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

Jiehua Chen*, Sven Grottke*, Jan Sablatnig*,
Ruedi Seiler* and Adam Wolisz†
Technical University of Berlin, Germany
*{chen, sfs, jon, seiler}@math.tu-berlin.de
†wolisz@tkn.tu-berlin.de

***Abstract*—We investigate the scalability of distributed virtual environments (DVEs) based on a structured peer-to-peer (P2P) overlay. We focus on network load and message routing latency. 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. [1]. Both our theoretical analysis and simulation results show that under constant population density, the network load is constant except for the overhead messages incurred by the overlay protocol. The overall network load per host as well as the message routing latency grows logarithmically with the number of hosts; this is in partial contradiction to Knutsson et al.'s results. We propose a resolution to this contradiction.**

I. Introduction

Typical distributed virtual environments (DVEs) such as online games, e.g. *World of Warcraft* or *Second Life*, are simulated virtual worlds shared by human participants. Interactions between participants are the main task of such DVEs. Each participant has a graphical representation of himself called *avatar* in the virtual world. He uses a host (computer) to control his avatar. Each host displays the virtual world (or a subset of it) to its participant and communicates with other hosts through the Internet using messages which cause network traffic and data transfer delays.

During the past few years, DVEs based on peer-to-peer (P2P) systems have been investigated since they promise better load balancing and robustness compared to the conventional client-server approach.

This paper focuses on the scalability of a P2P based DVE, i.e. how network load and message routing latency scale with the number of participants. We look into a structured P2P architecture using Pastry [2], a widely used P2P overlay, and Scribe [3], an application level multicast infrastructure built on top of Pastry. We study an existing prototypical game called SimMud [1] using this architecture. SimMud contains two important components of a virtual world, avatars and mutable objects. Four different kinds of basic activities that an avatar can perform are also included in SimMud.

How does a DVE like the SimMud game that is based on Pastry and Scribe scale with the number of participants? Two major factors, the design of the virtual world itself and the underlying network overlay, influence the network load and the message routing latency. If the network load incurred by the virtual world does not depend on the number of participants, we expect the scaling behavior of the overall network load per host to depend only on Pastry and Scribe, and hence to be $\mathcal{O}(\log N)$ ([2]) where N is the number of participants.

In this work however, we investigate the scaling behavior of the network load and the message routing latency under the influence of both, the virtual world design and the network overlay.

We assume the population in the virtual world to be uniformly distributed. We then investigate the network load incurred by the SimMud game, which we call the *subscriber message rate*. From our theoretical analysis in section IV, we see that the *average subscriber message rate* ([# subscriber messages/s/host]) is constant with a growing number of participants. This constant scaling behavior is confirmed by our experimental results. This allows us to build a virtual world like SimMud on top of a structured P2P architecture, which scales the same way as the P2P overlay itself.

Our theoretical expectation that the overall network load per host[1], i.e. the average number of messages received per second per host, scales logarithmically with the number of participants, is confirmed by our experimental results.

Our measurement of the average number of message routing hops agrees with our assumption and Knutsson et al.'s conclusion, that the message delay scales with $\mathcal{O}(\log N)$ hops on average, where N is the total number of hosts.

The idea of using Pastry and Scribe, and the SimMud game itself originate from Knutsson et al. [1]. In their work, they simulated SimMud and measured the total message rate at each host and the message routing latency for 1000 and 4000 participants. Their experimental results show that in both cases, the total message rate distributions are "very similar". The average number of multicast message routing hops decreases slightly when the number of hosts increases from 1000 to 4000.

Our result of a logarithmic scaling behavior of the average total message rate differs from Knutsson et al.'s. This discrepancy can, however, be resolved by assuming that Knutsson et al. did not take into account of the overhead messages incurred

978-1-4244-8953-4/11/$26.00

[1]We also call this the *average total message rate*.

by the overlay protocol. Unfortunately, it is not possible to verify this conjecture or to make a more detailed comparison, because the original code used in [1] is not available any more.[2]

II. System Model

As discussed in the introduction, our investigated DVE architecture employs a structured P2P approach which uses two existing components, Pastry [2] and Scribe [3]. Many applications such as online games can be implemented using this decentralized P2P model. In the following we will discuss two aspects of our system model: the aspect of the virtual world and the aspect of its realization, c.f. table I.

Virtual world	**Real world**
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 (within a region)	Client coordinator protocol
Inter-avatar interaction (within a region)	Direct communication

Tab. I: Two aspects of a DVE

A virtual world is a simulated world in which human participants, each represented by their own avatar, can interact with each other. Mutable objects, such as balls or doors, are another important component of virtual worlds. They are characterized by their states, e.g. position, color, closed/open. The difference between avatars and mutable objects is that each avatar is directly controlled by a human participant through his host.

In order to reduce network load and CPU consumption at each host, interest management [4] is applied, i.e. each host can express its own interest as the part of the virtual world in which the states of mutable objects and avatars are relevant to the avatar of this host. This field is called the *area of interest* (AOI) of this host. Typically, a host's AOI is derived from the position of its human participant's avatar in the virtual world.

Our system employs *partial replication*. More precisely, the whole virtual world is divided into fixed disjoint regions. Each host replicates only the objects and avatars in the same region as its own avatar. This region is the AOI of this host. Hosts, whose avatars are in the same region, form an interest group (a.k.a Scribe multicast group). For each region in the virtual world, there is a *multicast tree* in the real world, which contains all the hosts whose avatars are currently in the same region. These hosts are called *subscribers* (to this region/multicast group/multicast tree). The root of this multicast tree is called the *coordinator* of the region. It disseminates the states of avatars and objects located in this region. It also handles object update requests. Each host and each region has a unique ID. The host whose ID is numerically closest to a region's ID is assigned the coordinator of this region. The detailed description of the assignment of coordinators is described in [5].

Avatars can perform four different kinds of actions: An avatar can *move within a region*. Scribe is used to multicast regular position updates to the interest group to refresh the avatar position on other hosts of the group. An avatar can *change its current region*. This implies a multicast group change of the host. An avatar can *access any mutable object* in its own region. This is realized as a request to the coordinator host of the selected object. After a successful update, the coordinator multicasts the new object state to the interest group. An avatar can also *interact with another avatar* in the same region. The interaction is implemented through direct host communication.

III. Prototypical Game Scenario SimMud

SimMud is an abstraction of a simple online game and can be simulated on a single computer. SimMud covers the four activities of an avatar listed in section II. The state of each avatar includes its current position. The only kind of mutable objects are apples. Each apple has a nutrition counter. An avatar can move, eat an apple and fight with another avatar within the same region.

In a system with N participants (avatars), the virtual world of SimMud is divided into G fixed disjoint regions. An avatar enters a region with probability $1/G$. The average density of avatars of each region, which we call the *population density*, equals N/G avatars/region. The number of apples of each region equals N/G apples/region.

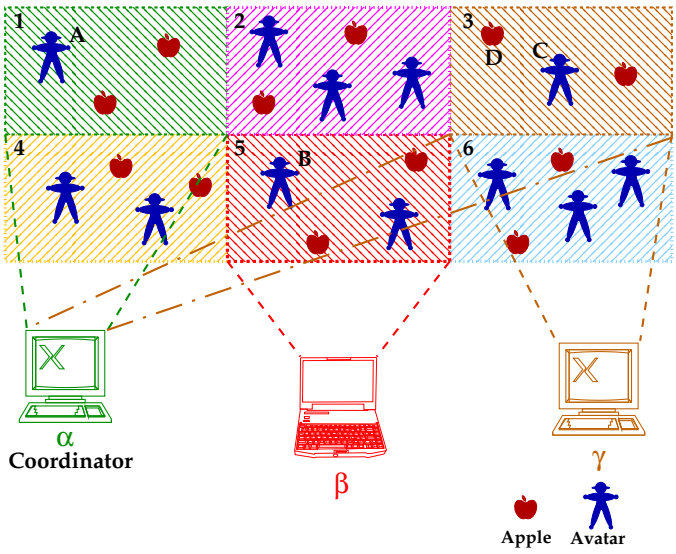

Fig. 1: A SimMud example

Figure 1 shows a game with 12 avatars, 6 regions each having $12/6 = 2$ apples, and a subset of the hosts involved, each replicating a region of the virtual world. Host α replicates region 1, because its avatar A is currently in this region. It also replicates region 3 because it is the coordinator for that region. If avatar C of host γ wants to eat apple D, γ must make a

[2]We thank Prof. Knutsson for kindly providing this information.

request to host α. If avatar B enters region 3, its host β notifies the coordinator of region 5 of its leave and the coordinator α of its join. α then sends β the current state of region 3 including the avatars and the apples. The hosts of other avatars in the same region are also informed of B's arrival. If avatar B wants to fight with another avatar in the same region, its host simply sends a fighting message to the opponent's host.

A. Types of messages received (Cf. figure 2)

A host is always a subscriber to the multicast group of the region in which its avatar is residing. It receives *subscriber messages* to this region including subscriber position messages, mutable object messages, messages relevant for region changes of any avatar within this region and fighting messages. Due to the way Scribe constructs a multicast tree, a host could also be a *forwarder* or the coordinator for another multicast group which it does not subscribe to. Messages which are only forwarded are called *overhead messages*.

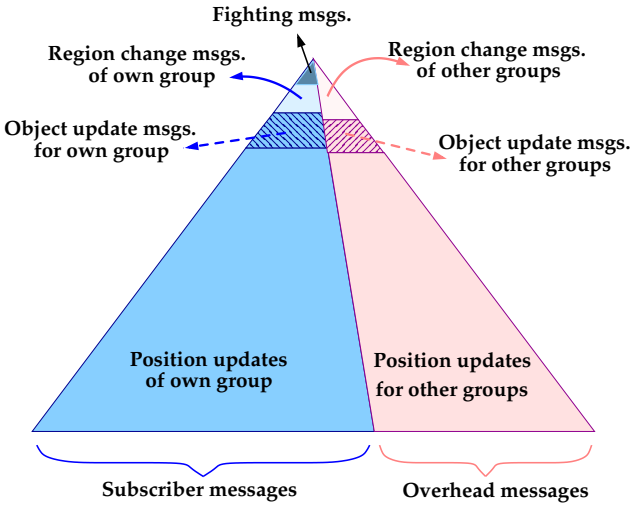

Fig. 2: Message types

Table II defines some important measurements. The *average total message rate* is defined as the mean value of the total message rates over all hosts in a DVE.

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. II: Message rates of different types at each host. T is the duration of a simulation run. We only consider messages received.

IV. Theoretical Analysis

The assignment of avatars to regions is random. In this section we analyze how probable it is that a region has a total of k avatars, and the distribution of the number of subscriber position messages received by a host.

A. Simplified probabilistic model of a host receiving subscriber position messages

We consider a situation which can be derived from the well-known urn model. In a system with N avatars and G regions, there are N hosts and G groups. At the beginning, each avatar joins any one of the G regions. We assume that the choice of each avatar is independent and identically distributed. If there are k avatars in a region, then the host of each of these k avatars receives k subscriber position messages about this region.

There are a total of G^N ways to distribute N avatars to G regions. Each region has the same probability ($= \frac{1}{G} := p$) to be chosen by each avatar. Let k_i with $i \in \{1, \ldots, G\}$ be the number of avatars which have chosen region i. We call $\omega := (k_1, \ldots, k_G)$ with $\sum_{i=1}^{G} k_i = N$ a possible selection event, with probability

$$p(\omega) = \frac{N!}{k_1! \cdots k_G!} \cdot p^N \tag{1}$$

1) Probability of a region having k avatars: Since all groups are equal, in the following, w.l.o.g, the probability $h(k)$ of the 1st region containing k avatars equals the probability of a region having k avatars:

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

2) Probability and Expectation of a host receiving k subscriber position messages: A region contains exactly k avatars with probability $h(k)$. Then the probability $s(k)$ that an avatar is in a region with k avatars equals

$$s(k) := p^{k-1} \cdot (1-p)^{(N-k)} \cdot \binom{N-1}{k-1} \tag{3}$$

, which is also the probability that a host receives k subscriber position messages.

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

$$E = d - \frac{d}{N} + 1 \tag{4}$$

With increasing number of N and constant population density $d := N/G$ the expectation of number of subscriber position messages received per host converges to $d + 1$.

3) Probability of a host receiving k position messages: A host is a subscriber for its own multicast group, i.e. the group of the region that its avatar is residing in. A host can also be a forwarder or the coordinators for other multicast groups. At this time, the host receives overhead messages. For example, if an avatar is in a region with x avatars, and its host is a forwarder for another multicast group with a total of y subscribers, then this host receives $x + y$ position messages. If a host is a forwarder or coordinator for a multicast group, then we expect that it receives on average twice as many position messages as a host which 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 multiplies accordingly.

If the distribution of the number of multicast groups a host is involved in is known, then we can convolute it with the theoretical distribution of the number of subscriber position messages as given in equation 3. Under the assumption that the number of position messages received by a host does not correlate with the number of multicast groups that a host is involved in, the resulting convoluted distribution should correspond to the distribution of the total number of position messages received by a host.

It is possible to analyze theoretically the probability of the number of multicast groups that a host is involved in. This theoretical analysis is part of the work of Pastry. In the current paper, network load of the P2P-based distributed virtual environments should be analyzed depending on the number of participating hosts and not the internals of the overlay used. So we *measure* the number of additional multicast groups that a host is involved in. This allows us to figure out the distribution of the number of additional multicast groups by a host and convolute it with the distribution of the number of subscriber position messages as given in equation 3.

B. Probabilistic model under region changes

Avatars in our simplified probabilistic model detailed above don't ever change their region. Now we suppose a more complicated situation, where each avatar can change its region. If an avatar moves from region i with $\hat{k_i}$ members to region j with $\hat{k_j}$ members (inclusive the newly joined avatar), its host receives firstly $\hat{k_i}$ subscriber position messages and then $\hat{k_j}$ subscriber position messages. We know from subsection IV-A that a host receives k subscriber position messages for the case without region change with probability $s(k)$ (Equation 3). Then the probability that it firstly receives $\hat{k_i}$ and then $\hat{k_j}$ subscriber position messages only about the region of his avatar is thus equal to $s(\hat{k_i}) \cdot s(\hat{k_j})$.

Now we investigate the probability of a host receiving $\hat{k}$ subscriber position messages and his avatar having been in Y regions in total. It is the same with a host receiving $\hat{k_i}$ subscriber position messages with $\hat{k}_1 + \cdots + \hat{k}_Y = \hat{k}$ for all $i \in \{1, \ldots, Y\}$.

Theorem 1 (Probability of receiving a total of $\hat{k}$ subscriber position messages after Y-1 region changes):

$$s_Y(\hat{k}) = p^{\hat{k}-Y} \cdot (1-p)^{Y \cdot N - \hat{k}} \cdot \binom{Y \cdot N - Y}{\hat{k} - Y} \tag{5}$$

For the proof of theorem 1 we refer to our technical report [5]. The expectation of the number of subscriber position messages received per host in this case is:

$$E' = Y \cdot (d - \frac{d}{N} + 1) \tag{6}$$

With increasing number of hosts N and constant population density d this expectation converges to $Y \cdot (d+1)$.

If each host sends in each time interval I a subscriber position message, then the average subscriber position message rate per host equals:

$$\tilde{E} = \frac{d - \frac{d}{N} + 1}{I} \tag{7}$$

With increasing number of hosts N and constant population density d this value converges to $(d+1)/I$.

V. Experiments

To evaluate the scaling behavior of a P2P based DVE, we measured two characteristics at each host: the total message rate and the average number of routing hops per message. We compare the simulation results on the one hand with our theoretical results, and on the other hand with the results in [1].

A. Experimental setup

We use our simulator **Adam**[3] [6] to experiment with networks of up to tens of thousands of hosts. On top of **Adam** we have implemented the Pastry routing protocol and the Scribe application level multicast protocol, the APIs of which are detailed in [2] and [3]. The game SimMud is simulated with a Pastry key length of 32 bit, a key base of 2^4 and a leaf set size of 32.

The network topology is fully connected. The link delay of any two hosts is between 3 and 100 ms with a uniform distribution. There is no packet loss. We investigate failure free environments, where there are no hosts leaving or joining in the overlay network during the game. Each simulated avatar changes its behavior at fixed times. Each experiment is run 5 times, each using a different random seed and lasting 300 (virtual) seconds.

During the game, each simulated host multicasts the position updates of its avatar every 150 ms to its interest group. Once every 150 ms, each avatar changes its current region with $\frac{0.15}{40} = 0.375\%$ probability, i.e. every 40 s on average, and eats and fights, each with probability $\frac{0.15}{20} = 0.75\%$, i.e. every 20 s. In this paper, each target region has the same probability to be picked. Furthermore, throughout all of the experiments in this paper, we choose the number of regions such that the average population density remains 10 avatars/region.

B. About the network load

Figure 3 shows three histograms for a simulation with 1000 hosts and 100 groups: the distribution of total message rate and subscriber message rate from our measurements, and from Knutsson et al. respectively[4]. Our measurements of the total message rate differs significantly from those given by Knutsson et al. We assumed that they considered only the subscriber messages rate as the total message rate received by a host. Every 150 ms each host multicasts the current position of its avatar to all members of its group, i.e. approximately 6.7 subscriber position messages/s/host. The average group density[5] is 10 hosts/group, so each host receives on average

[3] http://www.math.tu-berlin.de/condel

[4] Knutsson et al.'s data is taken from the plots in [1].

[5] This also equals to the population density from the view of the virtual world.

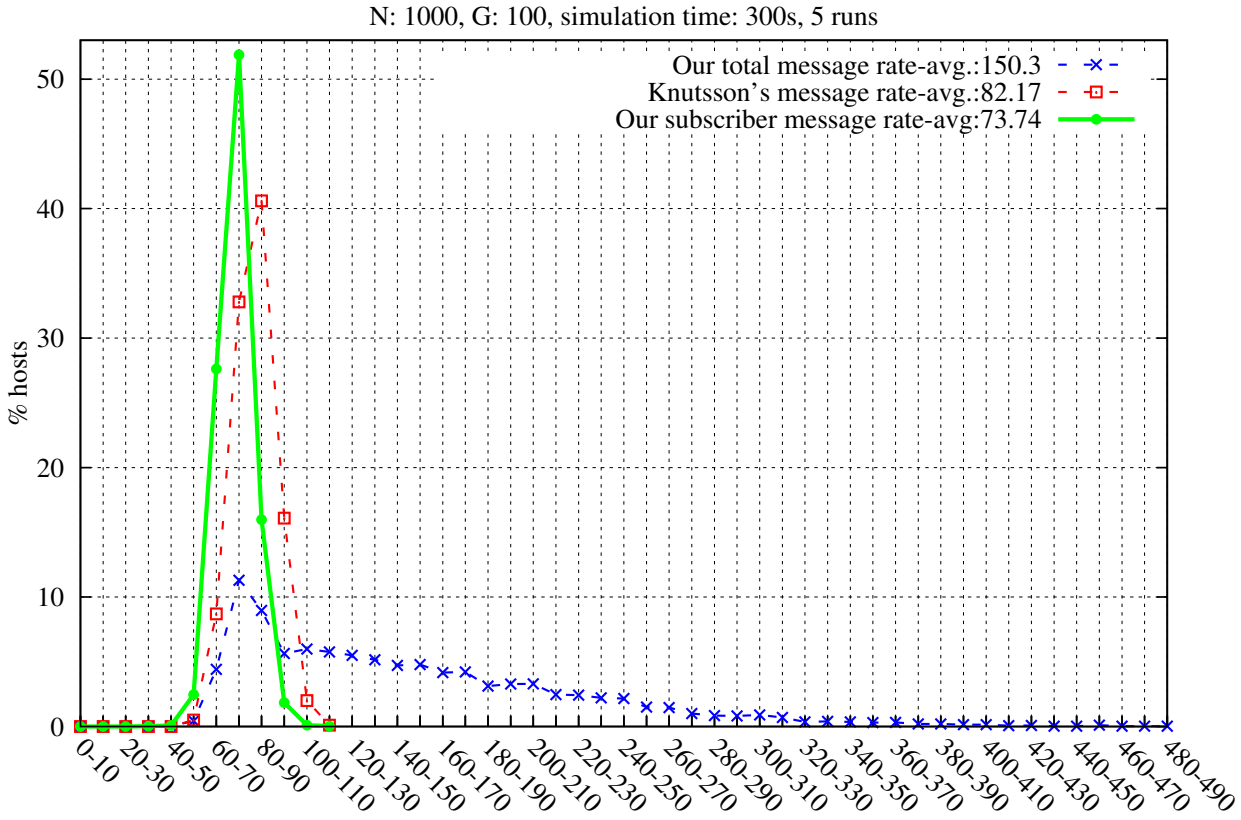

Fig. 3: Distribution of total message rates vs. distribution of subscriber message rates

66.7 subscriber position messages/s. If this host is also a coordinator for another multicast group, then it receives about $66.7 + 66.7 = 133.3$ position messages/s. Approximately 10% of all hosts are coordinators, which should receive at least 133.3 messages/s. This amount of hosts can unfortunately not be identified from Knutsson et al.'s result and evaluation. Furthermore, the subscriber message rate converges to $\frac{10+1}{0.15} \approx 73.3$ [subscriber messages/s/host] for simulations with increasing number of hosts[6], which resembles the "average total message rate" measured by Knutsson et al.

Knutsson et al.'s distribution of message rate agrees with ours of the subscriber message rate significantly better than with ours of the total message rate. The difference between Knutsson et al.'s distribution of message rate and our distribution of the subscriber message rate is only about 8 messages/s. However, this small deviation can not be further resolved, since on the one hand, Knutsson et al. did not describe how the messages are measured in their experiments. On the other hand, the original code used in [1] has been lost[7].

1) Effect of system size on the average subscriber messages rate: The average subscriber message rate is the mean value of the subscriber message rates over all hosts. As the whole network download traffic is dominated by the position messages, we predict from the theoretical analysis of the case with region changes[8], that under constant population density of $d = 10$ avatars/region the average subscriber message rate converges with increasing number of hosts to $\frac{d+1}{0.15} \approx 73.3$ [subscriber messages/s/host]. The experimental result in figure 4 shows an approximately constant behavior of the average subscriber message rate when the number of hosts increases. It is about 1 subscriber message/s/host more than the theoretical expectation. This is due to the fact that in addition to the subscriber position messages the subscriber messages also contain the messages relevant to the region changes, avatar fighting and mutable object update messages.

2) Effect of system size on the average total message rate: The average total message rate is the mean value of the total message rates of all hosts. We expect that in a system with N hosts, the average total message rate scales with $\mathcal{O}(\log N)$ [messages/s/host]. This is because the number of subscriber messages is constant and we use Scribe and Pastry to route messages. Both protocols deliver a message from sender to receiver in $\mathcal{O}(\log N))$ hops on average. Our expectation is matched by the experimental result in figure 4.

[6] 99% of the subscriber messages are subscriber position messages. See also section IV-B for the expectation of subscriber position messages. We still have to rescale it by dividing by $0.15\,s$.

[7] A private correspondence with Prof. Knutsson.

[8] Cf. subsection IV-B.

Our results on the average subscriber message rate and the average total message rate support our conjecture that Knutsson et al. really only considered the subscriber messages as the total messages received by a host. Therefore they observed "very similar rates" for different numbers[9] of hosts [1].

Table III gives a summary of our results on the network load.

	Average **total** message rate	Average **subscriber** message rate
Theoretical expectation	Eq. 7, [2]&[3]: $\mathcal{O}(\log N)$	This work: $\mathcal{O}(1)$
Experimental results	Knutsson et al.:1k, 4k hosts: "Quite similar"	
	This work: 100-42k hosts: $\mathcal{O}(\log N)$	$\mathcal{O}(1)$

Tab. III: Comparison: average total message rate and average subscriber message rate

C. Effect of system size on the message routing latency

Messages that are routed through Pastry are denoted as unicast messages. As [2] shows, the average number of routing hops of a Pastry ring with N hosts is $\mathcal{O}(\log N)$, so we expect that the unicast message routing latency also scales that way. The number of routing hops of a multicast message is measured from the root of the multicast tree to each subscriber to this multicast group. Because of the *reverse path* property [7], we know that the routing latency of such kind of messages should scale exactly the same as the one of unicast messages. To confirm our expectation, we measure the numbers of routing hops of unicast and multicast messages respectively.

The results of the average number of routing hops of unicast messages as well as of multicast messages from our simulations with different numbers of hosts are practically the same, as shown in figure 5. Both the unicast routing latency and the multicast routing latency scale logarithmically with the number of hosts.

A comparison to Knutsson et al.'s result is difficult since on the one hand, different implementations of Pastry and Scribe are used, and on the other hand it is not clear how the routing latency was measured in the original work. As can be seen in figure 5, the routing latency of both, unicast and multicast messages from 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 the simulation with 1000 hosts is larger than the one with 4000 hosts. This is contrary to their analysis, that the routing latency scales logarithmically with the number of hosts. We have no explanation for this contradiction.

Table IV summarize our theoretical and experimental results on the message routing latency. It also compares our results to Knutsson et al.'s.

[9]It should be kept in mind, however, that Knutsson et al. only provided two measuring points.

	Average number of routing hops of **unicast** messages	**multicast** messages
Theoretical expectation	[2]&[3]: $\mathcal{O}(\log N)$	
Experimental results	Knutsson et al.:1k, 4k hosts: Increase	**Slight decrease**
	This work:100-42k hosts: $\mathcal{O}(\log N)$	

Tab. IV: Comparison: average number of routing hops of unicast messages as well as multicast messages

VI. Related Work

Many works concerning the scalability of peer-to-peer (P2P) based distributed virtual environments (DVEs) have been published over the last ten years. For example, SOLIPSIS [8] and VON [9] use an unstructured P2P and zone free approach, i.e. there is no fixed partitioning of the virtual world. Hosts in SOLIPSIS and VON notify each other if new neighbors are entering their areas of interest (AOIs). One of the major problems of the above proposals is the persistency of non-avatar objects. This limits the design of DVEs. When simulating a system with up to 2000 hosts, the authors of VON observed a constant behavior of network load when the population density is fixed.

On the other hand, Knutsson et al. [1] and Colyseus [10] propose structured P2P architectures. Knutsson et al.'s system model is zone based. They simulated the SimMud scenario for up to 4000 hosts. Colyseus employs two alternative DHTs: Chord [11] and Mercury [12]. With simulations of only up to 225 hosts, however, it is hard to evaluate the scalability of their approach.

Aiming to improve the scalability of DVEs, the zoned federation model [13] and MOPAR [14] suggest to utilize the benefits of both unstructured and structured P2P. Both architectures are zone based. They use a *distributed hash table* (DHT) to find the coordinator for each zone. The main difference between them is the functionality of the coordinator. Coordinators in MOPAR also connect with each other to discover zone changes of avatars. The authors of [13] observe that the message delay grows (weak) exponentially with the number of zone members on only one single zone. The authors of MOPAR don't provide any measurements about their architecture.

VII. Conclusions and Future Work

This paper investigated the scalability of structured P2P-based DVEs. To this end, we researched a prototypical virtual world based on Pastry [2] and Scribe [3], which was originally described by Knutsson et al. in [1]. The two main characteristics of this DVE, i.e. the network load and the message routing latency, scale in the same way as the underlying network overlay under the conditions about the population density outlined in section V-A.

We find out both theoretically and experimentally that under constant population density, SimMud causes a constant average subscriber message rate with increasing number of

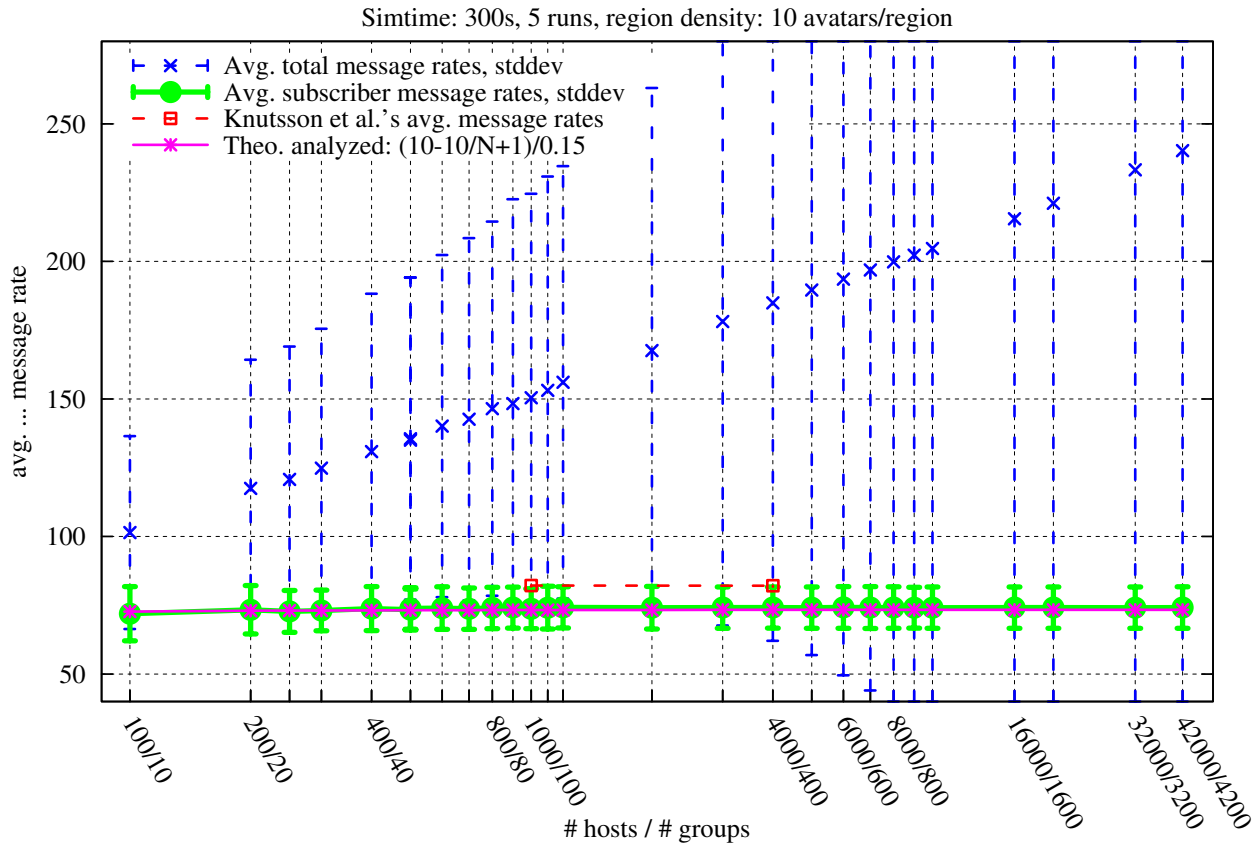

Fig. 4: # message/s/host depending on system size (semi-logarithmic scale)

hosts. Hence we expect both the network load and the message routing latency to also scale that way. This is confirmed by our experimental results, which show a logarithmic increase of the average total message rate and the average number of routing hops per message in the number of simulated hosts.

The discrepancies between our work and [1] summarized in tables III and IV can largely be resolved by a reinterpretation of Knutsson et al.'s results: They probably considered only the subscriber messages rather than all kinds of messages received by a host. As far as we know from Prof. Knutsson, the original code used in [1] is unfortunately no longer available, which makes it difficult to definitely verify our conjecture.

As the next step, we will study the problem of the scalability with increasing population density. In our work so far, we assume a uniform population distribution in the virtual world. For the future, we are looking into the cases with other population distributions, i.e. *crowding*. An evaluation of the Pastry/Scribe architecture with different application scenarios and the study of how the system handles *churn* [15] are in our research plan.

References

[1] B. Knutsson, H. Lu, W. Xu, and B. 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)*, vol. 1, 2004.

[2] A. Rowstron and P. 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)*, November 2001.

[3] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, Oct. 2002. [Online]. Available: citeseer.nj.nec.com/459613.html

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

[5] J. Chen, S. Grottke, J. Sablatnig, R. Seiler, and A. Wolisz, "Scalability of a Distributed Virtual Environment Based on a Selected Structured Peer-To-Peer Architecture," Telecommunication Networks Group, Technische Universität Berlin, TKN Technical Report Series TKN-10-003, Sep. 2010. [Online]. Available: http://www.tkn.tu-berlin.de/publications/reports.jsp

[6] J. Sablatnig, S. Grottke, A. Köpke, J. Chen, R. Seiler, and A. Wolisz, "Adam – A DVE Simulator," Telecommunication Networks Group, Technische Universität Berlin, TKN Technical Report Series TKN-08-004, Feb. 2008.

[7] Y. K. Dalal and R. M. Metcalfe, "Reverse path forwarding of broadcast packets," *Commun. ACM*, vol. 21, no. 12, 1978.

[8] J. Keller and G. Simon, "Solipsis: A massively multi-participant virtual world," in *The 2003 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2003)*. CSREA Press, 2003.

[9] S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "VON: A Scalable Peer-to-Peer Network for Virtual Environments," *IEEE Network Magazine*, vol. 20, no. 4, Jul/Aug 2006.

[10] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A distributed architecture for online multiplayer games," in *In Proc. Symposium on Networked Systems Design and Implementation (NSDI 06)*, 2006.

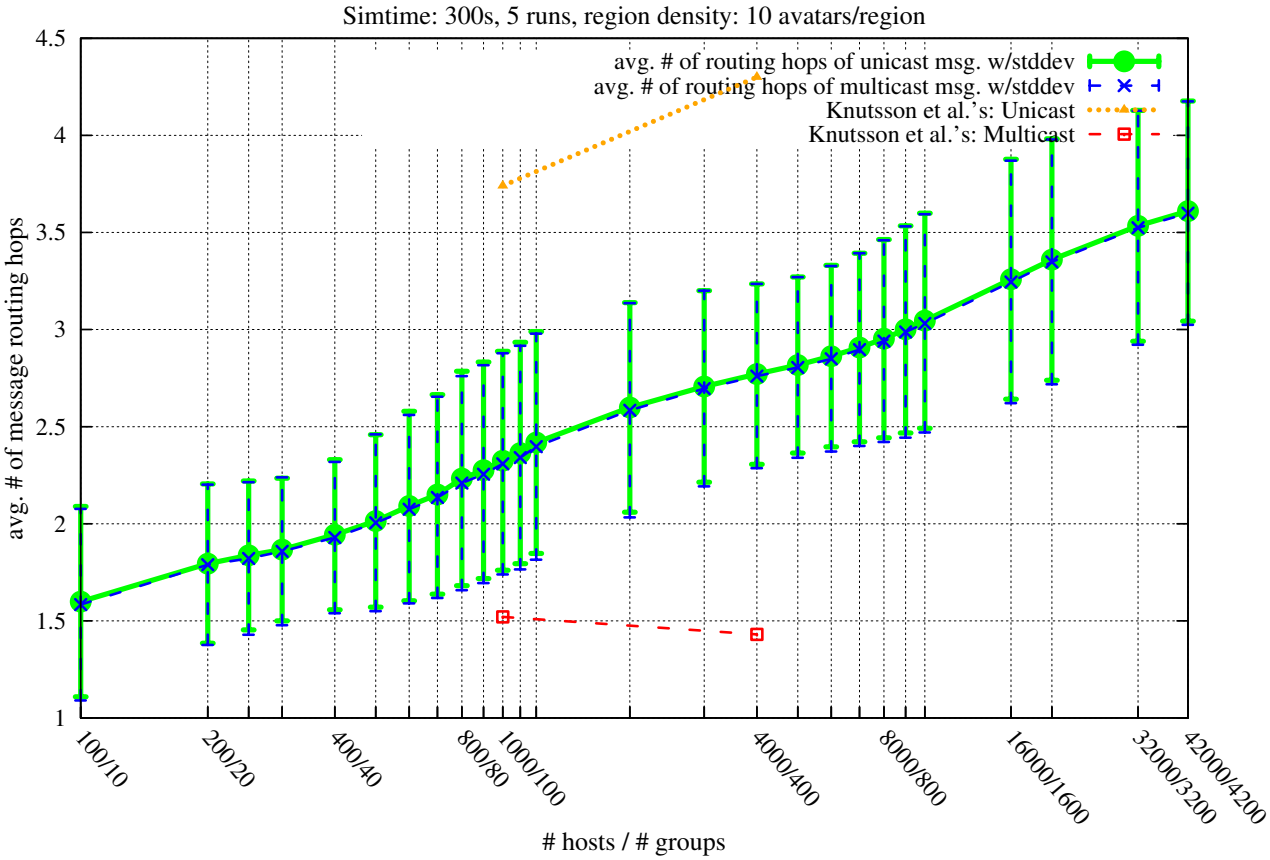

Fig. 5: Average routing latency of the dependent on the number of hosts (semi-logarithmic scale)

[11] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, August 2001.

[12] A. Bharambe, M. Agrawal, and S. Seshan, "Mercury: Supporting scalable multi-attribute range queries," in *In SIGCOMM 04*. Portland, Oregon, USA: ACM, August 2004.

[13] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned Federation of Game Servers: a Peer-to-peer Approach to Scalable Multi-player Online Games," in *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*. New York, NY, USA: ACM, 2004.

[14] A. P. Yu and S. T. Vuong, "Mopar: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games," in *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*. New York, NY, USA: ACM, 2005.

[15] S. Rhea, D. Geels, T. Roscoe, and J. K. z, "Handling Churn in a DHT," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-03-1299, 2003. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2003/6360.html