Keyword-Based Mobile Application Sharing

Abstract—**The advent and wide adoption of smartphones in the second half of '00s has completely changed our everyday mobile computing experience. Tens of applications are being introduced every day in the application markets. Given the technology progress and the fact that mobile devices are becoming strong computing devices, mobile *applications* are expected to follow suit and become computation-heavy, bandwidth-hungry and latency-sensitive.**

In this paper, we introduce a new mobile computing paradigm to alleviate some of the network stress that mobile applications are already putting into the network, *e.g.*, in case of crowded areas and events, where the mobile network effectively collapses. According to this paradigm, *users can share the applications that they have on their mobile devices with nearby users* that want access to *processed* information, which their own applications cannot provide. In a sense, then, the client application instance is also acting as a server instance in order to serve requests from nearby users. A representative example is a train or route information application in a busy station, airport, stadium or festival, or a gaming application onboard a flight. Our paradigm builds on Information-Centric Networking and uses *keyword-based requests* to discover shared applications in the vicinity.

I. Introduction

Mobile computing is currently led by smartphones and is largely application-centric. Users increasingly employ applications to gain access to information *e.g.*, the top-100 applications in most popular application markets are responsible for almost 90% of the access time and 80% of the traffic volume [1]. Through applications, users normally gain access to *processed information e.g.*, finding a route, searching for restaurants in an area, getting personalised social networking or news feeds, *etc.*, instead of only simply asking for static content. Although mobile devices have gained remarkable computing capabilities, the required resources for this processing are primarily provided by the cloud. Subsequently, using smart applications and the corresponding cloud-based service components *e.g.*, Facebook backend, depends on the availability of Internet access, increasingly stressing the (wireless) network infrastructure.

However, access to the cloud is in some cases not necessarily the best option, or not always possible. It is not uncommon the case where connectivity and access to the Internet is challenged in overcrowded areas (*e.g.*, airport lounges, festivals, stadiums, big conference-like events), or due to equipment failure (*e.g.*, in case of natural disasters), or complete absence of available connectivity (*e.g.*, onboard a flight, or in a train while in tunnel), or even due to high roaming costs. In these cases, the access bandwidth available to connect WiFi Access Points (APs) or Base Stations (BSs) to the Internet gets swallowed up very quickly, leaving users connected to the local AP, but with no or limited access further out. Naive approaches to deal with the above situations might suggest the increase of access capacity [2]. It is important to understand, however, that a very big proportion of the services that users are interested in during such events do not actually require access to the global network, but are rather targeting non-personalised services/content related to the local event itself. For instance, while in a festival, users are more likely to be interested in finding information on local restaurants, train times or local maps, rather than requiring VPN connection to their work email. Although the situation can be quite different in a conference-style, business-oriented event, we argue that the demand for local services is far from negligible [3]. In all cases, from a resource management point of view, dealing with demand for local services/content locally, increases the availability of Internet resources to those who do require access to remote content/services.

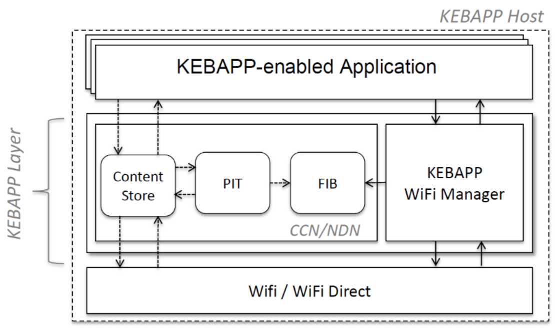
To this end, several solutions have been proposed to share information between mobile devices without using the Internet infrastructure *e.g.*, OpenGarden[1] and FireChat[2]. A significant body of work has focused on mobile ad-hoc networks, however inheriting the drawbacks of the underlying host-centric IP paradigm *i.e.*, location-identity coupling. Taking a data-centric approach, Haggle [4] first proposed a data-centric network that enabled seamless network connectivity and application functionality in dynamic mobile environments, separating application logic from transport bindings so that applications can be communication-agnostic. Trying to overcome IP limitations, other proposals focused on the Information-Centric Networking (ICN) paradigm *e.g.*, [5]. In [6], the authors propose Krowd to enable content sharing between users in crowded live events by realising a key-value store abstraction for applications.

Other approaches have been based on Delay-Tolerant Networks (DTN), exploiting both its inherent capability to exchange data in opportunistic environments, and its in-network storage functionality. For instance, a DTN-based content storage and retrieval platform is proposed in [7], enabling applications to make caching and forwarding decisions. In [8] maps of disaster areas are generated and shared over a distributed DTN-based computing system. ShareBox[3] enables the exchange of files over an opportunistic network. Similarly, the Floating Content [9] concept leverages ad-hoc communications among mobile users to share local information. According to Floating Content, message and information replication is limited in time and space. The proposed solutions so far aim at either enabling IP-based connectivity in mobile environments, or supporting the generic, application-agnostic exchange of content and computations often employing ICN primitives *e.g.*, name-based routing and forwarding [10]. Named Function Networking (NFN) [11] extends the resolution-by-name ICN primitives providing in-network data computations, but without enabling application sharing in mobile environments.

[1]http://opengarden.com

[2]http://opengarden.com/firechat

[3]https://play.google.com/store/apps/details?id=de.tubs.ibr.dtn.sharebox



Content Store

Data Chunck	Name
2a3b69e43f9bd48937	/d/e/f#t3#t4

PIT

In	Name	Out
Internal face #1	/a/b/c#t1#t2	BSSID1
BSSID2	/d/e/f#t3#t4	Internal face #2

FIB

Prefix	BSSID	Face
/a/b/c#t1#t2	BSSID1	Internal face #1
/d/e/f#t3#t4	BSSID2	Internal face #2

Fig. 1. KEBAPP-enabled host

In this article, we take a step further from content sharing and host-centric communications, focusing and building on the prevailing application-centric computation and communication model *i.e.*, explicitly enabling access to the desired *processed* information through the concept of *application sharing*. Namely, we leverage application-centrism to facilitate information discovery through application-driven and application-defined, hierarchical namespaces. Given the ad-hoc nature of the proposed computation framework, our approach further extends these namespaces by introducing the concept of *keywords i.e.*, free-text or application-driven (GUI) parameters used to enable the invocation of applications at co-located mobile devices. This enables the description, discovery and retrieval of *processed* information, further supporting variable accuracy results, instead of only exact matches *e.g.*, a search result that does not contain all search terms. Note that the invocation of remote processing (in co-located smartphone or WiFi AP devices) is central to our framework, as opposed to previous work on retrieving static content from nearby devices. Our *keyword-based mobile application sharing framework (KEBAPP)*, manages connectivity in an application-centric way *i.e.*, coupling connectivity options/opportunities to applications and their namespaces. KEBAPP extends existing ICN primitives, namely CCN/NDN, thus resulting in a generic solution across different applications and overcoming the pitfalls of IP.

II. THE KEBAPP FRAMEWORK

In this paper, we present KEBAPP, a new application-centric information sharing framework oriented to support and provide opportunistic computing to mobile devices (smartphones, tablets, etc.). Our approach targets scenarios where large numbers of mobile devices are co-located presenting the opportunity for localised collective information exchange, decoupled from Internet-access. In this context, KEBAPP employs application-centrism to facilitate/enable (i) *the exchange of processed information, in contrast to merely static content*, and (ii) *the discovery and delivery of information partially matching user interests*.

Figure 1, presents the structure of a KEBAPP-enabled host. KEBAPP provides a new layer between the application and the link layers exhibiting three major design features. Namely, (i) *application-centric naming, where applications share common name-spaces and further support the use of a keywords* (Section II-A), (ii) *application-centric connectivity management (KEBAPP WiFi Manager), where applications manage connectivity by defining and/or joining WiFi broadcast domains* (Section II-B), and (iii) *information-centric forwarding, extending CCN/NDN primitives* (Section II-C).

A. Naming

The discovery and invocation of services/applications in the networking vicinity of a user builds on a naming scheme that enables the fine-grained description of the desired processed information. To this end, KEBAPP builds on the observation that mobile computing is largely application-centric, in the sense that users tend to access information using purpose-built applications. Application-centricity presents a series of important characteristics:

- Applications inherently support the structuring of the namespace within their semantic context. In turn, instances of the same (or similar) application can share the same namespace in describing the related information *e.g.*, categories in a news application.
- Applications are inherently used for computation, enabling the (lightweight) processing of content/information *e.g.*, searching, sorting data or computing a route.

$$\overbrace{\underbrace{\texttt{/a/b/c/}}_{\substack{\text{App} \quad \text{Market} \\ \text{App Developer}}}}^{\substack{\text{Hierarchical} \\ \text{Part}}} \oplus \overbrace{\underbrace{\texttt{\#tag1, \#tag2}}_{\text{App Developer}}}^{\text{Hash Tags}}$$

Fig. 2. Keyword-based Names

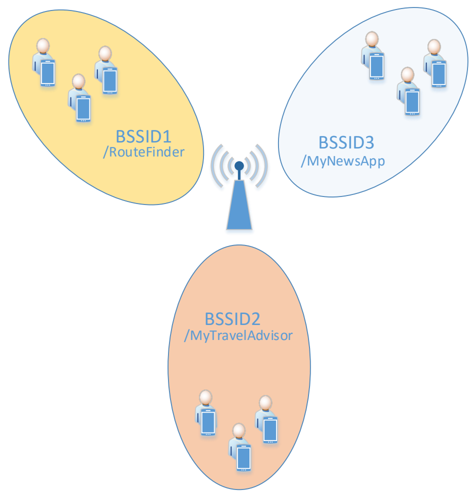
Taking these features into account, KEBAPP names are composed of two main parts (see Figure 2):

Fixed Hierarchical Part. It follows the hierarchical naming scheme of CCN/NDN and its purpose is to guarantee compatibility between instances of the same or different services/applications. Application developers can define their own hierarchical namespaces, enabling communication between different instances of the application *e.g.*, `/NewsApp/politics/international`.

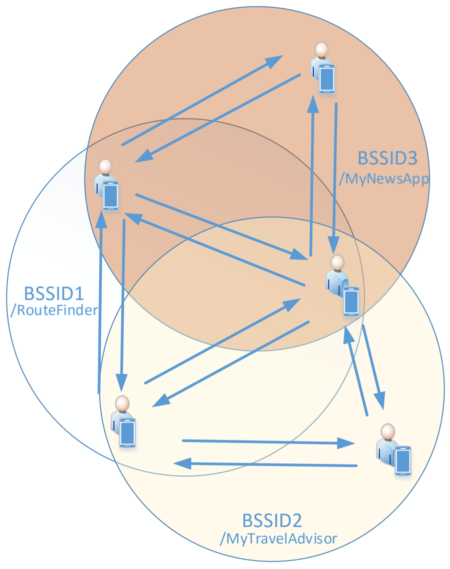
Moreover, application developers can also define suffixes corresponding to specific functionalities within their applications (in addition to static content), enabling this way the sharing of computation *e.g.*, the name `/MyTravelAdvisor/Top10Restaurants` is used to identify the list of the top-10 restaurants in a certain area.

According to our initial design the hierarchical part of the name will have to be an exact (longest prefix) match in order for a request to be served. It is noted though, that this matching is performed by the KEBAPP layer, with the user simply interacting with the application GUI *i.e.*, users need not be aware of the exact naming conventions.

Hashtags. The second part of the name comprises of hashtag-like free keywords, which the application developer can add to the application. The exact semantics of the hashtags depend on



(a) AP-assisted scenario



(b) WiFi Direct scenario

Fig. 3. Connectivity Options

whether the fixed hierarchical part of the name corresponds to static content or an application function(ality). In the former case, these keywords are used to semantically annotate the static content. This feature enables the partial matching of requests with available cache or routing/forwarding entries *i.e.*, given an exact match in the fixed hierarchical part of the name, hashtags can be used to support approximate matching, in turn enabling the search of information in nearby devices.

When the fixed part of the name identifies a certain application function(ality), the hashtag part of the name enables the passing of adequate parameters. In the aforementioned example of the *MyTravelAdvisor* application, the complete name included in a user request can have the fixed hierarchical part `/MyTravelAdvisor/Top10Restaurants` and the hashtags `#userrating, #London, #indian` indicating that the user is interested in the top-10 of the indian restaurants in London, according to users ratings. The submission of hashtag values is guided by the application GUI and can include both predefined value ranges *e.g.*, the sorting criteria for the top-10 restaurants, and free text fields *e.g.*, a user requests `/MyNewsApp/politics/search #Syria #negotiations` to use the search function of *MyNewsApp* and find anything related to negotiations for Syria.

B. Connectivity Management

Connectivity management plays a vital role in KEBAPP. In this work we focus on WiFi-enabled (IEEE 802.11) connectivity. This also includes WiFi Direct, which enables mobile devices to act as APs by forming communication groups. In KEBAPP, we propose the creation and use of 802.11 broadcast domains for the support of particular applications *i.e.*, KEBAPP-enabled hosts or APs advertise one or more Basic Service Set(s) (BSSs) for the support of one or more application(s). The creation of application-specific BSSs aims at enabling mobile devices to connect only when their counterparts support the same application and/or namespace. Within a BSS, hosts communicate employing CCN/NDN primitives, as described in Section II-C.

The advertising AP or host, through a WiFi Direct Group, acts as a mediator to connect different users willing to share the same application in a single broadcast domain. In the case of APs, functionalities such as access control, association, encryption, *etc.*, can be supported without imposing computation and/or battery overheads to mobile devices. Note however, that APs in this case need not provide access to the Internet. Figure 3a represents an AP-assisted scenario where different users share different applications.

The creation of an application-specific BSS requires the ability of mobile devices to identify the mapping between the BSS and the corresponding application. The recently announced WiFi Neighbour Awareness Networking (NAN) protocol [12] can support this requirement. Namely, WiFi NAN supports a low energy consumption device discovery mechanism enhanced with publish/subscribe primitives that can serve to retrieve what application is available in a certain BSS. Other technical approaches are also possible *e.g.*, employing the Access Network Query Protocol (ANQP) of IEEE 802.11u or using pre-defined SSIDs. It is noted that a device can be connected to more than one BSSs at the same time (*e.g.*, [13]), thus acquiring or providing information across several applications (Figure 3b).

C. Forwarding Operation

The basic forwarding operation of a KEBAPP node is a modified version of Named Data Networking (NDN) architecture [14]. The KEBAPP modifications aim at reflecting the forwarding of messages within the various BSSs a node may participate. As explained in the following, since we consider single-hop broadcasting domains, forwarding decisions lead to either the broadcasting of a message in the BSS or its delivery to a local application instance. As such, broadcast domains are considered as (inter)faces of a KEBAPP node.[4]

The KEBAPP forwarding scheme, similarly to NDN/CCN, has three main data structures: FIB (Forwarding Information Base), CS (Content Store) and PIT (Pending Interest Table). The FIB is used to forward Interest packets toward potential sources of matching data. The WiFi manager (see Figure 1) populates the FIB table with the name prefixes (hierarchical part of the name scheme detailed in Section II-A) advertised by the wireless networks in the vicinity *e.g.*, through WiFi NAN. As in NDN, a FIB table entry comprises the name prefix and a list of output face(s). In KEBAPP, the latter list (of output faces) includes the Basic Service Set IDentifiers (BSSIDs) advertised by other nodes. Moreover, when a KEBAPP node acts as information producer, providing information to other nodes, the output face list is further augmented with the `Internal_Face`, which enables a node to forward an Interest message to the local instance of the application.

The second main data structure, the CS, is responsible of caching the information requested by the users, providing fast fetching for popular information and avoiding to recompute information already requested by other users. Replacement policies of the CS is out of the scope of this paper.

Finally, the third data structure of a KEBAPP-enabled node, the PIT, keeps track of Interests forwarded to any BSS. KEBAPP keeps a PIT entry for every application request. Depending on whether an Interest message comes from the local application instance or another node in the corresponding BSS, the Requesting Faces list contains a handle to the local application instance (*i.e.*, the `Internal_Face`) or the BSSID of the corresponding BSS. In order to support delay tolerant communications, KEBAPP extends NDN's functionality, by allowing the creation of PIT entries even when no suitable destination *i.e.*, forwarding entry, has been found for the Interest message. At the same time, a PIT entry is extended to further indicate the Destination Face *i.e.*, the BSSID, it has been broadcast to or the corresponding `Internal_Face`. This serves the purpose of (re-)issuing Interest messages upon the discovery of a BSS that is associated with a matching name (in the FIB).

Figure 4 provides a representation of the KEBAPP packet forwarding engine.

In the following, we detail the operation of KEBAPP framework for an *information requester*:

1) The application requesting for information creates a new Interest.
2) The application looks for the information in the local CS. If the information exists locally, the data is sent to the application.
3) If the information is not found in the CS, the KEBAPP layer inserts an entry in the PIT (`<Internal_Face, name_prefix + keyword_list, null>`). As in NDN, we use the term "Internal Face" to point to the local application involved in the transaction (either as requester or as provider).
4) The KEBAPP (network) layer checks if there is a BSSID entry in the FIB matching the `name_prefix` of the PIT.[5] If an entry for the requested name prefix exists, the WiFi manager connects the WiFi interface to the BSSID in the FIB and broadcasts the Interest message with a corresponding time-out value.
5) Each time a new FIB entry is added because a new prefix name is discovered on a new BSS (*e.g.*, through WiFi NAN), the KEBAPP layer checks if a *pending* PIT entry for this prefix exists. As mentioned above, this corresponds to PIT entries created for Interest messages that could not be forwarded. In case an entry exists, the Interest is sent through the recently added BSSID, and the entry is updated with the BSSID value.
6) When a response is received with the information requested, the KEBAPP layer looks for the internal face that points to the application in the corresponding PIT entry and fordwards the response to it. The PIT entry is removed and the information requested is cached in the CS.

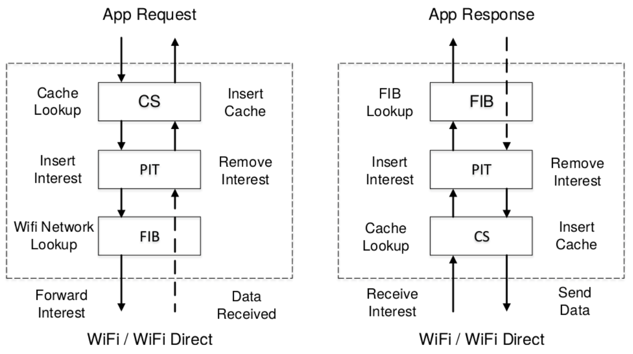


Fig. 4. Forwarding operations

Next, we describe the operation of the KEBAPP framework for an *information provider*:

1) The user receives an Interest through the interface connected to a certain BSS related to an application.
2) The KEBAPP layer checks the CS for a matching entry.
3) In case there is no entry in the CS matching the Interest, a PIT entry is first created. This entry allows the provider device to serve multiple, concurrently arriving, identical requests with a single message *i.e.*, applying multicast, as in original CCN/NDN. In this case, the Requesting Face list of the entry includes the BSSID of the current BSS. Subsequently, the FIB table is looked up and the `Internal_Face` is used to forward the Interest message to the corresponding application. For completeness, the `Internal_Face` is also added to the PIT entry as an output face.
4) The response from the application is cached in the CS and sent back to the broadcast domain indicated

[4]We focus on the KEBAPP functionality; details relating to the coexistence of KEBAPP with original NDN are out of the scope of this paper.

[5]Note that a local "Internal_Face" will never be used since this is a local request.

by the BSSID value of the local PIT entry, which is subsequently removed.

III. Use Case: RouteFinder App

When realising a *RouteFinder* application, the KEBAPP framework will have to deal with one of the two following cases. The first case is when some other device (either a client device or an AP) has previously setup a BSS, using WiFi in the case of an AP, or a WiFi Direct group in the case of a client device, advertising the corresponding application. The second case, is when no other BSS advertising the service required can be detected in the vicinity. In this second case, the user sets up a new BSS with the service *required* and waits for other users willing to connect to the BSS to share the requested application/service.

Let's assume user Alice is a tourist that just arrived in London by train. When she arrives at the train station, she wants to use her *RouteFinder* application to calculate the route to arrive at the hotel. However, Alice does not have Internet connection available because she does not want to pay expensive roaming data plans. In the station, there is an AP advertising the fixed hierarchical part of the name of the *RouteFinder* application Alice is running. To advertise the service name, the AP uses the WiFi NAN protocol. The *RouteFinder* application detects no Internet connection is available, but the BSS of the AP is advertising the service. The *RouteFinder* decides to connect to the BSS. Once connected, the *RouteFinder* application sends a request including their keyword preferences to the BSS. The user request should be like the following, `/RouteFinder/search #LondonStation #LondonHotel #Underground #Bus`, where the fixed hierarchical part of the name describes the service and the first hashtags are the origin and the destination of the route. The hashtags `#Underground` and `#Bus` are optional and define the route preferences.

User Bob, is a local passenger waiting for his train in the station, and he is running the *RouteFinder* application on the background in his mobile device. Bob is willing to share the *RouteFinder* service with other users. Since he has unlimited local data plan, he can run the service and calculate routes for other users. The application connects to the BSS and receives the request sent by Alice. Based on the keywords provided by the user, Bob's device executes the application and returns the result to Alice. Alice receives the calculated route on her handheld device transparently, without having Internet connection *i.e.*, without having to look for any public hotspot Internet connection or pay for expensive roaming data plans.

KEBAPP computations consume battery resources and possibly data resources of devices, as requests from neighbour nodes/users arrive. Therefore, an incentive system would likely motivate users to share their applications and their device resources, accordingly. A variety of existing approaches can be exploited to incentivise users to share their applications. For instance, in [15], the authors propose some form of micropayment system where users get a reward for every service they provide to their neighbours. Though it is out of the scope of this paper to explore incentives systems, we still note that embracing the application-centric model of current mobile computing is expected to facilitate such schemes. Mobile applications and the corresponding on-line services typically keep track of user behaviour through user accounts *e.g.*, to improve user experience or provide personalised services/advertisements. Though KEBAPP focuses on scenarios of limited Internet access, still, we envision a complementary role of both on-line and off-line application components to support such incentives schemes *e.g.*, keeping track of off-line user application sharing activity/contribution so as to support micro-payment schemes.

IV. Preliminary Results

A. Evaluation framework

For the evaluation of the proposed framework, we consider a KEBAPP-enabled tube map application that provides information on train lines and their respective schedule, as well as real-time information regarding delays, closed stations etc. We assume that a significant percentage of users commuting by tube have installed at least one such app on their smartphone. In particular, we consider a common scenario where a commuter wants to know if there are any delays on the lines that he/she needs to take in order to reach his/her destination, as well as the estimated time of arrival to the destination. By exploiting the proposed shared application framework, a KEBAPP user can take advantage of different applications (within the same context) shared by other users that have recently entered the tube station. In particular, by using as input simple information such as current location and destination of a user, the devices of nearby users compute the estimated arrival time, along with the route that the user needs to follow, and send a reply back to the user.

To evaluate the proposed application, we perform simulations based on mobility traces, from a subway station in downtown Stockholm[6].

For different numbers of KEBAPP users (which reflect the KEBAPP-enabled application penetration rate), we examine to what extent route calculation requests are successfully responded by other user applications. In particular, we randomly select a subset of the trace nodes, as the KEBAPP users, and randomly generate route calculation requests during their short stay in the station. We assume that a request can be successfully responded by only a small percentage (1, 5, or 10%) of the other KEBAPP users, which are also selected in random; this percentage corresponds to the users that have enabled the sharing capabilities of their KEBAPP application, and have access to the requested data, as well as sufficient battery level. We should also note that each commuter visits the tube station for a short period of time (*i.e.*, with an average stay of around three minutes), so the probability that two users that run a KEBAPP application do not exist in the subway station simultaneously is high.

All sets of simulations are repeated 1000 times. We measure the successful responses of route calculation requests, in terms of probability and response time. In particular, we measure: *i)* the *average Response Ratio (RR)*, *i.e.*, the fraction of the total generated requests that receive a successful reply, and *ii)* the *average First Response Time (FRT)*, which is the average time between each request issue and the first

[6]Trace was obtained from http://crawdad.org/kth/walkers/20140505/

reply, for those that are successfully responded. Note that in this preliminary experiment, we do not consider computation processing time and protocol-specific delays, as we do not focus on protocol implementation-details.

B. Evaluation results

Our evaluation results show that, as expected, the response ratio increases with the KEBAPP-enabled application penetration rate, as well as the percentage of the sharing users. In particular, even with a small penetration rate, (when only 100 out of the 3300 commuters are KEBAPP users), the KEBAPP users have a chance to get a successful response. As shown in Figure 5, the lowest response ratio of 9% is achieved when only one other passenger (during the simulation period of one hour) can successfully respond to the user request. The probability of a successful response gets significantly higher when more users are willing to share their resources; this probability is increased by a factor of 6.5 (from 9% with only one user sharing resources to 58.8% with ten users sharing resources out of 100 KEBAPP users). As the number of users increases, the response ratio approaches 100%.

As far as average response time is concerned, we notice that response time decreases as the number of users that are sharing resources increases, since the user that issues a request is more likely to encounter such a user. In particular, the maximum average delay is observed at the lowest sharing ratio of 1%, and it spans from 17-29 seconds. For increased information availability, a user request can almost immediately find a match, within a few seconds.

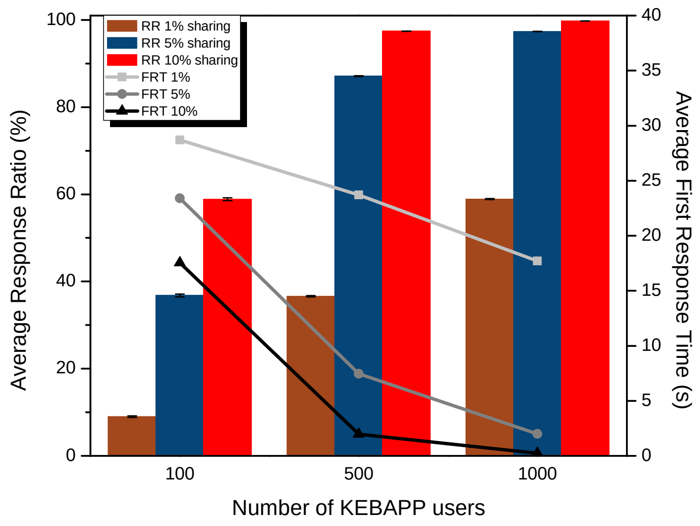


Fig. 5. Evaluation results

V. Conclusions

We have introduced the concept of *Keyword-Based Mobile Application Sharing* (KEBAPP), according to which mobile clients can make use of applications in nearby devices, *i.e.*, not necessarily only in their own mobile device. This requires applications installed in mobile devices to also act as servers, apart from their normal role as clients. KEBAPP builds on ICN principles and forms requests based on keywords and hashtags in order to invoke computation in nearby devices. As a last step, *processed information* is returned back to the requesting client.

Our proof-of-concept simulation results, built on a trace of commuters in Stockholm metro network, shows that even a tiny number of KEBAPP-enabled users can provide a significant service coverage to nearby users. With this study we attempt to start a new thread in the information-centric mobile computing area and trigger further research from both implementation and evaluation perspectives.

References

[1] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, “Identifying diverse usage behaviors of smartphone apps,” in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC ’11, (New York, NY, USA), pp. 329–344, ACM, 2011.

[2] A. Valcarce, T. Rasheed, K. Gomez, S. Kandeepan, L. Reynaud, R. Hermenier, A. Munari, M. Mohorcic, M. Smolnikar, and I. Bucaille, “Airborne base stations for emergency and temporary events,” in *Personal Satellite Services* (R. Dhaou, A.-L. Beylot, M.-J. Montpetit, D. Lucani, and L. Mucchi, eds.), vol. 123 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 13–25, Springer International Publishing, 2013.

[3] I. Wakeman, S. Naicken, J. Rimmer, D. Chalmers, and C. Fisher, “The fans united will always be connected: building a practical dtn in a football stadium,” in *ADHOCNETS 2013, 5th International Conference on Ad Hoc Networks*, (Barcelona, Spain), October 2013.

[4] J. Scott, P. Hui, J. Crowcroft, and C. Diot, “Haggle: A networking architecture designed around mobile users,” in *Proceedings of the Third Annual IFIP Conference on Wireless On-Demand Network Systems and Services (WONS 2006)*, IEEE, January 2006.

[5] C. Anastasiades, T. Braun, and V. Siris, “Information-centric networking in mobile and opportunistic networks,” in *Wireless Networking for Moving Objects* (I. Ganchev, M. Curado, and A. Kassler, eds.), vol. 8611 of *Lecture Notes in Computer Science*, pp. 14–30, Springer International Publishing, 2014.

[6] U. Drolia, N. Mickulicz, R. Gandhi, and P. Narasimhan, “Krowd: A key-value store for crowded venues,” in *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, MobiArch ’15, (New York, NY, USA), pp. 20–25, ACM, 2015.

[7] J. Ott and M. J. Pitkanen, “Dtn-based content storage and retrieval,” in *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pp. 1–7, June 2007.

[8] E. Trono, Y. Arakawa, M. Tamai, and K. Yasumoto, “Dtn mapex: Disaster area mapping through distributed computing over a delay tolerant network,” in *Mobile Computing and Ubiquitous Networking (ICMU), 2015 Eighth International Conference on*, pp. 179–184, Jan 2015.

[9] J. Ott, E. Hyytia, P. Lassila, T. Vaegs, and J. Kangasharju, “Floating content: Information sharing in urban areas,” in *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pp. 136–146, March 2011.

[10] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos, “A survey of information-centric networking research,” *Communications Surveys Tutorials, IEEE*, vol. 16, pp. 1024–1049, Second 2014.

[11] M. Sifalakis, B. Kohler, C. Scherb, and C. Tschudin, “An information centric network for computing the distribution of computations,” in *Proceedings of the 1st International Conference on Information-centric Networking*, ICN ’14, (New York, NY, USA), pp. 137–146, ACM, 2014.

[12] D. Camps-Mur, E. Garcia-Villegas, E. Lopez-Aguilera, P. Loureiro, P. Lambert, and A. Raissinia, “Enabling always on service discovery: Wifi neighbor awareness networking,” *Wireless Communications, IEEE*, vol. 22, pp. 118–125, April 2015.

[13] A. J. Nicholson, S. Wolchok, and B. D. Noble, “Juggler: Virtual Networks for Fun and Profit,” *IEEE Transactions on Mobile Computing*, vol. 9, pp. 31–43, Jan. 2010.

[14] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT ’09, (New York, NY, USA), pp. 1–12, ACM, 2009.

[15] D. Syrivelis, G. Iosifidis, D. Delimpasis, K. Chounos, T. Korakis, and L. Tassiulas, “Bits and coins: Supporting collaborative consumption of mobile internet,” in *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pp. 2146–2154, April 2015.