A Socio- And Locality-Aware Overlay for User-Centric Networking

Martin Florian, Fabian Hartmann, Ingmar Baumgart
Institute of Telematics, Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{martin.florian,fabian.hartmann,ingmar.baumgart}@kit.edu

***Abstract*—The recent success of online social networks hints at the general interest in socio-aware, user-centric networking applications. At the same time, mobile consumer devices are becoming both more powerful and more widespread. As an implication of these trends, user-centric networking can be realized directly between the users' devices, i.e. without depending on a server-based infrastructure. In this paper, we propose a novel hybrid overlay to facilitate the direct communication between user devices and enable the deployment of complex user-centric applications in a fully decentralized manner. In contrast to traditional overlay designs, our approach leverages social context to achieve a high interconnection between the devices of befriended users. Our overlay ensures the locality of user traffic and supports the delivery of messages to devices that are unavailable at the time of sending. Simulation results show that our approach achieves close to optimal latencies and is scalable in terms of network size.**

I. Introduction

Communication in today's computer networks is heavily influenced by two trends: the ubiquitous use of powerful mobile devices like laptops and smartphones and the demand for networking services among people sharing a certain social context (e.g. friends or co-workers). Currently such communication services are provided by companies like *Facebook* or *Dropbox* using a client/server architecture.

This paradigm has three drawbacks for the user. First, *Internet access is always required*, even if both communication partners are in the same local network or in geographical vicinity to each other. Second, even if mobile Internet access is available, *mobile data traffic is expensive* and a growing challenge for providers, which pass increasing costs down to their customers. Third, *privacy concerns* exist for storing large amounts of sensitive data central servers.

A novel approach to tackle these problems is a communication paradigm called *user-centric networking*: We assume that each user possesses several (mobile or stationary) devices which are powerful enough to provide services in a peer-to-peer manner, without depending on servers in the Internet. In contrast to classic peer-to-peer approaches we define the user as a communication target, not the device which he is currently using.

Since communication between users within a social context is often geographically local, mobile devices may span wireless ad-hoc networks or use a local area network to exchange data in a decentralized way, which avoids inefficient triangular routing via a central server.

Currently, the development of user-centric peer-to-peer applications is complex and error-prone, as developers face non-trivial challenges like the discovery and selection of communication partners and the establishment of efficient communication links. Addressing these difficulties, the idea of a middleware called *SODESSON* was proposed in our previous work [2]. However our previous publication gives only a rough overview of SODESSON's architecture and design goals. Specifically, no specific techniques were proposed for achieving the vision stated in it. Also, neither the locality of communication nor the prospects of leveraging social information for improving aspects of the system had been discussed previously.

In this paper we propose a novel peer-to-peer overlay that can form the foundation for the SODESSON middleware, enabling message delivery and distributed storage between user devices in a fully-decentralized manner. In our approach, we exploit properties inherent to user-centric communication, which are neglected by traditional overlay designs:

- *The social context of users:* Communication patterns and information sharing in user-centric networking are heavily influenced by social factors. We leverage information from the social graph to improve overall performance and security and to enable socio-aware services.
- *The locality of communication:* Socially connected people tend to be geographically close to each other [18]. Thus, our system enforces the locality of communication processes and keeps user traffic as local as possible (e.g. within the local LAN or an enterprise network).
- *The mobility of users:* Mobile devices may become temporarily disconnected from their communication partners or simply switched off by their owners. This leads to a partitioning of the overlay network which raises the need to temporarily store messages and deliver them at a later time - a challenge we address in the presented design.

II. Key Concepts

The SODESSON middleware provides a simple and consistent interface for the rapid development of fully decentralized user-centric applications. This interface is based on the *publish/subscribe* pattern of communication. To illustrate the use of publish/subscribe for user-centric networking we consider an instant messaging (IM) application and scenario: Alice wants to send a message to her friend Bob. Using SODESSON, she would do this by publishing her text to the

topic "IM@Bob" to which all IM-enabled devices of Bob are subscribed. The actual delivery of messages is than handled by the SODESSON middleware. Application developers are not burdened with addressing individual devices or choosing means of message delivery.

Based on this design, we make following key contributions:

- For dealing with network partitioning and the shifting availability of user devices, we propose an approach for realizing the *offline delivery* of messages in a decentralized publish/subscribe system. Messages to currently unavailable subscribers are persisted by the overlay and delivered once the recipients can be reached.
- In order to leverage the frequent geographic proximity of socially connected communication partners, we enhanced our basic publish/subscribe approach towards enabling the direct message exchange between publishers and subscribers. Such a *Direct Publish* achieves a significant improvement of communication locality while preserving the decoupling between publishers and subscribers to a great extend.
- In user-centric networking, communication is based on social connections and thus usually directed at known (and trusted) contacts of a user. Addressing this, we propose a novel overlay data structure called *Social Table* that keeps track of devices belonging to befriended users. Our contribution enables the efficient and low-latency lookup of likely communication endpoints.

III. Design

In the following, we will present the specific designs of our contributions in this paper.

A. Basic System

At its core, the system we propose consists of a *key-based routing/distributed hash table (KBR/DHT)* layer and a decentralized publish/subscribe system running on top of it. The publish/subscribe component is based on the concept of rendezvous-points (RP) in a KBR overlay, similarly to the approach used in [17]. In this scheme, each publish/subscribe topic is mapped to an overlay node that is to be responsible for it - a *rendezvous point* (RP). RPs collect and store subscriptions (in the DHT) and disseminate publications. Following holds for each RP to a given topic $\mathcal{T}$:

$$rp(\mathcal{T}) \equiv responsible(hash(\mathcal{T}))$$

This approach was chosen over publish-/subscribe systems based on multicast trees, like SCRIBE [7], as it is better suited towards scenarios with few[1] subscribers per topic and smaller amounts of transmitted messages.

For the KBR component, we propose the use of *R/Kademlia* [10], an enhanced version of *Kademlia* [13] with locality-enhancing modifications like recursive routing, *Proximity Routing* (PR) (biasing routing decisions towards next-hops that are closer in terms of RTT) and *Proximity Neighbor Selection* (PNS) (preferring close nodes when filling routing tables). In addition to the locality-enhancing modifications from R/Kademlia, the key-based routing component was extended with an additional *sibling table* as described in [4]. This measure improves the performance of R/Kademlia in the context of supporting a DHT running on top of it.

In the context of user-centric networking, the use of a basic overlay-based publish/subscribe approach leaves a number of open challenges: First, realizing offline delivery, which means the delayed delivery of messages to contacts that are unavailable at the time of sending. Second, the outlined publish/subscribe system has suboptimal locality properties, as all messages need to be routed via a possibly far-away RP. Lastly, classical KBR overlays like R/Kademlia are agnostic towards the social fabric between users. In this way, opportunities for leveraging social context are missed.

[1]In contrast to a microblogging service like Twitter we focus on communication between users which personally know and trust each other. Typically the average number of contacts per user is in the order of 100 (e.g. [18]).

Event: $rp(\mathcal{T})$ receives a publication p_j for topic $\mathcal{T}$
  for each S_i in $subscribers(\mathcal{T})$ **do**
    $DHT.put(pack(\mathcal{T}, S_i, p_j))$
  end for
  if $|subscribers(\mathcal{T})| > 0$ **then**
    $DHT.put(p_j)$
  end if

Event: $rp(\mathcal{T})$ receives an ACK from S_i for publication p_j
  $DHT.delete(pack(\mathcal{T}, S_i, p_j))$

Event: $rp(\mathcal{T})$ is notified by the inactive subscriber (to $\mathcal{T}$) S_i, that he has become reachable again
  $packs \leftarrow DHT.getPacks(\mathcal{T}, S_i)$
  for each $pack$ in $packs$ **do**
    $pub \leftarrow DHT.getPublication(\mathcal{T}, pack.publicationID)$
    send pub to S_i
  end for

Fig. 1. Pending Acknowledgment (PACK) life cycle and use.

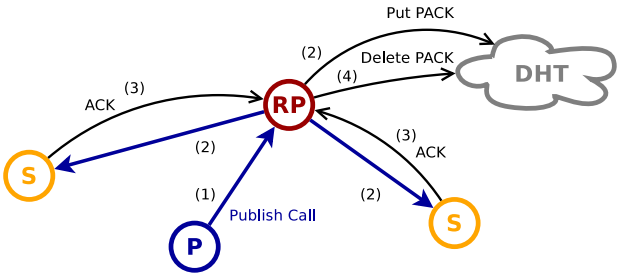

Fig. 2. Publish mechanism using pending acknowledgments (PACKs).

B. Offline Delivery

Communication services with *offline delivery*, i.e. services where messages can be delivered at a later time if a receiver is unavailable at the time of sending, are highly desirable in the context of user-centric networking. Instant messaging applications, for example, are expected to support the delayed

delivery of messages to temporarily unavailable contacts. However, the provision of offline delivery is not trivial in a fully decentralized system. Also, according to our knowledge, no mechanisms for enabling offline delivery in overlay-based publish/subscribe systems have been proposed yet.

Here, we suggest a novel approach based on *pending acknowledgment* objects, or *PACKs*, stored in a DHT. Acknowledgments (*ACKs*) are usually used to signal the successful arrival of a message to the message sender. Since in an rendezvous-based publish/subscribe system the RP is responsible for disseminating messages to the final subscribers, the RP is also proposed here as the collector of ACKs. With this setup, the PACK $pack(\mathcal{T}, S_i, p_j)$ is defined as an object stored in the DHT, that signals to $rp(\mathcal{T})$ that an ACK from the subscriber S_i needs to be received for the publication p_j. In the context of enabling offline delivery, this enables the persistence of p_j in the DHT until it has been received by all proper recipients. Since PACK objects are stored in the DHT and make use of the DHT's replication system, the PACK mechanism does not depend on the availability of one particular RP node. Once the responsibility for a topic changes (e.g. when the old RP leaves the network), the new RP simply requests the relevant set of PACKs from the DHT. The algorithm for our PACK approach is presented in Fig. 1. Additionally, Fig. 2 contains a graphical representation of the approach.

C. Direct Publish

The publish/subscribe system outlined so far provides a reliable, decentralized infrastructure for instant communication supporting offline delivery using the PACK mechanism. In the context of user-centric communication however, it has one major drawback: the existence of triangular routing in the way publications are distributed. Every time a node wants to publish a message to a topic $\mathcal{T}$, it has to pass it to $rp(\mathcal{T})$ first, introducing an additional routing hop. Triangular routing has an especially strong impact in user-centric networking scenarios where communication partners are assumed to be socially connected. Closeness in the social graph was shown to correlate with geographical proximity [18] and thus likely also with proximity in the Internet graph. In a user-centric networking scenario, communication partners are thus expected to be close to each other in the underlying network while the RP-based dissemination of publications always routes messages over random and likely far-away nodes. In this way, a high increase in latency is induced in comparison to direct communication and a significant amount of avoidable non-local traffic is being generated (especially when large amounts of data are transferred, e.g. in file transfer applications).

Addressing this, we propose a novel mechanism for enabling locality-optimized communication in rendezvous-based publish/subscribe systems, called *Direct Publish*. Our goal is the elimination of the additional routing hop over the RP, thus enabling the direct data exchange between publishers and subscribers. With our solution, we still maintain a certain degree of decoupling between publishers and subscribers, in the sense that a publisher to $\mathcal{T}$ *may* know the subscribers to

Event: P wants to publish a publication p_j to topic $\mathcal{T}$
  $subscriberHash = 0$
  for each S_i in $P.subscribers(\mathcal{T})$ **do**
    send p_j to S_i
    $subscriberHash \leftarrow subscriberHash \oplus S_i.id$
  end for
  send $(p_j, subscriberHash)$ to $rp(\mathcal{T})$

Event: $rp(\mathcal{T})$ receives $(p_j, subscriberHash)$ from P
  for each S_i in $subscribers(\mathcal{T})$ **do**
    $DHT.put(pack(\mathcal{T}, S_i, p_j))$
    $subscriberHash \leftarrow subscriberHash \oplus S_i.id$
  end for
  if $|subscribers(\mathcal{T})| > 0$ **then**
    $DHT.put(p_j)$
  end if
  if $subscriberHash = 0$ **then**
    { P's view was correct. }
    return
  else
    if $\exists S \in subscribers(\mathcal{T})$ so that $S.id = subscriberHash$ **then**
      { Only S was wrong in P's view. }
      send p_j to S
      notify P about S
    else
      { More than one subscriber was wrong in P's view. }
      send p_j to all $S_i \in subscribers(\mathcal{T})$
      notify P about all $S_i \in subscribers(\mathcal{T})$
    end if
  end if

Fig. 3. Direct Publish mechanics.

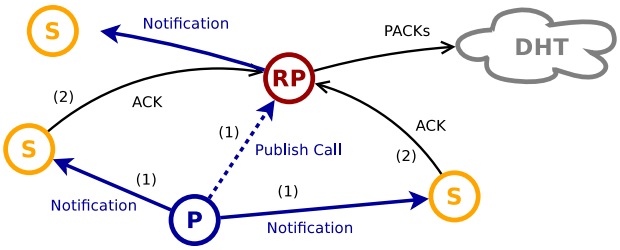

Fig. 4. Publish mechanism using Direct Publish.

$\mathcal{T}$ (allowing him to communicate with them efficiently) but is not *required* to. Whenever a publisher's knowledge on his topic's subscriber set is incomplete or wrong, the responsible RP jumps in to ensure correct publication delivery.

As a preliminary step to the presented algorithm, $rp(\mathcal{T})$ informs potential publishers about the subscribers to $\mathcal{T}$. Dedicated *subscription notification* messages are used for this purpose. For simplicity, we now assume that all potential publishers to $\mathcal{T}$ are also subscribers to $\mathcal{T}$[2]. Then, $rp(\mathcal{T})$ sends

[2]This is not a compulsory constraint and can be relaxed at will, resulting in a slightly more complex design.

subscription notifications to a subscriber S to $\mathcal{T}$ whenever he senses that S's view on $subscribers(\mathcal{T})$ is inconsistent.

A detailed overview of the complete Direct Publish mechanism is presented in Fig. 3. As can be seen, $rp(\mathcal{T})$ is still notified about P's publication, despite the fact that it's transmitted directly to $\mathcal{T}$'s subscribers as well. This is necessary for various reasons. Firstly, it can never be guaranteed that P's information on the subscriber set is up to date. P might, for example, have missed a subscription notification because of routing failures, or an additional subscriber might have joined the overlay shortly before the publication was sent. Secondly, in the case of unavailable subscribers, the publications must be persisted for later recovery. Additionally, PACKs must be generated. In the current design, all those tasks are carried out by $rp(\mathcal{T})$, so it is unavoidable that it gets informed about the publication as well. Concerning the calculation of a *subscriberHash* for each publication, this serves mainly the conservation of bandwidth, as only one value needs to be transmitted instead of a list of all subscribers. The simple XOR-based hashing function we use has the advantage that, if only one subscriber is missing or wrong in P's view (which is a likely case), the ID of that subscriber can be discovered without any additional communication overhead (see Fig. 3). The presented mechanism can be extended to allow the discovery of several missed nodes, using additional knowledge about the history of changes to the subscriber set. If no match can be found, the publication must be retransmitted to all subscribers.

The proposed Direct Publish does not interfere with the previously introduced PACK mechanism. RPs still generate PACK objects and subscribers still send ACK messages for a topic $\mathcal{T}$ to $rp(\mathcal{T})$, regardless from who they received the respective publication. Fig. 4 visualizes this in the context of a publish operation using the Direct Publish mechanism. In general however, responsibilities are shifted away from RP nodes when using Direct Publish, thus also improving the overall resilience of the system.

D. Social Routing Table

The presented core functionality of the SODESSON middleware and its improvements for a better locality of communication were additionally enhanced by a social dimension, by providing an integration point with the social graph. In the following, we introduce a novel data structure for Kademlia-based KBR overlays - the *Social Table*.

The Social Table of a node holds the node IDs of all devices that belong to friends of the node's owner or to the node owner himself. In R/Kademlia's bucket maintenance logic, it has a lower priority than the sibling table but a higher priority than ordinary buckets. This enables a biasing of overlay routing decisions towards socially connected nodes used to improve the overlay's security. Furthermore, the Social Table is an approach to achieve forwarding of sensitive data via trustworthy nodes, without modifying the actual routing algorithm. Since this approach is agnostic to the actual routing algorithm, it can also be applied to other routing mechanisms, both structured and unstructured. As demonstrated in [9], unstructured overlay techniques like flooding loose some of their disadvantages when restricting the set of queried nodes to only such nodes with whom social connections exist.

We now propose two basic approaches for populating Social Tables - *passive* and *active* friend device discovery:

Passive Discovery: Whenever a node S receives a publication directly from a publisher P (when using Direct Publish), it checks whether the owner of P is a friend of S's owner. If so, S adds P to its Social Table. Additionally, it sends a *friend device notification* to P. Since friendship links are assumed to be bidirectional, P can then add S to its own Social Table as well.

Active Discovery: In this approach, the identifiers of all devices belonging to a user are stored in one location in the DHT, namely under the user's ID. Thus, whenever a device joins the overlay it first adds its own device ID to this collection. Additionally, it retrieves the node IDs of the devices of both his owner and that of his owner's friends. Upon receiving the identifiers, it sends friend device notification messages to all those devices, informing them about its own presence. Upon receiving a response from a contacted friend device, it can be added to the Social Table. Device entries are persisted only for a limited amount of time in order to better adapt to churn. Thus, available devices refresh their own entries periodically by reinserting their own device IDs into the DHT.

IV. Evaluation

We implemented a prototype of our system using the overlay simulation framework *OverSim* [3]. We then used this prototype for performing an extensive simulation study of our design. In the following, we will present our simulation scenario, including several novel enhancements to OverSim, and the results from our measurements.

A. Test Scenario and Additions to OverSim

In order to model user-centric networking scenarios, we first enhanced OverSim by introducing user entities and device ownership. Additionally, we realized the support for a social network between user entities and provided a generator for social graphs based on the small-world graph generation algorithm by Watts and Strogatz ([19]). The parameters for the algorithm were chosen to model the properties of real social networks.

To model user behavior, we designed a SODESSON test application that mimics the communication patterns of a fictional SODESSON-based instant messaging application. We developed our test application with realism in mind, setting its parameters according to studies on instant messaging behavior (e.g., [1]). All presented simulations were run in conditions of moderate churn, with churn parameterizations based on measurements taken for the *KAD* network [16]. We used OverSim's *SimpleUnderlay* underlay abstraction for modeling the underlying network. It offers a network coordinate-based approximation of IP routing topologies. Node coordinates were generated based on real Internet latency measurements, using the *Skitter* [11] datasets.

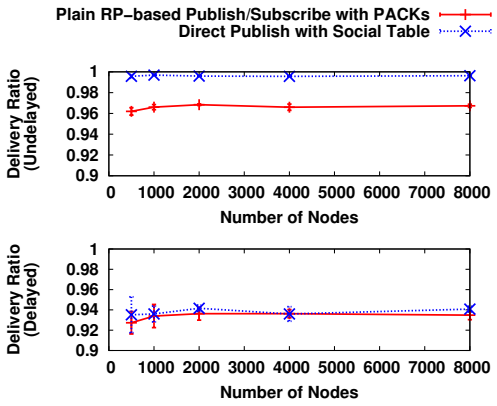

Fig. 5. Delivery ratios for undelayed (i.e. directly deliverable) and delayed (i.e. persisted for later delivery due to unavailability of recipient) publications.

The number of users in the system was set to be equal to the average size of the active node population, i.e. the number of nodes running both SODESSON and the SODESSON test application. Thus, with the churn model we use, every user owned an average of two active devices over the course of the simulation of which one device on average was online at every given moment. The social graph between users was generated in such a way that users have an average of 100 friends (corresponding observations in Facebook [18]).

The following plots show average values and 95% confidence intervals from 5 simulation runs with 20000 seconds of observation time per run.

B. Overall Performance

For determining the basic properties of our proposal and the impact of the different modifications we suggest, we evaluated a wide range of possible configurations of our system with node populations varying between 500 and 8000 nodes. In each simulation, 10% of the node population consisted of active nodes, being associated with users and running the SODESSON test application. The other 90% performed only RP-related tasks and were not owned by users. In the following, we will present the results measured for two specific configurations that we consider most descriptive, namely:

- A plain rendezvous-based publish/subscribe system using only the PACK mechanism described in this paper.
- A rendezvous-based publish/subscribe system using the PACK and Direct Publish mechanisms introduced here as well as Social Tables with both passive and active friend device discovery. This configuration bundles all techniques and modifications we propose in this paper.

The results for these configurations can be seen in Fig. 5 and Fig. 6. Fig. 5 focuses on the measured delivery ratios, i.e. the percentage of publications that were delivered correctly and within a justifiable time frame. The upper diagram depicts the delivery ratio for *undelayed publications*, i.e. publications that could be delivered right after the time of publication, while the lower one shows the same ratio for publications that first had to be persisted by the system as the receiver was not available at the time of publication (*delayed publications*).

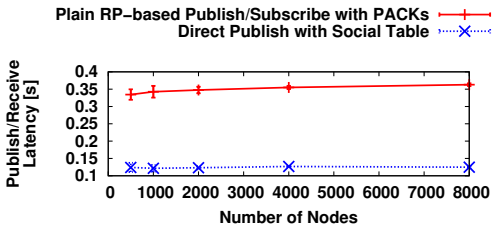

Fig. 6. Latency for undelayed publications.

For undelayed publications, the modifications proposed in this paper induce a significant improvement in delivery ratios. This can be explained through the additional layer of control over the RP exercised by publishers when using Direct Publish, as well as through the better overlay interconnection achieved through the use of Social Tables. Concerning the measured delivery ratios, the reported message losses are by a great margin due to the effects of churn on the availability of DHT entries, as was confirmed by reference simulation studies. We are currently working on ways to improve the DHTs performance under churn. Concerning the measured communication latencies (Fig. 6), a significant improvement can be observed from the use of the modifications we propose in this paper. This is mostly due to the elimination of triangular routing realized through the use of Direct Publish.

C. Effects of Locality Improvements

An additional simulation study was performed to further evaluate the impact of the locality improvements proposed in Section III-C. Again, 90% of the node population were not associated with a user and performed only RP-related tasks. The remaining 10%, i.e. the nodes actually engaging in user-centric communication over SODESSON, were clustered in the underlay so that the RTT between any two of them never exceeded 20 ms. This setup reflects the assumption that socially connected users are likely to be geographically close to each other as well.

As a comparative measure for the locality of communication, we calculated the *latency stretch* for each delivered message. We define the latency stretch for a delivered message as follows:

$$\text{stretch} := \frac{\text{publish/receive latency}}{\text{theoretical latency for direct transmission}}$$

The results of the latency stretch evaluation can be seen in Fig. 7. In the upper half, the stretch was calculated in a simulation with random node placement like in Section IV-B, while the lower half shows the results for the configuration introduced in this section. Again, the large locality improvement through the use of Direct Publish becomes evident. The order of magnitude of the improvement in a scenario where the communicating nodes are close to each other in the underlay is especially notable. On a side note, the latency stretch in the scenario with clustering of befriended nodes appears higher than when placing nodes randomly, even for configurations employing Direct Publish and Social Tables. This is because even when using Direct Publish, occasional triangular routing

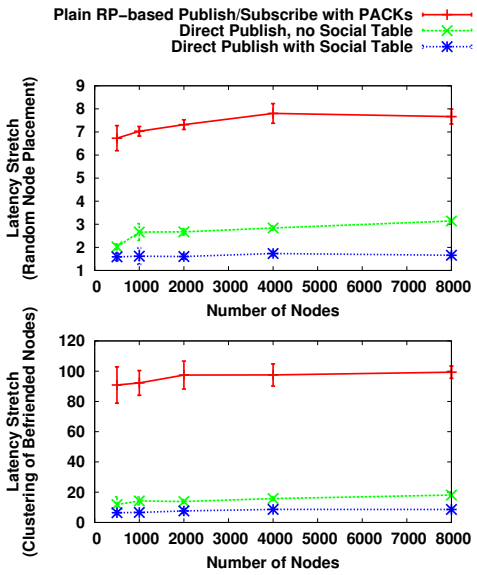

Fig. 7. Latency Stretch in two scenarios: (1) Nodes have random underlay coordinates. (2) Befriended nodes are close to each other in the underlay.

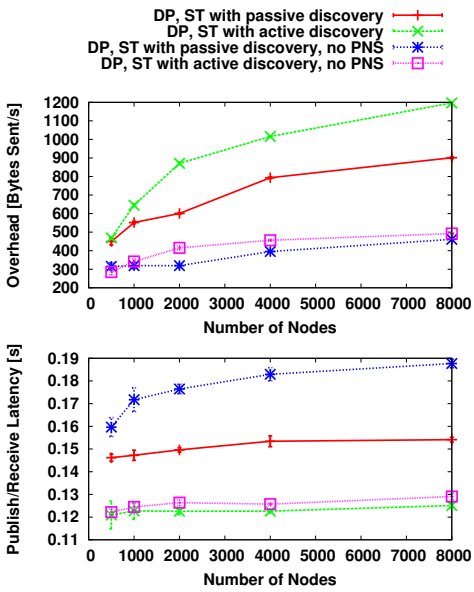

Fig. 8. Bandwidth consumption and communication latency. For brevity of representation, we abbreviated Direct Publish as *DP* and the use of Social Tables as *ST*.

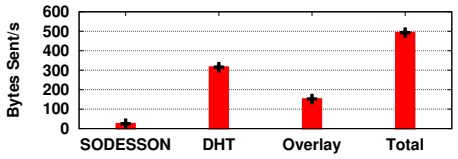

Fig. 9. Bandwidth distribution for a configuration with Direct Publish and Social Tables with active discovery. PNS is disabled.

can not be avoided, e.g. when a publisher does not know the full subscriber set for a topic. When communication partners are close to each other, such rare detours over a far away RP can influence the average latency stretch result. Additionally, SODESSON latencies are always a little higher than fictional ideal latencies, as the encapsulation in SODESSON messages induces a slight increase in size.

D. Bandwidth Consumption

Our last series of simulations aims at assessing the scalability of our solution and the modifications proposed in this paper, focusing mainly on the bandwidth consumption of our designs. Fully active node populations were used this time, i.e. all nodes were associated with a user entity and ran an instance of the SODESSON test application. Based on the previous results and the desirability to support socio-aware services, only configurations with Direct Publish and an activated Social Table were considered here. Both the active and passive approaches for discovering befriended nodes were evaluated in the simulations.

As an experiment, additional runs were also made where *proximity neighbor selection* (PNS) was turned off in the overlay routing component. PNS is a mechanism used in R/Kademlia and other KBR protocols to prefer proximate nodes when filling the routing table, thus promoting a greater locality of communication [10], [14]. The extra runs were motivated by the suspicion that a large part of the current prototype's bandwidth consumption is due to ping traffic induced by PNS. The results from our bandwidth evaluation are depicted in Fig. 8.

Our results confirm the suspicions concerning PNS: the bandwidth overhead generated by our solution is significantly lower when PNS is deactivated. The same effect can likely be achieved by employing a network coordinate system like Vivaldi [8] for assessing the proximity of nodes instead of sending ping messages. However, this was not attempted in the scope of this work. Communication latencies in our scenario deteriorate only slightly when PNS is deactivated, as can be seen in the lower half of Fig. 8. This can be explained by the fact that (1) *Proximity Routing* is still active and (2) messages are passed directly between sender and receiver (thanks to Direct Publish) and communication partners are socially connected, thus residing in each others Social Tables and able to route messages between them in one hop. Without PNS, the bandwidth consumption of our solution appears stable with growing populations. The use of an active friend device discovery mechanism leads to no significant bandwidth overhead, while improving latencies and leading to a more complete knowledge of a node's friends in the overlay.

For the most promising parameter combination we simulated (Direct Publish with Social Tables and active friend device discovery), we also evaluated how much the different system components contribute to the total bandwidth. The results of this evaluation, depicted in Fig. 9, show that actual SODESSON messages are responsible for only a negligible part of the overall bandwidth consumption in comparison to the DHT component and the overhead produced at the overlay routing layer. The DHT component is by far the

largest "bandwidth eater", which is mostly attributed to the maintenance traffic involved for assuring a sufficient number of replicas for each dataset.

V. Related Work

The desire for user-to-user communication in contrast to pure information retrieval is as old as the Internet itself. Communication applications like email, instant messenger and popular modern OSNs are tools to this end. However, classical systems based on the client/server paradigm do not provide a generic user-centric networking middleware that can be used for developing arbitrary new applications and enables the flexible service provision by user devices, while regarding the user's individual behavior.

The *Unified Internet Architecture* [9] (UIA) is a step towards the latter goal, while also solving many of the problems found in centralized architectures. However, UIA is a device-centric system, with device addressing instead of user addressing and no real consideration for the social element of communication. Several decentralized OSN projects exist that are user-centric in this respect, for example *PeerSoN* [6] or *SuperNova* [15]. Among other things, these proposals also demonstrate ways of leveraging social context in decentralized systems. However, they focus almost exclusively on privacy and classical OSN functionality. Recent work in the area of opportunistic networking [12] [5] uses the social context to detect communities which are used to improve data dissemination. Neither opportunistic networking nor the decentralized OSN proposals mentioned before aim at providing a general-purpose communication middleware based on the service provision by powerful user devices.

The problem of communication locality in a user-centric networking context was also not addressed in any of the aforementioned works. Many contributions exist for improving the locality qualities of overlay networks in general, resulting for example in KBR protocols like *Bamboo* [14] However, according to our knowledge, no approaches have yet been proposed for improving the locality properties of overlay-based publish/subscribe systems or similar systems. Finally, no suggestions for supporting offline delivery in an overlay-based publish/subscribe system are known to the authors.

VI. Summary and Future Work

Online social networks and socially motivated communication in general is as much a modern trend as the ubiquity and growing capability of mobile consumer devices like smartphones. In this paper, we presented a novel overlay-based publish/subscribe system enabling the deployment of generic user-centric networking applications and services in a fully decentralized manner by user devices. Our approach supports the persistence and delayed delivery of messages to currently unavailable recipients (offline delivery) and achieves a close to optimal locality of communication links, as simulation results show. Our hybrid overlay approach, which combines a structured overlay with a social routing overlay, significantly helps to reduce latency and keep traffic local.

In the future, we plan to focus on the privacy and security aspects of our design. Additionally, we envision to automatically learn from users' behavior to improve the availability of data and reduce communication costs.

References

[1] D. Avrahami and S. E. Hudson, "Communication characteristics of instant messaging: effects and predictions of interpersonal relationships," in *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, ser. CSCW '06, 2006, pp. 505–514.

[2] I. Baumgart and F. Hartmann, "Towards secure user-centric networking: Service-oriented and decentralized social networks," in *Proceedings of the 5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*, Ann Arbor, Michigan, USA, Oct. 2011, pp. 3–8.

[3] I. Baumgart, B. Heep, and S. Krause, "OverSim: A flexible overlay network simulation framework," in *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, May 2007, pp. 79–84.

[4] I. Baumgart and S. Mies, "S/Kademlia: A Practicable Approach Towards Secure Key-Based Routing," in *Proceedings of the 13th International Conference on Parallel and Distributed Systems (ICPADS '07)*, Dec. 2007.

[5] C. Boldrini, M. Conti, and A. Passarella, "ContentPlace," in *Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems - MSWiM '08*, 2008, p. 203.

[6] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta, "Peerson: P2P social networking: early experiences and insights," in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, ser. SNS '09, 2009, pp. 46–52.

[7] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1489 – 1499, Oct. 2002.

[8] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '04, 2004, pp. 15–26.

[9] B. A. Ford, "Uia: A global connectivity architecture for mobile personal devices," Ph.D. dissertation, Massachusetts Institute of Technology, Boston, MA, USA, Sep. 2008.

[10] B. Heep, "R/Kademlia: Recursive and Topology-aware Overlay Routing," in *Proceedings of 2010 Australasian Telecommunication Networks and Applications Conference (ATNAC 2010)*, Auckland, New Zealand, Nov. 2010, pp. 113–118.

[11] B. Huffaker, D. Plummer, D. Moore, and K. Claffy, "Topology discovery by active probing," in *Symposium on Applications and the Internet (SAINT)*. Nara, Japan: SAINT, Jan 2002, pp. 90–96.

[12] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 11, pp. 1576–1589, 2011.

[13] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. London, UK: Springer-Verlag, 2002, pp. 53–65.

[14] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, Boston, MA, USA, Jun./Jul. 27–2, 2004, pp. 127–140.

[15] R. Sharma and A. Datta, "Supernova: Super-peers based architecture for decentralized online social networks," in *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, jan. 2012, pp. 1–10.

[16] M. Steiner, T. En Najjary, and E. W. Biersack, "Long term study of peer behavior in the KAD DHT," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, Dec. 2009.

[17] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '02, 2002, pp. 73–86.

[18] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the facebook social graph," *CoRR*, vol. abs/1111.4503, 2011.

[19] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, Jun. 1998.