MobiTribe: Cost Efficient Distributed User Generated Content Sharing on Smartphones

Kanchana Thilakarathna, Henrik Petander, Julián Mestre, and Aruna Seneviratne

Abstract—Distributed social networking services show promise to solve data ownership and privacy problems associated with centralized approaches. Smartphones could be used for hosting and sharing users data in a distributed manner, if the associated high communication costs and battery usage issues of the distributed systems could be mitigated. We propose a novel mechanism for reducing these costs to a level comparable with centralized systems by using a connectivity aware replication strategy. We develop an algorithm for grouping devices into tribes for content replication among intended content consumers and serve it using low-cost network connections. We evaluate the performance of the algorithm using three real world trace data sets. The results show that a persistent low-cost network availability can be achieved with an average of two replicas per content. Additionally, cellular bandwidth consumption and energy consumption of users are evaluated analytically using user content creation and consumption modeling. The results show that the proposed mechanism lowers monetary and energy costs for users compared to non-mobile-optimized distributed systems irrespective of the content demand model.

Index Terms—Mobile distributed decentralized social networking, user generated content sharing, mobile data traffic off-loading

1 INTRODUCTION

There has been significant growth in the use of free social networking services, such as Facebook, and as well as services that support the distribution of user generated content (UGC), such as YouTube. With advanced capabilities of smartphones, users increasingly use mobile devices to access these services, while creating virtual social communities through mobile social networking. All of these popular social networking and UGC sharing services use mobile device as a thin client with all the data being centrally hosted on the servers of the service provider. Even though the cost of storage and distribution is high when using centralized architectures, the service providers prefer to host the service centrally as it allows the mining of user data and generating revenue from other sources, e.g., advertisements. Advantages for the user are; being able to access the content from any device, and not having to worry about maintenance. However, these advantages come at the cost of reduced privacy and losing the ownership of their data.

There have been a number of proposals aimed at addressing issues of privacy and the user losing control of their data [11], [18], [23], [27]. These proposals use distributed decentralized social networking (DDSN) architectures that enable an individual user or a community of users to host data. In general, these proposals require the users to have always on devices for hosting the data or rent resources. The hosting service may also be provided by a third party as a cloud service. However, if the hosting service is provided by a third party, there is potential for leaks of privacy and loss of control of the data as user data can be mined [25], [39]. Also there is the inconvenience and the costs of hosting, which will limit its appeal and usability.

Smartphones and tablets of today have sufficient computing power, storage capacities to host users content and are always connected to a network. Furthermore, users of these devices are likely to be the biggest consumers and producers of content. Therefore if the capabilities of these device could be leveraged, they could form the platform for providing DDSN services without the drawbacks of communication costs and loss of privacy. However, the availability of user mobile devices vary depending on time and location. Existing DDSN proposals have not been designed with this variability in mind, drastically limiting their applicability in mobile computing environments. For example, Safebook [11] replicates content on a number of devices without consideration of communications costs or power usage. Moreover, sharing of high volume multimedia content over cellular networks exacerbates the problem associated with the exponential increase in mobile data traffic that has been widely predicted [9].

MobiTribe architecture provides DDSN services using smartphones without negatively impacting the communication costs and battery usage of mobile devices [33], [34]. We showed that it is possible to exploit the time elasticity of social networking content, harness the advanced capabilities of mobile devices and the fact that the user will have access to high speed low-cost network such as WiFi networks, to provide social networking services on smartphones.
In this paper, we extend the work with predictive pre-fetching that replicates content on devices of users who are highly likely to view the same content. This pre-fetching minimizes the replication overhead and provides high quality user experience with instant accessibility for services such as on-demand mobile video through a DDSN service. Further, this paper provides a deeper analysis of the architecture with realistic content creation and consumption modeling. The results show that peer-to-peer store and forward architectures can provide the same functionality as centralized architectures, but with lower distribution costs and energy consumption. Moreover, it is always more cost efficient than content sharing via non-mobile-optimized distributed social networking schemes.

This paper makes the following contributions:

- Shows that the content replication problem in distributed peer-to-peer architectures is \( NP\text{-Hard} \) and then present a new algorithm based on a combination of a bipartite b-matching and a greedy heuristic, which minimizes content replication and maximizes content availability whilst ensuring fairness.

- Using real data traces, we show the viability of the proposed algorithm, namely that it is possible to achieve persistent low-cost network availability with only two replicas.

- For the particular traces, we evaluate the performance of MobiTribe for sharing content in distributed social networks by analytically modeling content creation and consumption behavior.

- We benchmark the performance of MobiTribe against alternative social networking approaches and show that MobiTribe saves 76\% of the uplink and 46\% of the downlink cellular bandwidth compared to current widely used centralized server based architectures.

The remainder of this paper is organized as follows: Section 2 presents related work. This is followed by an overview of the MobiTribe system architecture in Section 3. In Section 4, we formally define the content replication problem in peer-to-peer architectures. Section 5 presents our device grouping algorithm followed by the evaluation in Section 6. Then, in Section 7 system performance is evaluated in terms of communications cost and energy efficiency. Finally, Section 8 concludes the paper.

2 RELATED WORK

2.1 Distributed Social Networking

Diaspora [18] can be considered as the only widely used DDSN service. It is fully funded by user donations and estimated to have over 1.5 million users as of March 2012 indicating a demand for privacy-aware social networking. In Diaspora, users need to set up their own server for hosting their content. PrPl [28] is another proposed service which enables a user to run the service on a home server, or rent a service to host their own data. In both systems if a mobile device is used to host the content, either the availability has to be compromised or will result in increased communications costs.

Safebook [11] is based on the concept of decentralization and collaboration among friends and creates a secure social network. Similar to MobiTribe, friends are assumed to be cooperative and friends’ devices are used for replication to increase the availability. MyZone [23] also deploys user profile replicas on the devices of trusted friends to increase the availability of the profile. SuperNova [29] is another recently proposed DDSN that uses content/profile replication similar to MobiTribe. The idea of all these systems is to increase the online availability of content. They do not attempt to lower the communication cost. None of the systems explicitly consider mobile devices, and availability is dependent on the number of replicas. Hence, similarly to Diaspora and PrPl, if used with mobile devices, it would result in increased communication costs and energy consumption. Contrail [31] is a cloud-based architecture to enable distributed social networking on smartphones which is again conceptually similar to MobiTribe. The cloud relays content between users and stores content for offline recipients to increase the reliability, however this can lead to loss of privacy and user control.

2.2 Distributed Caching for Content Sharing

Ioannidis et al. [19] proposes a distributed caching mechanism to optimize the delivery delay in a opportunistic content dissemination system. The system model assumes that users exchange content only when they are directly connected, whereas in MobiTribe we propose to share content when users are connected to low-cost networks irrespective of the location. Furthermore, since the system is fully distributed, it does not ensure minimum replication. Tribler [27] is a peer-to-peer file sharing system, where peers replicate their contextual information related to similar interests with other peers. Tribler does not consider methods of increasing the availability of the content or minimizing communication costs.

Sharma et al. [30] proposed a friend-to-friend content replication strategy to ensure minimal replication and maximal availability. The proposed method does not ensure fairness of resource usage of devices. An erasure coding based friend-to-friend storage system is proposed in [16]. These coding based redundancy techniques are not suitable for social networking content due to relatively small content size and frequent access of the content as discussed in [29].

2.3 Predictive Mobile Data Traffic Offloading

In ActiveCast [22], the content is pre-fetched and cached in different locations including end user terminals via low-cost networks. ActiveCast does not address uploading or hosting of UGC, and relies on accurate prediction of future demand [22]. Han et al. [17] have proposed a target set selection mechanism for propagation of the content with opportunistic communication. This mechanism will only work well for popular content, and the focus of all this work is on dissemination of data from a “centralized” data store and as a result does not address user privacy.

Lee et al. [21] presents a quantitative analysis of offloading 3G data traffic. Wiffler [3] augments mobile 3G capacity, taking advantage of delay tolerant applications. It predicts near future WLAN availability and delays the transmission to offload data into WLAN. In contrast MobiTribe
is a distributed architecture, which additionally supports on demand uploads. Further, none of these mobile data offloading schemes address the issues of loss of control of data or user privacy.

3 THE MOBITRIBE ARCHITECTURE

The proposed MobiTribe architecture is shown in Fig. 1. To preserve the privacy, a Creator distributes the content to be shared to two groups of devices from the creators’ social networking friends as shown in Fig. 1(c), 1). Mobile Private Storage Tribe - mTribe, as described in Section 3.3, 2). Predicted Consumers, i.e. devices that are predicted to consume the content. Each user device identifies potential shared content to consume, based on a modified content recommendation algorithm presented in [37]. This is achieved by each device maintaining and periodically updating three matrices for content recommendation, 1). user-user social relationship matrix, 2). content-content similarity matrix based on keywords and tags of contents, 3). user-content matrix, which shows the relationship between content access and users. Then the content shared by friends are ranked according to the relevance to the user for pre-fetching as described in [37].

The content replication on the mTribe is carried out as described in Section 3.1. A Consumer would then access the shared content pre-fetched or stored on the mTribe as detailed in Section 3.2. The architecture relies on a centralized entity (Content Management Server - CMS) to track the addresses and current connectivity of devices hosting content and to manage the peer-to-peer content distribution.

3.1 Sharing of Content

A creator sharing content via a social networking application such as Facebook would result in MobiTribe initiating the content sharing as a two step process.

1). Registration: The content is registered with the CMS immediately after creation. The registration is done through any available network as shown in Fig. 1(a). Then, a link to the content, pointing to the CMS, is advertised to the users who subscribed to receive updates from the content creator through a social networking service, e.g. Facebook.

2). Replication: Each subscribed device then decides whether to pre-fetch the content, based on predicted demand and informs the CMS of their intention of downloading the content. The CMS decides whether these devices, based on the availability patterns, provide sufficient availability of the content for the other (on-demand) consumers. If not, an extra set of devices, namely an mTribe is created as shown in Fig. 1(c) to replicate the content. The replication of the content takes place when the creator and the devices in the mTribe have access to low-cost networks. The low-cost network connections are considered to be networks with spare capacity, WLANs or device to device connections. The CMS coordinates the replication process acting as a tracker. The replication strategy ensures that the devices in the mTribe have complementary low-cost network access patterns, thus making it highly probable that one of the devices in the mTribe can serve the content over a low-cost network connection.

3.2 Accessing of Content

A consumer initiates the content downloading process by clicking on an advertised notification appearing in the social networking application as shown in Fig. 1(b). This triggers the content downloading process through:

1). Local device (Cache hit): If the consumer’s device has successfully predicted the content access and the pre-distribution process has been completed, the requested content would have already been downloaded to the device. If the content is not available on the device, i.e a cache miss, the link to the CMS which is included in the notification feed, triggers the content downloading process.

2). Mobile tribe (Cache miss): When the CMS receives a request for content, it directs this request to the devices in the mTribe that could serve the requested content. The choice of the mTribe devices to forward the request is based on the real-time network connectivity and the load of available devices in the mTribe at the time of the request.

3). Download from custodian: When a request for content is received by the CMS, if the content is not available in the mTribe, the request is forwarded to the content creator. The content creator has the option of delivering the content over the lowest cost data connection available to the creator or make it unavailable until the pre-distribution of content to mTribe has been completed.

A content consumer has the option of downloading the content over whatever network connection available or the content can be scheduled to be downloaded through a low-cost network. In contrast, the upload path, i.e. from mTribe to fixed network edge, always uses low-cost network capacity.
Upon downloading the content, the consumer also acts as a seeder to share the content for a defined period of time, thus helping the system to scale with demand. To perform the content sharing between the content creator, devices in the mTribe and content consumers, an augmented BitTorrent peer-to-peer protocol, with the CMS acting as a tracker to balance the traffic load, is used in MobiTribe. The tracker considers the real-time resource constraints of mobile devices such as battery and storage limitations, when selecting peers for the content sharing process.

### 3.3 Replicating of Content

An mTribe device, may not be able to provide access to the content continuously due to three primary reasons, namely the communications costs, battery usage, and not being “on” all the time. This is overcome by replicating content on devices that have complementary connectivity, sufficient battery power and being “on”. Moreover, through this process, if the traffic is routed through a low-cost and/or less congested network, it will result in significant benefits to both the service providers and the users. The downsides of the content replication are the overheads associated with replication. These disadvantages can be minimized as the mTribe members, where possible, are chosen from the potential consumers, i.e. subscribers to receive updates from the creator. Then, there is no overhead as they are likely to consume the content.

Device grouping for content replication is performed by the CMS, as it has a global view of the system state in terms of both mobile devices and network infrastructure. If the CMS is hosted by the mobile network operator, the communications cost can be further optimized by analysing network load information. The CMS carries out the grouping periodically or in response to a significant degradation of the availability of a tribe’s content. The device grouping is done using the context information of devices and network infrastructure as described in Section 4.

The storage on mTribe devices is managed by removing the old replicated content using standard cache replacement strategy, namely LRU. The creator has the option to remove/archive or keep its own content. If old content becomes popular, it will be treated as a newly generated content.

### 3.4 Privacy and User Control of Data

In online content sharing services, when a user accesses the service, there is some loss of privacy through direct and indirect channels as described in [10]. Thus, it is impossible to completely prevent leakage of privacy. Our aim is to minimize the loss of privacy compared to the existing centralized content sharing architectures.

This is done in MobiTribe by keeping user data away from third party service providers thereby limiting the ability to mine user data or profile users. The CMS only facilitates the interactions among friends and does not have access to any user data stored on mTribe devices. The users can retrieve any content hosted on the mTribe back through the MobiTribe service. As a result the privacy of the user and ownership of the data are preserved. It is possible to argue that the content replication results in a loss of privacy. In MobiTribe, this is minimized by specifically selecting the devices of friends and family of the content creator to create the mTribe, i.e. it only replicates on devices of the people with whom the content creator intends to share the content with.

The device context information such as available network types, the battery level, AC power availability and the amount of spare storage capacity on the mobile device are required to determine the ability of a device to host others’ content. To avoid leakage of privacy, these user context information is aggregated to a single bit which indicates the availability of the device to host data, *Device Availability*, as described in Section 4. Although storing of the aggregated context information on a centralized server may result in some loss of privacy, the design allows using of non user accessible information such as cellular network load information to lower the cost of content delivery.

### 3.5 User Incentives

User privacy and the user control of data is inherent in the system design which is the primary incentive to use the service. One of the major reasons behind the low acceptance of the previous DDSNs is that they only focus on privacy preservation and do not address the fundamental issues of cost, energy usage and ease of use. In contrast, MobiTribe addresses these issues in addition to privacy.

Predicting the consumption of content allows prefetching of the content to the device, when there is access to a low-cost network. Firstly, this reduces the monetary and energy costs of delivering the content as shown in Section 6. Secondly, the user experience would improve dramatically both in terms of the startup delay and for the re-buffering, which according to Bitmobile [5], caused stalling of 5-40% of videos. Finally, due to the operator managed data offloading and traffic localization capabilities, the operator could provide price incentives such as family and friends data plans as in the case of voice communication. Thus, we believe that MobiTribe would be able to incentivize mobile users to take part in distributed mobile social networking.

### 4 Content Replication in MobiTribe

The device grouping algorithm identifies mTribe devices to maximize the availability of a content via low-cost networks. The content replication process is managed by the CMS. Each device periodically (daily) updates the CMS with its binary *Device Availability* pattern, i.e. its ability to host others’ content. We showed in [34], that the history of device availability patterns can be effectively used to infer future availability patterns of a device and groups of devices.

Let the availability of a device $i$ be $A_i^t$, where $A_i^t = 1$, if the device is capable of hosting others’ content during the time slot $t$ and $A_i^t = 0$ otherwise. Then, the probability of

1. Description of the practical implementation is provided in [32].

2. We compared playback of pre-fetched video on an Android device from local storage with streaming over loaded and unloaded 3G cellular, and the startup delays were 0.5s, 7s and 25s.
low-cost network availability of a tribe of devices (G) can be defined as follows, where |T| represents the number of time slots considered to infer future availability, the “Training Period”.

\[ P_a(G) = \frac{\sum_{i \in T} \bigcup_{e \in G} A_i^e}{|T|}, \quad A_i^e = \begin{cases} 1 & \text{Available} \\ 0 & \text{Not available} \end{cases} \]

If users have regular behavioral patterns, it is possible to use \( P_a(G) \) as a metric to predict future availability as discussed in [34]. If \( P_a(G) \) is higher than a threshold, the “Threshold of Pairing” (\( P_{th} \)), we consider that tribe G satisfies the availability constraint and can be used to host others’ content in the future. However, we cannot implicitly select tribes based on threshold of pairing because replication consumes scarce resources. Therefore, we consider two additional constraints in device grouping to ensure the resource limitations are taken into consideration.

1) **Minimum Replication:** A high level of replication will disrupt the system performance by creating additional traffic flows and overheads, wasting battery life and storage of mobile devices. Hence, the device grouping trades-off replication to availability. The minimum replication is ensured by defining a Limit of Replication, i.e. the maximum allowed size of a tribe.

2) **Fairness:** If a device is selected by many users to host data, it would drain energy and storage of that device. The fairness of the system is ensured by defining the Limit of Hosting, i.e. the maximum number of tribes that a single device can belong to.

When these two constraints are satisfied, a device included in the mTribe is said to be covered. Thus, the content replication problem becomes “how to find a maximum cover while satisfying minimum replication and fairness constraints”.

**PROBLEM DEFINITION**

A hypergraph is a pair \((V, E)\), where \(V\) is a ground set of elements and \(E\) is a collection of subsets of \(V\). The rank of a hypergraph \(H = (V, E)\) is defined as \(\Delta_H = \max_{e \in E} |e|\); we will drop the subscript \(H\) when the hypergraph is clear from the context. For a subset \(C\) of \(E\) and a vertex \(u\) \(\in V\), we denote the degree of \(u\) in \(C\) by \(\deg_C(u) = |\{e \in C : u \in e\}|\).

In the DEVICE GROUPING (DG) problem, we are given a hypergraph \(H = (V, E)\) and a capacity function \(c: V \rightarrow \mathbb{Z}^+\). The objective is to find a subset \(C \subseteq E\) that maximizes the number of vertices spanned by \(C\), namely;

\[
\text{Maximise} \quad \left|\{u \in V : \deg_C(u) \geq 1\}\right|
\]

subject to \(\deg_C(u) \leq c(u)\) for all \(u \in V\).

The relation between DG and our application is as follows. The vertex set \(V\) represents the set of devices. The hyperedge set \(E\) is the collection of subsets of \(V\) satisfying the availability constraint, \(P_{th}\). The parameter \(\Delta\) is the maximum allowed group size, i.e. the limit of replication. Finally, the capacity function \(c\) represents the limit of hosting. The objective is to perform a grouping that covers (spans) as many devices as possible.

**Theorem 1.** DG is NP-Hard even when \(\Delta = 3\) and \(c(u) = 1\) for all \(u \in V\).

**Proof.** Let \((V, H)\) be a hypergraph. A matching is a subset \(C\) of \(E\), where the sets in \(C\) are pairwise disjoint. The matching is perfect if every vertex is covered by some edge in \(C\); namely, \(\cup_{e \in C} e = V\). When \(\Delta = 3\), testing if such a matching exists is known as the 3D-matching problem and is one of Karp’s original 21 NP-Hard problems [20].

The hardness of DG follows immediately since the case specified in the theorem statement is the problem of finding the largest subset \(C \subseteq E\) such that sets in \(C\) are pairwise disjoint. \(\square\)

5 DEVICE GROUPING ALGORITHM

5.1 A Tractable Special Case of DG

Even though the problem is hard when \(\Delta \geq 3\), we can show that the problem is solvable in polynomial time when \(\Delta = 2\), i.e. when \((V, E)\) is a graph. This shows that the hardness proof in the previous section cannot be represented as a simpler problem.

In most cases our problem can be reduced to a single bipartite \(b\)-matching computation for which highly tuned implementations are available (the problem reduces directly to maximum flow). An instance of the \(b\)-matching problem is specified by a bipartite graph \((A, B, E)\) and a function \(b: A \cup B \rightarrow 2^+\). The objective is to find a maximum cardinality subset \(M\) of \(E\) such that \(\deg_{bm}(u) \leq b(u) \forall u \in A \cup B\).

**Theorem 2.** When \(\Delta = 2\) and \(c(u) > 1\) for all \(u \in V\), we can reduce DG to a single bipartite \(b\)-matching computation in linear time.

**Proof.** Let \((V, E, c)\) be our DG instance. We construct a bipartite graph \((A, B, E)\) where \(A = \{v : v \in V\}\) and \(B = \{v' : v' \in V'\}\); for each \((u, v) \in E\) we include two edges \((u, v')\) and \((v, u')\) in \(E\). In addition, we set \(c(u) = 1\) and \(c(u') = b(u) \forall u \in V\).

Any solution to the DG instance induces a solution to the \(b\)-matching problem with the same value, and vice versa. To prove the forward direction, let \(C \subseteq E\) be a solution to DG. For each vertex \(u\) covered by \(C\), choose an arbitrary edge \((u, v) \in C\) incident on \(u\), and add \((u, v')\) to the matching. Then it is trivial to show that the cardinality of the matching equals the number of devices covered by \(C\) and that the matching is feasible. In the reverse direction, let \(M\) be some \(b\)-matching. Construct a directed graph \((F, V)\) such that if \((u, v') \in M\) by adding the directed edge \((u, v)\) to \(F\). These edges represent that device \(u\) is being covered by \(v\). The goal is to find a solution for DG that covers the same number of devices as \(M\). Let \(\deg_{F}^{+}(u)\) denote the number of edges in \(F\) going out of \(u\), and \(\deg_{F}^{-}(u)\) denote the number of edges coming into \(u\). Note that for every vertex \(u \in V\), \(\deg_{F}^{+}(u) \leq 1\) and \(\deg_{F}^{-}(u) \leq c(u)\); this follows immediately from \(b\)-matching constraints. Therefore in the underlying undirected graph induced by \(F\), each connected component will be made up of one-trees (a tree plus one edge, or equivalently, a graph with a single cycle).

It is not possible to disregard the directions of the edges in \(F\) and use this as our DG solution, because the combined in and out degree of a vertex \(u\) can be \(c(u) + 1\). First, we prune unnecessary edges. A node is a leaf in \(F\), if \(\deg_{F}(u) = 0\). Then, if there is a node \(u\) such that
degre^+(u) = 1 and degre^-(u) ≥ 1 and its in-neighbourhood is made up of leaves, delete the edge going out of u from F iteratively. When it is not possible to delete any more edges, the remainder is a collection of disjoint stars and directed cycles with some additional edges pointing to vertices in the cycles as shown in the second graph in Fig. 2. If a vertex u in a cycle has an incoming edge from outside the cycle, then remove its outgoing edge from F as shown in the third graph in Fig. 2.

It can be easily shown that if a vertex u still has its outgoing edge, u is a leaf or it was a cycle node with a single incoming edge from another vertex in a cycle. Note that in either case degre^+(u) + degre^-(u) ≤ 2. For all other vertices with no out-going edges, degre^+(u) + degre^-(u) ≤ c(u). Therefore, if the directions of edges in F are disregarded, the solution is feasible for DG because the c(u) > 1 ∀u ∈ V by the assumption made in the theorem statement.

Now the only requirement is to show that every vertex covered by the b-matching is also covered by the DG solution. Note that every time an edge (u, v) is removed from F, there is always another edge (v, u) with degre^+(v) = 0; thus, it guarantees that (x, u) will survive further pruning and remain in F until the end. Therefore, vertex u will be covered by the (undirected) edge (x, v) in the proposed DG solution.

### 5.2 A Greedy Algorithm for General Instances

For general hypergraphs (V, E), the problem cannot be solved in polynomial time, since it is NP-Hard. However, it is possible to approximate the problem of covering the device efficiently using a greedy heuristic with an approximation ratio bounded by Δ, the size of the largest set in E. The algorithm works in iterations by building a solution C. In each iteration, it finds an edge e that can be safely added to C without violating the capacity constraints so as to maximize the number of newly covered elements.

Note that the set systems defined by the feasible solutions of the DG was shown to be Δ-extendible [24] and the maximization objective is submodular. For such a problem, Chekuri et al. [6] showed that greedy is a Δ + 1 approximation. However, for our special objective function, a slightly better approximation is possible.

**Theorem 3.** Let (V, E, b) be an instance of DG, then GREEDY is a Δ-approximation algorithm.

**Proof.** Let O be an optimal solution. The idea is to assign devices covered by O to devices covered by C, so no device covered by C is assigned more than Δ devices covered by O. First, assign sets in O to sets in C.

Suppose that GREEDY picks a set e in a given iteration. For each u ∈ e, find some set fu in O, if any, such that u ∈ fu; assign fu to e and remove fu from O. At the end of the algorithm all sets in O must be assigned. Indeed, let f be some set in O. If degre^+(u) = c(u) for some u ∈ f, then clearly f must have been assigned. Otherwise, degre^+(u) < c(u) for all u ∈ f, but this means that GREEDY left the while-loop prematurely because f can be safely added to C, a contradiction.

Because of the greedy choice, when adding e to C the number of newly covered devices cannot be larger than if fu is added to C. More specifically, |e \ D| ≥ |fu \ D| for the set D just before adding e to C. Thus, it is possible to distribute the devices covered by {fu} to devices covered by e so that each newly-covered device in e receives at most Δ devices from {fu}. Therefore, the overall number of devices covered by O is no more than Δ times the overall number of devices covered by C.

### 5.3 Summary of Device Grouping

DG will be carried out at a central entity, namely the CMS. Each device periodically updates the central entity with its recent device availability patterns (A^i). From these availability patterns, an undirected graph among all pairs of devices is generated to solve DG for the case of Δ = 2, i.e. the maximum tribe size of two. Assume that there exists an edge between two devices u and v, if P_v(u, v) > P_{th}. Fig. 3(a) shows an instance of a simple graph created from availability patterns of five devices. Next, a bipartite graph is constructed to implement a single bipartite b-matching reduction as explained in Theorem 2. For each edge (u, v) in the original graph, edges (u, v) and (v, u) are added in the bipartite graph. Since the b-matching can be reduced to a maximum flow computation, a flow network is generated by introducing two extra nodes s and t as in Fig. 3(b). Each incoming edge to node t from a device u is assigned the capacity equal to c(u), i.e. the limit of hosting. All the other edges are assigned the capacity value of one. In the example network in Fig. 3(b), though device 2 is connected to all other devices, it can only host the content of two others. Thus, the limit of hosting ensures that device 2 only takes part in a maximum of c(2) = 2 tribes. Once the maximum flow from s to t is computed, we prune extra edges from the resulted directed graph as illustrated in Fig. 2. In this particular example, edges (4, 2) and (2, 1) are pruned from the resulting graph as shown in Fig. 3(c). Finally, the direction of the edges are ignored and the resulted graph (Fig. 3(d)) is considered as the solution for the DG problem for the case of Δ = 2. As the next step, if there are any ungrouped devices, the GREEDY approximation algorithm (Algorithm 1) proved in Theorem 3 is used to find tribes for the remaining devices.

### 6 Performance of Device Grouping

Effectiveness of device grouping can be determined by 1) the aggregated low-cost network availability of the
selected groups, namely the mTribes, 2) the amount of replicated content and 3) the number of tribes that a single device belongs to. The higher the low-cost network availability, the lower the replication, the smaller the tribe membership of devices, and the more effective the device grouping. Since these three parameters depend on user behavior, time and location, the analysis needs to consider real mobile users connectivity patterns. We used three data sets to evaluate the effectiveness of the proposed DG algorithm. The context information used for low-cost network selection was limited to WiFi connectivity.

6.1 Data Sets and Simulation Setup
The Rice Community [1] trace data set has WiFi connectivity information with a one minute granularity for 14 users in Houston for 6 weeks. The CoSphere trial [1] also contains WiFi connectivity patterns of 12 users for 6 weeks. Since these data sets are individually too small to evaluate device grouping, we combined these two data sets and created one data set (DS1). We assume that these users are social networking friends in the same time zone for this evaluation. The data set from South Korea (DS2) has WiFi connectivity patterns of 97 iPhone users spanning over 18 days with a 3 minute granularity [21]. When a device has WiFi connectivity, the device is considered to be available via a low-cost network. Neither data set has consistent connectivity patterns, i.e. there are time instances when the exact state of WiFi connectivity is “unknown”. We consider all unknown states as representing WiFi unavailability to evaluate the worst case performance. The worst case average WiFi availability for DS1 is 50%, and for DS2 43%. This indicates that there are devices having very low WiFi connectivity. Altogether, the data sets contain 1470 days of connectivity patterns from 105 users in 3 different sites at 3 different times.

The trace data sets are divided into training and evaluation periods. The training period is used to perform the tribe calculation and the performance of the selected tribes are evaluated during the evaluation period3. The device grouping is carried out daily. If one device in the selected tribe is available via WiFi, the content stored in the tribe is considered to be available. The availability of all selected tribes are calculated and the same procedure is repeated over the entire trace data period to obtain average system availability, Availability of MobiTribe. It has been observed that the fairness constraint described in Section 4, i.e. the maximum number of tribes that a single device belongs to, namely the Limit of Hosting, affects the efficiency and the performance of grouping. The main reason for this is that the devices with high availability would be preferred by many others to host their content. Hence, we measured the availability of a tribe against the limit of hosting. We evaluated the device grouping algorithm for all three data sets thoroughly in our previous work [33] and here we present the summary of the evaluation of the device grouping algorithm.

6.2 Availability-Replication-Fairness Trade-Off
Fig. 4(a) shows the availability of content hosted on mTribes for DS1 and DS2 with respect to maximum number of others’ content hosted on a device. Note that both data sets provide similar availability despite the different connectivity characteristics of the data sets. This indicates that the DG algorithm has the potential to adapt to different types of mobile environments. Even for the worst case scenario, almost 100% availability can be achieved when a device is allowed to take part in only three tribes.

3. In our previous work [34], the effect of training period and evaluation period selection was analysed with a heuristic device grouping algorithm.
Average Replication of the system is the average number of replicas of the original content ($R_o$). When the limit of hosting is lower, some devices may not be able to find a tribe, which lowers the $R_o$ as shown in Fig. 4(b). When the limit of hosting is higher, devices are allowed to select a single device with high availability instead of several devices with low availability, which again lowers $R_o$. The maximum average replication required to satisfy the availability requirement of $P_a = 0.95$ is $R_o = 1.19$ with the standard deviation of 0.39.

This analysis shows that nearly 100% availability of content over low-cost networks can be achieved with approximately 2 replicas, if the devices are grouped into tribes as described in Section 5. This minimizes the overhead of replication in terms of network bandwidth, storage and battery consumption. Other systems, such as Safebook [11] or MyZone [23], do not select devices for replication based on the connectivity patterns. Instead, they use random replication, which results in a higher number of replicas than MobiTribe to provide a similar level of availability. To validate this we performed a random group selection. As shown in Fig. 4(c), random selection requires 7 replicas to provide the same availability level that is provided by MobiTribe with 1.19 replicas. Although the results are for the small scale trace data sets used, the number of replicas will be the same or even lower for large scale real environments because the probability of matching complementary connectivity patterns increase with the number of users due to the availability of variety of connectivity patterns of devices.

6.3 Scalability of MobiTribe

As summarized in Section 5.3, the device grouping process consists of three phases. We first evaluate the asymptotic worst case time complexity for each phase.

1) Hypergraph creation: Similar to the notation in Section 4, let a hypergraph $H = (V, E)$ where the vertex set $V$ represents the set of devices. The hyperedge set $E$ is the collection of subsets of $V$ satisfying the availability constraint, $P_a > P_{th}$. Fig. 4(b) shows that we can cover the majority of devices by using bipartite b-matching algorithm, i.e. the maximum allowed group size $Δ = 2$. Hence, we limit the rank of the hypergraph to 2 in the initial graph creation phase, which makes it a simple undirected graph.

In general, if a system has $n$ devices, there can be a maximum of $\frac{m(m-1)}{2}$ number of pairs that we need to calculate $P_a$ for the creation of edge set $E$. $P_a$ calculation does not require heavy computation as it only requires the calculation of the union of two binary signals and one division for each pair of users. Therefore, the upper bound for hypergraph creation would be $O(n^3)$ in general. In the data sets considered, there were $|E| = 2905$ sets out of 3528 pairs, which satisfied the $P_{th} = 0.95$ for $|V| = 84$ devices.

2) Bipartite b-matching algorithm: The scalability of the DG is dominated by the calculation of the bipartite b-matching algorithm. Bipartite b-matching is directly reduced to the maximum flow, thus the complexity is bounded by the maximum flow computation. There are many fine tuned algorithms available for maximum flow computation in the literature [15]. The bipartite flow network $(A, B, E)$ consists of $|A \cup B| = 2|V| + 2 = 2n + 2 \approx 2n$ vertices and $|E| = 2(\frac{n(n-1)}{2})$ edges. Hence, if we use Goldberg’s algorithm [15], the upper bound for the bipartite b-matching would be $O(\left|A \cup B\right|E\log \frac{|A|\cdot|B|^2}{|E|}) \approx O(m^3)$.

3) Greedy replication: We perform greedy replication selection only for the devices which are unable to group together when $Δ = 2$. Let there be $m$ remaining devices. Then, for each $m$ there are $\binom{n-1}{Δ-1}$ combinations to evaluate the threshold of pairing $(P_{th})$ and threshold of hosting $(c(u))$. Hence, the upper bound for the greedy algorithm would be $O(m\binom{n-1}{Δ-1})$. We perform greedy selection until $m = 0$ while incrementing $Δ$.

Altogether, the scalability of the device grouping would be $O(n^2) + O(n^3) + O(mn\binom{n-1}{Δ-1})$ for all three phases. For the case of $Δ \leq 3$, the asymptotic time complexity is upper bounded by $O(n^3)$ because the fastest growing component is the bipartite b-matching. For the case of $Δ > 3$, the time complexity of device grouping is dominated by greedy replication and is given by $O(mn\binom{n-1}{Δ-1})$. Since it is possible to cover majority of devices in phase two, $m$ would be considerably low compared to $n$ and drastically decrease with each increment of $Δ$. In addition, $Δ$ does not need to increase more than 4 in real-life networks. However, $m$ can be as large as $n$ in theory. Therefore, the theoretical worst case time complexity increases to $O(n^3)$. This is highly unlikely in real-life networks because of high WiFi availability as observed in this evaluation.

It has been shown that the active relationships a person can maintain is about 150 (Dunbar’s number [12]) and validated for many online social networks. Even in large systems like Facebook, the median friend size is approximately 100 [36], i.e. $n$ would be around 100. Since the device grouping is performed at the CMS, where there are no processing power and energy limitations compared to mobile devices, $O(n^3)$ and $O(n^3)$ can be considered as practical asymptotic complexities for the NP-Hard DG problem. Furthermore, it is expected that the selected tribes in MobiTribe will become quasi static over the time due to the regular behavior of people [13]. Hence, the tribe calculation will be carried out only for users who have changed their behavioral patterns such that the tribe attributes go below the required thresholds. In practice, the number of users of the service increases gradually and as a result tribes will only be calculated for newly added users. Therefore, it is evident that the device grouping algorithm will scale up with the increasing number of users.

Content popularity is another factor which could impact scalability during content sharing. As the size of friends will be limited in human social networks (Dunbar’s number [12]), the content popularity is also limited by friends. The user population of data sets used for the evaluation of MobiTribe is similar to the current number of friends of Facebook users. Moreover, only 20% of them account for 70% of interactions [38]. Therefore the content popularity among groups of friends will not be a scalability issue for MobiTribe. When the same content is popular among a number of groups, it is considered as separate content for each group. In addition, the group of friends in MobiTribe are assumed to be collaborative, i.e. once someone downloaded content she will be willing to share it with others participating in the peer-to-peer content distribution.
process (seeding). For highly popular content and for creators with a large number of friends, there will be a higher number of copies available on mobile devices. Therefore, the peer-to-peer protocol scales with demand similar to BitTorrent.

In addition, the number of different content items generated by users within a group could impact scalability because of storage capacity. Although the usage of spare storage capacity in a mobile device has very low relative cost for both the users and operators, it is obviously not comparable to the storage available via centralized services. Therefore, to avoid accumulation of content, the old replicated content will be removed from user caches using a standard LRU cache replacement scheme as described in Section 3.

7 Performance of MobiTribe Architecture

The overall performance of MobiTribe can be quantified by the cellular bandwidth and the energy consumption of devices. Therefore, the performance of MobiTribe is evaluated by comparing it with the following:

1) Central Server: where the content will be uploaded to a centralized server at the time of sharing. With a centralized server, the shared content will be immediately available to the consumers. Majority of the current content sharing mechanisms via social networking services such as Facebook, YouTube and Google+ fall under this category.

2) Mobile Server: where the content will not be uploaded to a centralized server. With a mobile server, the content consumers access the shared content via creator’s own server which is a smartphone in this evaluation. The distributed social networking architectures such as Diaspora [18] and PrPl [28] fall under this category.

3) Mobile Peer-to-Peer: where all content consumers will also host the downloaded content to help others to access the content (seeding). Many services which make use of protocols like BitTorrent fall under this category.

For this evaluation, we assume that both content creators and consumers are mobile devices, and the mobile devices are always-on and always-accessible via the cellular network. In addition, we consider that on-the-spot data traffic offloading via WiFi networks is possible in all architectures. Since all of the above architectures support immediate content availability, we do not consider delayed offloading to a centralized server. Note that even in MobiTribe, although the content pre-fetching takes time, the shared content is immediately available via the creator’s device. Therefore, delayed offloading is not directly comparable with MobiTribe. On the other hand, the high replication overheads of the non-mobile-optimized replicating systems, such as SafeBook [11] and MyZone [23] would almost certainly lead to higher bandwidth and energy consumption. This is due to the large number of replicas that would be needed to provide consistent availability of the content via low-cost networks, when using a replication algorithm which is not aware of network availability patterns. Hence, we do not consider those architectures for this evaluation.

7.1 Content Creation and Consumption Model

Table 1 summarizes the content creation and consumption model parameters. To analyse the impact of UGC sharing in the future, we use the projected mobile network data traffic figures of Cisco [9]. It is predicted that an average smartphone will consume 2.6GB of data per month by 2016. Since the uplink (UL) traffic percentage is around the 23-25% mark [8], we assume that the average smartphone user generates 612MB (23% of 2.6GB) per month and the generated content are Gamma distributed [2] with the mean content size of 2MB. It has been shown that the internet access time of mobile users of DS2 are exponentially distributed in [21]. Therefore, the inter-arrival-time for the content generation is obtained from an exponential distribution such that it spreads throughout the whole trace duration.

It has been observed that 10% of the top popular videos in YouTube account for 80% of views [7]. Hence, we use the Pareto type II distribution for content popularity such that each generated content is accessed by a number of randomly selected consumers from a Pareto distribution. Then, these selected consumers will access the content after a content access delay which again has a Gamma distribution. We evaluate the performance by varying the mode value of the Gamma distribution, namely “Peak Content Access Delay” (CADpeak) to understand the performance under different content access delay conditions. We evaluate the performance against the true positive rate (TPR) of prediction which is the portion of correctly predicted consumers out of the actual consumers and we assume the TPR=0.7.

We simulate a content sharing network among the mobile users in DS2. We consider WiFi data rate as 1.97Mbps, which is the mean data rate observed in the experiment [21] and 3G data rate as 200Kbps, which is the same data rate used in the simulation for this environment in [21]. We use this as the communication model for this simulation along with the workload in Table 1 to evaluate the effectiveness of each architecture.

7.2 Cellular Bandwidth Consumption

If the creator is not able to pre-distribute the content, it is served from the creator’s device immediately after its creation either through a cellular or a WiFi network. MobiTribe was designed on the premise that there will be a time lag between content creation and consumption (content access delay), and therefore it will be possible to take advantage of the availability of low-cost networks. Note that there can be many instances that the content is accessed right after sharing, even if the CADpeak=1 day, because CADpeak is only...
Fig. 5. Cellular UL and DL bandwidth usage. (a) UL usage vs peak content access delay and (b) DL usage vs peak content access delay.

Fig. 6. Bandwidth saving compared to central server.

Fig. 7. Cellular network usage of devices compared to central server. CAD_{peak}=1 day.

In Fig. 5, alternative architectures do not show a significant change in uplink or downlink cellular usage with increasing content access delay. In contrast, MobiTribe provides considerable reduction of cellular network usage when there is higher content access delay. This is because a large content access delay allows content creators to pre-distribute the content to the mobile devices via low-cost networks. The mobile server architectures consume the highest uplink bandwidth as shown in Fig. 5(a), which is higher than the generated content by approximately 1.5 times. This is because of the creator serving all content requests. Downlink bandwidth usage of alternative architectures are similar to this as shown in Fig. 5(b). This is because none of them perform predictive pre-fetching and thus entirely depend on the connectivity pattern of the downloading device. Content access prediction in MobiTribe leads to significant downlink bandwidth saving due to successful pre-distribution of content to the consumers via low-cost networks. The mobile server architectures consume the highest uplink bandwidth as shown in Fig. 5(a), which is higher than the generated content by approximately 1.5 times. This is because of the creator serving all content requests. Downlink bandwidth usage of alternative architectures are similar to this as shown in Fig. 5(b). This is because none of them perform predictive pre-fetching and thus entirely depend on the connectivity pattern of the downloading device. Content access prediction in MobiTribe leads to significant downlink bandwidth saving due to successful pre-distribution of content to the consumers via low-cost networks. The mobile server architectures consume the highest uplink bandwidth as shown in Fig. 5(a), which is higher than the generated content by approximately 1.5 times. This is because of the creator serving all content requests. Downlink bandwidth usage of alternative architectures are similar to this as shown in Fig. 5(b). This is because none of them perform predictive pre-fetching and thus entirely depend on the connectivity pattern of the downloading device. Content access prediction in MobiTribe leads to significant downlink bandwidth saving due to successful pre-distribution of content to the consumers via low-cost networks.

76% of the uplink cellular bandwidth compared to a central server architecture, if the CAD_{peak}=1 day. Though the percentage saving for the downlink is 46%, the amount of bandwidth saved by the downlink is higher than the uplink because of the larger downlink data volume. In total, MobiTribe saves 54% of the cellular bandwidth compared to a centralized server architecture with on-the-spot data traffic offloading, when the peak content access delay is one day.

Fig. 7 illustrates the cellular bandwidth consumption of each device in MobiTribe normalized by cellular bandwidth consumption of a central server. In this particular social network, 90% of the devices consume only 60% of the cellular bandwidth compared to the central server, i.e. 90% of these devices can save at least 40% of the cost of cellular communications. On the other hand, in both mobile server and mobile peer-to-peer architectures there are devices that will see an increase in cellular bandwidth consumption. In systems like Diaspora, 85% of the devices will consume more cellular bandwidth compared to the central server architecture, which affects both monetary cost and device energy consumption. Thus, MobiTribe satisfies its key design goal of cellular bandwidth saving. In the following sections, we evaluate how the cellular bandwidth saving of MobiTribe varies with algorithmic and content consumption model parameters.

7.2.1 Effect of the Content Access Prediction

Since replication of content is done via low-cost networks, the false predictions do not affect cellular bandwidth consumption. Hence, we evaluate the performance against the true positive rate (TPR) of content access prediction. Fig. 8(a) shows that cellular uplink bandwidth saving is not heavily dependent on TPR, whereas bandwidth saving significantly increases with the content access delay. In contrast, downlink shows significant saving only when both the TPR and the content access delay are higher as shown in Fig. 8(b). However, the TPR has more impact on downlink saving than the content access delay.

Since the content is to be pre-distributed to devices of users who are also likely consumers, users can access the content instantly. Fig. 8(c) depicts the variation of cache hit ratio with respect to the two main limiting factors - TPR and content access delay. According to the Fig. 8(c), the TPR significantly improves the cache hit ratio, whereas the impact of content access delay is relatively low. When the TPR is
zero, the cache hit ratio is as low as 2% and increases up to 78% for completely accurate prediction. Note that even with the 15min content access delay, 31% of users are able to access the content instantly for a TPR of 70%. Overall, a high average cache hit ratio, higher than 50%, can be achieved by MobiTribe, which can be considered as an extra benefit to the users on top of the cost savings.

7.2.2 Effect of the Content Popularity Model

The worst possible content popularity model in terms of bandwidth consumption is the one where everyone in the group is accessing all generated content. In Fig. 9(a) and (b), cellular bandwidth consumption for the worst case content popularity model and the Pareto model is presented. The uplink shows significant increase in bandwidth usage for the worst case model, downlink follows similar usage for both models. In contrast, the cellular bandwidth usage of centralized server architectures do not change with the content popularity model since it uploads the content to a central server regardless of the demand. Fig. 9(c) depicts the cellular bandwidth saving of MobiTribe for the worst case content popularity normalized by a centralized server architecture. The total bandwidth saving does not drop significantly for the worst case. When the CADpeak is 1 day, the cellular bandwidth saving for the worst case is 47% compared to the 54% for a realistic Pareto distribution. Thus, in this particular social environment, MobiTribe will be able to save approximately 50% of the cellular bandwidth irrespective of the content popularity distribution. Since the data trace contains 1470 days of connectivity patterns from 105 users in three different cites in three different times, we believe that the performance of MobiTribe improves independent of the social environment.

7.3 Energy Consumption

Content replication incurs more data transfers among devices than a centralized architecture. However, this overhead traffic is mainly transferred through low-cost networks in MobiTribe. Hence, the energy gain in MobiTribe depends on the energy efficiency of WiFi over cellular network. It has been shown that the energy consumed by WiFi, $(E_w)$ is significantly lower than those of 3G networks $(E_{3g})$ [4], [26]. This is mainly due to the heavy traffic load in cellular networks leading to lower speeds and thus taking longer time to complete data transfers. It has been shown that $E_{3g}/E_w$ can vary up to 100 depending on status of the cellular network at the time of the transfer [26]. To overcome these, the analysis assumes that the energy consumption is proportional to the time spent on each network and evaluates the performance with respect to the energy ratio $E_{3g}/E_w$.

Fig. 10 shows the energy consumption of each architecture, normalized by the energy consumption of on-the-spot offloading to a centralized server. The mobile server architecture always consumes approximately 1.5 times more energy than a centralized server. The mobile peer-to-peer architecture performs slightly better when $E_{3g}/E_w > 20$. In
contrast, normalized energy consumption of MobiTribe has an exponential decay and has a 50% lower consumption than a central server architecture for higher $E_3g/E_w$.

Even though MobiTribe consumes almost the same energy as a centralized server architecture when $E_3g/E_w = 20$, approximately 70% of devices still consume lower energy than a centralized server architecture as shown in Fig. 11(a). Less than 10% of devices consume more than 1.5 times energy. Though the mobile peer-to-peer architecture has a similar energy consumption pattern as a central server architecture, 80% of the devices consume more energy than the MobiTribe when $E_3g/E_w = 20$. In the mobile server architecture, 80% of the devices consume more energy than the centralized server architecture. Thus, it is clear that mobile server architectures such as Diaspora and fully distributed peer-to-peer architectures are not suitable for mobile environments. In summary, MobiTribe has an energy efficiency similar to the centralized server architectures. However, it is more efficient than all alternative distributed content sharing architectures.

### 7.3.1 Effect of the Content Popularity Model

The worst case content popularity model turns out to be more energy efficient than the realistic Pareto model as shown in Fig. 11(b). In the worst case, when everyone in the group is accessing all the generated content, the false positive rate (FPR) becomes zero, which is the portion of incorrectly predicted consumers out of the actual consumers. Hence, there will not be any redundant data transfers or any replication overhead. The energy consumption becomes steady after $E_3g/E_w > 20$. Thus, the content popularity distribution stands out as the controlling factor for the trade-off between cellular bandwidth and energy consumption.

### 7.3.2 Effect of the False Positive Rate of Prediction

False positives in content access prediction creates redundant data transfers and results in extra energy consumption. Fig. 11(c) shows the variation of energy consumption of MobiTribe with the FPR of the prediction algorithm, which could be as much as 2.1 times higher than the centralized server when $E_3g/E_w = 20$. Hence, the FPR can be considered as a critical factor for energy consumption. However, a higher FPR increases the availability of the content via low-cost networks. Also, amount of false positives can be reduced at the cost of increasing false negatives.

It is possible to extend MobiTribe to pre-fetch the content when devices are connected to a power source, which will reduce the impact of replication energy. In fact, mobile device charging analysis presented in [14] illustrates that the majority of users charge devices during the night. Also, the majority of the users have WiFi access during the night because of the availability of home WLANs.

## 8 Conclusion

Smartphones could be used for UGC sharing and host/share distributed social network data, if the challenges of high communication costs and battery usage were solved. We proposed a novel mechanism for addressing these challenges by providing continuous availability of content over low-cost network connections. We showed the content replication problem in distributed peer-to-peer architectures such as MobiTribe is NP-Hard. We then developed a new algorithm which minimizes content replication and maximizes content availability whilst ensuring fairness. The performance of the algorithm was evaluated using three real world data sets and realistic content creation and consumption models. Even with these small scale data sets, it was shown that persistent availability of content via low-cost network can be achieved with approximately two replicas, which will become lower with larger variety in real scale environments. In addition, the results from 97 iPhones with realistic content creation and consumption modeling showed that MobiTribe saves 76% of the uplink and 46% of the downlink cellular bandwidth compared to current widely used centralized server based content sharing architectures. Moreover, we showed that device energy consumption decays exponentially with energy consumption ratio between low-cost networks and cellular networks. Therefore, we conclude that MobiTribe can provide a mobile optimized distributed content sharing platform for mobile social networking.

## References


Kanchana Thilakarathna is a PhD candidate at the School of Electrical Engineering and Telecommunications, University of New South Wales and a Graduate Researcher at NICTA. He worked as a mobile network optimization Engineer for three years, before pursuing postgraduate studies. He has also worked as a research intern at INRIA, France for six months in 2012. His current research interests are content distribution and privacy preservation in mobile social networks.

Henrik Petander received his MSc in Electrical Engineering from Helsinki University of Technology in 2002 and his PhD degree in Computer Science from the same university in 2007. He has worked as a researcher at NICTA since 2005, and as a cofounder at Incoming, a mobile video company, since 2013. His research interests include mobile content delivery and networking in a heterogeneous mobile environment.

Julian Mestre received his PhD degree in Computer Science from the University of Maryland in 2007. After graduating, he worked for three years at Max Planck Institute for Informatics in Saarbruecken, Germany. Since 2010, he works at the University of Sydney where he is a Lecturer in the School of Information Technologies. His area of research is Theoretical Computer Science.

Aruna Seneviratne is the foundation Chair in Telecommunications and holds the Mahakanorn Chair of Telecommunications, and the leader of the Networks Research Group at NICTA. His current research interests are in mobile content distributions and preservation of privacy. He received his PhD degree in electrical engineering from the University of Bath, U.K. He has held academic appointments at the University of Bradford, U.K., Curtin University, and UTAS.