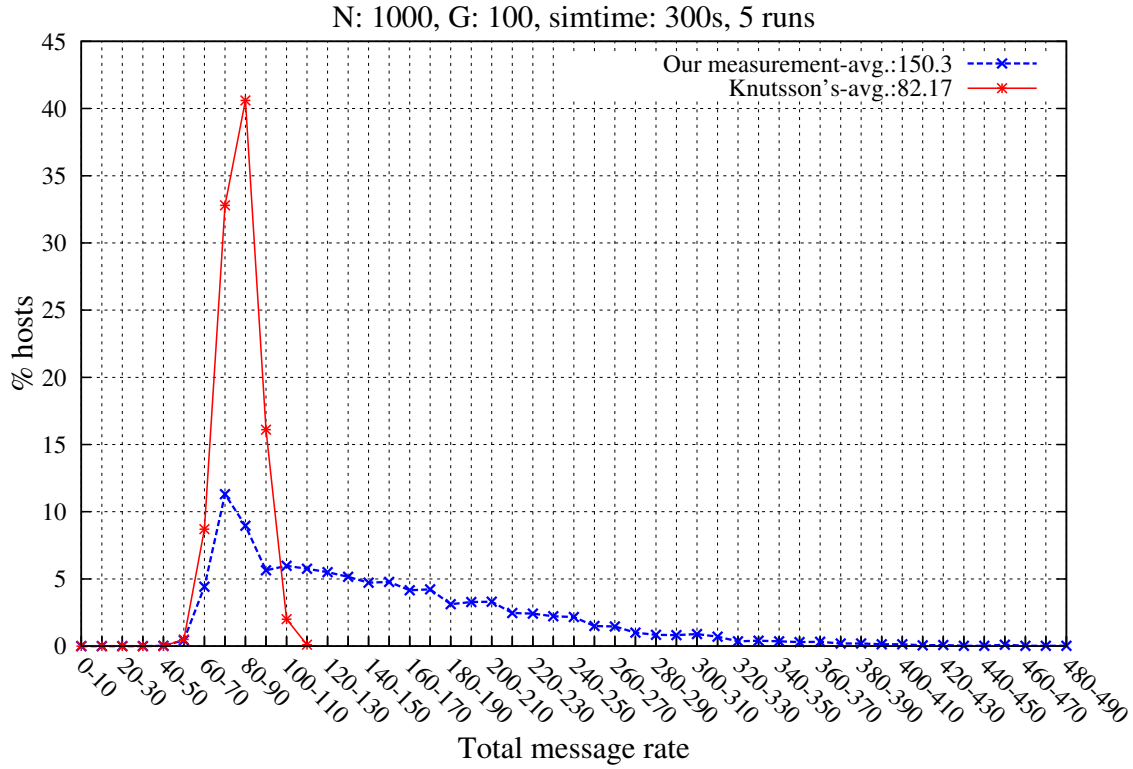


Fig. 7.1: Comparison: distribution of total message rate

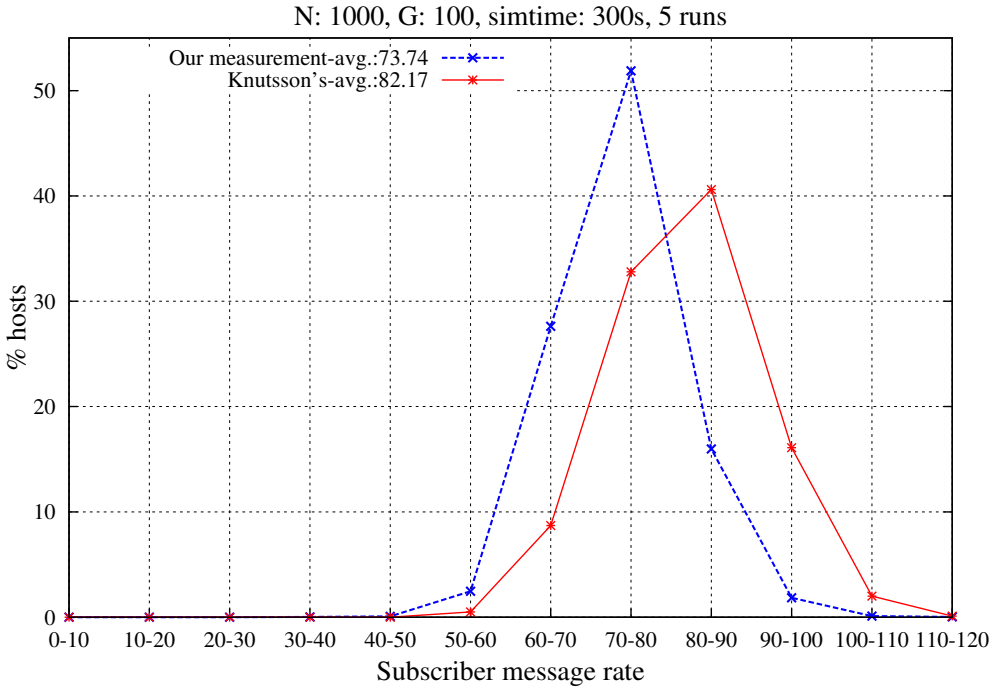


Fig. 7.2: Comparison: distribution of subscriber message rate

Figure 7.1 shows two distributions of the message rate from our own measurements and from Knutsson et al.'s work[1]. The two results are very different, so the assumption seems likely that two different sizes are measured. As discussed in section 3.4.1 on page 16, the message rate that Knutsson et al. have measured contains probably not really all messages received by a host, but only the subscriber messages[2]. Our argumentation is, there are coordinators in the DVE and some of them should receive at least 133.3 messages per second. But Knutsson et al.'s result shows a maximum of message rate at $120 - 130$ messages/s and this maximum makes up not more than 1%. Although it is not clear how a Scribe multicast tree of Knutsson et al. really look like and so we can not estimate how many and how likely hosts are forwarders for a Scribe multicast group, but it is clear that approximately 10% of the hosts are coordinators. So there have to be an notable amount of hosts receiving at least 133.3 messages per second. But it is regrettably not the case for Knutsson et al.

Knutsson et al. have probably only taken into account the subscriber messages for this measurement. Subscriber messages received by a host are messages about a region, which this host's avatar is currently residing in. This kind of messages forms only a fraction of the total messages that a host can receive. To make it clear, we see figure 7.2 comparing the distribution of subscriber message rate of our measurement (the blue curve) and the distribution of Knutsson et al.'s total message rate (the red curve, see also their original figure in 3.3 on page 14). The agreement between the two curves is significantly higher than the two in figure 7.1. Because of the inaccurate description of the message measurement from Knutsson et al. an exact match of the two measurements is but not to be expected. Knutsson

[1] Knutsson et al.'s data come from figure 3.3 on page 14.
[2] Subscriber messages are defined in definition 19 on page 11.

et al.'s curve shows about 8 messages/s over the curve of our measurement.

7.1.2 About the message latency

Knutsson et al. have given no average routing latency of unicast or multicast messages. We averaged therefore them from figure 3.4 and 3.5 respectively and show the averaged values[3] in table 7.1. The comparison with Knutsson et al. is difficult since on the one hand, different implementations of Pastry and Scribe routing algorithms are used, and on the other hand it is not clear, what has been measured accurately in the original work for the routing latency. As you can see here, the routing latency of two message types with Knutsson et al. show no similarities, while our own measurements are the same within each other's statistical deviation. Moreover, the average number of routing hops of Knutsson et al.'s multicast messages for simulation with 1000 hosts is larger than the one with 4000 hosts. This is contrary to their analysis, the routing latency scales logarithmically with the number of hosts. One explanation for this contradiction is difficult.

	Unicast messages		Multicast messages	
# hosts / # groups	1000/100	4000/400	1000/100	4000/400
Knutsson et al.	3.7	4.3	1.5	1.4
our measurement	2.32	2.77	2.31	2.76

Tab. 7.1: Comparison: avg. numbers of routing hops (in # hops)

7.2 Extensions

We have made two main extensions compared to Knutsson et al:

1. We have analyzed theoretically, how the message rate distribution should look like.
2. Many more experiments for both smaller and larger number of hosts are simulated. A maximum of 42,000 hosts were simulated with 4200 Scribe groups.

Section 7.2.1 compares the message rate distribution from our own measurements with the theoretical analysis in chapter 5. In section 7.2.2, we compare the results from our experiments with those of Knutsson et al. under constant group density for 1000 as well as 4000 hosts.

7.2.1 Comparison to theory

Knutsson et al. have made no theoretical analysis on the message rate distribution. An extensive theoretical analysis of the present work was described in chapter 5. Our theoretical results agree very well with our own measurements. More detailed discussions on the comparison between the theoretical analysis and experimental results were given in section 6.1 on page 32.

[3] See also figure 6.11 on page 41.

7.2.2 Comparison to Knutsson et al.'s results

On the different message rates We can calculate the mean and the standard deviation from the distribution of the number of subscriber messages rate from all hosts (the blue curve in figure 7.2) and repeat this process for simulations with different hosts. These results are then recorded into the blue curve of figure 6.7 on page 38. As discussed in 6.2.1, the mean value of these message rate should converge at high number of participating hosts with a constant density to ≈ 73.3 [subscriber messages/s/h][4]. Moreover, Knutsson et al.'s reported average message rate (See also table 3.2 on page 15) is entered here in the red curve. Knutsson et al. have measured only two points and their message rates are really very similar, as they have described.

Furthermore, we can calculate the mean and the standard deviation of the distribution of the total message rate (the blue curve in figure 7.1 on page 42) and repeat this process for simulations of different participating hosts. These results are then entered in the blue curve in figure 6.9 with semi-logarithmic scale on page 40. As discussed in section 6.2.2, the average message rate scaled at constant density logarithmically with the number of hosts. Knutsson et al. have measured only two points and results are still quite similar.

These all confirmed the suspicion that they have not really measured the total message rate. Hence comes the presumption that they have not taken into account all types of messages for their measurement.

About the routing latency Similarly, we can calculate the mean and standard deviation of each routing latency of unicast and multicast messages respectively (See figure 6.10 on page 41). This process is repeated for simulations with different number of hosts. The results are recorded in figure 6.11 with semi-logarithmic scale. As you can see, the two routing latencies scale logarithmically with the number of participating hosts, while the green and the red curve, which are estimated from figure 3.4 and 3.5, show a different behavior. In particular, a decrease of number of routing hops of multicast messages is visible. It contradicts Knutsson et al.' s own expectation of a logarithmic increase. This contradiction can not not be resolved. Further discussions can be found in section 7.1.2 on page 44.

[4] Compare to note 5 on page 37.

Chapter 8

Conclusion

This work aimed to investigate the scalability of a P2P-based distributed virtual environment. We researched Knutsson et al.'s proposed P2P architecture and their game abstraction SimMud. Contradictions between their experimental and our expected results were investigated and simulations were expanded with more participating hosts.

Our main result is that both the average message rate as well as the average routing latency per message scales logarithmically with the number of participating hosts in the studied system under constant population density. The discrepancy that Knutsson et al.'s average total message rate is constant from simulations with different number of hosts can be resolved by reinterpretating their measurement: if one only considers the subscriber messages received by a host, then the average number of this kind of messages received per host per second is constant both in our theoretical as well as in our experimental result. So we assume, that Knutsson et al. have only considered the subscriber messages as the whole network download by a host. Knutsson et al.'s measured scaling behavior of message routing latency is slightly abnormal and could not be explained. However, our experimental measurements are extensive enough to confirm our and their expected logarithmic increase in the number of participating hosts.

In two major respects, this work goes well beyond the original work from Knutsson et al. On the one hand, we analyzed the received message rate of a host extensively, which agrees very well with our experimental result. On the other hand, we have made measurements for a wide range from 100 to 42000 hosts, which goes well beyond the two points, 1000 and 4000 hosts, of Knutsson et al.

Bibliography

[1] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):100–111, October 2002.

[2] Yogen K. Dalal and Robert M. Metcalfe. Reverse path forwarding of broadcast packets. *Commun. ACM*, 21(12):1040–1048, 1978.

[3] Björn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins. Peer-to-Peer Support for Massively Multiplayer Games. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 1, 2004.

[4] K. Morse. Interest management in large-scale distributed simulations. Technical Report ICS-TR-96-27, University of California, Irvine, 1996.

[5] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[6] Jan Sablatnig, Sven Grottke, Andreas Köpke, Jiehua Chen, Ruedi Seiler, and Adam Wolisz. Adam – A DVE Simulator. TKN Technical Report Series TKN-08-004, Telecommunication Networks Group, Technische Universität Berlin, February 2008.

[7] Marc Shapiro, Karthik Bhargavan, Yek Chong, and Youssef Hamadi. A formalism for consistency and partial replication. Technical Report MSR-TR-2004-58, Microsoft Research, Cambridge, UK, June 2004.

[8] Summary on Peer-to-Peer Support for Massively Multiplayer Games website. `http://www.cs.ubc.ca/~malam/cs538/presentation2`, März 2004. Zuletzt besucht: 2010-03-01.

Appendix A

Proof of Theorem 25

Beweis 25 (For theorem 24):

Base case Given by the equation 5 on page 26.

Induction hypothesis: Let the probability of receiving totally k subscriber position messages under $Y-1$ times region changes equals

$$s_Y(\hat{k}) = p^{\hat{k}-Y} \cdot (1-p)^{Y \cdot N - \hat{k}} \cdot \binom{Y \cdot N - Y}{\hat{k}-Y}$$

Inductive step: The probability of k received subscriber position messages with Y times region changes is found as follows:

$$\begin{aligned}
s_{Y+1}(\hat{k}) &:= \sum_{\hat{k}_1+\hat{k}_2=\hat{k}} s_Y(\hat{k}_1) \cdot s(\hat{k}_2) \\
&\overset{IV.+(5)}{=} \sum_{\hat{k}_1+\hat{k}_2=\hat{k}} p^{\hat{k}_1-Y} \cdot (1-p)^{Y \cdot N - \hat{k}_1} \cdot \binom{Y \cdot N - Y}{\hat{k}_1 - Y} \\
&\qquad \cdot p^{\hat{k}_2-1} \cdot (1-p)^{(N-\hat{k}_2)} \cdot \binom{N-1}{\hat{k}_2-1} \\
&= \sum_{\hat{k}_1+\hat{k}_2=\hat{k}} p^{\hat{k}_1+\hat{k}_2-(Y+1)} \cdot (1-p)^{(Y+1)N-(\hat{k}_1+\hat{k}_2)} \cdot \binom{YN-Y}{\hat{k}_1-Y} \cdot \binom{N-1}{\hat{k}_2-1} \\
&= p^{\hat{k}-(Y+1)} \cdot (1-p)^{(Y+1)N-\hat{k}} \cdot \sum_{\hat{k}_1=Y}^{\hat{k}-1} \binom{YN-Y}{\hat{k}_1-Y} \cdot \binom{N-1}{\hat{k}-\hat{k}_1-1} \\
&\overset{\hat{k}'_1:=\hat{k}_1-Y}{=} p^{\hat{k}-(Y+1)} \cdot (1-p)^{(Y+1)N-\hat{k}} \cdot \sum_{\hat{k}'_1=0}^{\hat{k}-(Y+1)} \binom{YN-Y}{\hat{k}'_1} \cdot \binom{N-1}{\hat{k}-(Y+1)-\hat{k}'_1} \\
&\overset{(9)}{=} p^{\hat{k}-(Y+1)} \cdot (1-p)^{(Y+1)N-\hat{k}} \cdot \binom{YN-Y+N-1}{\hat{k}-(Y+1)} \\
&= p^{\hat{k}-(Y+1)} \cdot (1-p)^{(Y+1)N-\hat{k}} \cdot \binom{(Y+1)\cdot N-(Y+1)}{\hat{k}-(Y+1)} \qquad (13)
\end{aligned}$$

□

Appendix B

Description of the Software Supplied

In the accompanying software you can find our simulator **Adam** plus the two routing protocols, and the implementation of SimMud, and the measurement scripts. The implementation of Scribe, and the SimMud and the scripts were produced as part of this thesis.

B.1 Installation

1. Prerequisite: A Linux on a machine with Boost libraries
2. Installation:
   - Extraction:

```
tar xvzf adam-1.0.0-chen.tgz
cd adam-1.0.0-chen
```

   - Compilation:

```
./bootstrap.sh -c -s
make
```

B.2 Execution

A simulation with 1000 participating hosts and a group average density of 10 hosts/group can be accomplished as follows:

```
cd experiments/simmud_knut

./bin/simulation.sh -f simmud_knut.ini
```

For a simulation with another number of hosts, e.g. 4000, or group density, for example 20 hosts/group, some points in files simmud_knut.ini simmud_knut.ned must be changed:

In simmud_knut.ini:

simmud_knut..no_hosts = 1000;* ⇒ *simmud_knut.*.no_hosts = 4000;*
simmud_knut..group_density = 10;* ⇒ *simmud_knut.*.group_density = 20;*
simmud_knut..yNoRegions = 100;* ⇒ *simmud_knut.*.yNoRegions = 200;*
In simmud_knut.ned:
player: cSimMudKnutHost[1000]; ⇒ *player: cSimMudKnutHost[4000];*
for i=0..999 do ⇒ *for i=0..3999 do*

B.3 Informationen for the codes

The software model of a simulated host in this study is shown in figure B.1. Basic concepts were described in chapter 2.

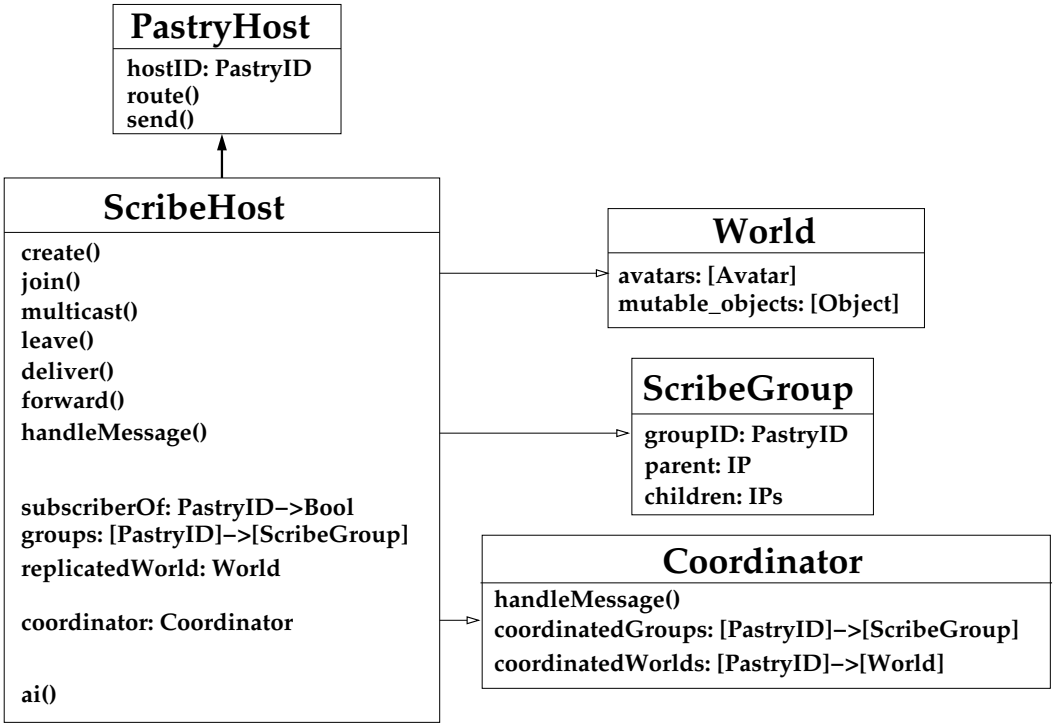


Fig. B.1: SimMud & Scribe Class Diagramm

Each Scribe host replicates the region (World), in which his avatar is just located. The replication of each region is always implemented as a list of avatars (Avatar) and food objects (Object). Each object has a position state. An object have an additional state called a food counter that says how many food units it still has.

Each Scribe host (ScribeHost) is also a Pastry host (PastryHost), so it has an ID and can use the API of Pastry, such as *route(msg, groupID)*. He keeps a list of Scribe group descriptor (groups) for which he acts as a forwarder. A Scribe group descriptor (ScribeGroup) with ID *groupID* will be stored by a Scribe host when the host gets a SCRIBE_JOINmessage for a Scribe group with ID *groupID* (=PastryID) from another Scribe host and is not yet a forwarder or subscriber for this group. He adds the sender (and its IP-address) of this SCRIBE_JOINmessage in the children list of the newly created Scribe group descriptor

and routes with a new SCRIBE_JOINmessage with ID *groupID* through Pastry. The old SCRIBE_JOINmessage is terminated. Pastry ensures that the next Scribe host on the routing path towards the coordinator of this group receives this new SCRIBE_JOINmessage.

If a Scribe host is a subscriber for a particular group, it can handle a multicast message through *handleMessage(msg, groupID)* or receive messages that are sent to him directly and relevant for the interaction of his avatar in the corresponding region.

If a Scribe host is a coordinator for a particular Scribe group with group ID $groupID'$, then he replicates in addition to the region, in which his avatar resides, also the region with ID $groupID'$. He also manages a Scribe group descriptor as a forwarder for this additional group. All multicast messages that go to the Scribe group are first sent to him. It decides by *coordinatorHandleMessage(msg, groupID)*, if a multicast message can be delivered to the members. The avatar activities of a host will be realized by means of the method *ai()*. Section 4.3.2 specifies when and how the decisions for the activities in each tick are made.

B.3.1 Statistics of own codes

As part of this thesis a total of 12651 lines of code (see enclosed CD) were written in addition to the already existing implementations in **Adam**. Table B.1 lists the number of rows for the self-written files:

Filename	LOC
general/gen.cc	174
scenarios/simmud/food.cc	118
scenarios/simmud/food.hh	83
scenarios/simmud/player.cc	162
scenarios/simmud/player.hh	114
scenarios/simmud/region.cc	89
scenarios/simmud/region_factory.cc	121
scenarios/simmud/region_factory.hh	113
scenarios/simmud/region.hh	79
scenarios/simmud/simmud_scenario.cc	312
scenarios/simmud/simmud_scenario.hh	156
consistency/knut/consistency_knut.cc	492
consistency/knut/consistency_knut.hh	246
consistency/knut/coordinator_knut.cc	552
consistency/knut/coordinator_knut.hh	150
consistency/knut/knut.msg	37
consistency/knut/knut_msg.cc	91
consistency/knut/knut_msg.hh	78
consistency/knut/knut_mutable_object.msg	18
consistency/knut/knut_mutable_object_msg.cc	64
consistency/knut/knut_mutable_object_msg.hh	65
consistency/knut/knut_region_state.msg	18
consistency/knut/knut_region_state_msg.cc	65
consistency/knut/knut_region_state_msg.hh	88
scribe/scribe_group.cc	79

scribe/scribe_group_factory.cc	126
scribe/scribe_group_factory.hh	82
scribe/scribe_group.hh	96
scribe/scribe.msg	39
scribe/scribe_msg.cc	94
scribe/scribe_msg.hh	66
scribe/scribe_multicast.msg	26
scribe/scribe_multicast_msg.cc	86
scribe/scribe_multicast_msg.hh	66
scribe/scribe_router.cc	968
scribe/scribe_router.hh	266
experiments/simmud_knut/src/simmud_knut_consistency.cc	652
experiments/simmud_knut/src/simmud_knut_consistency.hh	125
experiments/simmud_knut/src/simmud_knut_host.cc	1031
experiments/simmud_knut/src/simmud_knut_host.hh	141
experiments/simmud_knut/src/simmud_knut_router.cc	459
experiments/simmud_knut/src/simmud_knut_router.hh	124
experiments/simmud_knut/src/simmud_knut_scenario.cc	153
experiments/simmud_knut/src/simmud_knut_scenario.hh	62
experiments/simmud_knut/scripts/calc_avg_measurements.pl	48
experiments/simmud_knut/scripts/calc_moment.pl	124
experiments/simmud_knut/scripts/calc_msg_distribution.pl	89
experiments/simmud_knut/scripts/calc_stddev.pl	43
experiments/simmud_knut/scripts/calc_ws_distribution.pl	49
experiments/simmud_knut/scripts/check_mem.sh	30
experiments/simmud_knut/scripts/compute_avg.pl	167
experiments/simmud_knut/scripts/compute_coordinator_avg_results.pl	165
experiments/simmud_knut/scripts/conf_exp.sh	223
experiments/simmud_knut/scripts/conf_exp_withRegionChange.sh	203
experiments/simmud_knut/scripts/convolution.pl	151
experiments/simmud_knut/scripts/distrib_no_hosts.sh	228
experiments/simmud_knut/scripts/distrib_no_hosts_withStepsize.sh	233
experiments/simmud_knut/scripts/distribution_hopCounts.pl	166
experiments/simmud_knut/scripts/falten.pl	67
experiments/simmud_knut/scripts/groupSize_distrib.pl	77
experiments/simmud_knut/scripts/hopCounts_noHost_relation.pl	122
experiments/simmud_knut/scripts/make_lists.pl	36
experiments/simmud_knut/scripts/make_msgs_table.pl	114
experiments/simmud_knut/scripts/many_runs.sh	155
experiments/simmud_knut/scripts/messages_distribution_confidence.pl	176
experiments/simmud_knut/scripts/messages_distribution.pl	209
experiments/simmud_knut/scripts/messages_noHosts_relation2.pl	96
experiments/simmud_knut/scripts/messages_noHosts_relation.pl	152
experiments/simmud_knut/scripts/msg_distribution.pl	219
experiments/simmud_knut/scripts/plot_theory_numeric_measurements.pl	158

experiments/simmud_knut/scripts/quantisise.pl	49
experiments/simmud_knut/scripts/single_no_hosts.sh	172
experiments/simmud_knut/scripts/theory_numeric_measurements.sh	126
experiments/common_files/combine_sca_files.pl	156
experiments/common_files/compute_avg_results.pl	190
experiments/common_files/distrib_run2.sh	232
experiments/common_files/distrib_search.sh	117
experiments/common_files/single_search.pl	426
Insgesamt	12651

Tab. B.1: My own implemented source files and code size