Adaptive Routing in a Contextcast Overlay Network

Lars Geiger, Frank Dürr, Kurt Rothermel
Institute of Parallel and Distributed Systems (IPVS), Universität Stuttgart
Universitätsstrasse 38, 70569 Stuttgart, Germany
{geiger, duerr, rothermel}@ipvs.uni-stuttgart.de

***Abstract*—Context-based communication allows for the dissemination of messages to mobile users with a specified context, i.e. at a location and with certain attribute values. This enables, e.g., a message to students on campus attending a certain class, with information about a study group for an upcoming exam. An overlay network of context-aware routers efficiently disseminate the messages to all matching receivers. Directed forwarding of such messages requires that the routers maintain knowledge about the contexts of connected users. Global knowledge, i.e., each router knowing about every user, scales poorly, though, because of the necessary updates.**

To overcome this challenge, a router can selectively propagate context information that actually allows its neighbors to prune a message distribution tree. In this paper, we present an approach to adaptively propagate only those user contexts that offer a reduction in overall system load. The algorithm automatically and locally adapts to the observed messages and user contexts on each node.

Our solution significantly improves the scalability of the system by reducing the overall load by almost 50%.

I. Introduction

Context-aware communication (contextcast) allows for a dissemination of messages to mobile receivers with matching contexts. Possible applications for such a technology include the dissemination of concert information for people interested in a certain musical style or invitations to study groups for students attending the same university class (cf. [1]). To achieve this goal, a system can use one of two forwarding strategies: (1) Broadcast a message, thus reaching every possible recipient. (2) Use knowledge about user contexts for a directed forwarding.

Obviously, constantly broadcasting messages limits the performance of such a message dissemination scheme. Thus, to efficiently distribute contextcast messages, we use an overlay network with context-based routing algorithms. Users connect to routers in whose access network they currently reside. Context updates are propagated into the network for the routers to maintain their routing tables. The contextcast routers make forwarding decisions by comparing the addressed context to the contexts of users in access networks.

However, maintaining the contexts of all users on all routers causes a high update load in the network, shifting the load from message broadcasts to context update broadcasts. There are two possible solutions to reduce this context update load: On the one hand, we can aggregate similar user contexts to reduce the amount of redundant information in propagated contexts. We investigated this in our previous work [2]. On the other hand, we can selectively propagate only those contexts that are actually addressed, which is the focus of this paper.

As main contribution, we present an adaptive, selective context propagation approach, which propagates user contexts according to the messages the system observes. The rationale here is that propagating contexts in the system is unnecessary if they are only rarely addressed. For example, if a context class is addressed only once a day, propagating context updates to the routers is much more costly than simply broadcasting this one message when it occurs. It can also be combined with the aggregation approach to reduce the amount of redundancy in the information that gets propagated.

We introduce a distributed algorithm, with each node deciding locally what context information would reduce the amount of false positive messages from its neighbors. Each node keeps a statistic on the observed false positives for this purpose. With this statistic, it identifies false positive classes that generate more load than the context updates to prevent them and propagates this information. In addition, each node also monitors the context information it has received to determine when a context is no longer useful to maintain. This is the case when messages change enough that the false positives, which this context information prevented, no longer occur. In this case, it improves overall load when such information is invalidated and the system falls back to assuming the presence of a matching receiver, speculatively forwarding these messages.

Our results show that the adaptive context propagation reduces the overall system load by almost 50%. This is achieved by a drastic reduction in context updates at the expense of additional, speculatively forwarded messges.

The remainder of this paper is structured as follows: In Section II, we give a brief overview of our system model, before introducing the statistical measurements and the Adaptive Context Propagation in Section III. Section IV presents the results of our evaluation of a prototype implementation of these concepts, and in Section V we summarize our work and conclude with a brief outlook on future work.

A. Related Work

The concept of context-based routing (or contextcast) [1] is in its idea similar to content-based routing [3]. Distributed content-based pub/sub systems such as Siena [4], REBECA [5], or PADRES [6], use a broker network to deliver notifications to all interested clients. To this end, the brokers forward clients'

Published in 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). Shanghai, China, 10-12 October 2011.
© IEEE 2011
http://dx.doi.org/10.1109/WiMOB.2011.6085340

subscriptions, and then use this knowledge for their forwarding decision. Without this knowledge, the best approach is to broadcast all notifications and let clients filter based on their subscriptions. With the propagation of subscriptions to brokers, this broadcast shifts from notifications to subscriptions: Each broker needs to know about the clients' interests for a directed forwarding. To reduce the load of these subscriptions, "cover" and "merge" try to replace several subscriptions for overlapping sets of notifications by fewer, potentially larger subscriptions.

The use of *"advertisements"* provides a further improvement to this scheme. These broadcasts describe the notifications a source may send out. Brokers then forward subscriptions only towards sources that might produce a matching notification [7]. In publish/subscribe, this was shown to significantly improve the scalability of a system [8]. However, our system supports senders that send only a few messages, for which explicit advertisements would not work. Nevertheless, our solution subsumes the use of *advertisements* in pub/sub. Our system uses the observation of speculatively forwarded false positives as a form of implicit advertisements: Context information is only forwarded over links towards routers from where false positives originate, thus enabling routers in-between to use directed forwarding. This reduces these false positives without broadcasting all context updates or explicit advertisements.

Contextcast employs a very similar routing scheme: Messages in our system contain constraints on the intended receivers' contexts. User contexts are propagated to the routers, which use them to forward messages in the direction of matching receivers. This reduces the amount of needlessly transmitted messages compared to the broadcast forwarding (i.e., broadcast of messages, with local filtering depending on receiver context and the constraints). Unfortunately, since user contexts are very dynamic, propagating and updating these user contexts causes a rather high load in the network. We have shown an approach where similar contexts are aggregated and only the combined information is propagated. The resulting coarser information reduces the update load.

While this may sound similar to 'cover' and 'merge' in pub/sub systems, there are two key differences: the fact that user context are essentially point values in a context space and it is necessary to add and remove contexts all the time with very little overhead. These points require a more sophisticated approach than the relatively simple concepts behind cover and merge. A more detailed discussion of these differences and our aggregation approach is available in [2].

In contrast to context aggregation, with its goal of reducing redundant information in propagated user contexts, in this paper we show an approach to only propagate context information that is actually addressed by messages and can thus lower message dissemination load.

II. System Model and Contextcast Semantic

In this section, we introduce the Contextcast system model as well as the message dissemination semantic, which forms the basis of our Adaptive Context Propagation algorithm in Section III.

A. System Model

Contextcast employs a distributed system of context-aware routers, or ContextRouters, organized in an overlay network, depicted in Figure 1. The links in the overlay network form an acyclic undirected graph; for any arbitrary overlay network, a routing algorithm can ensure the acyclic property. The overlay links follow a locality principle, i.e., connections between geographically close routers are more likely than between ones that are far apart. This design allows us to exploit existing locality with regard to user contexts, e.g., a concentration of students on a campus. We assume that some attributes (such as *location*) and attribute combinations are used more often than others, especially from high volume senders, while spontaneous messages from regular clients show greater variations in their addressing.

In addition to routing functionality, some routers provide access to the network for clients covering a certain service area. When this distinction from the ContextRouter is significant, we call the nodes with added access functionality ContextNodes. The ContextNodes maintain information about the contexts of connected users within their service areas.

B. Contextcast Semantic

A user context c in the system consists of an arbitrary number of context attributes α_i. $\mathcal{A}(c)$ denotes the set of all the attributes that make up a context c, i.e., $\mathcal{A}(c) = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$. Each attribute α_i is a tuple *(type, name, value)*. Figure 2a shows an example of such a context c.

A context message m addresses clients using constraints ϕ_i on context attributes. All these constraints need to be fulfilled for a user (or rather their context) to actually match and thus receive a message. A constraint ϕ is a tuple *(type, name, predicate, value)*, where type$_\phi$ is the attribute type, name$_\phi$ is the attribute name, and predicate$_\phi$ can be any predicate that is defined on the type of the attribute. Additionally, messages usually have a payload, which is the actual message content. Figure 2b shows such a message m.

We designed the contextcast forwarding mechanism to achieve a "no false negatives" semantic. This means that a user should not miss any message whose addressing their respective context information matches. The matching semantic between messages and user contexts as well as the changes necessary to support an aggregation of user contexts are described in detail in our previous publications [1], [2].

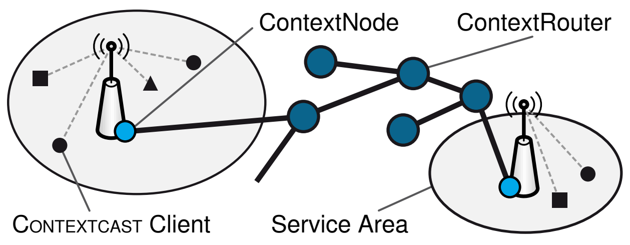


Figure 1: The Contextcast system

c:	WGS84: loc	$=$	48.12N, 9.10E
	hierarchy: class	$=$	"pedestrian"
	enum: gender	$=$	"female"
	int: age	$=$	29
		$\vdots$	

(a) Example of a user context

m:	WGS84: loc	$\in$	48.0N–48.4N, 9.0E–9.2E
	enum: gender	$=$	"female"
	int: age	$>$	15
	int: age	$<$	35
	payload	$=$	[questionnaire & voucher]

(b) Example of a contextcast message

Figure 2: User Contexts and Contextcast Messages

III. Adaptive Context Propagation

In contextcast, we can observe two types of load on the overlay links: *message load*, which results from the dissemination of messages, and *context update load*, which is caused by the propagation of context information to routers. Message load can be divided into legitimate message load (those messages for which a receiver exists) and false positive message load. There is little one can do about the legitimate message load if the system is to retain its previous semantic. False positive load, however, is generated whenever the system forwards a message for which no matching receiver exists. This happens when a node has imprecise information, e.g. due to incomplete knowledge about contexts, and must assume the presence of a matching receiver in the direction of a link. Therefore, propagating context knowledge reduces false positive load, by giving a router the necessary information to employ its directed forwarding algorithm. However, this comes at the price of the load it takes to propagate and maintain context information. The goal is to minimize the overall system load, i.e., from contextcast messages and context updates.

Our approach adapts to the actual messages and user contexts in the system. Propagating context information is only useful for attributes/combinations that actually occur in constraints. At the same time, we need to consider the dynamic of contexts, which requires updates to keep propagated information current.

Therefore, we only propagate context information *iff* it reduces the overall combined message and update load. In other words, the reduction in false positives must outweigh the additional update load for the context information.

Since we no longer propagate each context, the system needs to be able decide when it has the necessary information for a directed forwarding. To this end, we allow incomplete context information (the details are explained in the next section), which can be used for a directed forwarding if available, but fall back to speculative forwarding otherwise. Each node can decide in a decentralized manner what information to forward to a neighbor, based on statistics it keeps on false positive messages and context updates.

In the following sections, we present the necessary changes to handle such incomplete knowledge, metrics and statistics to estimate the load of context updates and false positives, and the algorithms that use these statistics to adaptively forward context information when it benefits overall load.

A. Incomplete Context Knowledge

With complete context knowledge available to Context-Routers, i.e., the user contexts of all downstream receivers, a directed forwarding decision is straightforward: A router forwards a message over a link *if and only if* it knows of a matching recipient in that direction. However, for our adaptive context propagation, the system must be adjusted to maintain the semantic without this complete knowledge. The following simple example illustrates the challenges when supporting incomplete context knowledge.

Figure 3 shows two ContextRouters A and B. A knows contexts c_1, c_2, c_3, and c_4 (locally within its service area). A has forwarded only c_2 and c_4 to router B, which has entered this information in its routing table for the link to A. From the perspective of B, we call A the *origin* of the contexts. If B now evaluates a message, which would match c_1, it would not forward it since it does not know about c_1. Without additional care, this would require falling back to broadcasting messages since every router would have to assume the presence of a matching recipient for any given message.

To prevent this, every router must be able to decide whether it has enough knowledge to evaluate a given message's constraints or needs to fall back to speculative forwarding. To support incomplete knowledge in our system, we allow routers to select *attribute sets* $A_p = \{\alpha_i : i \in \{1, \ldots, k\}\}$, for which complete knowledge is maintained on a link. (The actual selection is based on observed messages and discussed in Section III-B.) The attribute knowledge for such a set A_p is propagated as a *composite context*. Such a composite context contains the context information of several user contexts for the attribute set A_p. A router that receives such a composite context has the complete picture of values for the attributes in A_p and can fully evaluate messages with constraints on the attributes in A_p or a subset thereof.

A composite context is constructed from all contexts whose attribute sets intersect A_p. More formally, let $A_p = \{\alpha_i : i \in \{1, \ldots, k\}\}$ be a set of attributes a router wants to propagate. For every context c_i with $\mathcal{A}(c_i) \cap A_p \neq \emptyset$, we construct partial contexts $c_{p,i}$ by removing all attributes from c_i that are not in A_p. These resulting partial contexts are then merged into a composite context for A_p and propagated. (Currently, this merging is a simple list of all partial contexts;

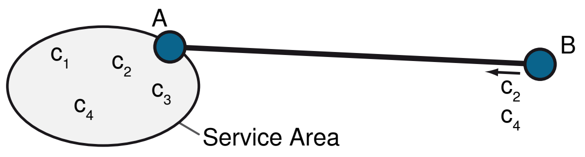


Figure 3: Context knowledge and forwarding

Published in 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). Shanghai, China, 10-12 October 2011.
© IEEE 2011
http://dx.doi.org/10.1109/WiMOB.2011.6085340

later, it could also be more sophisticated, such as an aggregation of the partial contexts.) This composite context then needs to be continuously updated by the node that created it so its neighbors can use the information for directed forwarding. The receiver of the composite context has complete knowledge about the values of the attributes in A_p and can use it for its directed forwarding decision.

B. System Load Statistics

In this section, we present a number of statistics, which we use to determine whether to propagate certain attribute sets A_p or invalidate them once they are no longer useful. All the statistics are taken during a configurable time window t_w.

1) False Positive Rate: The *false positive rate fp* is a measure of how many false positives arrive at a node of a certain type. It directly measures the amount of false positive load that could be avoided in the system by propagating certain context information.

Whenever a router receives a contextcast message M, with constraints $\{\phi_{\alpha_1}, \ldots, \phi_{\alpha_k}\}$, one of two things can happen: (1) The router knows another entity to which it needs to send M. This can be either a neighboring router or a matching receiver in its own access network. (2) The router does not need to forward the message to a neighbor and knows no local receiver. In case (1), M is a legitimate message since the router either has a local or a downstream receiver. In case (2), M is a false positive message and needs to be counted as such.

For a given constraint combination $\{\phi_{\alpha_1}, \ldots, \phi_{\alpha_k}\}$, we count the false positive rate for the attribute combination $\{\alpha_1, \ldots, \alpha_k\}$ during a time window t_w as:

$$\begin{aligned} &fp\left(\{\alpha_1, \ldots, \alpha_k\}\right) \\ &\quad = \frac{\text{number of false positives for } \{\alpha_1, \ldots, \alpha_k\}}{t_w} \end{aligned} \quad (1)$$

2) Context Update Rate: The *context update rate* u reflects the amount of load that results from propagating and updating a certain piece of context information, i.e., a composite context.

For an existing composite context c it can be observed directly as:

$$u\left(c\right) = \frac{\text{number of updates for } c}{t_w} \quad (2)$$

However, when evaluating the propagation of a new composite context, it is necessary to estimate the update rate, since it cannot be observed yet. In this case, the update rate can be estimated from the new composite context's attributes and their update rate, which each node can observe directly. Formally, the update rate of a context c_{new} consisting of the attributes $\alpha_1, \ldots, \alpha_k$, i.e., $\mathcal{A}(c) = \{\alpha_1, \ldots, \alpha_k\}$ can be estimated as the sum of the updates observed for its individual attributes:

$$\begin{aligned} &u\left(c\right) = u\left(\{\alpha_1, \ldots, \alpha_k\}\right) \\ &\quad = \frac{\sum_{i=1}^{k} \text{observed number of updates for } \alpha_i}{t_w} \end{aligned} \quad (3)$$

3) Distribution Prune Rate: The distribution prune rate p is a measure of the usefulness of a piece of context information *after* it has been propagated. After a node propagates a composite context to a neighbor, the origin node no longer receives the respective false positive messages (the reason for propagating the information). We measure this effect by counting the number of messages pruned by a context c:

$$p\left(c_{\text{composite}}\right) = \frac{\text{number of pruned messages for } c_{\text{composite}}}{t_w} \quad (4)$$

Obviously, this can only be measured at the pruning node rather than the node that sent the respective context update.

4) Smoothing: To limit the influence of short term changes, we smooth the statistics using an exponential moving average with a factor β. Thus, e.g., the smoothed false positive rate $fp_{S,t}\left(A\right)$ for an attribute set A at time t is calculated from the previous observed and smoothed values as:

$$fp_{S,t}\left(A\right) = \beta fp_{t-1}\left(A\right) + (1-\beta) fp_{S,t-1}\left(A\right) \quad (5)$$

We can calculate the smoothed context update rate $u_S(c)$ and the smoothed distribution prune rate $p_S(c)$ in the same manner.

5) Statistic Overhead: The impact of the statistic on the routers is relatively minor. We can use Bloom filters, for example, to efficiently store false positive statistics for attribute sets. Also, the nodes regularly remove old attribute sets that have not seen a false positive in some time. This cleans out rare combinations and keeps the statistic size manageable.

The statistics for propagated composite contexts increase the size of each context by a fixed amount: two counters for the prunes and updates in the current observation window and two floats for the respective rates that are updated after the window ends.

C. Per-link Adaptive Context Propagation

Using the metrics from the previous sections, we can now in detail describe our adaptive context propagation algorithm, the necessary changes in message forwarding and our approach to invalidate old contexts.

Each node observes the rate of false positives for the messages it receives. It also monitors the update rates for all the contexts it knows or estimates them from the individual attributes' update rates. Using these two rates, it can calculate the benefit of a certain piece of context information.

Definition 1 (Context Benefit): Let $c_{\text{composite}}$ be a composite context. Using Equations 2 and 4, a router can calculate the *context benefit* $B(c_{\text{composite}})$ as the smoothed distribution prune rate $p_S(c)$ over the smoothed context update rate $u_S(c)$ required for $c_{\text{composite}}$:

$$B\left(c_{\text{composite}}\right) = \frac{p_S(c_{\text{composite}})}{u_S(c_{\text{composite}})} \quad (6)$$

Intuitively, the context benefit gives us a measure for the amount of messages saved by a composite context $c_{\text{composite}}$ and the load required to update it.

For a set of context attributes $A_{\text{cand}} = \{\alpha_1, \ldots, \alpha_k\}$ that was not propagated, yet, it is possible to estimate the benefit using Equations 1 and 3:

$$B\left(A_{\text{cand}}\right) = \frac{fp_S(A_{\text{cand}})}{u_S(A_{\text{cand}})} = \frac{fp_S(\{\alpha_1, \ldots, \alpha_k\})}{u_S(\{\alpha_1, \ldots, \alpha_k\})} \tag{7}$$

Obviously, only propagating attribute sets with $B(A) > 1$ improves the load on the system, as it saves more false positives than the load it generates for updating the context information.

1) Propagating Composite Contexts: The nodes need to decide for which attribute sets they propagate a composite context. Since the nodes cannot compute the power set of all attributes, they need an efficient method to select candidate sets for propagation. Our selection of candidate attribute sets is based on the observed sets of attributes used in constraints and their propagation benefit.

After each time window, when the statistics are updated, a node calculates the expected benefit of propagating an attribute set $B(A)$ using Equation 6. The attribute sets with a benefit above a *propagation threshold* $B_{\text{th,P}}$, are propagated towards the neighbor. This threshold limits context propagation to those candidates that actually offer an adjustable benefit and prevents candidates from propagation that arise due to small fluctuations in the statistic. Without this, we may observe candidates where, in one time window, the false positives outweigh the updates slightly (and would thus be propagated); and in the next time window, the updates dominate, thus contradicting the decision to propagate the candidate set. After such a propagation, the node must update the composite context in the future; at least until it is invalidated because the messages have changed sufficiently so the information is no longer useful.

Additionally, a newly propagated attribute set A_{new} may be a superset of another attribute sets A_k. Since a composite context for A_{new} contains all the information of composite contexts for its subsets, a node can then stop updating these subsets. The node receiving the new composite context can also determine all subset contexts and remove them from its routing table. These steps are formally summarized in Figure 4

```
Require: A list FP of candidate attribute sets for neighbor $n$.
Ensure: Composite context for the attribute sets with sufficient
  benefit propagated to $n$.
  $C_{\text{prop}} \leftarrow \emptyset$
  for all $A_{\text{cand},i} \in$ FP do
    if $fp(A_{\text{cand},i}) > fp_{\text{th}}$ and $B(A_{\text{cand},i}) > B_{\text{th,P}}$ then
      for all $A_{\text{prop}} \in C_{\text{prop}}$ do
        if $A_{\text{cand},i} \subseteq A_{\text{prop}}$ then
          continue
        else if $A_{\text{prop}} \subseteq A_{\text{cand},i}$ then
          $C_{\text{prop}} \leftarrow C_{\text{prop}} \setminus A_{\text{prop}}$
        end if
        $C_{\text{prop}} \leftarrow C_{\text{prop}} \cup A_{\text{cand},i}$
      end for
    end if
  end for
  for all $A_{\text{prop}} \in C_{\text{prop}}$ do
    Propagate a composite context for $A_{\text{prop}}$
  end for
```

Figure 4: Composite Context Propagation

2) Message Forwarding: The message forwarding algorithm requires only a minor change, shown in Figure 5. Nodes use directed forwarding if they have the necessary knowledge to evaluate the constraints of a Message M and fall back to speculative forwarding if not. Finding the relevant composite contexts is not the focus of this paper. It could, however, be implemented efficiently using an approach similar to [9].

```
Require: A message $M$, routing tables for all neighbors.
Ensure: $M$ forwarded to satisfy "no false positive" semantic.
  for all $n \in$ Neighbors do
    if $\exists C_{\text{composite}} : \mathcal{A}(M) \subseteq \mathcal{A}(C_{\text{composite}})$ then
      Forward $M$ to $n$ if any $c_{\text{partial}} \in C_{\text{composite}}$ matches $M$
    else
      Forward $M$ to $n$ speculatively
    end if
  end for
```

Figure 5: Message Forwarding

3) Context Invalidation: After some time, the messages a composite context is supposed to prevent may no longer occur. In this case, we must invalidate a composite context so the originating node no longer needs to update it, thus again lowering the overall system load. Obviously, the messages this context stopped, should they occur occasionally, will then be propagated as false positives, since the node must assume the presence of a matching receiver.

Similar to the initial propagation, a router sends back an invalidate message for each composite context whose benefit drops below a configurable *invalidation threshold* $B_{\text{I,th}}$. A value $B_{\text{I,th}} \leq 1$ causes the invalidation of all those contexts for which the updates are more costly than the messages stopped. This is formally shown in Figure 6.

4) Propagation/Invalidation Hysteresis: The two different values for the propagation threshold $B_{\text{P,th}}$ and the invalidation threshold $B_{\text{I,th}}$ serve as a hysteresis for the propagation and invalidation. This way, an administrator can configure the system to only propagate information if it saves 30% more false positives than the updates it causes ($B_{\text{P,th}} = 1.3$). At the same time, the system may be configured to invalidate

```
Require: The list $C$ of composite contexts received from a
  neighboring node $n$.
Ensure: All contexts invalidated for which updates dominate
  the system load.
  for all $c \in C$ do
    if $B\left(c\right) < B_{\text{I,th}}$ then
      Send invalidation message for $c$ to $n$
    end if
  end for
```

Figure 6: Composite Context Invalidation

those contexts which save 10% less messages compared to the necessary updates ($B_{\text{I,th}} = 0.9$). If the two values were too close or the same, the system would respond rather nervously to small changes during an observation window since they might raise or lower a given context's benefit above or below the threshold, causing a propagation or invalidation, respectively.

IV. Evaluation

We have implemented our adaptive approach, shown in the algorithms in Figure 4, 5, and 6, in a prototype implementation to evaluate its performance. The goal of our approach is to reduce the overall load in the overlay network. Therefore, we focus our experiments on that and disregard things such as underlay latency. For this reason, we implemented the prototype in the Peersim [10] network simulator, which does not consider the underlay network topology.

For the simulation, the network consists of 500 Context-Routers, placed on an area of $[0, 1] \times [0, 1]$ and connected according to the Heuristically Optimized Trade-off model (cf. [11]). We selected the weight parameter γ (the authors of [11] use the letter α) as $\sqrt{n}$ for a network of n nodes. This leads to an acyclic undirected network graph, which exhibits an Internet-like power-law distribution for the node degrees.

A fraction of 60% of these nodes are then selected as access nodes (ContextNodes). They get assigned a rectangular service area with edge lengths between $[0.05, 0.06)$. Each service area is then placed on the simulation area with its corresponding ContextNode in the center. Areas extending beyond the $[0, 1]$ interval in either direction are cut off to remain within the boundaries of the simulation. This leaves a number of routers with degree 1 without an access network. Even though no user can ever connect to such a node, they can still send messages, e.g., a provider of commercial messages might operate its own overlay node as gateway.

We simulate an average of 9000 clients in the system, each with a location identical to its access network's service area. Additionally, each contains between six and twelve numeric attributes, uniformly chosen out of a set of 30 different ones. The values of the numeric attributes follow normal distributions with different means and standard deviations per attribute.

The network load is determined by the rates of updates and messages. To not favor either update or message load, we simulate update and message rates of 5 per simulation cycle, each. Updates are generated according to the following instructions, which models users both changing their context and users leaving and new ones joining the system: A fraction of 70% of all updates results from a change of a single attribute. If the changed attribute is the user's location, the simulation uniformly selects one of its ten nearest neighboring ContextNodes as its new access node; the value of a numeric attribute is changed in the same manner as it is created for new contexts. For the remaining 30% of updates, a uniformly selected context disconnects and a different random context – created in the previously described way – connects to the system. Context messages originate from $\frac{1}{10}$ of our routers, which are uniformly chosen as senders. Messages are created in a manner similar to context updates: A fraction $\lambda = 0.5$ contains a target location with edge length between 0.2 and 0.3. In addition, they contain between one and three numeric constraints; the selection of the numeric constraints follows a Zipf distribution (with parameter 1.5, using different attribute permutations for the different senders). This ensures that messages of a sender show a bias towards similar addressing.

For our evaluation, we compare our adaptive algorithm (*A*) with a baseline approach (*B*) that propagates all updates into the network. We look at both the steady-state and dynamic performance of our adaptive approach. For each experiment, we run ten simulations with different random seeds and then compute the arithmetic mean of the simulation runs. Unless stated otherwise, the standard deviation of the different runs was negligible. Table I contains an overview of our setup.

A. Load Reduction: Impact Of Propagation Threshold

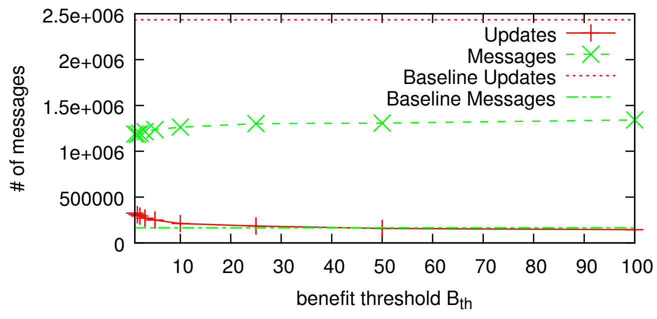
To illustrate the merit of our approach, we compare the adaptive context propagation with the baseline over a sampling period of $1,000$ simulation cycles. We observe the system in a steady state to ensures that the initial propagation of user contexts does not give the baseline approach an unfair disadvantage. Figure 7a shows the update and message load for the baseline and the adaptive approach for various *propagation thresholds* $B_{\text{th,P}}$. The *invalidation threshold* $B_{\text{th,I}}$ was set to a constant 0.9 for this experiment.

The results show that our adaptive approach reduces the overall load by between 42% and 43% compared to the baseline algorithm. Looking at the individual load elements, one can see that the adaptive approach lowers the amount of updates by between 87% and 94%. At the same time, since nodes no longer have complete context information, they need to fall back to speculative forwarding for more messages, thus increasing the number of forwarded messages by 614% up to 709%. The difference in message is due to false positives, of course, since the baseline approach forwards and delivers a message *if and only if* there is a matching receiver. However, even with this increase in messages, the overall load on the system is almost cut in half, due to the reduction in updates. This meets our expectations, since one of the major goals in the design of this algorithm is to trade off update load against additional false positives, while preserving the "no false negatives" semantic. Context invalidations play virtually no role in this experiment since the system is in a steady state and messages do not change significantly enough.

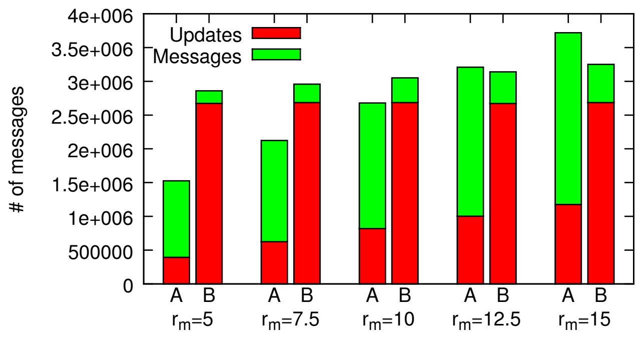
Parameter	Value
# of routers	500
γ	$\sqrt{500}$
# of access nodes	300
# of users	9000
# of numeric attributes/user	6 - 12 (uniform)
average update rate	5/cycle
# of numeric constraints/message	1 - 3 (uniform)
average message rate	5/cycle

Table I: Summary of simulation parameters

Published in 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). Shanghai, China, 10-12 October 2011.
© IEEE 2011
http://dx.doi.org/10.1109/WiMOB.2011.6085340

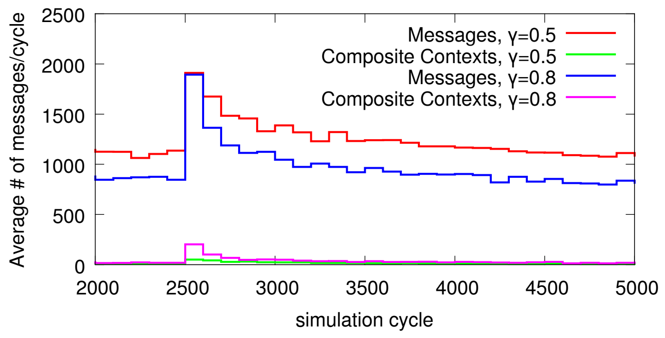


(a) System load for various values of $B_{\text{th,P}}$

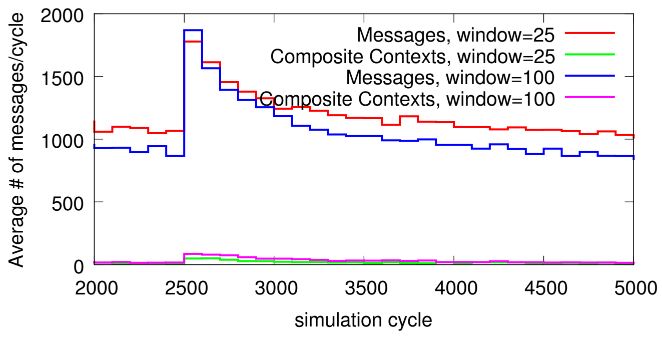


(b) System load for $r_u = 5$ and various r_m

Figure 7: Static Behavior of the System



(a) Stabilization for different values of γ



(b) Stabilization for different window lengths

Figure 8: Dynamic Behavior of the System

The propagation threshold $B_{\text{th,P}}$ determines what context information is forwarded between routers. The higher this number, the fewer information neighbors exchange, thus increasing false positive load, while at the same time reducing the amount of update load. In this scenario, with each node using a relatively similar attribute set for all its messages, the system soon reaches a steady state: all nodes have propagated composite contexts for the prevalent messages in the system. The attribute sets of these composite contexts have a high benefit value. Thus, with a lower threshold, the system establishes additional composite contexts for rarely addressed attribute sets. They increase update load without actually pruning many distribution trees; higher thresholds removes these from the system without causing many additional messages, thus lowering the overall system load. For very high thresholds, the effect reverses and the additional false positives start dominating the total system load.

B. Load Reduction: Impact Of Message & Update Rates

As the previous experiment showed, our adaptive approach reduces update load at the expense of an increase in message load. If a given percentage of all forwarded messages are overhead in the form of false positives, increasing the message sending rate also increases the absolute amount of false positives. To investigate the influence of the message rate on our system, we observe the system with $B_{\text{th,I}} = 1.5$, a constant update rate $r_u = 5/\text{cycle}$ and message rates $r_m \in \{5, 7.5, 10, 12.5, 15\}$, for $1,000$ cycles in a steady state.

Figure 7b shows the amount of messages and updates for the different message rates, for the adaptive (A) and baseline approaches (B). Clearly, higher message rates increase the message load on the system, both legitimate and false positive messages. Once the message rate reaches 2.5 times the update rate, the overall load of the adaptive approach is higher than the overall load of the baseline algorithm. Our algorithm is tailored more to handling dynamic context attributes such as location or mode of transportation than to rather static attributes (e.g., subscriptions to news feeds).

Also, note that the higher message load causes an increased number of updates for the adaptive approach. This is due to the higher message rates causing more attribute combinations to reach the propagation threshold and thus the system to create composite contexts for these combinations.

C. Stabilization: Impact Of Exponential Moving Average

The continuous measurements enable our system to adapt to changes in the addressing of messages. To measure how quickly our system adapts, we introduce a drastic change after $2,500$ simulation cycles: Every sender constructs messages using a certain preference of attributes, basically a Zipf distribution over a permutation of attributes. In this experiment, for the first $2,500$ cycles, the nodes select numeric constraints from the first 15 attributes. At $2,500$ cycles, each node selects a new permutation of the attributes from the second half of the available 30 attributes. Thus, at this instant, the complete addressing in the system changes, requiring the distribution of new composite contexts to match the new messages.

To measure the impact of the parameter γ of the exponential moving average, we vary this parameter between 0.5 and 0.8, with a constant window length of 100 cycles. Figure 8a shows the amount of messages and new composite contexts over time for $\gamma = 0.5$ and $\gamma = 0.8$. (To maintain the legibility of the figure, we omit the measurements for the updates in the system as well as for values between 0.5 and 0.8; the curves show the same trend, they just differ in the number of messages.) For both values, at $2,500$ cycles, the amount of messages increases sharply to the same amount of speculatively forwarded messages; at this time, previously established composite contexts can no longer be used for directed forwarding. It then decreases again as new composite contexts are propagated and used to reduce the speculatively forwarded messages. The higher γ shows a quicker reaction, with more new composite contexts and a faster decline of message load, since the influence of the last measurement on the moving average is higher. It also exhibits a lower overall number of messages (i.e., speculative forwarding) once the system stabilizes since nodes propagate composite context faster; with a lower value γ, the same false positive needs to be observed for several observation windows before the benefit is high enough to actually propagate a composite context.

An administrator should set γ to the highest possible value that still sufficiently filters out transient changes in message addressing, to allow for a quick adaptation to changes and an overall lower load due to speculative forwarding.

D. Stabilization: Impact Of Window Length

We also observe the stabilization for different window lengths $t_w \in \{25, 50, 75, 100\}$ (in simulation cycles). We introduce the same change to the system as in the previous section. Figure 8b shows that the window length t_w influences the time it takes for the system to react to changes (again, for legibility, we limited the figure to $t_w = 25$ and $t_w = 100$). A higher window length, e.g., 100 cycles, causes a higher spike of messages right after the change. This is due to the fact that with $t_w = 25$, the nodes can establish the first new composite contexts after 25 cycles, which then immediately reduce message load, while with $t_w = 100$, the nodes need to speculatively forward for 75 more cycles before reacting to the change. However, for $t_w = 100$, we also see that the system establishes more new composite contexts over a 100 cycle period. Thus, even though a drastic change causes more speculatively forwarded messages for higher t_w, the number decreases faster as well and reaches an overall lower number. This is caused by a better observation of messages over the longer time window.

Just as for γ, we recommend that an administrator sets t_w to the highest value that offers a good balance between agility when reacting to changes and longer and thus more accurate statistics. Obviously, this depends on the rate of messages and updates that nodes observe.

V. Conclusion & Future Work

In this paper, we presented an approach for adaptive routing in a context-based communication system. The nodes of an overlay network for context-based message forwarding adaptively propagate context information in the overlay network and automatically adapt their routing tables. This reduces the overall system load by over 40%, owing largely to the drastic reduction in context updates in our system.

In the future, we are planning to investigate a self-tuning improvement to the presented algorithm. This would allow the routers to automatically adjust the various parameters of the approach to improve overall system load. Another direction we are planning to investigate are negative comparisons with partial information. A prime example here is the location information: if routers have the information of what locations can be reached over a link, they can test the location constraint, directly pruning all branches which have no overlap with the destination location.

Acknowledgments

The work described in this paper was partially supported by the German Research Foundation (DFG) within the Collaborative Research Center (SFB) 627.

References

[1] L. Geiger, F. Dürr, and K. Rothermel, "On Contextcast: A Context-Aware Communication Mechanism," in *Proceedings of the IEEE International Conference on Communications (ICC '09)*, 2009.

[2] ——, "Aggregation of User Contexts in Context-based Communication," in *Proceedings of the 6th Euro-NF Conference on Next Generation Internet (NGI) 2010*, 2010.

[3] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, 2003.

[4] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Content-Based Addressing and Routing: A General Model and its Application," Dept. of Computer Science, University of Colorado, Boulder, USA, Tech. Rep. CU-CS-902-00, jan 2000.

[5] G. Mühl, "Generic Constraints for Content-Based Publish/Subscribe," in *Cooperative Information Systems*, ser. Lecture Notes in Computer Science. Berlin and Heidelberg: Springer, 2001, pp. 211–225.

[6] E. Fidler, H.-A. Jacobsen, G. Li, and S. Mankovski, "The PADRES distributed publish/subscribe system," in *Feature Interactions in Telecommunications and Software Systems VIII, ICFI'05, 28-30 June 2005, Leicester, UK*, S. Reiff-Marganiec and M. Ryan, Eds. IOS Press, 2005, pp. 12–30.

[7] G. Mühl, "Large-Scale Content-Based Publish/Subscribe Systems," Ph.D. dissertation, TU Darmstadt, 2002.

[8] G. Mühl, L. Fiege, F. C. Gärtner, and A. Buchmann, "Evaluating Advanced Routing Algorithms for Content-Based Publish/Subscribe Systems," in *Proceedings of 10th IEEE Intl. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002.*, 2002, pp. 167 – 176.

[9] A. Carzaniga and A. L. Wolf, "Forwarding in a content-based network," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2003, pp. 163–174.

[10] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, "The Peersim Simulator," `http://peersim.sf.net`.

[11] A. Fabrikant, E. Koutsoupias, and C. H. Papadimitriou, "Heuristically Optimized Trade-Offs: A New Paradigm for Power Laws in the Internet," in *ICALP '02: Proceedings of the 29th Intl. Colloquium on Automata, Languages and Programming*. London, UK: Springer-Verlag, 2002, pp. 110–122.