Krowd: A Key-Value Store for Crowded Venues

Utsav Drolia, Nathan Mickulicz, Rajeev Gandhi, Priya Narasimhan
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, USA
{udrolia,nmickuli}@andrew.cmu.edu, {rgandhi,priyan}@ece.cmu.edu

ABSTRACT

Attendees of live events want to capture and share rich content using their mobile devices, during the events. However, the infrastructure at venues that host live events provide poor, low-bandwidth connectivity. Instead of relying on infrastructure provided by the venue, we propose to stand up a temporary "infrastructure" using the very devices that need it, to enable content-sharing with nearby devices. To this end, we developed Krowd, a novel system that provides a key-value store abstraction to applications that share content among local, nearby users. We evaluated Krowd using over 200 hours of real-world traces from sold-out NBA and NHL playoffs and show that it is 50% faster and consumes 50% less bandwidth than alternative systems. We believe that Krowd is the only decentralized and distributed system to provide a key-value store made for neighboring mobile devices and of neighboring mobile devices.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Distributed Systems; H.3 [**Information Storage And Retrieval**]: Miscellaneous

Keywords

cooperative; mobile; key-value; crowd; localized

1. INTRODUCTION

It's common to hear of tens of thousands of users at live events (concerts, games, commencement) producing and consuming significant and increasing quantities of data using their mobile devices. Simultaneously, the number of mobile devices at these live events is also growing. As captured in Figure 1, there is a trend for *an ever-increasing number of users at live events to expect high-bandwidth connectivity and the ability to share content via their mobile devices.* Let's look at data from the NFL Super Bowl, an annual high-profile event:

MobiArch'15, September 7, 2015, Paris, France.
© 2015 ACM. ISBN 978-1-4503-3695-6/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2795381.2795388.

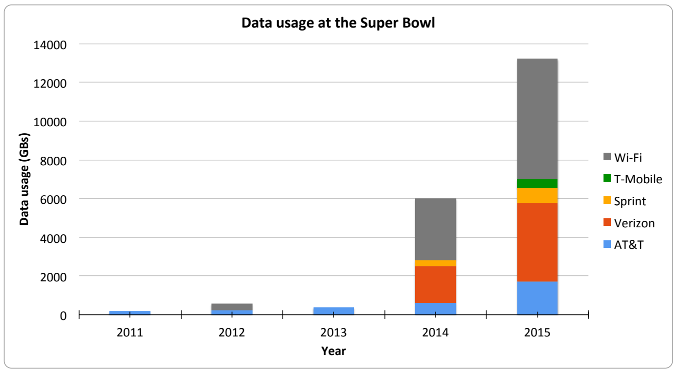


Figure 1: Increasing in-stadium Wi-Fi and cellular data usage over mobile devices at the NFL Super Bowl.

- At the 2015 Super Bowl [4], over 6.23TB of data was transferred over the in-stadium Wi-Fi network, with social networks consuming 10% of all bandwidth.
- At the 2015 Super Bowl [3], Verizon reported 4.11 TB of cellular data transferred, including 122,000 videos, while AT&T logged 1.7TB. Sprint's cellular network transferred over 750GB of data with 25% for social networks, 15% for sharing content, and 5% for streaming videos.
- At the 2014 Super Bowl, over 5 photos/sec were uploaded inside the stadium via a single image-sharing application, over 90,000 photos in all.
- The number of in-stadium devices has skyrocketed, from 8000 at Super Bowl 2012 to 68,500 at Super Bowl 2014 [12].

Another trend is the increased focus on improving the attendees' in-venue experience. For example, stadiums and sports teams now provide multiple live camera angles and on-demand replays viewable on fans' smartphones [2]. Indeed, by extension, one could think of every attendee's perspective as a unique "camera angle," with the associated content (of, say, a replay captured by a fan's smartphone camera) being of value to other attendees in the venue. Hence, there can be a lot of content created in the venue, of value to other people inside the venue, and this trend is likely to increase.

The challenges in supporting this trend lie with the current infrastructure. For example, only 35% of major stadiums have in-venue Wi-Fi networks [7], provisioned largely for coverage, not capacity, which means that, while every seat is covered, the networks cannot handle a large density of attendees. Hence, at most venues, attendees will not be able to share content at their desired rates or on their desired channels. Installing or upgrading in-venue wireless networks can be cost-prohibitive, e.g., it would cost some organizations "$2 million each to improve stadium cell phone and Wi-Fi connectivity." [12] Clearly, in-venue wireless infrastructure, in its current form, will not be sufficient to handle the (content and user) demands that will likely be placed on it.

However, *what if more attendees at a live event actually meant better infrastructure?* What if instead of straining the venue-provided infrastructure, each device augmented it? Could we stand up a temporary infrastructure built on the very devices that need it? Could we provide attendees with the capability to share their in-venue-generated content with other attendees, without relying on the venue-provided infrastructure?

To enable this we present Krowd, a temporary "infrastructure" supported by the attendees' devices themselves. We propose to use the devices' computation, communication and storage capabilities to power this "infrastructure". These devices have communication capabilities which allow them to talk to other nearby devices without going over the Internet, e.g. through access points (AP). Krowd leverages this and handles the communication and coordination between nearby mobile devices to provide an easy-to-use key-value store abstraction for applications.This paper highlights how it can be used for content distribution. Krowd is made for neighboring mobile devices and of neighboring mobile devices. In this paper, we evaluate Krowd using multiple real-world traces from attendees at popular NHL and NBA games and show that it lowers the latency and bandwidth consumption by approximately 50% as compared to alternative systems.

Contributions. This paper makes three contributions:

- A key-based addressing technique for clusters of neighboring mobile devices.
- A key-value store abstraction using the addressing technique to enable coordinated content sharing.
- Insights from experiments based on over 200 hours of real device-activity traces from NHL and NBA games.

The next section sheds light on the requirements and design of Krowd. The components of Krowd are described in Section 4 and it is evaluated in Section 5 through traces obtained from popular sporting events. Section 6 gives an overview of the related work. We conclude in Section 8.

2. PROBLEM STATEMENT

The primary focus of Krowd is to *enable mobile-device users to easily share content with and retrieve content from other neighboring mobile-device users in a resource-efficient manner.*

2.1 Requirements

A content sharing service for mobile devices has the following high-level requirements:

Descriptive Data. The system should allow user applications to search based on meta-data, e.g. tags.

Efficiency. Given that these devices are mobile and battery-powered, efficient use of resources is necessary.

Responsiveness. Since these services are user-facing, they need to have low end-to-end latency.

3. DESIGN OVERVIEW

The basic goal of Krowd is to provide a distributed key-value store for content sharing, discovery and retrieval. We do this through the use of network-based discovery to find nearby devices and consistent hashing to guide the share-content and retrieve-content requests made on the key-store to the correct nearby device. The rest of the section describes the design choices made for Krowd to meet the requirements mentioned in Section 2.1.

Guiding principles for the overall design were decentralization, to avoid any one device becoming "more equal" than other devices, and autonomy, since there can be no system administrators for such a system.

To enable storing and searching data based on attributes, Krowd provides a key-value store abstraction for the application. The application can use attributes of the content as keys and the URI (Universal Resource Identifier) of the content as the value when storing and retrieving from the key-value store. If other users also store values with the same key, these values get appended into lists. When fetching, the application would need to provide an attribute for the content it is looking for and would receive the consolidated list of content URIs. Note, the key-value store only stores meta-data, not the actual content.

We made multiple design choices to ensure efficiency in Krowd. To keep the workload equal amongst participating devices, the key-value store needs to be distributed and decentralized. A consistent hashing scheme is used to partition and distribute the keys of the key-value store. This ensures each device is responsible for a unique subset of the keys and no single device is overloaded. The hashing scheme used by Krowd is *rendezvous hashing*, also known as highest random weight hashing [13]. This hashing scheme, along with network device discovery, ensures that Krowd is decentralized yet coordinated, and the workload is distributed. Since the devices use wireless communications, which is energy-intensive, minimizing communication is essential for energy efficiency. The use of rendezvous hashing helps achieve this requirement and we delve into this further in Section 4.

By using rendezvous hashing, Krowd also ensures low end-to-end latency. All lookups in this scheme are resolved in one hop, hence key retrieval is fast. As there is no single "master" device, lookups and stores are spread across all devices and no node becomes a bottleneck.

Krowd chooses to not handle churn implicitly. This is because handling churn implicitly, i.e. without knowing the application's intent, consumes bandwidth, e.g. DHTs handle churn implicitly by replicating data. Moreover, most of the attendees at an event largely stay in their places during the event. If the application needs to tolerate churn, it can simply re-issue its key-value pairs periodically.

3.1 Background: Distributed Hash Tables

Distributed hash tables (DHTs) [10, 9] are a class of structured peer-to-peer systems. They use key-based routing and consistent hashing to provide a lookup service similar to a hash table, but where the (key, value) pairs are distributed across participating nodes. They were designed to function on peers (desktops) distributed across the Internet, for cooperative Web caching, distributed file systems, domain name services, instant messaging, multicast, and also peer-to-peer file sharing. On the surface, Krowd and DHTs look similar and in fact, Krowd draws inspiration from DHTs. Both provide a key-value based lookup service, route based on keys, use some form of hashing, and distribute the workload across all participating nodes. The key design choices were made for DHTs to work across the wired Internet. However, the goal for Krowd is to work across a *local, wireless* network, and this is the key difference. Section 5 compares Krowd with Kademlia and provides key insights that help understand why DHTs are not suitable for local, wireless networks.

3.2 Assumptions

Single-hop communication. Krowd relies upon 1-hop ad-hoc networks provided by wireless access points or Wi-Fi Direct.

Broadcast support. We assume that the communication technology, i.e. access points, Wi-Fi Direct etc. supports the broadcast channel and mobile devices are allowed to use it.

Session ID. Krowd assumes that for a given session, i.e. the duration of the event, the IP address of a device does not change. In the future, this can be easily overcome by using universal IDs instead of IP addresses to identify the devices.

4. Krowd'S APPROACH

As can be seen in Figure 2, the system is divided into three main components: Discovery and Communication, KRoute, and KVStore; the application on top interacts only with KVStore. An instance of Krowd and the application runs on each participating device in the group. For this implementation we consider every mobile device connected to the same access point as part of the group (the number of nodes connected to an access point varies from 20 to 50, depending on the manufacturer), as seen in Figure 2. This paper focuses on the creation and management of a single group.

Discovery and Communications. This module is responsible for announcing a device's presence, discovering other devices in the vicinity, and communicating with them. To alert other devices about its presence, this module sends UDP beacons to the access point's broadcast address once every five seconds. This beacon also contains the network endpoint that the device is listening on. On receiving a beacon, this module opens a connection with the beacon's source. Since these beacons are periodic, they are used for detecting device departures as well. If a beacon (or any other data) is not received from a discovered device for T seconds, where T is configurable, this module tries to ping the device. If this is unsuccessful, the device is considered to have left the vicinity. Therefore at steady state each device has a list of all other devices in the group, i.e. connected to the same access point. It might seem like this discovery scheme uses a large share of precious bandwidth. However, by keeping the beacons small (13 bytes on average) and using the broadcast mechanism, the bandwidth consumed is approximately 255 bytes/s per device, for a group of 20 devices and T=5s, which amounts to less than 0.01% of a nominal 54MB/s Wi-Fi network. By using this discovery mechanism and tracking devices' presence, this module provides a set of nearby devices and an interface to communicate with them. It preserves symmetry and avoids having a centralized registry for storing membership and device presence information.

KRoute: Partitioning and routing. This is the key-based routing module. Given a key and data, it sends the data to the correct device responsible for that key. It uses *highest random weight* hashing [13] to accomplish this. Given a key and a list of known servers, this scheme consistently maps the key to a unique server from the known servers. In [13], this scheme was used by clients trying to access resources over the Internet. If all the clients had the same list of servers, they would all always fetch a resource at a given URL from the same server. Consequently, KRoute uses the set of available nearby devices provided by the discovery module, and consistently maps a key to a device. Given that the discovery layer on each device discovers every other device in the group, this hashing scheme is consistent and KRoute will always map a key to the same device. KRoute presents itself to other components as a high-level communication layer, where instead of requiring endpoints to communicate with, it requires keys. Hence, if two different devices provide the same key to their respective KRoute modules, they end up communicating with the same device. This is essential for efficient coordination. It obviates the need for any dynamic agreement protocol when multiple devices want to talk to the same device. Neither does it need multiple hops to find the correct device. This was key because multiple hops would cause congestion in a local dense wireless network.

KVStore: Key-value interface. Armed with the key-based communication provided by KRoute, this module builds a thin remote procedure call (RPC) layer on top of it. It also contains the device-local hash table for storage of keys and values. The RPC layer implements the two procedures required for a key-value store, namely *put(key, value)* and *get(key)*. When the application stores a key-value pair, KVStore creates and serializes a *put* RPC object and sends it using KRoute by providing the key from the key-value pair. On receiving a remote *put* call, KVStore extracts the key-value pair from the serialized RPC and stores it in its local hash table. When the application requests a key, KVStore provides the *get* RPC object and the requested key to KRoute, which sends it to the correct remote device. On the remote device, when KVStore receives the RPC, it extracts the key, checks its local hash table and replies with the associated value. By providing a simple key-value interface, KVStore makes Krowd easy to use for applications, and specifically makes sharing and searching content through descriptive tags easy.

Sharing content using Krowd. Lets look at an example scenario to see how an application would use Krowd to enable content-sharing. Suppose user A wants to share a video taken during a game and tags it with "dunk". The application calls the *put("dunk", URI)* method in KVStore, where URI is the URI of the video. KVStore creates a RPC object and submits it to KRoute, providing "dunk" as the key. KRoute then maps this key to a device in the vicinity, e.g. device X, and sends the RPC object to it. On device X, the respective key and value is stored. Now if another nearby user B searches for content tagged as "dunk", the application calls the *get("dunk")* method of KVStore, which submits the respective RPC object to KRoute with the key as "dunk". KRoute again maps this key to device X and sends the RPC

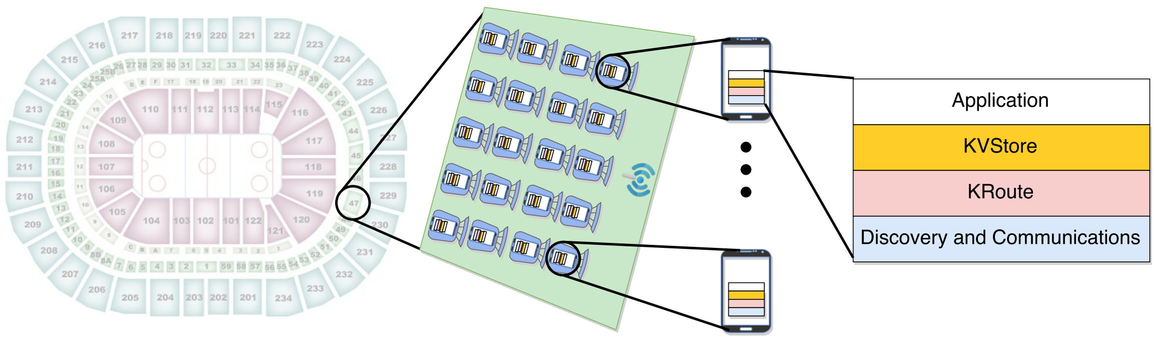


Figure 2: Fans with devices in a Wi-Fi enabled section of a stadium. Each device runs the Krowd architecture

object to it. On device X, the URI associated with this key is sent back to user B's device. The application then uses the URI to contact user A's device and get the requested video.

Thus, by using device discovery along with key-based communication, Krowd enables decentralized yet efficient coordination and the RPC layer provides an easy-to-use key-value store for content sharing and discovery.

5. EVALUATION

Goal. The goal of the experiments was to evaluate and compare how Krowd and Kademlia [9], a popular DHT, function in groups of wireless mobile devices. The metrics for comparison were end-to-end latency for getting the value of a specified key and the overall bandwidth used, i.e. inclusive of actual fetching/storing and maintenance. End-to-end latency measures the responsiveness of the system, while bandwidth consumption measures efficiency, e.g. higher bandwidth consumption would lead to faster discharge of batteries in mobile devices.
Real-world Traces. Krowd is meant for scenarios when users come together at popular places and events. To simulate such a scenario, evaluation was done based on real-world traces collected from attendees' mobile devices at popular, sold-out NHL and NBA games. The traces contain sequences of timestamped logs of all attendees' requests to view content on their devices during the entire live event. Each request log contains the time of request, user ID and content requested. We use these traces to represent how attendees would use their devices to share and view content at live events. Each request made by a user in the trace is treated as a request to retrieve content from nearby devices, and hence as a get request on the key-value store. For the experiments, the request logs for each unique user ID was replayed by a unique device according to the timestamps. We used traces from over 50 games to evaluate Krowd.

5.1 Experiments

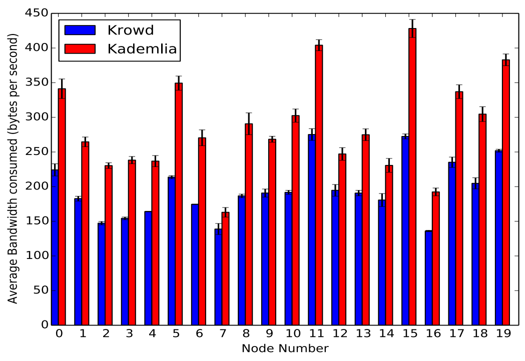
Setup. Our experiments were setup to emulate a section of an NHL stadium where the section seats 20 attendees in it. Such a section is generally served by one access point in a configuration similar to the one shown in Figure 2. Each user's mobile device was emulated by a virtual machine configured with 1 CPU and 2GB RAM. These 20 virtual mobile devices were connected to a simulated access point, over a simulated wireless-medium, using NS3 [5] to reconstruct the scenario shown in Figure 2.
Procedure. To use our real-world traces in this setup, some filtering was performed. For each trace, 20 of the most active users were selected and their logs were partitioned to form 20 separate user-traces, each containing the logs of a unique user. Each of these user-traces was assigned to a different virtual mobile device. During the experiment, the requests made by a virtual mobile device was in accordance with its assigned user-trace. The procedure followed by each device for each experiment:
1. Discover all devices and stabilize
2. Store keys from the user-trace in key-value store
3. Issue get requests as per its user-trace

Note, the metrics are measured during the get-requests phase only since put requests are asynchronous and do not contribute to user-perceived latencies. Also, only the value of the key is fetched from the key-value store, i.e. the URI of the content, not the content itself because fetching the content will cause the same bandwidth consumption and latency on both systems.

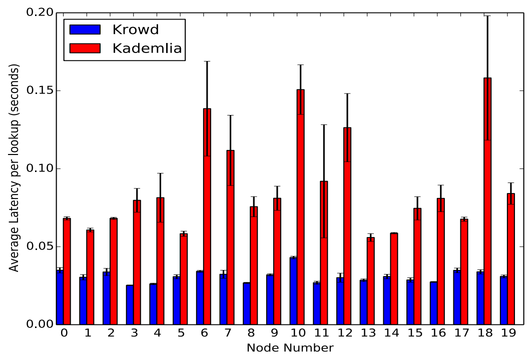
5.2 Modifying Kademlia

Bootstrapping. Kademlia, and any DHT in general, starts up using at least one "well-known" node which every new node contacts to join the DHT. Having such a node is infeasible in the scenarios that Krowd is meant for. Hence, the same discovery layer as Krowd is used in Kademlia for the experiments. The discovered nodes are treated as neighbour nodes and the normal Kademlia protocol for joining takes over.
Bucket size. Kademlia has a tunable parameter, k, which controls the size of each bucket. A bucket is a list of neighboring nodes at an equal "distance" from the given node. Each node contains multiple buckets, each for a fixed distance from this node[1]. To make a fair comparison we set k to 20. This way every node in the network knows every other node in the network and hence effectively it would not need multiple hops to reach a specific node.

[1]For further details about the distance metric and how it is established, please see [9]



(a) Bandwidth consumption



(b) End-to-end latency

Figure 3: **Comparing Krowd and Kademlia.**

5.3 Insights

The results seen in Figure 3 are from the trace of the game that saw the most requests among all games. Results from other traces are similar and are omitted for brevity.

Bandwidth Consumption. In Figure 3a we see that for each node, Krowd consumes lesser bandwidth than Kademlia. The bandwidth measured is the overall bandwidth, i.e. including bandwidth consumed for maintenance during the experiment. Kademlia does parallel lookups when processing a get request as explained in [9]. It was designed to do so to accelerate finding the correct node which held the requested key. This leads to a higher bandwidth consumption. Moreover, to accelerate future lookups for a key, when a node completes a get request successfully, it stores the key-value pair on a known neighbor closest to the key. This further increases bandwidth consumption. The multi-hop nature of DHTs would contribute to higher bandwidth consumption as well, but as mentioned above, Kademlia is configured to always resolve a lookup in a single hop. On the other hand, Krowd was designed for a local, wireless network, and to resolve each lookup in a single-hop. Hence it does not need to parallelize lookups nor accelerate them any further.

Latency. In Figure 3b we see that for each node, Krowd has lower end-to-end latency than Kademlia. The latency is measured as the time taken starting from issuing a get request to when a value is returned to the application. The graph shows average latency per request. Note, Krowd not only is much faster but is also consistent across all nodes, approximately taking the same time per request on each node, compared to Kademlia. This is because of multiple reasons. Firstly, as mentioned before, Kademlia employs heuristics for accelerating lookups such as parallel lookups and storing fetched key-values in nearby nodes for faster subsequent lookups. On a local, wireless medium these heuristics cause harm - every additional lookup or store blocks other requests/lookups. A second reason is the periodic pings used by Kademlia for liveness. This again causes interference with other operations on the wireless medium. Krowd, instead, cleverly uses the broadcast property of the wireless medium. Moreover, this mechanism has the by-product of being used as a presence protocol as well. Krowd does not need to ping each node separately to check if it is still available.

DHTs were designed for the Internet and wide-area wired networks and a number of heuristics were used to lower latency of requests across the Internet. However in a local, wireless network, these heuristics are ineffective and in fact degrade performance.

Since Krowd is designed specifically for a local, wireless network from the ground up, we make sure there is no duplication of effort. Krowd uses the network's properties and is thus faster and more efficient than systems designed for the Internet and wired networks.

6. RELATED WORK

6.1 Mobile content dissemination and search

[8] presents a system for locating content in the vicinity based on attributes, similar to Krowd. However, in this work, each device broadcasts its query tags/attributes to all devices in the vicinity. There is no coordination and hence every query, from every device, needs to be sent to every other device in the vicinity.

[14] proposes MobiTribe - a system for disseminating content, generated on mobile devices, across the Internet. It requires a central server for content discovery, peer registration and meta-data management. Krowd is completely decentralized and targets groups of nearby devices.

[11, 6] present systems for content search in mobile devices. Both consider registered smartphones as distributed databases and allow a third party to compose queries on a central server and push them onto these smartphones to find out photos that match the query. We would like to enable similar content-based search on Krowd but without the need of a central server.

6.2 DHTs on mobile ad-hoc networks

There has been considerable work on using DHTs on mobile, ad-hoc networks as surveyed in [1]. Such systems try to use DHT routing protocols at the network level. This is entirely different from Krowd, which builds on top of a one hop network, uses key-based routing for coordination and provides a key-value store abstraction to applications.

7. FUTURE WORK

One of the first next steps will be accommodating churn, i.e. users joining and departing while the application is active. Although this can be done easily at the application level by republishing its keys periodically, we would like to design a more efficient solution.

This paper highlights how Krowd can be used for proximal content-sharing. However, Krowd can be used for multiple other applications as well. Any application that requires efficient decentralized cooperation among proximal mobile devices can make use of Krowd. For example, we envision it being used for topic-based localized chat-rooms.

By allowing coordination and cooperation among neighboring mobile devices, Krowd takes a step towards building "edge-clouds" of mobile devices. To further scale-up, we are looking into how to connect individual groups formed by Krowd by exploiting the fact that access-points at venues may be interconnected to form a venue-specific intra-network.

We are building a prototype compatible with smartphones and will soon test it in real venues such as stadiums and arenas.

8. CONCLUSION

Numerous attendees want to share content that they create while attending the event. Venues have tried enable this by adding infrastructure for high-bandwidth connectivity. However, the demand for connectivity and bandwidth, and the increase in density of devices will likely overwhelm venue-provided infrastructure. Instead, we proposed Krowd, a system that uses all the nearby devices' computation, communication and storage capabilities to provide an easy-to-use, yet powerful key-value store abstraction for applications that share content with other nearby devices. The novel key-based communication mechanism coupled with network service discovery show how Krowd is designed for local, wireless networks and the RPC layer around these provides an easy-to-use interface. We evaluated the system using over 200 hours of real-world traces obtained from attendees' devices at NHL and NBA games to show the efficacy of these design choices. We believe that Krowd is a step in the right direction to build coordinated wireless clouds of neighboring mobile-devices.

9. REFERENCES

[1] Abid, S. A., Othman, M., and Shah, N. A survey on dht-based routing for large-scale mobile ad hoc networks. *ACM Computing Surveys (CSUR)* (2014).

[2] Cisco. Connecting Fans in New Ways to Deliver the Ultimate Fan Experience.

[3] Dano, M. Super Bowl traffic stats. `http://goo.gl/uzMD2B`.

[4] Extreme Marketing Team. In-Stadium Wi-Fi Analytics Reveal Fan Engagement At Super Bowl XLIX. `http://goo.gl/piMmtQ`.

[5] Henderson, Thomas R et al. Network simulations with the ns-3 simulator. *SIGCOMM demonstration* (2008).

[6] Jiang et al. Mediascope: selective on-demand media retrieval from mobile devices. In *ACM International Conference on Information Processing in Sensor Networks* (2013).

[7] Kapustka, P., and Stoffel, C. State of the Stadium Technology Survey. Tech. rep., 2014.

[8] Kooh, T. G. C., Lv, Q., and Mishra, S. Attribute based content sharing in mobile adhoc networks of smartphones over wifi. In *Computer Communications and Networks, IEEE International Conference on* (2012).

[9] Maymounkov, P., and Mazieres, D. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems*. Springer, 2002.

[10] Rowstron, A., and Druschel, P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware* (2001).

[11] Sani et al. Opportunistic content search of smartphone photos. *arXiv preprint arXiv:1106.5568* (2011).

[12] Steinbach, P. Wi-Fi Service Increasingly Seen As a Must-Have Stadium Amenity. `http://goo.gl/LCGCDv`, July 2013.

[13] Thaler, D. G., and Ravishankar, C. V. Using name-based mappings to increase hit rates. *IEEE/ACM Transactions on Networking* (1998).

[14] Thilakarathna, K., Petander, H., and Seneviratne, A. Performance of content replication in mobitribe: A distributed architecture for mobile ugc sharing. In *Local Computer Networks, IEEE Conference on* (2011).