# Towards Socio- and Resource-Aware Data Replication in User-Centric Networking

Fabian Hartmann and Ingmar Baumgart
Institute of Telematics
Karlsruhe Institute of Technology
Karlsruhe, Germany

***Abstract*—Recently, there has been a lot of research on decentralized social networks motivated by privacy concerns with centralized systems. However, an open challenge with decentralized systems is the actual data replication and retrieval among the participating devices. We argue this aspect can be greatly improved in terms of efficiency by taking social relationships, user behavior and locality into account. User-centric networking is a paradigm which includes the users' own devices – from smartphones to highly available personal clouds – and targets at a socio-aware data storage, based on the users' behavior and their devices' availability. In this paper, we introduce the concept of a Decision Engine that chooses the replication devices based on multi-dimensional input parameters, from momentary conditions to long-time learned user behavior and social relationships.**

## I. Introduction

Besides pure information retrieval, personal user-to-user communication belongs to the most important applications of the Internet today. Even though applications like email and instant messaging exist for decades, service providers like *Facebook* or *Google* now offer popular comprehensive communication services in the form of *Online Social Networks*. The same applies to easy-to-use file sharing solutions for small private groups like *Dropbox*. Usually, these services use a client/server architecture and do not feature a decentralized, federated architecture. This brings three major drawbacks for the user:

- First, *privacy concerns* exist as large amounts of sensitive data are stored centrally on one provider's servers, resulting in a loss of control for the user.
- Second, *Internet access is always required*, even if both communication partners are in the same local network or in geographical vicinity to each other. This imposes unnecessary obstacles to easy direct communication (e.g. when traveling by train without free WiFi access).
- Third, even if mobile Internet access is available, *mobile data traffic is still expensive* and a growing challenge for providers which pass increasing costs down to their customers. Depending on the data size and access network, data exchange might be unacceptable in certain situations.

A recent approach to tackle these problems is a communication paradigm we call *user-centric networking*: As the name implies, we define not devices, but the users themselves as communication parties. We assume that each user possesses one or 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.

Privacy can be enforced to different degrees in such a system, depending on the structure of the peer-to-peer network. In a KBR/DHT network [1], a node which is responsible for popular data item can monitor metadata like access frequencies from other nodes, even if the data item itself is encrypted. For example, if two co-workers edit a document in collaboration, their devices concurrent accesses (from 9 to 5 on Mondays to Fridays only) might reveal a social relationship between the devices' owners.

A set of users can prevent metadata leaks if exclusively devices under these users control communicate end-to-end in regard to a data item that is only meant for these users. We define this communication paradigm as *self-sufficient*. Typically the amount and types of such devices differ from user to user, ranging from smartphones to highly available personal servers. This has a strong impact on device availability and heterogeneity, yet all data storage and distribution have to be realized by these devices only.

With the three mentioned drawbacks in mind, we therefore need to decide on which devices the data should be replicated and when is the best moment for doing so. In order to make these decisions towards a satisfying user experience, we need to take the users' behavior in multiple terms like mobility, application and device usage and social context into account. Self-sufficient communication with its additional constraints relies even more on suitable decisions.

In this paper, we introduce the required mechanisms that map multi-dimensional input parameters given by the users' behavior onto replication decisions.

## II. Challenges in User-Centric Networking

In this section we present our view on the challenges in *user-centric networking (UCN)*. The basic goal in user-centric networking is the data exchange, especially the storage and retrieval of persistent data items, between users that know and trust each other via their very own devices. This gives us three main aspects we have to cover:

- Availability
- Resource conservation
- Privacy

### A. Availability and resource conversation

Different devices from a smartphone to a highly available and well-connected personal cloud server can be involved in

each user's resource pool. Each of them has different demands in regard to availability, costs and energy management.

Based on the information which defines the target user group, the question that remains is *when* and *to what devices* the data items should be replicated. This is the main problem we regard in this paper.

To some extent this question is similar to the problem in opportunistic networking, since device connectivity can be limited due to user mobility. However, since we focus on persistent data retrieval and long time storage, the challenges are different. Furthermore, classic opportunistic networking relies on liberal use of emerging forwarding links, while self-sufficiency enforces hard constrains in this regard.

### B. Privacy

There is a trade-off between privacy and availability which has to be defined per use-case in UCN. The resulting constraints define the set of possible replication choices. We identify four degrees of privacy in the UCN context, with degrading privacy levels and increasing opportunities for availability:

- Publisher-to-subscriber: A publisher device communicates directly with all subscriber devices. This way, no subscriber learns about other subscribers and cannot deduce if and which other users are recipients. This also applies to other devices in the overlay network.
- Subscriber-to-subscriber: Subscribers are informed about other subscribers. This equals to a meeting of a closed group and gives us a lower level of privacy. Since the published information stays inside the subscriber group, we also define this scheme as self-sufficient. The advantage over the publisher-to-subscriber scheme is the possibility that devices of different subscribed users can distribute and store the data item among them as well, hence relieve the original publisher device and provide a better level of availability.
- Contact-to-subscriber: Devices of mutual contacts between a publisher and a subscriber help to distribute and store a data item, even if they are not subscribers. This breaks our definition of self-sufficiency, but further increases the opportunities for distribution and storage. Given that we trust a contact more than an unknown user, such a scheme can still provide an acceptable level of privacy.
- Any-to-subscriber: All devices on the overlay network can be used to distribute and store a data item. This enables well-defined replication schemes, such as in a KBR/DHT network, but gives us the lowest level of privacy and enables metadata leaks as described earlier.

Note that a contact-to-subscriber scheme assumes that a user fully trusts all his contacts. In a closed user group where all participants know each other, an attacker might be even more interested in communications patterns than a random attacker from a large, mostly unknown user group, for example to verify gossip behind one's back or incriminate a cheating spouse.

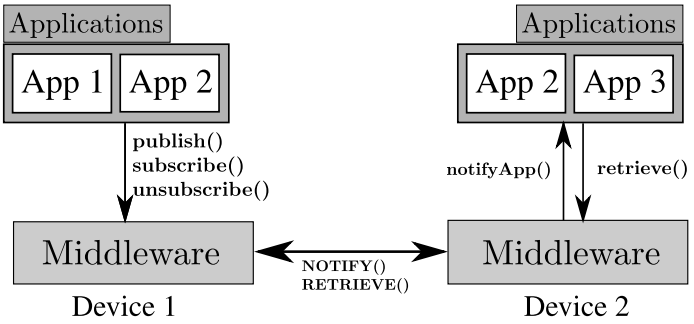



Fig. 1: Architecture overview

## III. Publish/Subscribe for User-Centric Networking

A communication service for UCN should support the *dissemination* and *distributed storage* of data among users' devices. It should fulfill the following requirements:

- **Multiple devices per user**: In UCN users should have ubiquitous access to their data independent of the device they are currently using (e.g. desktop computer, laptop or smartphone).
- **Offline-delivery** Due to user mobility and offline devices we cannot assume permanent connectivity between the sender and all potential receivers of a message. Therefore we should store data on intermediate (trusted) nodes to support delivery as soon as a device gets available again.
- **Broad range of application**: The communication service should fulfill the requirements for various UCN applications (e.g. sharing of documents, profile, instant messaging, gaming).
- **Privacy**: Personal data should only be stored on trusted devices.

We argue that that publish/subscribe paradigm is well suited for UCN, since it

- supports one-to-many communication
- is device-agnostic
- supports one time delivery as well as permanent distributed storage
- is feasible, because the number of subscriber is limited in UCN[1].

Such a publish/subscribe service could be provided by a middleware running on each device but for performance and security reasons the direct integration into the network stack could also be considered. We presented an example for such a user-centric middleware in our previous work [2].

Figure 1 shows the architecture layout which is described in the following paragraphs.

### A. Addressing

The primary target for data in UCN is a user and not a specific device. Therefore all topics have the form *app@uid*.

[1]A typical user in an online social networks has about 100 social contacts on average.

*uid* is the user id of the user owning the topic and *app* is the name of an application.

Depending on the application, either all devices of a user or devices of his social contacts are subscribed to a topic. E.g. for instant messaging Bob publishes an instant message to the topic $im@uid_{alice}$ and all of Alice's devices are subscribed to this topic. When Bob wants to receive Alice's latest photo albums, he subscribes to $photos@uid_{alice}$.

### B. Decoupled retrieving

According to predicted user behavior (see section IV) and currently available device resources, such as battery level and connectivity, the Decision Engine on a receiving device might decide to delay the data transfer. Therefore, we decouple the actual transfer of a data item from the notification about its creation / update. Hence, the signaling protocol requires an additional message which we assume small in size, so it can be distributed fast without further negotiation, as opposed to e.g. a movie which is several gigabytes in size. As a further optimization, small data items such as an instant message, can be piggybacked in its notification.

After notification, the retrieval of the data item can be triggered in two places: a) the user reads the notification about a new publish and decides that he wants to receive the data right now, hence overriding all automatic decisions and b) the Decision Engine has detected a preferable situation for data transfer and triggers the retrieval by itself. In that case, the corresponding data is already cached on the target device when the user decides to access it.

### C. Application interface

For the data dissemination control and the communication between applications and middleware we propose a publish/subscribe interface. In contrast to traditional pub/sub interfaces we have a strong focus on the decoupling between the notification of subscribers and the actual transfer of the published data towards the subscribers. Another important aspect that is usually not covered in traditional pub/sub systems is the difference between non-persistent message dissemination and persistent distributed data storage.

Our interface provides the following methods:

- *handle* ← **subscribe**(*topic*)
  This method is used to subscribe to *topic*. The Decision Engine should notify a subscriber immediately, if there are any persistently stored data objects for this topic.
- **unsubscribe**(*topic*)
  This method is used to unsubscribe from *topic*.
- **publish***(topic, metainfo, data, permanent, ttl)*
  This is used to publish data object *data* with corresponding meta information (e.g. an object id) to *topic*. The flag *permanent* indicates, if the data object should be deleted, once it was received by all subscribers. *ttl* is the minimum time an persistent object needs to stored (respectively delivery of a non-persistent message should be pursued).
- ⟨*data[], metainfo[]*⟩ ← **retrieve***(topic, [filter])*
  This is used to enforce a retrieve of a data object, overriding the Decision Engine. Retrieves all data objects for *topic*, which *metainfo* matches *filter*.
- → **notifyApp***(topic, metainfo, [data])*
  This is used to notify a subscriber about a new data object. Small data objects are piggybacked with the notification.

Note that this interface between applications and middleware does not define *when* and *to what devices* a published data item should be replicated. This is the task of the communication protocol which we propose to anticipate user behaviour. In the following section we sketch the idea of a Decision Engine and its possible input parameters.

## IV. Decision Engine

We introduce the idea of a decentralized data replication mechanism that we call the Decision Engine (DE). The DE runs on each device and complements the pub/sub service. Whenever a new data item gets published, the DE on the sending device takes multi-dimensional aspects (from present circumstances like connectivity to learned long-time behavior) into account to decide:

- Time of sending: Now or later
- Set of recipients: All of them or just a subset
- Relay support: Handle transmissions alone or ask for help

Even if the sender's DE decides that right now is an adequate situation for sending, this does not necessarily apply to every or any recipient. For example, if a recipient device is currently in a subway with bad mobile reception, it probably does not want to download large movie file, even if the sender has a fast connection. This means that the recipient's DE takes similar aspects into account before it deems the current situation adequate for receiving the data. Hence, before any data replication takes place, the DEs on the sending and receiving ends both need to agree that it is a suitable situation for doing so.

### A. Goals

In order to define efficient and smart data replication in UCN, we need to set actual goals that are to be fulfilled in this context. We have identified three major goals: Resource conservation, data availability and privacy.

These goals are mutually exclusive as shown in Figure 2. A high *data availability* can be established by flooding all data items on all devices, which is not resource conserving at all. Additionally, it violates *privacy* considerations since the responsibility for storing data items does not get limited to specific devices. High privacy constraints and specific device choices might put a high load on single devices, which in return violates the goal of *resource conservation*.

Depending on the application context, set of subscribers and data item, these goals have to be prioritized and a suitable replication strategy must be chosen.

### B. Input parameters

In order to make its decisions, a device's DE takes multiple types of input parameters into account. For these parameters we identify three classes of longevity:

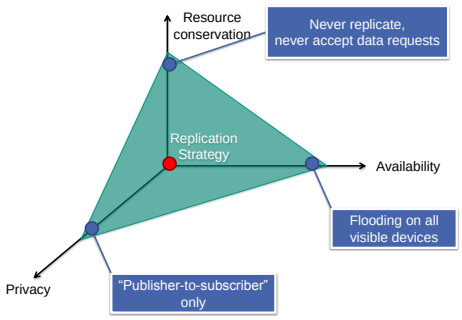



Fig. 2: Decision Engine play field

*1) Immediate:* Participating devices might be limited or enabled in ways that can be detected right here and now – possibly after a one-time manual setup by the user: a mobile connection linked to a volume-based contract, high data roaming costs abroad, a weak battery or low disk storage. Depending on the sender and recipient devices' locations, there could be an opportunity for local ad-hoc communication.

*2) Short history:* Mobile devices tend to fluctuate strongly in their availability. Among other possible issues, mobile reception might be spotty, it might change from high-speed LTE / 4G to a slow EDGE connection in an instant or the battery might run empty. Besides the immediate status of the device's conditions, it makes sense to monitor these conditions over a longer period like several hours. For example, if for the last several hours a mobile device has not been moved, has been plugged into an AC charger and has had a continuous strong WiFi reception, it is safe to assume, that it is able to receive a larger data transfer that might last several minutes. Furthermore, it can even be used as a relay device to distribute the data to other recipients. On the other hand, if its owner is carrying the device while traveling, it might not be best choice for big data transmissions, even if the immediate, current connection might be favorable.

*3) Long history:* The daily behavior of an average person is not completely random, but features periodic patterns like being at the workplace from 9 to 5 on workdays, doing grocery shopping afterwards or going to the gym each Tuesday and Thursday evening. Many of these activities are correlated to specific device accesses, as for example the work PC or the tablet PC in the living room. Moreover, in terms of social contexts these activities are strongly linked to meeting the same people (and their devices) again, like family members, work colleagues or sports partners. When it comes to transferring data between devices in such a context, there is not necessarily an Internet connection required if local communication is possible. This applies to immediate communication opportunities as described above, but also to long term patterns, which we further discuss in section IV-C. Periodically running pattern recognition and machine learning algorithms can help detecting these opportunities and improving the DE's set of input information.

### C. Learning

In the following, we focus on the long history aspects the DE can learn over time. According to our considerations, these include – but might not be limited to – device availability, the user's mobility and application usage behavior and the long-term social graph. The learning process is based on established machine learning and pattern recognition algorithms, like Naïve Bayes or Eigenbehaviors [3]. Input data for these algorithms is a permanent data sampling and collection a device performs for itself for the mentioned aspects. If decisions require negotiations between the devices of different users, the summarized conclusions instead of the raw data should be exchanged to protect users' privacy.

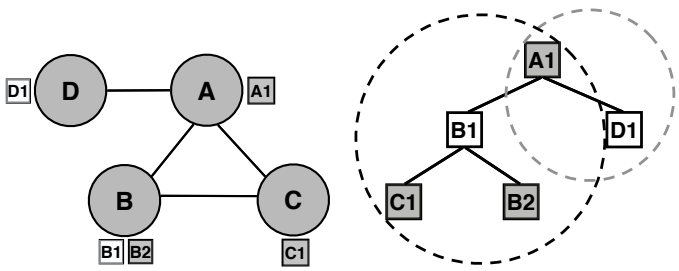
*1) Device availability:* As an example, assume that Alice decides at 3pm to send a photo to Bob with her smartphone. If her smartphone's DE has learned that she meets Bob (i.e. one of Bob's devices is locally available) every day on 4pm, it might be sensible to hold back the data transfer for one hour if resources are otherwise limited. Given that delay-tolerant networks even establish multi-hop routing via independent face-to-face meetings, this deems a plausible scenario.

Availability is also an important factor to relaying and retrieving persistent data items. For data items to be persisted permanently (especially for new social contacts in the future), data items should be preferably stored on devices with a long history of high availability.

*2) Mobility behavior:* Similarly, assume that Alice wants to take an important work document home to continue editing it there. Since her home PC is switched off while she is away, her work PC cannot communicate directly with her home PC. However, she carries her smartphone from home to work and back each day, hence the smartphone's DE has learned that it regularly has local contact to both other devices on different locations. In this case, the work PC would search for a relay device, negotiate with the smartphone and the DEs recognize the smartphone as a useful relay for carrying the work document home.

*3) Application usage behavior:* Now let us assume that Alice wants to share the photo from her smartphone (see section IV-C1) not only with Bob, but also with Carol, Dave, and many other friends of hers. This is comparable to a typical OSN scenario, where a photo is made available to hundreds of friends. Depending on Alice's current connection and the availability of relays, her smartphone might be too limited to provide all recipients at once. However, the affected DEs might have learned that in similar situations Bob instantly looks at Alice's photos as soon as he receives the notification. Dave on the other is much less interested in Alice's photos and he just browses through the Photo app once a week. In this case, Bob should be prioritized before Dave to use the limited resources optimally and to deliver the data primarily to the users who actually care.

*4) Social graph:* In this section, we want to elaborate further on the impact of the social graph on the data dissemination and replication. Figure 3a shows an example social




(a) Social graph between users (b) Social multicast tree

Fig. 3: Social user graph (circles) and their devices (boxes). White devices are resourceful, grey devices are limited. The dashed circles enclose the device communities defined by cliques in the social graph.

graph and the device possessions. Alice, Bob and Carol form a fully-meshed social sub-graph (*clique*). If Alice publishes a data item for all her social contacts (Bob, Carol, Dave), the connection between Bob and Carol can also be leveraged for data replication and storage, unless Alice has defined publisher-to-subscriber privacy for this use-case. Assuming that Bob has a strong device which could relay Alice's data item, he possibly has a social interest in Carol receiving the data item, hence Bob's relay is used to disseminate it to Carol's devices as well. The edges in the social graph can even be weighted, since there are approaches to quantify and measure social closeness (for example [4]), which makes it possible to define a threshold for cooperation.

To detect such cliques, the DEs periodically exchange their own contact lists and build a local 2-hop social graph. A maximal clique detection algorithm then locally detects collaboration communities. The clique problem is NP-complete, but in our scenario we assume small social graphs that rarely change, so the computational overhead is seen as feasible. Such a collaboration community can now be transformed into $n$-nodes tree with $n$ being the overall number of devices in the community. Figure 3b shows such a tree from the view of Alice's device A1. As indicated by the dashed circles, this tree can be divided into two sub-trees, consisting of the community between Alice, Bob and Carol (left, $n = 4$) and Alice and Dave (right, $n = 2$) respectively. Given that a data item should be published with A1 being the source, for each sub-tree it can be disseminated via an application-layer multicast tree instead of A1 sending all the data by itself. With A1 being a resource-limited device, it makes sense for the left sub-tree to pull the strong device B1 up to a higher level. However, since D1 is not part of the community, A1 has to handle it by itself.

## V. Related Work

Due to privacy concerns there has been a major interest in distributed systems for *online social networking (OSN)* (e.g. [5]). However, these proposal neglect challenges like learning from user behavior or dealing with network partitioning due to user mobility. Recent work in the area of opportunistic networking [6][7] uses the social context to detect communities which are used to improve data dissemination. [8] and [9] employ publish/subscribe systems to disseminate messages to communities over a opportunistic network and try to identify good carriers based on mobility prediction. Finally [10] shows a centralized approach to calculate the optimal distribution of content updates for mobile social networks based on knowledge of the social graph.

UCN shows several similarities to socio-aware opportunistic networking like asynchronous communication due to mobile devices. However, there are several major differences: In UCN we focus on communication between users, which have a personal relationship and trust among each other. To protect users' privacy we utilize this trust to store data only on trusted devices. Finally we assume a set of very heterogeneous devices (from smartphones to personal cloud servers).

## VI. Conclusion

We sketched an approach to socio-aware data replication that ties in to the emerging scenario of user-centric networking. Our approach is based on a publish/subscribe service as communication abstraction as well as a novel socio- and resource-aware mechanism for smart replication choices, which we call Decision Engine (DE). We defined the goals of the DE, specified the corresponding input parameters and sketched an example decision flow. Our next step is the implementation and evaluation of a DE prototype. However the evaluation of socio-aware protocols is still a major research challenge, since multi-dimensional user models covering all relevant aspects of user-behavior are not available yet.

## References

[1] I. Baumgart, B. Heep, C. Hübsch, and A. Brocco, "OverArch: A common architecture for structured and unstructured overlay networks," *2012 Proceedings IEEE INFOCOM Workshops*, pp. 19–24, Mar. 2012.

[2] M. Florian, F. Hartmann, and I. Baumgart, "A Socio- And Locality-Aware Overlay for User-Centric Networking," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC 2014)*. Honolulu, HI, USA: IEEE, Feb. 2014, pp. 327–333.

[3] N. Eagle and A. S. Pentland, "Eigenbehaviors: identifying structure in routine," *Behavioral Ecology and Sociobiology*, vol. 63, no. 7, pp. 1057–1066, Apr 2009.

[4] S. Phithakkitnukoon and R. Dantu, "Mobile social closeness and communication patterns," *7th IEEE Consumer Communications and Networking Conference*, no. 2, pp. 1–5, Jan 2010.

[5] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: A privacy-preserving online social network leveraging on real-life trust," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 94–101, 2009.

[6] 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.

[7] 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.

[8] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft, "A socio-aware overlay for publish/subscribe communication in delay tolerant networks," in *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems - MSWiM '07*, Chania, Greece, 2007, p. 225.

[9] P. Costa, C. Mascolo, M. Musolesi, and G. Picco, "Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 5, pp. 748–760, Jun. 2008.

[10] S. Ioannidis, A. Chaintreau, and L. Massoulie, "Optimal and Scalable Distribution of Content Updates over a Mobile Social Network," in *IEEE INFOCOM 2009*, Apr. 2009, pp. 1422–1430.