An Adaptive Overlay Network for World-wide Geographic Messaging

Frank Dürr, Kurt Rothermel
Institute of Parallel and Distributed Systems (IPVS), Universität Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany
{*frank.duerr,kurt.rothermel*}*@ipvs.uni-stuttgart.de*

Abstract

In this paper, we propose an overlay network supporting world-wide geographic messaging. Our approach is based on hierarchical symbolic coordinates like `/usa/fl/miami/`*. Although hierarchical network topologies lend themselves to the implementation of such overlay networks, they may lead to bottlenecks at the root of the hierarchy, long message paths, and inefficient bandwidth utilization. To avoid these problems, we propose an overlay network that adapts its structure to the users' communication patterns by dynamically adding "shortcut" links to the hierarchy leading to a routing mesh. We present an algorithm that carefully selects shortcuts based on their utility to assure short message paths on the one hand and to reduce the induced overhead on the other hand. Through simulations we show that this approach decreases the average path length significantly and reduces network load to about 50% compared to hierarchical routing.*

1. Introduction

Geographic messaging (geocast) allows for sending messages to hosts located at certain geographic target areas. This communication paradigm is useful for a wide range of applications, for instance, dissemination of information with geographically limited relevance like traffic and tourist information, location-based advertisements, or warning messages.

Our work is based on hierarchical symbolic coordinates like `/usa/fl/miami/`, which denotes the city of Miami in the State Florida in the USA. Such symbolic coordinates have proven to be an intuitive, pragmatic, and at the same time efficient alternative to geometric addressing. Moreover, they are based on simple hierarchical location models rather than complex geometric models [1, 2].

For a large-scale geocast service used by a large number of hosts spread all over the world, efficient geocast message distribution becomes an important issue. Different types of overlay networks have been proposed in the literature for this purpose, e.g., [2–4]. Because of the hierarchical nature of symbolic coordinates, one could simply use a routing tree based on a geographic partitioning of the service area, for instance, a routing tree of country routers, state routers, city routers, etc. However, such hierarchical topologies may lead to bottlenecks at routers near the root of the tree and long message paths resulting in a waste of bandwidth.

Therefore, the approaches in [2–4] start with a routing tree and add "shortcut" links between routers allowing for direct connections to certain target areas. This turns the routing tree into a mesh increasing bandwidth-efficiency and scalability. However, the previously proposed approaches are static in the sense that they do not adapt the routing mesh to the users' current communication needs dynamically. Shortcuts are added according to simple heuristics. Either shortcuts are installed "top down", i.e., shortcuts to top-level locations like countries are preferred to shortcuts to smaller locations like cities [2], or shortcuts to nearby locations are preferred to distant shortcuts [3,4]. Although these are reasonable heuristics for certain communication patterns—in particular, patterns where most messages are sent to areas close to the sender—they may be inefficient for situations not fitting the anticipated pattern.

The main contribution of this paper is an *adaptive* shortcut selection algorithm. In contrast to static schemes, shortcuts are selected based on the current popularity of geographic target areas, i.e., shortcuts are added and removed dynamically depending on the users' current communication needs. The result is an adaptive overlay network in form of a routing mesh that dynamically adapts links to *any* communication pattern. Our evaluation results show that even for small numbers of shortcuts this strategy significantly decreases path lengths, thereby reducing the overall network load to about 50% compared to tree-based routing. At the same

Published in Proceedings of the 22nd IEEE International Conference on Advanced Information Networking and Applications (AINA 2008); pages 875-882, Gino-wan, Okinawa, Japan, March 2008.
© IEEE 2008
http://dx.doi.org/10.1109/AINA.2008.67

time, it only induces small communication, space, and computational overhead for shortcut management.

Although we focus on geocast, the basic principle of our approach is also beneficial to other fields. For instance, overlay networks of servers managing large numbers of mobile objects [5] can be optimized by adding shortcuts to servers of frequently queried areas. Moreover, peer-to-peer overlay networks can be improved by adding links to frequently addressed nodes.

The remainder of this paper is structured as follows. In Sec. 2, we present related work. In Sec. 3, the underlying addressing scheme and system model are introduced, followed by the basic routing algorithm and network architecture in Sec. 4. Our adaptive shortcut selection algorithm for optimizing the overlay network is presented in Sec. 5. Finally, the approach is evaluated in Sec. 6, before the paper is concluded in Sec. 7.

2. Related Work

Our goal is a geocast routing infrastructure in form of an overlay network that can be used for world-wide geographic messaging (in contrast to approaches for mobile ad hoc networks [6], which operate in limited geographic domains). Different geocast infrastructures have been proposed in the literature. *Native geocast routing* algorithms are implemented on the network layer. In [7], different geographic extensions to distance-vector and link-state routing protocols as well as a multicast-based geocast routing algorithm are proposed. These approaches lead to "optimal" distribution trees with respect to the topology of the IP network, however, the IP network has to be modified, which is not an easy task as the slow adoption of IPv6 shows.

Overlay geocast routing avoids this problem by implementing geocast on the application layer using an overlay network of geocast routers on top of the IP network. For geometric addressing, [7] and [8] proposed hierarchical and flat geocast overlay networks, respectively. Although geometric addressing is powerful since almost any target area can be addressed, a detailed, possibly three-dimensional model is required and routers have to use complex geometric operations to forward messages. Therefore, we focus on symbolic addressing as a pragmatic and intuitive alternative without the need for complex geometric models and operations.

For symbolic geocast, different hierarchical overlay networks and heuristics for adding shortcuts have been proposed as already described in Sec. 1 [2–4]. The basic idea of adding shortcuts has also been applied to peer-to-peer overlay networks. In [9], so-called "expressways", which correspond to shortcuts, are added to a content-addressable overlay network (CAN) based on a n-dimensional space. A node sets up expressways to certain predefined zones of this space. Similar to [3, 4], a node adds more links to nearby zones than to distant zones. In contrast to these "static" heuristics for shortcut selection, we want to adapt shortcuts dynamically to the current popularity of targets. Our adaptive shortcut selection strategy may either be used stand-alone, or it may supplement the above heuristics to make them robust to unanticipated communication patterns.

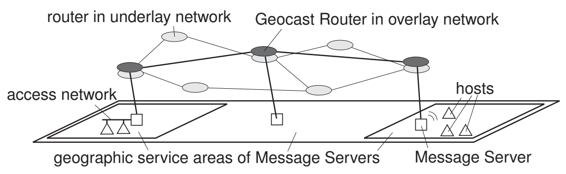
router in underlay network
Geocast Router in overlay network
access network
hosts
geographic service areas of Message Servers
Message Server

Figure 1. System model

3. Addressing Scheme and System Model

Our approach is based on a hierarchical symbolic location model [1]. The location hierarchy consists of a set of locations L and is structured according to the spatial inclusion relation between locations. For two locations l_1 and l_2 it holds $l_1 < l_2$, if l_2 spatially contains l_1. l_1 is called a *descendant* of l_2; l_2 is an *ancestor* of l_1. Direct descendants or ancestors are called child and parent locations, respectively. Each location has a unique *symbolic address* like `/us/ny/new-york-city/` denoting New York City. Locations are used to define service areas and message target areas.

The three components of our architecture are hosts, message servers, and routers (cf. Fig. 1): *Hosts* represent the mobile or stationary recipients of geocast messages. Messages have to be delivered to all hosts within the addressed target area. *Geocast Message Servers (GMS)* are responsible for message distribution to all hosts within certain access networks. A GMS has a geographic *service area* whose size is equal to the area covered by its assigned access network. *Geocast Routers* are responsible for forwarding geocast messages from the sender to the GMSs whose service areas overlap with the target area of the message. Geocast routers are arranged in an overlay network and exchange messages using the UDP service offered by the underlying IP-based Internet infrastructure.

In this paper, we focus on the efficient forwarding of messages via the geocast overlay network from the sender to GMSs, i.e., to the local networks containing the hosts. Efficient local message distribution protocols for GMSs are beyond the scope of this paper.

4. Overlay Network and Forwarding

In this section, we present the detailed architecture of the overlay network, the mechanisms to set it up, and the algorithm for forwarding messages.

The basic idea of our geocast approach is to have a set of overlay routers with geographic services areas. Each symbolic location is associated with one overlay router. We call this router the *designated router* of the area. For instance, there exists an earth router, country routers, state routers, city routers, etc. The symbolic location associated with a router, say r, is called its service area, $\mathrm{s}(r)$, which is defined when a router is configured. Since it is not reasonable to assign a different physical router to each single location, a physical router can implement multiple virtual routers. For instance, a building router also implements virtual floor routers for this building as long as no designated floor routers have been configured. The term “router” denotes a virtual router unless otherwise noted.

Basically, these geocast routers form a *routing tree* according to the spatial inclusion relation between service areas. That is, the designated router of some location, l, has a link to the designated router of the parent location of l and vice versa. For instance, state routers are child routers of country routers, city routers are child routers of state routers, etc. To add a new router to the tree, a bootstrapping process is performed. After this process, the newly integrated router knows the UDP addresses and service areas of parent and child routers and vice versa. In [3] we describe this process in detail.

Over time, the routing tree is extended by shortcuts that are used to send messages directly to certain target areas, by-passing routers of the routing tree. Adding shortcuts leads to a *routing mesh*. This mesh may change continuously as shortcuts may be added and removed dynamically. Within this mesh the routing tree is sort of a backbone guaranteeing that a path to every target area exists. Links of the routing tree are used whenever no path via a shortcut exists.

When a router decides to set up a shortcut to a target area, say τ, it sends a TargetRouterSolicited(τ) request. This request is passed through the routing tree according to the forwarding algorithm sketched below. In order to reduce communication overhead, the request is piggy-backed onto the next message forwarded to this area. When the designated router of τ receives the request, it returns a reply including its UDP address to the requester. Upon receipt of this reply, the source router adds a new entry to its forwarding table to establish the shortcut. Since shortcuts are unidirectional, only the source router records this shortcut’s information.

Messages are forwarded in two phases (for a detailed description see [3]). In phase 1, the message is forwarded to the designated router of the target area. For instance, a message to New York City is forwarded to the New York City router. The forwarding algorithm chooses the locally known router whose service area address is the longest prefix of the target area address as next hop. This can be either a router reachable via the routing tree or a shortcut.

In phase 2, which is started by the designated router of the target area, the message is distributed among all designated routers with service areas within the target area by simply forwarding the message down the sub-hierarchy rooted at the designated router of the target area. For instance, the New York City router forwards the message to all borough routers like the Manhattan router, these routers forward the message to district routers, etc. In phase 2, the routers also forward the message to the GMSs whose access networks cover parts of the addressed area.

5. Adaptive Shortcut Selection

The main objective of shortcuts is to reduce the network load by shortening the average communication path length. However, shortcuts do not come for free. They cause extra communication overhead for TargetRouterSolicited reply messages. Moreover, they increase the length of forwarding tables, which adds extra space and computational overhead. Consequently, the number of shortcuts is to be limited and only those with the highest expected utility are to be established.

One important factor for a shortcut’s utility is the popularity of its target area. If shortcuts are used frequently, the extra set-up costs amortize rapidly. Of course, the popularity of targets may differ from router to router and may change over time. Therefore, routers are required to individually monitor the popularity of targets and dynamically decide which shortcuts to establish and to replace. Another factor is the number of hops that can be saved by using the shortcut. Obviously, the utility of a shortcut increases with the number of saved hops and its popularity.

5.1. Target Utility

Next, we introduce the utility of a target to express the estimated benefit should a shortcut be established to a target, say τ. For this, we use the utility function, $\mathrm{util}_t(\tau)$, proposed by Lee et al. for web caching in [10]. In our context, $\mathrm{util}_t(\tau)$ defines the utility of target τ at time t. Each router, r, calculates utilities independently, i.e., τ’s utility may differ between different routers. We say a target is *referenced* at time t if r forwards a message to τ at time t. t is defined by r's logical clock ticking whenever r forwards a message. Assume that r forwards a message to target τ' at time t'. Then the utility of another target τ at t' is defined based on the previous reference time t of τ as follows:

$$\mathrm{util}_{t'}(\tau) = \begin{cases} 0 & \text{if } \tau \text{ has never been referenced} \\ \mathrm{weight}(\tau) & \text{if } \tau = \tau' \text{ and } \tau \text{ is referenced for the first time} \\ \mathrm{util}_t(\tau) \times \mathrm{F}(t'-t) + \mathrm{weight}(\tau) & \text{if } \tau = \tau' \text{ and } \tau \text{ is referenced for the } 2^{nd}, 3^{rd}, \ldots \text{ time} \\ \mathrm{util}_t(\tau) \times \mathrm{F}(t'-t) & \text{if } \tau \neq \tau' \end{cases} \tag{1}$$

Function $\mathrm{F}(x) = (\frac{1}{2})^{\lambda x}$ with parameter λ defines the influence of reference recency and frequency. For $\lambda = 0$, only the frequency of references is considered by counting the number of past references. For large values of λ, emphasis is put onto recency. With this definition, each reference in the past contributes to the current utility of τ, although only the utility at the last reference time has to be stored. This reduces space overhead considerably since the reference history of a target can be stored with constant space complexity.

Function weight(τ) assigns an individual weight to each target τ. We define the weight of τ as the number of saved overlay hops if a shortcut is established to τ. The problem is that weight(τ) has to be defined *before* a shortcut to τ is installed. However, without detailed topological knowledge about the overlay network, a router can only estimate path lengths. Still we can give a good estimation of the saved hops based on a router's symbolic service area address and the target address. Consider the router with the service area `/usa/fl/miami/` and the target area `/usa/ny/new-york-city/`. From the structure of these addresses, we can derive that the message has to traverse 3 (virtual) routers to reach the target via the routing tree without using any shortcuts, namely the Florida State Router, the US country router, and the New York State router. Thus, a direct shortcut to New York City would save 3 hops. This is only an estimation since a physical router may implement several virtual routers and shortcuts between other routers may exist reducing the number of hops actually saved by the potential shortcut.

5.2. Shortcut Selection Algorithm

Based on target utilities, router r can decide for which targets shortcuts are to be established. For that we distinguish between hot, warm, and cold targets. There is only a limited number of *hot targets*, for which r sets up shortcuts. *Warm targets* have been referenced in the past, but they are not considered hot yet. For warm targets, which may become hot later, r records a utility but does not establish a shortcut. For *cold targets*, r maintains no information either because they have been referenced never before or a long time ago and hence have been evicted from the targets buffer.

Hot and warm targets are maintained in the so-called *targets buffer*, B, which is divided into two areas, the hot and warm targets area. B_{hot} denotes the *hot targets area*, which is of limited size $|B_{\text{hot}}| \leq maxsize_{B_{\text{hot}}}$. Typically, $maxsize_{B_{\text{hot}}}$ is small (< 50) to strictly limit the communication and computational overhead induced by shortcuts. B_{warm} denotes the *warm targets area*, which is sort of a "warm-up" area allowing targets to heat up and possibly become hot. In particular, this is important if parameter λ is set such that the reference frequency strongly impacts the utility of targets since then a shortcut needs several references to increase its utility value significantly. Each entry e in the targets buffer contains the following fields: the symbolic address of the target, the utility value of this target, and the target's shortcut, i.e., the UDP address of the target's designated router, provided the target is hot.

Since the utility values of targets change over time, warm targets may become hot and vice versa. Consequently, we need a replacement condition that defines when a warm target in B_{warm} replaces a hot target in B_{hot}. The replaced target is moved form B_{hot} to B_{warm}. For the new hot target, a shortcut is added; the shortcut of the replaced target is removed. An intuitive approach would be to keep the $maxsize_{B_{\text{hot}}}$ targets with the highest utility values in B_{hot}. The target with the minimum utility in B_{hot} would be replaced when there exists a target with a higher utility in B_{warm}. The problem with this approach is that a shortcut is set up immediately when a warm target's utility exceeds the minimum utility in B_{hot}. Since the utility of a target decreases monotonically between two references to that target, it may happen that targets that are referenced *sporadically* have already been removed from B_{hot} before they are referenced a second time. Obviously, for such "volatile" targets the overhead for setting up a shortcut does not pay-off since the shortcut is replaced before it can be used. This is especially the case if the utility function stresses recency as this leads to high utility values that drop rapidly if no further reference follows shortly. However, considering recency is important to be able to react rapidly to changing target popularity.

The basic idea of our replacement strategy is to test whether a target's utility is "stable" enough to constantly stay in B_{hot} for a period of $minref$ consecutive references to the target before it is considered hot and a shortcut is set up to it. In other words, the target has to "prove" *minref* times that is has the potential to be a stable shortcut target that is unlikely to be evicted before the costs of shortcut setup have amortized. The distinction of volatile and stable shortcuts is the main difference to traditional cache replacement strategies used in web caching, database buffer management, etc. Caching web or database pages does not cause communication overhead since the page has to be fetched from the server or database anyway before it can be accessed. In contrast, setting up a shortcut is optional since messages can always be sent via the routing tree. Deciding about the stability of a target *before* actually setting up the shortcut has impact on communication overhead.

We now give a precise definition of the so-called *hot target condition* that defines when a previously warm target becomes hot and hence replaces a target from B_{hot}. First, we define the *Least-Hot Target*, $\tau_{\min,t}$, at time t as the target with the minimum utility in B_{hot} (for a concise description, we only consider the steady state when B_{hot} is already filled):

$$\forall \tau \in B_{\text{hot}} : \text{util}_t(\tau_{\min,t}) \leq \text{util}_t(\tau) \qquad (2)$$

Hot Target Condition: Let $t_{\tau,j}$ be the time of the j-th reference to target τ. n denotes the the number of times τ has been referenced so far. Then target τ changes from the warm to the hot state at time $t_{\tau,n}$, if

$$\forall t \in [t_{\tau,n-minref+1}, t_{\tau,n}] : \text{util}_t(\tau) > \text{util}_t(\tau_{\min,t}) \qquad (3)$$

If the hot target condition is fulfilled for some target τ at time t, τ from B_{warm} replaces $\tau_{\min,t}$ in B_{hot} and vice versa. For greater *minref*, the chance of selecting only stable shortcuts increases, thus, the overhead decreases. On the other hand, the algorithm reacts slower to changes of a target's popularity. For $minref = 1$, no stability check is performed at all.

The definition of the utility function assures that this check can be implemented efficiently. Note that the utility of target τ decreases monotonically between two consecutive references to τ. However, two targets that are not referenced do not change their order defined by their utility values! Therefore, we only need to compare the utility of a warm target τ to the utility of the Least-Hot Target when τ is referenced and not on each reference to other targets between two consecutive references to τ. In other words, if util(τ) is greater than util$(\tau_{\min})$ at times $t_{\tau,n-1}$, $t_{\tau,n}-1$, and $t_{\tau,n}$—i.e., at the last reference time, just before and at the current reference—then it also must have been greater than util$(\tau_{\min})$ at any time between $t_{\tau,n-1}$ and $t_{\tau,n}$.

We implemented B_{hot} as a heap in order to be able to find the Least-Hot Target to be evicted and compared quickly. This heap has a complexity of $O(\log|B_{\text{hot}}|)$ for update operations and therefore can be considered efficient to manage the small number of hot targets even for high message rates. Note that computational efficiency is crucial, since our algorithm for shortcut selection is an online algorithm deciding in real-time about relevant shortcuts whenever a message is forwarded.

Also the size of B_{warm} is inherently limited by router memory, which leads to situations, where a warm target is to be replaced by a formerly cold target. Usually, $maxsize_{B_{\text{warm}}}$ is much greater than $maxsize_{B_{\text{hot}}}$ and may contain thousands of entries. Therefore, we used a simple LRU buffer for B_{warm} with a complexity of $O(1)$ for replacement operations. Using LRU, it is not guaranteed that the target with the least utility is evicted from B_{warm}. However, it is very likely that a target with a small utility is evicted, since targets that have not been referenced for a long time also have a small utility.

The following algorithm shows the complete shortcut selection algorithm execute by each router r:

```
On receiving message with target area $\tau$ do
  $t_{\text{now}} \leftarrow t_{\text{now}} + 1$ // increase logical time
  // update utility of target
  if no entry $e \in B$ with $e.target = \tau$ exists then
    // $1^{st}$ reference of $\tau$
    create new buffer entry $e$ with
      $e.target \leftarrow \tau$; $e.shortcut \leftarrow undef$
      $e.util \leftarrow \text{weight}(\tau)$; $e.t_{n-1} \leftarrow t_{\text{now}}$; $e.refcnt \leftarrow 0$
    put $e$ into $B_{\text{warm}}$ // LRU replacement policy
  else
    // $2^{nd}, 3^{rd}, \ldots$ reference of $\tau$
    $e \leftarrow$ buffer entry $\in B$ with $e.target = \tau$
    $util_{t_{\tau,n}-1} = e.util$
    $e.util \leftarrow e.util \times \text{F}(t_{\text{now}} - e.t_{n-1}) + \text{weight}(\tau)$
    $e.t_{n-1} \leftarrow t_{\text{now}}$
  fi
  // update target position in buffer
  if $e \in B_{\text{hot}}$ then
    update position of $e$ in $B_{\text{hot}}$
  else // $e \in B_{\text{warm}}$
    // check whether $\tau$ fulfills hot target condition
    if $e.refcnt = 0$ then
      if $e.util \geq \text{util}_{t_{\text{now}}}(\tau_{\min})$ then
        $e.refcnt \leftarrow 1$ fi
    else // $e.refcnt > 0$
      if $util_{t_{\tau,n}-1} > \text{util}_{t_{\text{now}}-1}(\tau_{\min})$ and
         $e.util > \text{util}_{t_{\text{now}}}(\tau_{\min})$ then
        $e.refcnt \leftarrow e.refcnt + 1$
      else
        $e.refcnt \leftarrow 0$ // e is volatile
      fi
    fi
    if $e.refcnt \geq minref$ then
      // $\tau$ is hot now; replace $\tau_{\min}$ by $\tau$
      $e' \leftarrow$ buffer entry $\in B_{\text{hot}}$ with $e'.target = \tau_{\min}$
```

```
            move $e'$ from $B_{\text{hot}}$ to $B_{\text{warm}}$
            $e'.refcnt \leftarrow 0$
            move $e$ from $B_{\text{warm}}$ to $B_{\text{hot}}$
            set up shortcut to $\tau$ (set $e.shortcut$)
          fi
        fi
```

5.3. Indirect Shortcuts and Covered Shortcuts

Not only a shortcut directly aiming at the target area router shortens the path to the target area. Any other shortcut targeted at routers on the path between the source and the target area router also leads to a shorter path. We call such shortcuts *indirect shortcuts*.

Moreover, we can imagine situations where a larger area can be considered to be more popular than smaller areas below this area in the hierarchy. If for instance the Miami router forwards one message to each city in Texas, then the cities are not popular targets from the Miami router's perspective. However, the state Texas can be considered to be popular since many messages are sent to areas within Texas. Therefore, an indirect shortcut to Texas is a good choice for the large number of messages sent to "warm" city targets in Texas.

We modify our shortcut selection algorithm as follows to allow for indirect shortcuts. If a message to target τ, arrives at the designated router r of area $\mathrm{s}(r)$, then *all* locations on a path between τ and $\mathrm{s}(r)$ in the location hierarchy are referenced starting at τ. Exceptions are child and parent locations of $\mathrm{s}(r)$, which are not considered since links to parent and child routers already exist through the routing tree. For instance, the Miami router references the following targets on receiving a message to New York City: New York City, New York State, USA. Florida is not referenced since it is the parent location of Miami.

6. Evaluation

6.1. Simulation Set Up

For our simulation we use the network simulator ns-2. The underlay network is a real Internet topology of one autonomous system in the USA from [11] consisting of of 2942 routers and their geographic positions. The overlay network consists of country, state, city, and city district geocast routers. Each overlay router is linked to a single underlay router such that the designated geocast router of a geographic area is linked to an underlay router located in this area.

We address target areas at the city district level. The geocast traffic outgoing from each city district follows a Poisson distribution with an average rate of one geocast message per second. Each curve is the result of five simulation runs of 200 s duration.

We compare our adaptive shortcut selection algorithm (Adaptive Shortcut Routing (ASR)) to the static shortcut selection algorithm (Static Shortcut Routing (SSR)) proposed in [3] and to hierarchical routing using a routing tree according to the spatial inclusion relation between service areas without any shortcuts (Basic Routing (BR)). In SSR, each router statically sets up shortcuts to all ancestor locations of its service area as well as to child locations of ancestor locations. So SSR sets up more links to areas close to a router's service area and few to distant areas, whereas ASR dynamically adapts shortcuts to target area popularity.

We use traffic patterns with different target area popularity distributions: *Uniform*: All targets are equally popular. *Concentrated-1*: The popularity of targets follows a Zipf distribution. Each sender orders target areas randomly or by distance to the sender (see below). Then, the popularity $\mathrm{P}(X = i)$ of messages targeted to area τ_i with $1 \leq i \leq N_{\text{targets}}$ is defined as $P(X = i) = \frac{i^{-s}}{\sum_{n=1}^{N_{\text{targets}}} \frac{1}{n}}$. For Concentrated-1, we set $s = 1$, i.e., a sender sends about 80% of its messages to 25% of all targets. *Concentrated-2*: The popularity of targets is defined by a Zipf distribution with $s = 2$, i.e., a sender sends 95% of its messages to 1% of all targets.

In order to evaluate the influence of the fact that SSR sets up more links to geographically close areas, we combine the above popularity distributions and different geographic target distributions: *Distance-dependent target popularity*: The popularity of targets decreases as distance to the sender increases. Each sender orders targets according to their distance to the sender. Then, the popularity of targets is calculated according to the above Zipf distribution. *Distance-independent target popularity*: Each sender orders targets randomly and defines the popularity of target areas according to the above Zipf distributions.

6.2. Evaluation of Path Length

First, we evaluate ASR path lengths. As performance metric we use the *stretch factor*, which denotes the factor by which underlay network paths achieved by ASR, SSR, or BR are longer than the optimal underlay path lengths. The optimum is defined by shortest path trees rooted at the sender and pruned such that they only contain branches to GMSs in the target area.

6.2.1. Impact of Buffer Size. First, we vary the buffer size of B_{hot} for ASR. B_{warm} has a fixed size of 1000 entries. Shortcuts are only established for targets in B_{hot}. For space restrictions, we only present the results for the distance-independent distributions. Parameter λ is set to 0.05. Different geographic distributions and λ values only showed minimal differences.

Figure 2(a) shows the stretch factors for increasing sizes of B_{hot}. Without shortcuts ($|B_{\text{hot}}| = 0$), the path is about 4.7 times longer than the optimal path. Path lengths decrease rapidly for increasing buffer sizes. Even small buffer sizes of 20 shortcuts lead to paths that are on average 50% to 75% shorter than without shortcuts. If all targets are equally popular—the worst case for our adaptive shortcut selection algorithm ASR—path lengths are almost halved for $|B_{\text{hot}}| = 20$ entries. In this case, B_{hot} is filled with the (few) top-level areas like countries and states. For the Concentrated-2 distribution, ASR selects the (few) very popular city districts and almost reaches the optimal stretch factor of 1 for 20 shortcuts; for the Concentrated-1 distribution, the optimum performance is reached for 50 shortcuts.

6.2.2. Comparison of ASR to SSR and BR. Next, we parametrize ASR as follows and compare it to SSR and BR: $|B_{\text{hot}}| = 50$, $|B_{\text{warm}}| = 1000$, $\lambda = 0.05$. Figure 2(b), 2(c), and 2(d) plot the stretch factor over 200 s simulation time for the distance-independent target distributions. At $t = 0$ all routers using ASR start with an empty buffer, which then is populated during the simulation. All shortcuts are set up from the beginning for SSR.

The results show that the stretch factors of SSR do not change over time since shortcuts are not adapted, while with ASR stretch factors decrease rapidly as B_{hot} is populated with shortcuts. For both distance-independent target distributions, SSR cannot achieve a stretch factor less than 2.3 (cf. Fig. 2(c), 2(d)). Since a significant portion of message is sent to distant targets, SSR can often only use shortcuts to top-level areas like states. In contrast, ASR continuously adapts its shortcuts and shows significantly smaller stretch factors. For the uniform distribution, ASR achieves paths that are about 10% shorter on average than using SSR (cf. Fig. 2(b)). For the Concentrated-1 and Concentrated-2 distributions ASR clearly outperforms SSR achieving paths that are about 30% to 50% shorter on average than using SSR (cf. Fig. 2(c) and 2(d)). Compared to BR, ASR paths are about 55% to 75% shorter on average.

Figures 2(e) and 2(f) show the results for distance-dependent target distributions, i.e., the communication patterns SSR is specifically designed for. The performance of SSR improves significantly compared to distance-independent distributions. Remarkably, ASR reaches SSR's performance quickly by adapting shortcuts although ASR sets up fewer shortcuts (50 per router for ASR compared to more than 500 for SSR). BR on average doubles the path lengths of SSR and ASR.

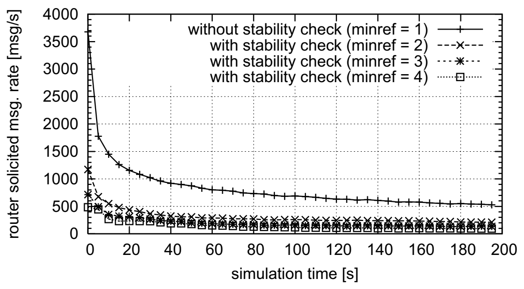

Figure 3. Average TargetRouterSolicited **msg. rate**

6.3. Communication Overhead

Especially volatile shortcuts evicted fast from the buffer cause overhead. With the stability check (Sec. 5.2) we try to reduce this overhead. Here, we analyze the efficiency of this check. The following simulations are ran with and without stability check. The check is parametrized using parameter *minref*. Higher *minref* values increase the probability of setting up stable shortcuts only. The other values of this simulation are: $|B_{\text{hot}}| = 50$, $|B_{\text{warm}}| = 1000$, $\lambda = 0.05$, Distance-independent/Concentrated-1 target distribution.

Figure 3 shows the overall message rate of TargetRouterSolicited requests over time. We see that the rate without stability check ($minref = 1$) is significantly higher than with stability check ($minref > 1$). Already for a stability check with a small period of two consecutive references ($minref = 2$), the overhead is only about 30% of the overhead without stability check when the system is in the steady state. Longer periods, i.e., $minref > 2$ reduce the overhead only marginally. The overhead rapidly settles down at about 225 messages per second ($minref = 2$) when the majority of shortcuts do not change anymore. This amounts to 15% of all messages (geocast + TargetRouterSolicited messages). However, the savings due to reducing the stretch factor compensate for this overhead by far. For the distance-independent distribution, ASR leads to about 80% of the *overall* network load (UDP datagrams of geocast messages *and* TargetRouterSolicited messages) compared to SSR and 45% compared to BR.

7. Summary and Future Work

We presented an adaptive overlay network for world-wide geographic messaging. Based on a tree-shaped network of geocast routers, shortcut links are added to build a routing mesh with short message paths. In contrast to existing algorithms for shortcut selection, we dynamically adapt shortcuts according to target pop-

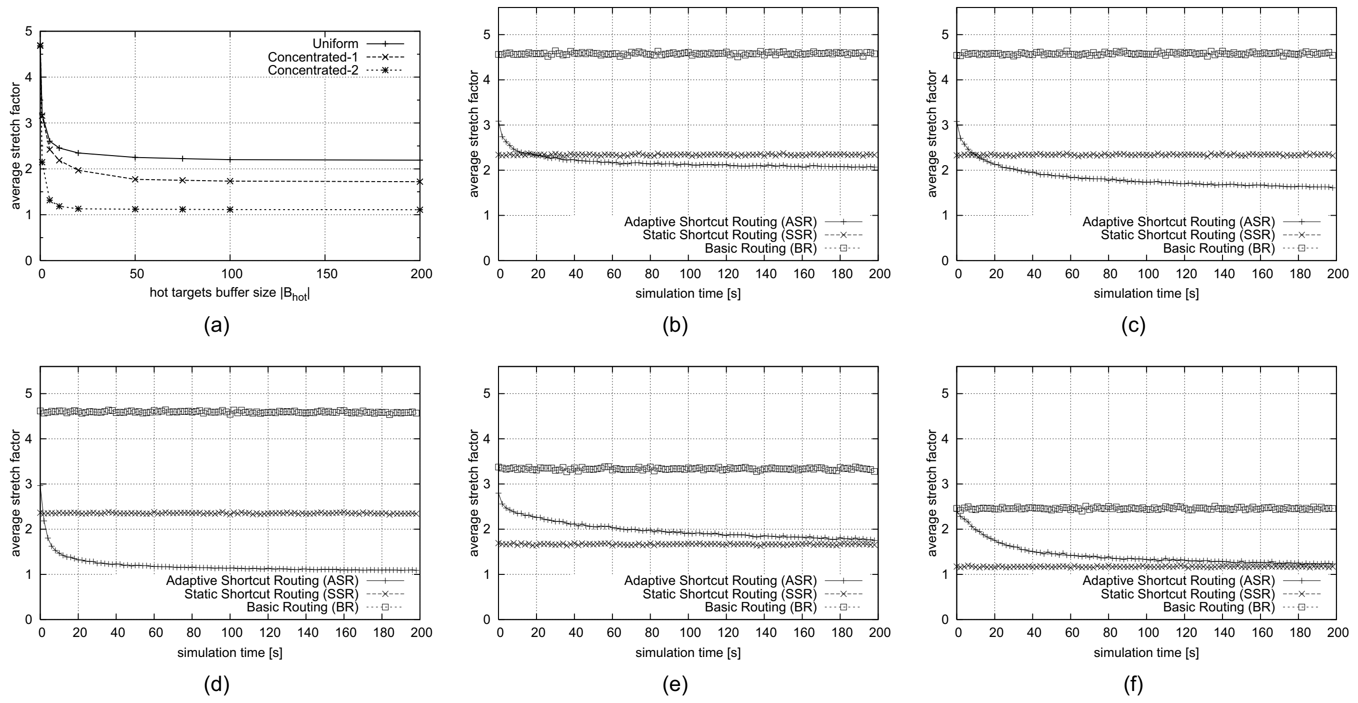

Figure 2. Stretch factors: (a) ASR with varying hot buffer size (b) Uniform target distribution (c) Distance-independent/Concentrated-1 distr. (d) Distance-independent/Concentrated-2 distr. (e) Distance-dependent/Concentrated-1 distr. (f) Distance-dependent/Concentrated-2 distr.

ularity. We showed that this approach reduces path lengths and network load significantly compared to hierarchical routing or non-adaptive shortcuts.

Mechanisms for making the routing mesh—shortcuts in particular—robust to link or router failures are an important extension of the presented approach. We have already developed such mechanisms. However, for space restrictions we have to refer the reader to our technical report [12] for further details.

References

[1] F. Dürr and K. Rothermel, "On a location model for fine-grained geocast," in *Proceedings of the Fifth International Conference on Ubiquitous Computing (UbiComp 2003)*, Seattle, WA, Oct. 2003, pp. 18–35.

[2] J. Roth, "Semantic geocast using a self-organizing infrastructure," in *Innovative Internet Community Systems (I2CS)*, Leipzig, Germany, Jun. 2003, pp. 216–228.

[3] F. Dürr and K. Rothermel, "An overlay network for forwarding symbolically addressed geocast messages," in *Proceedings of the 15th International Conference on Computer Communications and Networks (ICCCN 2006)*, Arlington, VA, USA, Oct. 2006.

[4] X. Wei and K. Sezaki, "Gnet: A peer-to-peer protocol for internet scale location-based applications," in *Proceedings of the 2nd International Conference on Embedded Software and Systems*, Xi'anm, China, Dec. 2005.

[5] J. Roth, "A decentralized location service providing semantic locations," FernUniversität Hagen, Hagen, Germany, Tech. Rep. 323, Jan. 2005.

[6] C. Maihöfer, "A survey on geocast routing protocols," *IEEE Commun. Surveys Tuts.*, vol. 6, no. 2, 2004.

[7] J. C. Navas, "Geographic routing in a datagram internetwork," Ph.D. dissertation, Rutgers University, Department of Computer Science, May 2001.

[8] D. Heutelbeck, "Distributed space partitioning trees and their application in mobile computing," Ph.D. dissertation, FernUniversiät Hagen, Germany, May 2005.

[9] Z. Xu and Z. Zhang, "Building low-maintenance expressways for p2p systems," Hewlett-Packard Labs: Palo Alto, Tech. Rep. HPL-2002-41, 2001.

[10] H. Bahn, K. Koh, S. L. Min, and S. H. Noh, "Efficient replacement of nonuniform objects in web caches," *Computer*, vol. 35, no. 6, pp. 65–73, 2002.

[11] M. Liljenstam, J. Liu, and D. M. Nicol, "Development of an internet backbone topology for large-scale network simulations," in *Proceedings of the 2003 Winter Simulation Conference*, New Orleans, LA, Dec. 2003.

[12] F. Dürr and K. Rothermel, "An adaptive overlay for world-wide geographic messaging," IPVS, Universität Stuttgart, Tech. Rep., 2007, http://www.ipvs.uni-stuttgart.de/abteilungen/vs/abteilung/mitarbeiter/eigenes/duerrfk/duerr-tr_adaptive_geocast_routing.pdf.