On the Trade-off between Efficiency and Congestion in Location-aware Overlay Networks - Example of a Vertical Handover Support System

Thomas Zinner, Simon Oechsner, Tobias Hossfeld, Phuoc Tran-Gia
University of Würzburg, Institute of Computer Science, Department of Distributed Systems
Am Hubland, 97074 Würzburg, Germany
{zinner,oechsner,hossfeld,trangia}@informatik.uni-wuerzburg.de

Abstract

Introducing location-awareness in overlay networks allows for a higher efficiency of the offered services, e.g., shorter search times or higher throughput. This is especially important for business applications supported by overlays. In this paper, we will illustrate how such an increase in efficiency causes congestion in an otherwise well-dimensioned locality-aware overlay using a Distributed Hash Table (DHT). In particular, we will use a Vertical Handover (VHO) support system using a Pastry substrate as an example. We will describe the occuring congestion problem and identify the underlying reasons. Thus, we describe the basic trade-off between efficiency and load-inequality. Furthermore, we introduce a solution for the specific issue observed, and conduct a performance evaluation identifying the major influence factors and optimization potential for future work.

1 Introduction

Locality-aware mechanisms in overlay networks offer several advantages: less overhead in the underlay, lower costs due to a lower link utilization, and better service characteristics, e.g., shorter search times. This makes locality-awareness interesting for cost-effective business applications.

In addition, topology-aware overlays are ISP-friendly, as they might keep most traffic in the ISP's own domain and, thus, reduce costs due to a decreased amount of transit traffic.

An example for a locality-aware application is a VHO support system, which uses a distributed database to store relevant coverage information for handover decisions [1]. In particular, we consider in this work cellular wireless networks containing UMTS and WLAN technology with the vertical handover taking place between these two technologies. A key requirement for such a vertical handover support system are short answer times in providing the required information for the handover decision itself.

In this work, we consider a Pastry-based DHT used to store and retrieve the radio coverage information at the UMTS nodeBs and WLAN access points which act as peers. In order to speed up the retrieval process, we modified the Pastry algorithm by taking into account the locality of the mobiles requesting a vertical handover [2]. In an earlier work, we showed the feasibility of this location-aware approach with a simplified model of the message processing at a peer [3].

Here, we use a more sophisticated queueing model to reflect the scarceness of resources at a peer. With this queueing model, we found that although the total system capacity provided by the peers in the overlay is excessively sufficient to handle the total load, the specific location-aware overlay layout of our approach may lead to congestion at single peers. We will here present an analysis of this issue, and show in detail how the locality-aware routing process is disturbing the load distribution in the network.

We use this example to illustrate the general trade-off between efficiency and congestion in locality-aware overlay networks. The main issue here is the high degree of specialization of the overlay we consider. However, it just serves as an instance for the problems occurring when designing and optimizing an overlay for a business application. In particular, it provides an illustration of the pitfalls of locality-aware overlay construction.

While the commonly known DHTs consider stability and load distribution in their design from the start, they are either not offering a performance necessary for critical applications, or are applied in areas where there are few constraints on their performance. For example, locating files might be an important part of a file-sharing network, but there exists no tight bound on the response time of such a system. This allows for the luxury of offering a high reliability and good load balance at the cost of performance.

The example presented in this paper shows the existance of the basic trade-off between efficiency and stability, in this case represented by the absence or presence of congestion, and that this trade-off becomes more pronounced in overlays optimized for their application.

As a result of this observation, we developed a self-optimizing algorithm in order to influence this trade-off in favor of efficiency. We will show by the results of simulation studies the feasibility of this new algorithm. In addition, we point out the major parameters which allow for further optimization.

The rest of this paper is structured as follows. Section 2 shows the background of this work and reveals the related work done on locality-aware overlay networks.

The basic system model and the enhanced queueing model, as well as the description and identification of the problem are described in Section 3. In Section 4 we describe the newly developed Congestion Avoidance Algorithm on which we conduct a performance evaluation, presented in Section 5. Finally the paper is concluded in Section 6.

2 Background

2.1 Basic System to Support VHO

The basic system considered in this paper is a database storing measured radio coverage information that is used to support VHO decisions. Such information ranges from received signal strength indicators (RSSI) to block error rates (BLER) and is location-dependent, i.e., a measurement is connected to a specific location in the complete coverage map. Instead of scanning the spectrum themselves, mobiles that want to do a VHO can now query this database for information about available radio networks around their current geo-location. This query is first sent to the attachment point the mobile is connected to, i.e. a UMTS nodeB or WLAN access point, or even originates there, in case of a network initiated handover. This attachment point then forwards the query to the database. This approach saves considerable time which would otherwise be spent by the mobile or the attachment points by scanning the environment. A possible implementation of this system is described in [1].

2.2 Location-Aware Routing Mechanism

While being described as a central database in [1], the connection of measurements and their location offers the possibility to distribute the database and exploit locality in order to make the lookup process more efficient. To this end, we proposed in [2] a distributed database based on a DHT that uses the attachment points themselves as peers, on which the measurement data is stored. These attachment points, such as nodeBs and WLAN access points, form an overlay that is used to route queries from the attachment point where they enter the system to the storage location of the queried data.

To make this overlay locality-aware, the identifiers of the peers are not chosen randomly, as known from most DHTs, but are constructed using the peers' geo-location as follows. Assume that an attachment point has the two-dimensional coordinates $(X, Y) = (x_1x_2 \dots x_n, y_1y_2 \dots y_n)$, and offers a certain radio access technology T (e.g., UMTS, WLAN), which is encoded by $t_1t_2 \dots t_k$. All x_i, y_i, t_i are encoded to base b. We consider only the two-dimensional case here for readability, although there is no such limitation in the algorithm itself.

Then, a peer's ID is constructed as $t_1 \dots t_k x_1 y_1 x_2 y_2 \dots x_n y_n$. The same process is applied to the location and the technology of the measurements that are stored at the peers in order to construct the key for each measurement. Finally, a Pastry overlay is used to form the overlay connections and to route queries. The metric to compute the distance between two peers p and q is

$$d(p, q) = \left| \sum_{i=0}^{m} (p_i - q_i) \cdot b^i \right|, \qquad (1)$$

with m being the total number of digits of an ID, and p_0, q_0 being the least valued, i.e., rightmost, digit of their respective IDs.

As a consequence, peers are responsible for and store measurements that a) were taken close to their location and b) are from the same technology as the peer. This is due to the fact that with our ID generation process, geographic proximity is translated to numerical proximity in the overlay, which is used in Pastry to determine which node is responsible for which document. Thus, the overlay is based on geographic layout.

Since Pastry uses prefix matching, the technology part as the part of the ID with the highest significance would be matched first. In order to prevent queries to be routed via geographically distant nodes, *shortcut links* were added that connect close peers of different technologies. The shortcut links are found in the following way. A peer of technology $tech(A)$ located at position (X, Y) searches for the closest peers B of different technologies $tech(B) = \overline{t_1 \dots t_k} \neq \overline{tech(A)}$ by sending a request with the key identifier $\overline{t_1 \dots t_k} x_1 y_1 x_2 y_2 \dots x_n y_n$. These shortcut links are always used first to process VHO requests, as in that case a mobile connected to $tech(A)$ wants to change to technology $tech(B)$ and will likely connect to the closest attachment point nearby.

Our first evaluations in [3] showed the basic feasibility of this approach, with traffic load being low on the vast majority of the peers. However, a small fraction of the peers received a higher load. Up until now, we did not investigate whether the reason for this was a random issue of the layout, or whether the cause for this was design-inherent, i.e., whether the locality-awareness of our approach was the reason for this. In order to show the applicability of such an overlay to business critical applications, this question has to be answered. By introducing a more detailed queue model, we could show that these local traffic hotspots are indeed consequences of our locality-aware layout, and what the reason for this development is. This is one of the contributions of this paper. Additionally, an algorithm to circumvent this problem is introduced and discussed, which is the second part of the paper.

2.3 Related Work

Topology-awareness has been a topic of interest in the P2P community. The fact that overlays as logical networks may only make inefficient use of the underlay, i.e., the physical network, has been addressed in several papers. Approaches like proximity routing, proximity neighbor selection and geographic layout have been described and covered in, e.g., [4] and [5].

Proximity neighbor selection is already a part of the original specification of Pastry [6] and was analyzed in more detail in [5]. However, its effect in the described implementation is somewhat diminished by the fact that node IDs are still generated by a hash function, and therefore the choice between peers is limited by the routing table structure.

In [5], also the problem of load imbalance is addressed on an abstract level. It is shown that in an optimized Pastry overlay, there are nodes that have to handle much larger load than others. A reactive approach with backoff messages generated by overloaded peers is proposed. In this paper, we will show how severe this problem can become in a heavily optimized network and give an example for the abstract solution presented in [5].

A topology-aware binning approach is presented in [7]. An exemplary application of this concept to the CAN [8] overlay is presented. However, the focus lies on the route lengths produced by this method, with the load distribution aspect being excluded. Topology- and locality-awareness is not only an issue in DHT-based search networks, but also in the widely used content-distribution networks, such as BitTorrent. In [9], a locality-aware scheme for selecting neighbors in this network is presented that aims at having neighbors in the same ISP domain. Projects like SmoothIT [10] or P4P [11] have similar aims, albeit in a more general context.

3 Model and Problem Formulation

3.1 Modeling the Network and the VHO Requests

Queries of VHO requesting mobiles are generated by first deciding to which technology the mobile in question is currently connected. The queries are distributed evenly among the two technologies. In the next step, a random attachment point from that technology is selected which is located at (X_{AP}, Y_{AP}). Then, a random (x, y)-coordinate in the coverage area of this attachment point is chosen. This resembles the mobile requesting coverage information for its position and therefore the location part of the query key. Due to the sake of simplicity, we use a simple disc model with radius $r = 500\,\mathrm{m}$ for UMTS and $r = 180\,\mathrm{m}$ for WLAN around the center (X_{AP}, Y_{AP}), respectively. Since the mobile is modeled to be connected via the randomly selected attachment point offering technology T_1, e.g., UMTS, it queries for information of the other technology T_2, in this case WLAN.

After such a query is generated at the selected AP, it is turned over to the locality-aware Pastry routing process, until it reaches the peer responsible for the queried coverage information, which then is modeled to look up, process and send back the information directly to the peer where the query originated.

The network layout used throughout the paper is a heterogeneous distribution of 1783 UMTS nodeBs, based on real antenna locations in an area of $250\,\mathrm{km} \times 200\,\mathrm{km}$ of a German radio network provider. The WLAN access points are placed by selecting a random nodeB and positioning the WLAN AP at a random location in the coverage area of the chosen nodeB. This algorithm generates a higher concentration of wlanAPs in areas with many nodeBs in order to reflect areas with higher traffic demands. The ratio of WLAN APs to UMTS nodeBs is fixed at roughly $2 : 1$, with 879 WLAN APs in the network. In our simulation study, the VHO requests follow a Poisson process with rate $\lambda_{total} = 100,000\,\mathrm{s}^{-1}$, which means that each peer gets around 40 messages per second in this scenario. The size of search and answer messages is assumed to be 204 bits.

3.2 Queueing Model at a Peer

The model used for the evaluation in [3] allows to approximate the search delay for finding the appropriate peer responsible for a specific VHO information. This basic model does not take queuing and request processing time into account. To remedy that we refine this model by introducing a queue and a single processing unit at each peer, illustrated in Figure 1. Now each request needs a service time t_s to be served at a peer. The service time is composed of a processing time t_p, needed for checking the routing states, and a network transmission delay t_n. The transmission delay between any two nodes is derived from their geographic distance.

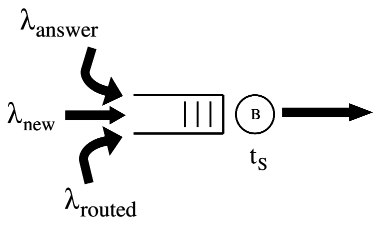

Figure 1: Queuing Model of a peer

For each peer we assume three different arrival processes. First each AP gets VHO requests from the mobile users connected to it with rate λ_{new}. Then, since each AP participates in an overlay, he also gets requests from other peers which either have to be answered or forwarded. This happens with rate λ_{routed}. Finally, it receives answers for all requests that originated at this peer. This rate λ_{answer} is equal to λ_{new}.

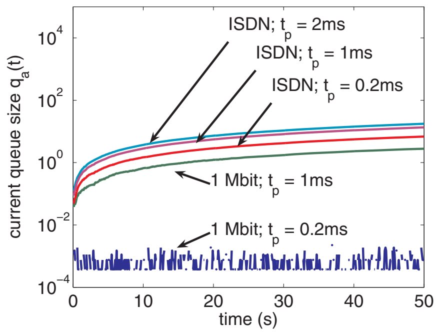

Figure 2: Average queue length over all peers

3.3 Problem Description

One result of our first evaluation was the phenomenon that a very small number of peers had to handle a much larger query load than the rest [3]. This may lead to congestion problems, depending on the amount of resources a peer has to process these messages. A more detailed investigation applying the above introduced queueing model shows that this is the case for different realistic processing speeds and connectivity of the peers. In Figure 2, the average queue size $q_a(t)$ over all peers in the network is shown during one simulation run. It is plotted for several combinations of connectivity, chosen between 1 Mbit/s and ISDN as exemplary values, and processing time t_p for one message, which is either 0.2 ms, 1 ms or 2 ms. Here, we assume that the queue size is not limited.

It can be observed that, except for the most generously dimensioned system, the queue sizes grow over time with every one of the tested combinations of resources allocated to the peers. This means that in these scenarios, at least one peer in the network is congested, leading to increasingly longer response times for queries in that region of the network.

Since the network was designed to support a time critical application, we are interested in finding out whether this occurred due to a random feature of the peer locations, which would be hard to circumvent in a real system, or whether there was an system-inherent fault at work here. In the latter case, this systematic error can be corrected. To answer this, the following questions have to be considered: How many and which nodes receive too much load? Where in the network are these nodes? Can this happen to any node, or do they have to exhibit specific characteristics?

3.4 Finding the root cause

In the first step of our analysis of the overload problem, we want to know how many peers are actually overloaded. To this end, we observed the queue sizes of each peer during a simulation run. Figure 3 shows the current queue size $q(t)$ of individual peers. For clarity, only peers with queue sizes larger than 10 messages are shown. Note that the maximum queue size q_s is limited here to 200 resulting in a maximum waiting time per peer of $200 \cdot E[t_s]$ on average. Messages which arrive at a peer at time t when the peer's waiting queue is already full, i.e. $q(t) = q_s$, have to be dropped.

Figure 3 shows that only a small number of peers, in this case five peers, experience a continuing overload situation.

A closer look at one of the congested peers reveals

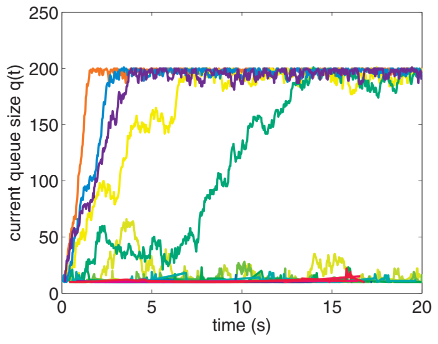

Figure 3: Queuesizes of overloaded peers

that the location aware-routing is at fault here. In order to explain how the overload situation developes, we will give a detailed analysis of one exemplary scenario that is applicable for at least all Pastry-based location aware overlays using the proximity metrc given in Section 2.

Figure 4 shows a detailed birds-eye view of a part of the used network layout. The symbols mark the geo-locations of the peers, with crosses and rings being WLAN access points and triangles being UMTS nodeBs. The overloaded peer in this scenario is the red dot, a WLAN access point. The solid blue lines illustrate zone borders where one digit in the ID space changes. In this case, the shared prefix between the peers in zone marked as Zone A and the peers in Zone B is quite short, resulting in a long zone border.

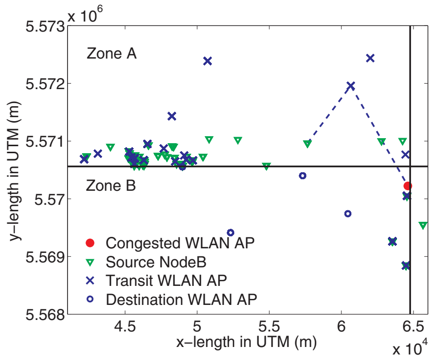

Figure 4: Map of an example overload scenario

The congested AP receives traffic from all WLAN APs marked by crosses, with the majority of these being located in Zone A, neighboring the congested APs' Zone B. The large number of peers forwarding traffic to the overloaded peer causes the congestion.

The congested peer is, due to its position and the proximity metric used in our algorithm, the primary choice for the routing table entry of the WLAN APs in the upper zone, resulting in all traffic destined for the lower zone to be routed via the congested peer. This is due to the fact that our metric assigns more weight to high-value digits of the ID, resulting in one peer being closest to all peers in another zone according to this metric. Therefore, we can identify the proximity metric used in our algorithm as the root cause for the congestion problem.

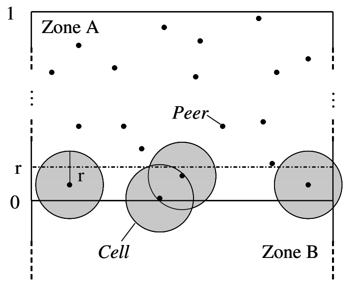

Figure 5: Abstract model for an overload situation

Since only a small number of peers is affected by the drawbacks of the chosen metric, we implemented a Congestion Avoidance Algorithm with backoff messages instead of changing the metric itself and turning down the positive effect of locality-awareness. This solution is also more generally applicable to similar problems, and therefore offers more insight. Our implementation of this algorithm is described in Section 4.

3.5 Basic Analytical Estimation of the Load at a Peer

In this section, we will analyze the probability that a node in a given zone receives queries for measurement data from one specific neighboring zone. Since this is exactly the traffic that leads to overload in our network, we can thus approximate the probability for an overload situation for a more general network scenario.

We consider a given zone A of quadratic form with a normalized side length, cf. Figure 5. A number N of peers is uniformly distributed in this zone. Each peer is the center of a circular cell with radius r. Since we want to compute the traffic a specific overload candidate in zone B receives, we can restrict the examination to the zone border between zone A and B. Only peers that have an maximum distance of r to this zone border are able to generate traffic for the zone B. For a peer P in this subset, the exact share of traffic that is actually routed to zone B depends on the size of the overlap of Ps cell with zone B.

If P has distance d to the zone border, then the cell area C that overlaps with zone B is

$$C(d) = \frac{r^2 \cdot 2 \cdot acos(\frac{d}{r})}{\pi} - d\sqrt{r^2 - d^2}. \qquad (2)$$

Accordingly, the share of traffic from this peer routed to zone B is $\frac{C(d)}{r^2\pi}$. Thus, the probability p_{AB} of a request originating in zone A to be routed to zone B can be computed as

$$p_{AB} = r \cdot E[\frac{C(D)}{r^2\pi}], \qquad (3)$$

where D is the distribution of distances to the zone border in the strip of breadth r at the zone border. For an uniform node distribution, $d(x) = \frac{1}{r}$ for $0 \leq x \leq r$. We have neglected here the smaller overlap of cells that are at the corners of zone A, however, for a large zone, where the described problem is most severe, this error is not significant.

As a result, this means that for a sufficient number of nodes, a fraction of p_{AB} of all requests originating in zone A reach one peer in zone B, due to the prefix matching of Pastry coupled with our location-aware ID generation. In general, an analogue derivation can be used to compute the load in locality-aware overlays. In that case, the amount of relayed traffic determined here by $C(d)$ has to be estimated.

E.g., in a zone with side length 50km containing 400 nodeBs that are uniformly distributed in this zone, the probability for a request to be routed to one neighboring zone is $p_{AB} = 0.002$. This may not seem much, but for a high total load in the complete zone, such a fraction may suffice to overload single peers.

4 Self-Organizing Routing Mechanism

In order to avoid the local congestion of peers, the location-aware overlay routing mechanism introduced in Section 2.2 is enhanced by a congestion avoidance algorithm (CAA). The CAA reorganizes the structure of the overlay and reroutes traffic on application layer. In particular, we consider a congested peer A of technology $tech(A)$ which receives messages from a set of peers $\mathbb{B}$. To master the overload situation, parts of the incoming traffic originated by the peers in $\mathbb{B}$ are blocked by A and have to be handled by other peers. For the selection which traffic can be routed apart A, we dinstinguish the incoming traffic at peer A into four classes:

1. *Source traffic.* Requests generated at peer A by a mobile user of $tech(A)$ requesting a VHO to $tech(B)$. This traffic has to be forwarded to a peer C of $tech(C) \neq tech(A)$ via the shortcut link $S_A = C$ of A.
2. *Shortcut traffic.* Traffic generated by peers $B \subseteq \mathbb{B}$ which send messages to peer A via their shortcut links $S_B = A$. Thus, $tech(A) \neq tech(B)$.
3. *Transit traffic.* Traffic a peer A gets from peers $B \subseteq \mathbb{B}$ of similar technology, i.e. $tech(A) = tech(B)$, forwarded via peer B's leafset L_B or routing table R_B. This traffic has to be forwarded to another peer.
4. *Destination traffic.* Requests for VHO information for which the peer A is responsible for. In that case, A has the relevant information and sends an answer message directly back to the requesting peer.

While the source and the destination traffic have to be handled by the responsible peer A, the transit and the shortcut traffic can be distributed on other peers located near the congested peer.

Our self-organizing routing mechanism works as follows. A peer A measures and predicts its current load. The simplest way is to use the current queue size $q(t)$. Note that complex filter functions, e.g., a Kalman filter, can be used to optimize the performance of the algorithm. This is out of scope of this paper where we want to describe the fundamental relationships.

As soon as $q(t)$ exceeds a certain threshold τ, peer A assumes to become congested and applies the CAA. The basic idea is that a congested peer A informs other peers B that it cannot process their messages any more. This is done by sending an overload message to these peers. This message contains close peers stored in the leafset L_A of peer A. These overload messages get a higher priority in the system such that they are processed first. Peer B then scans L_A and replaces peer A in its routing table with the most suitable peer found in L_A. As a result, peer B now uses an alternative route to relieve the congested peer. However, this might cause a larger transmission delay pointing out the trade-off between efficiency and congestion.

Our approach is to distribute the load peer A gets in two steps. First we restrict the number of peers relaying transit traffic via peer A to at most a_r. Furthermore, since the current peer might also have to serve much shortcut traffic it is not destination for, we also limit the number of allowed peers to a_s. Therefore, a local peer has to keep two additional sets O_r and O_s cointaining the peers allowed to use peer A for transit or shortcut traffic, respectively. The details on the

Algorithm 1 Congestion Avoidance Algorithm CAA

```
input : $A$ = id of local node
        $D$ = key of arriving message
        $B$ = id of last transmitter
1: if  $A$ is destination for $D$
   or $B \in O_r \cup O_s$
   or $(tech(B) = tech(A)$ and $length(O_r) \leq a_r)$
   or $(tech(B) \neq tech(a)$ and $length(O_s) \leq a_s)$
   then
2:     location-aware Pastry routing: LAPR(msg)
3:     if  $A$ is not destination for $D$
       then
4:         if $tech(B) = tech(A)$ and $length(O_r) \leq a_r$
           then
5:             add $B$ to $O_r$ {allow B in relay set}
6:         end if
7:         if $tech(B) \neq tech(A)$ and $length(O_s) \leq a_s$
           then
8:             add $B$ to $O_s$ {allow B in relay shortcut set}
9:         end if
10:    end if
11: else
12:    send overload message to $B$ {inform peer $B$ that
       it has to reorganize its state tables}
13: end if
```

current implementation of the congestion avoidance algorithm is given as pseudo code in Alg. 1.

This algorithm is a problem-specific implementation of a generally applicable congestion avoidance algorithm as it is also sketched out in [5]. This abstract algorithm would use a function $f(k_1, k_2, \ldots, k_l)$ on one or more locally measurable parameters to determine whether the local peer is experiencing an overload situation. If this is the case, the local peer responds with backoff messages to all peers not included in the sets of allowed connections.

More than one level of congestion may be defined, with different restrictions applying to the sets of allowed traffic senders. This general procedure is given in Alg. 2, with three different congestion levels as an example. In this paper, we use simply the queue length $q(t)$ as indicator for current load at a peer, i.e. $f(q(t)) := q(t)$. For the first level of congestion, we consider different settings of τ. The second level of congestion is defined via υ which is set to the maximimum queue size q_s in this paper, i.e. $\upsilon = q_s$. For this level, we use $a'_r = 0$ and $a'_s = a_s$. This means we deny any transit traffic and only allow shortcut traffic. More sophisticated adaptation strategies of both parameters might lead to a further optimized system which is a matter of future work.

Algorithm 2 Self-Organizing Routing Algorithm

```
input : $A$ = id of local node
1: if  $(f(k_1, k_2, \ldots k_l) < \tau)$ then
2:     normal routing procedure
3: else if $(\tau < f(k_1, k_2, \ldots k_l) < \upsilon)$ then
4:     Congestion Avoidance Algorithm with $(a_r, a_s)$
       {example algorithm used in this paper}
5: else if $(f(k_1, k_2, \ldots k_l) \geq \upsilon)$ then
6:     Congestion Avoidance Algorithm with $(a'_r, a'_s)$
       {Redefine thresholds in such a way that
       $a'_r \leq a_r$,$a'_s \leq a_s$ }
7: end if
```

5 Numerical Results

The numerical results were produced using an event-discrete simulator. First, we consider qualitatively the system behavior and compare the plain location-aware routing with the enhanced congestion avoiding algorithm in Section 5.1. After that, we take a closer look on the impact of the starting threshold τ of the CAA and the maximum queue size q_s in Section 5.3 and quantitatively derive the main performance indicators from the user's point of view. These are the sojourn times of VHO requests within the system and the ratio of dropped messages. The sojourn time of a VHO request is the time interval when the mobile users requests a VHO at its attachment point until the reception of the answer message at the attachment point. Both performance measures describe the subjectively experienced quality by a user when a VHO shall be conducted. As for a smooth VHO, short response times of the system are required, the maximum queue size is limited by q_s in all simulation studies.

5.1 Temporal Evolution due to Self-Organization

The functioning of the interaction between the location-aware Pastry routing and the congestion avoidance algorithm has to be characterized over time. The temporal evolution of the current queue size $q(t)$ of the peers in the system triggers the start of the CAA in Alg. 2. The maximum queue size is restricted to $q_s = 200$, while the starting threshold τ of CAA is set to $\tau = q_s/2$.

Figure 6 illustrates the current queue size for all peers which showed a queue size larger than 10 during the simulation time of 20 s. It can be seen that the overlay starts to reorganize as soon as $q(t) > \tau$ for a single peer. This is also indicated by the number of overload messages sent over time. In Figure 7, the time is discretized into bins of length 100 ms and the total number of sent overloaded messages during each time interval is counted. This allows to easily determine how long the reorganization phase takes place. It can be seen that within a few seconds the overlay is reorganized and the additional workload due to overload messages is negligible. Note that during ten seconds, there are about one million search requests which are processed by the VHO support system.

Furthermore, Figure 6 reveals that only a few peers out of the around 2,600 peers experience high load.

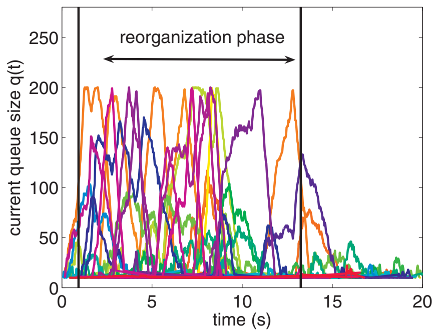

Figure 6: Current queue size of selected peers over time

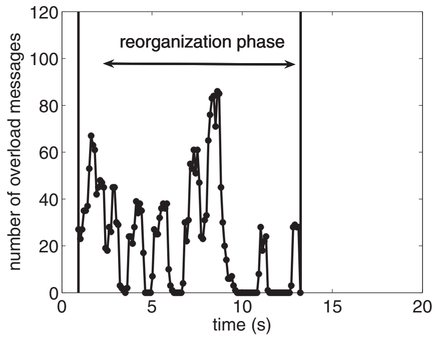

Figure 7: Number of overload messages over time

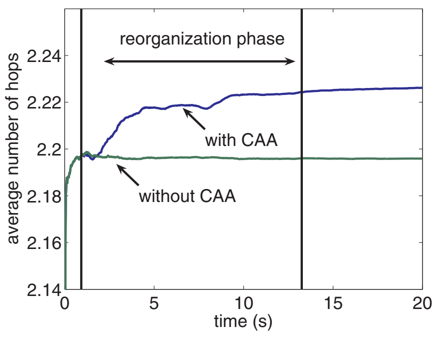

Figure 8: Avg. number of hops over time with & w/o CAA

This confirms the findings described and derived in Section 3. In contrast to Figure 3, we observe now that a few more peers are dealing with queue sizes $q(t) > 10$. This is caused by the fact that the CAA shifts the traffic from one peer to another. Since all decisions are performed using local knowledge, i.e. a distributed algorithm is applied, the structure of the overlay is adapted over time until all peers are no longer congested and remain stable, i.e. $q(t) < \tau$. As a drawback of this necessary load distribution, the search requests, i.e. the VHO messages, are not routed on the optimal path in overlay according to the overlay metric, but use slightly longer routes in term of hops. Figure 8 shows the cumulated average number of hops over time for the self-organizing routing mechanism with and without the CAA. At time t, we compute the average number of hops for all VHO messages from the beginning of the simulation until t. In a system without CAA, the overlay structure does not change significantly, if all peers in our system, i.e. the nodeBs and WLAN APs, have joined the overlay, since we do not consider churn here. Hence, the average number of hops will converge to a certain value which is around 2.20 hops here. When using the CAA, the average number of hops slightly increases around 1%-2% to roughly 2.23 hops and remains there over time. The curves in Figure 8 depict again the efficiency of the locality-aware routing algorithm, which was already shown for the plain locality-aware routing mechanism w/o CAA in [3].

5.2 Comparison of Location-Aware and Self-Organizing Algorithm

Next, we will investigate the performance gain of avoiding congestion at the cost of a slightly increased number of hops. In particular, we take a look at the sojourn times of search requests, as well as at the drop ratio of incoming messages at a peer. Therefore, we compare the performance metrics for the locality-aware Pastry routing (LAPR) algorithm and the self-organizing routing (SOR) algorithm with CAA.

Generally, the longer overlay routes established by the CAA prolong the search times, while the lower block probabilities and shorter waiting times in full queues lead to a higher QoS for the offered search service. Therefore, it is of interest which of these influences is stronger, especially regarding the request sojourn times.

Figure 9 shows the cumulative distribution function (CDF) of the sojourn time for the system using the locality-aware routing mechanism with and without CAA. In addition, we varied the maximum queue size q_s from 50 to 100 which affects the starting thresh-

Table 1: Blocking probabilities and average hops comparing location-aware routing and self-organizing routing

	$q_s = 20$		$q_s = 50$		$q_s = 100$	
	p_b	hops	p_b	hops	p_b	hops
w/o	3e-3	2.19	3e-3	2.20	2.8e-3	2.20
$\tau = q_s$	3e-4	2.22	1e-4	2.22	1.7e-5	2.22
$\tau = q_s/2$	3e-4	2.23	1e-4	2.22	1.4e-4	2.22

old $\tau = q_s/2$. Since only a small number of peers experience overload, the CDFs differ only for the last few percent of the cumulative distribution. Nevertheless, this range is of interest, since for the chosen application, the 'five nine' concept known from communication providers applies. Therefore, just the interval from 0.97 to 1 is shown on the y-axis.

It can be seen that the sojourn times for the system with CAA are clearly smaller, independent of the applied maximum queue size. The first plateau in the CDF curves of the system without CAA stem from the fact that newly arriving messages at congested peers will have to wait for q_s messages until they get served. The second plateau is caused by VHO requests which are generated in the coverage area of a congested peer. In that case, the congested peer has to process the VHO request and sends it via the shortcut to the different technology. After some time, the answer message is sent to him. This means this VHO request has to pass through the entire queue twice. Beyond this second plateau in the CDF curves without CAA, a combination of VHO requests generated at congested peers and relaying traffic via other congested peers occurs. Additionally, long end-to-end paths cause might of course increase the total sojourn time.

Table 1 shows that the drop ratio p_b is about a factor of 10 times smaller for the system with CAA than without CAA. On the other hand, the average number of hops is slightly smaller for the system with CAA than without CAA. It has to be noted that we consider here a simulation run of length 20 s while the reorganization phase with CAA takes about 15 s. This means that the impact of the reorganization phase during which the queues are more loaded is quite large. In the long term, the blocking probability will tend towards zero and the CDF of the sojourn time will converge.

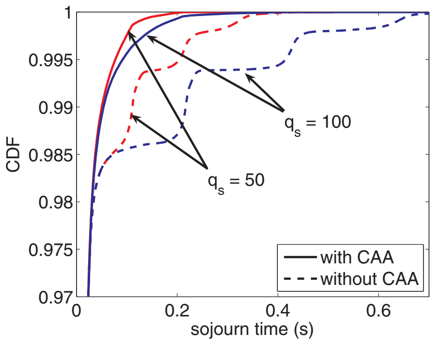

Figure 9: Sojourn times with and without CAA

5.3 Performance Study of Starting Treshold τ and Queue Size q_s

After the qualitative statements of the general behavior of the CAA algorithm, we now consider two parameters in more detail, the starting threshold τ and the maximum queue size q_s.

We consider the drop ratio and the 99% quantile of the sojourn time of messages and vary the maximum queue size q_s from 20 to 100 in steps of 10. We repeated each simulation scenario ten times in order to get statistically credible numerical results. Figure 10 shows the average drop rate when using CAA for $\tau = q_s/2$ and $\tau = q_s$. For the study with $\tau = q_s/2$ we additionally plotted the 99% confidence levels. As we can see, the average drop ratio clearly decreases for larger maximum queue sizes. However, there is no strict statement possible for the curves $\tau = q_s/2$ and $\tau = q_s$, as the average values for both curves lie within the confidence intervals. Nevertheless, we can clearly see that the average drop ratio with CAA is much better than for the same scenarios without CAA.

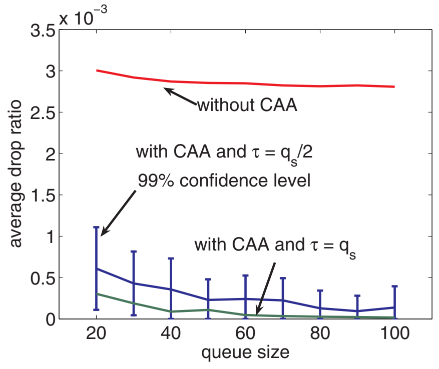

Figure 10: Average drop ratio of messages depending on maximum queue size q_s

The same is true when considering the 99% quantiles $t_{99\%}$ of the sojourn time. $t_{99\%}$ increases in this case with the maximum queue size. For the system without CAA, we can see a linear relationship which can be fitted by $t_{99\%} = 5.8\,\text{ms} + 2.1\,\text{ms} \cdot q_s$. For the system using CAA, the quantiles only slightly increase,

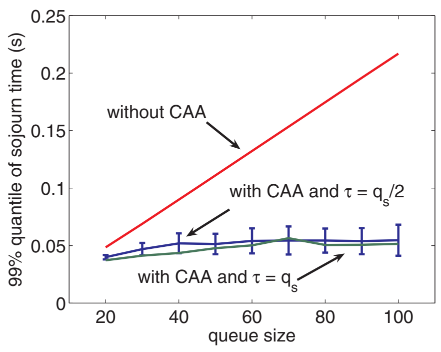

Figure 11: 99% quantile of sojourn time depending on maximum queue size q_s

whereas no clear differentiation is possible for different starting tresholds. Summarizing, we see that the CAA leads to a significant performance gain and can be used to cope with congestion in locality-aware overlays.

6 Conclusion

In this work we analyzed a location-aware Pastry-based P2P overlay architecture for supporting vertical handover in a Beyond 3G network. In contrast to existing work, a sophisticated queueing model was used to reflect the scarceness of resources at a peer. Due to the location-awareness of the system, congestion occured at a small number of peers, affecting the QoS of the offered lookup service.

We analyzed the reasons for this congestion and could show that it is a result of the locality-aware metric used in the overlay construction. The presented causes and effects are partially applicable to location-aware overlays in general. The shown trade-off between efficiency and congestion should be kept in mind when designing an performance-oriented overlay network.

We described an algorithm allowing the congested peers to distribute the transit traffic among other peers. Additionally, we conducted a performance evaluation of the enhanced model including the congestion avoidance algorithm. It turned out that the presented algorithm has the ability to reduce packet loss and overall sojourn times. This offsets the marginally longer routing paths introduced by the load redistribution.

Future work has to deal with a further optimization of the presented algorithm including on-the-fly estimations of the actual load and an improved identification of overload. Furthermore the limiting parameters for the confirmed peers should be allocated dynamically. Apart from that, we plan to investigate the described trade-off in other networks.

Acknowledgment

This work has been performed partially in the framework of the EU ICT Project SmoothIT (FP7-2007-ICT-216259) and of the DFG Project MobileP2P (TR 257/22-1).

References

[1] M. Siebert, M. Lott, M. Schinnenburg, and S. Göbbels. Hybrid information system. In *Proceedings of IEEE Semiannual Vehicular Technology Conference*, Milan, Italy, May 2004.

[2] T. Hoßfeld, S. Oechsner, K. Tutschku, F. Andersen, and L. Caviglione. Supporting vertical handover by using a pastry peer-to-peer overlaynetwork. In *MP2P'06*, Pisa, Italy, 3 2006.

[3] T. Hoßfeld, S. Oechsner, K. Tutschku, and F. Andersen. Evaluation of a pastry-based p2p overlay for supporting vertical handover. In *IEEE WCNC*, page 6, Las Vegas, NV, USA, apr 2006.

[4] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of dht routing geometry on resilience and proximity. In *SIGCOMM '03*, pages 381–394, New York, NY, USA, 2003. ACM.

[5] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, Topology-aware routing in structured peer-to-peer overlay networks, Tech. Rep. MSR-TR-2002-82, Microsoft Research, One Microsoft Way, Redmond, WA 98052, 2002.

[6] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218, 2001.

[7] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *INFOCOM'02*.

[8] S. Ratnasamy, P. Francis, M. Handley, R. M. Karpand, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.

[9] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *Proceedings of the 26th IEEE ICDCS*, page 66. IEEE, IEEE Computer Society Washington, DC, USA, 2006.

[10] The SmoothIT project. http://smoothit.org/, 2008.

[11] The P4P project. http://www.dcia.info, 2008.