# Crossroads: A Framework for Developing Proximity-based Social Interactions

Chieh-Jan Mike Liang[†], Haozhun Jin[*], Yang Yang[*], Li Zhang[⋆], Feng Zhao[†]

[†]Microsoft Research, [*]Tsinghua University, [⋆]USTC
{liang.mike, zhao}@microsoft.com, {genezetta, geraint0923, zlfenyang}@gmail.com

**Abstract.** Proximity-based Social Interaction (PSI) apps are emerging on mobile platforms. While both industries and academic communities have developed frameworks to simplify the PSI app development, our framework, Crossroads, brings a set of features to balance the development overhead and developer expressiveness. We argue that APIs with application hints give developers the expressiveness, and core services (such as virtual links over the star topology) simplify network maintenance. Finally, PSI-specific primitives (such as presence beaconing with interval decaying and group dissemination) improve the energy efficiency. Evaluation results on real smartphones show the energy efficiency gain, topology robustness, and lower group dissemination load.

## 1 Introduction

With rich connectivities and features, smartphones today are capable of extending our presence virtually. A simple example is the increasing popularity of social apps [21]. Building on this momentum, a disruptive form of social interactions is emerging on the market: *Proximity-based Social Interactions* (PSI). With PSI, people's virtual interactions become more location-centric and tied to their current physical neighborhood (typically $< 150$ ft). This is different from the rather static "friend lists" in typical online social interactions.

Industries and academic communities have developed several frameworks for PSI app development [14, 18, 13]. These frameworks recognize device resource management and network maintenance as the primary challenges. An example is the rich choice of physical radios on modern mobile devices, which have a wide range of characteristics (e.g., range, throughput, and energy). While existing frameworks represent a significant step forward, we argue that they lack a set of well-defined APIs and services that balance the development overhead and expressiveness. At one extreme, Windows 8's Proximity API [14] abstracts away many low-level intricacies, at the expense of app-specific tuning of certain parameters such as device presence beaconing frequency. On the other hand, Android exposes many low-level functionalities and controls, but the developers are taxed with the burden to use them properly. While Qualcomm's popular AllJoyn [18] sits in between these two extremes, its fundamental designs do not incorporate several mobile-specific optimizations.

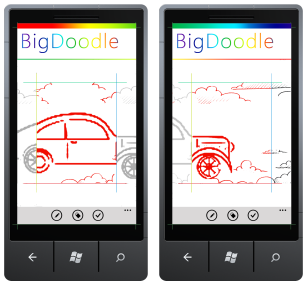



**Fig. 1.** Our PSI game – Big Doodle.

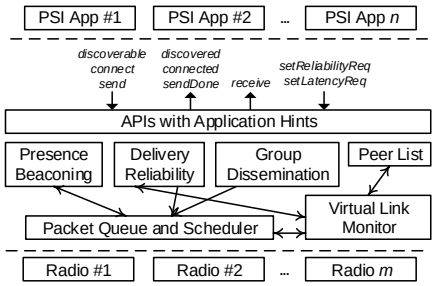



**Fig. 2.** Crossroads framework architecture.

The contributions of our work come from defining powerful PSI programming abstractions, robust networked PSI device management, and efficient operating system services. To this end, the paper presents our PSI framework, *Crossroads*. We drive design decisions from first-hand PSI app development experience and observations from existing frameworks. Crossroads has three main differentiators. First, based on PSI-specific application hints, Crossroads aggregates and schedules pending transmissions while matching the app requirement. Second, Crossroads combines star topology and interval-decaying beacons to better achieve link robustness and device efficiency. Finally, recognizing group communication as an important PSI primitive, we designed a group dissemination protocol that addresses the problem of load hot-spots in many existing solutions.

Evaluations on real smartphones show an energy reduction up to 66% with application hints, and a group dissemination completion time reduction by up to 50%. Finally, Crossroads is able to fix network disconnections via physical radio link migration within 3.5 sec.

Next, §2 first highlights the design patterns across PSI apps. §3 discusses design decisions and features of our framework, and §4 presents our current implementation. Then, §5 evaluates the performance of our framework. Finally, we present related work in §6 and conclude in §7.

## 2 Background

Hundreds of PSI apps [7, 16, 6, 9, 22] and games [1, 20, 23] are already available on the market. We first present two PSI apps we have developed, and then discuss the typical design patterns across most PSI apps to motivate our framework.

### 2.1 Our Proximity-based Apps

***Big Doodle.*** Big Doodle (c.f. Fig 1) explores the class of collaborative gaming, where a group of people work together to accomplish a task. In our case, the goal is to collaboratively doodle an object on a large virtual canvas. During the game

| | Traffic volume | Frequency | Reliability requirement | Delay tolerant |
|---|---|---|---|---|
| **Presence adv.** | Low | Periodic | Low | Yes |
| **Handshake** | Low | On-demand | Low | No |
| **Data transfer** | High | On-demand | High | Maybe |

**Table 1.** Traffic requirement of the three types of messages in PSI apps.

play, participants have limited real-time view of the canvas sections adjacent to theirs. The result from cooperation without explicit coordination can usually be unexpected and humorous.

***SyncUp.*** Exchanging ideas forms the basis of meetings and rendezvous discussions, and conversations can lead to impromptu sharing of files and documents. In contrast to related solutions [8], SyncUp builds on-demand and ad-hoc connections among participants, rather than relying on centralized back-end application servers. This design removes both the dependency on the cloud and the overhead of data transfer over the public Internet.

### 2.2 Traffic Patterns of Proximity-based Apps

Our experience suggests that PSI apps typically exhibit similar design patterns: presence advertisement, connection handshake, and data transfer. As similarities realize a framework, we now look at their network requirement (c.f. Table 1).

***Presence Advertisement.*** PSI starts by discovering neighboring devices and services. One approach is to periodically beacon on short-range radios to infer the relative proximity. And, the reception of beacons would suggest the receiver is near to the transmitter. The beacon frequency and range are tunable to satisfy the requirement of discovery speed and neighborhood area.

***Connection Handshake.*** Before app instances on two devices can exchange data, they need to agree and establish a logical link. A logical link specifies several parameters: the physical medium, end-point addresses, end-point roles, delivery reliability, etc. We use the notion of logical link here to abstract away many of the physical layer differences and intricacies. First, as most mobile platforms support multiple radio options, apps are free to choose the one that closely matches their requirement. Windows 8 Proximity API offers the option of connecting over Wi-Fi Direct or Bluetooth. Second, some network mediums assign different roles to each of the link end-points. For example, in the context of Wi-Fi Direct, one node acts as the soft access point (AP), and other nodes initiate the 802.11 association procedure to establish the link.

***Data Transfer.*** Finally, bi-directional data exchange happens on the established logical link. The data object can vary in size, delivery latency and reliability, number of receivers, etc.

## 3 Architectural Design Overview

### 3.1 APIs with Application Hints

We now present the architecture of our Crossroads framework (c.f. Fig 2). One challenge is to match the app requirement with system resources. At the extremes, the framework can either infer the traffic semantics with a generic model for all apps, or expose all the low-level network functionalities to apps. However, both do not balance between developers' burden and intention expressiveness. Crossroads adopts the approach of passing application-level hints via APIs to achieve the sweet spot. As one contribution, we identify four key PSI hints below.
***Destination(s).*** Knowing the destined group size allows optimizations for application requirement, such as the delivery reliability. For point-to-point transmissions, a simple retransmission mechanism is sufficient to achieve delivery reliability. However, for group dissemination, the same mechanism can overload the sender with retransmission requests.
***Delivery Reliability.*** As reliable transmissions have the cost of additional control transmissions, knowing the reliability requirement helps the framework to avoid unnecessary overhead. An example is device presence beaconing, where one missing reception does not significantly impact the overall operation. Under certain conditions, Crossroads can exploit data objects with low delivery reliability, e.g., artificial packet drops on senders with low remaining battery life.
***Delivery Latency.*** Latency represents room for the framework to schedule packet transmissions for the benefit of amortizing some costs. An example is the tail energy of many radios [2] (e.g., 3G, GSM, and Wi-Fi). Section 5 evaluates two approaches in achieving this goal: piggy-backing onto existing radio operations, and batch transfers.
***Delivery Frequency and Transmission Range.*** These two properties mostly define the behavior and the scope of neighborhood/service discovery. A higher delivery frequency for beacons speeds up the device discovery. While the property of transmission range can be used to set the radio transmission power, it can also be a hint of the physical radio to use.

### 3.2 Network Topology Management

**Star Topology** While many existing frameworks opt the bus topology, we argue that the star topology better fits the PSI communication pattern.

In the bus topology, all devices within a neighborhood connect to a single virtual bus. The virtual bus can span multiple hops via intermediate relay nodes. The star topology shifts the focus from being bus-centric to node-centric. The node-centric view implies that node $a$ can send packets to node $b$ only if there is a physical link between them. We note that this link is a one-hop asymmetric link, as most proximity-based interactions happen among direct neighbors. In cases where the requirement on neighborhood size is relaxed, mobile devices already support network mediums of large coverage, such as the cellular network.

We next elaborate the advantages of the star topology over the bus topology, in the context of PSI apps. First, the star topology has a lower topology maintenance overhead, as star nodes need to maintain only a list of their directly reachable neighbors. In contrast, bus nodes can address a packet to any other node on the same virtual bus, even nodes being relayed. This implies that every node require a globally consistent view of the virtual bus. While the bus topology is manageable in static and wired networks, we believe the overhead represents unnecessary costs in dynamic networks. Second, the lower topology maintenance overhead also translates into topology robustness. In other words, maintaining the neighbor list can be achieved by simply monitoring presence changes in the neighborhood [17].

**Persistent Virtual Links** The abstraction below topology is link, and Crossroads exposes the notion of virtual links to apps. Each virtual link sits on top of multiple physical links, and it hides the complexities of managing multiple radio interfaces. In addition, if the current physical link deteriorates, Crossroads can switch to another radio interface without interrupting the apps. This automatic interface migration addresses the fact that mobility is inevitable on mobile platforms, where network disconnection can happen if users move out of the range of each other or some infrastructure. Although fixing on a long-range radio mitigates the disconnection problem, radios typically have the trade-offs among range, bandwidth and energy.

### 3.3 Group Dissemination Support

Group dissemination is a primitive, especially that many PSI apps function in a group setting. The common approach for group dissemination is originator centric, where only the originator is responsible for reliable delivery. However, this puts a significant burden on the originator in terms of transmissions, and the originator will consume energy much faster than the receivers. To this end, our group dissemination protocol is based on the concept of peer-to-peer (P2P) [1].

Packet loss is generally not uniform across a group of mobile platforms in proximity. In an experiment where four Nokia N800 smartphones were connected to the same 802.11g access point (AP), we instrumented one to stream 10,000 (unreliable) 1,500-byte multicast packets to the other three at an interval of 50 ms. While only 7.51% of the packets were successfully received by all, 91.94% were received by at least one receiver. This suggests that originator is not the only node capable of retransmissions, which motivates the P2P design.

| Name | Arguments | Return values | Type |
|---|---|---|---|
| discoverable | {radio_types}, isDiscoverable, initAdvInterval | | Method |
| discovered | {device_IDs} | | Event |
| connect | device_ID, radio_types | ishandshakeStarted | Method |
| connected | device_ID, radio_types | | Event |
| getConnected | | {device_IDs} | Method |
| send | ProximityObj_ptr, device_ID | | Method |
| sendDone | ProximityObj_ptr, isSuccessful | | Event |
| setReliabilityReq | ProximityObj_ptr, reliability_class | | Method |
| setLatencyReq | ProximityObj_ptr, latency_seconds | | Method |
| receive | ProximityObj_ptr, | | Event |

**Table 2.** ProximityLink class.

| Properties | Descriptions |
|---|---|
| objPtr | Pointer to the data object in memory |
| objSize | Size of the data object |
| latencyReq | Latency class of the data object |
| reliabilityReq | Reliability class of the data object |

**Table 3.** ProximityObj class.

# 4 Current Implementation

## 4.1 Application-hints APIs

*ProximityLink* and *ProximityObj* class represent the logical link and the network data object, respectively (c.f. Table 2 and 3). Every PSI app has an unique app ID, and each app instance on a device first instantiates a copy of *ProximityLink*.

An app first calls `discoverable` to find neighboring devices of the same app ID, with constraints on radio types and the initial presence beaconing interval. The former constraint allows apps to specify the list of radio interfaces to maintain virtual links, and the latter enables an energy-efficient discovery (c.f. §4.2). The `discovered` event is signaled as neighboring devices are found. `connect` and `connected` allow an app to start the handshake with the found app instance on another device, and be notified when a virtual link has been established.

For data transfer, we support two common reliability classes: best-effort and reliable. In addition, the latency requirement translates to maximum queuing delay on the sending device.

## 4.2 Network Topology Management

**Presence Beaconing with Interval Decaying** Calling `discoverable` triggers beaconing all available app IDs on the device. In related work, all devices typically beacon with a fixed frequency. Our interval decaying minimizes this beaconing energy overhead. Considering the case of two devices, a discovery happens when any one device can hear the beacons from the other. In other words, it is theoretically feasible for only one of the two devices to beacon. Unfortunately, this naive solution does not perform well as devices do not know whether there is a neighbor actively beaconing.

[1] While modern smartphones support a wide range of physical radios, we illustrate with 802.11 networks for ease of discussion.

Our basic idea is for all devices to slowly decrease their beaconing frequency. The advantage is that, as the neighborhood stabilizes, all devices would beacon rather infrequently. The design does not add latency to discovering new neighbors, as new devices would beacon at high frequency. The initial beaconing interval is passed as an applicant hint to `discoverable`. Big Doodle uses an `initAdvInterval` of one second to ensure fast multi-player discovery, and then doubles the interval every five beacons.

**Topology Maintenance** `connect` establishes the virtual link with a two-way handshake between two devices, $x$ and $y$. The information exchanged during the handshake sets up both end points for physical link migration. The first message from $x$ declares the intent for virtual link establishment to $y$, with a list of radio interfaces to use. Upon receiving this message, $y$ acknowledges back if it can support the requested link parameters. This acknowledgment then triggers $x$ to update its peer list to reflect the new virtual link. During data transmissions over the virtual link, Crossroads translates destined device ID to the address of the physical radio interface currently in use.

Crossroads actively maintains live virtual links by monitoring the quality of underlying physical links. First, in the idle state, node $x$ assumes a disconnected virtual link if presence beacons from node $y$ have not been heard for a period of time. Second, during an active transfer, the sender assumes broken link if it does not receive acknowledgment after several tries. For receivers, the condition is a timeout since the last successful packet reception. Big Doodle sets the sender threshold to be 10 tries (once per sec), and the receiver wait timeout to be 10 sec.

If the underlying physical link is no longer usable, Crossroads immediately switches to a physical radio interface with longer range, if available. Big Doodle prioritizes by the radio coverage: Cellular > Wi-Fi (infrastructure mode) > Bluetooth. The challenge lies in the interface switching timing (c.f. § 5.2).

### 4.3 Group Dissemination

Crossroads divides the data object of size $S_{obj}$ into fragments of size $S_{pkt}$. Each network packet includes the 32-bit data object ID and the 16-bit fragment sequence number to identify the packet payload. After the dissemination originator advertises the 32-bit unique data object ID, file name, and file size ($S_{obj}$), it multicasts all fragments of the data object to the group. Then, the network enters the recovery phase.

**Reliable Packet Loss Recovery Phase** After a node stops hearing any multicast packet for some time, it scans received data for lost packets and broadcasts a request packet, *REQ*. *REQ* is sent via multicast to reach all Crossroads nodes in the group. Our current implementation sends each request three times to compensate for the lower delivery reliability of multicast.

Upon receiving the first *REQ*, node $x$ starts a 1-sec delay timer for additional *REQ*s from other nodes. Then, by aggregating all requests, nodes can estimate the network-wide reception ratio for each packet. After the delay timer fires,

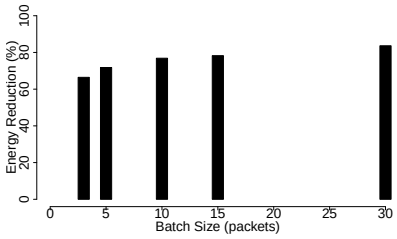



**Fig. 3.** The energy reduction due to batching on transmitting 600 UDP packets over Wi-Fi at 1 Hz.

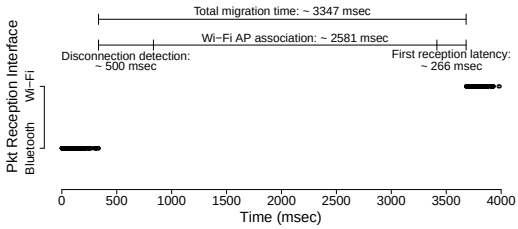



**Fig. 4.** Virtual link switches to Wi-Fi link after the Bluetooth link becomes unavailable.

node $x$ compiles a list of requested packets that it can fulfill. The challenge is to minimize the chance that multiple eligible nodes contribute to the same packet recovery. Our duplication suppression mechanism implements two techniques.

First technique is the queue randomization. Each node randomizes the ordering of requested packets that it plans to fulfill. In contrast to the naive sequential ordering, queue randomization minimizes the chance that two nodes send the same packet at the same time. Second, through overhearing, nodes can learn which data fragments have been sent on the network, and remove them from their queue. This effectively suppresses any duplicated effort.

The group may go through multiple rounds of requests and retransmissions before all nodes successfully receive the data object. After stopping receiving any multicast packet for 1 sec, nodes return to the request phase and send a new request packet with their current data reception summary.

## 5 Evaluations

### 5.1 Latency Hint

Relaxing the latency requirement allows the framework to delay pending transmissions until the energy cost is low, or when a duty-cycled radio is already up. We use Wi-Fi for the purpose of this discussion. Modern smartphones reduce energy consumption by putting energy-hungry radios to sleep whenever possible. An example is the Power Saving Mode (PSM) on Wi-Fi. When PSM is enabled, smartphones periodically wake up to check whether the associated AP has buffered packets destined to them [2]. As each wake-up incurs a fixed cost, we evaluate the usefulness of latency hints with two approaches below.

The first approach is to delay pending transmissions until Wi-Fi radio's periodic wake-ups to listen for beacons from the associated AP. The experiment consists of two Nexus S smartphones connecting to the same TP-Link 802.11g AP, and one device sent one 1,500-byte UDP packet to another device at an average interval of 30 seconds for an hour. We connected the sender to the Monsoon

[2] The Wi-Fi beacon listen interval on smartphones is typically 200 msec.

Power Monitor [15] for energy measurement. The comparison baseline is where the sender transmits according to a random timer with a mean of 30 seconds. Then, we instrumented the same sender to delay each transmission until the Wi-Fi radio is up. Results suggest the latter has an energy reduction of about 3%, and this improvement is from amortizing a fixed radio wake-up cost.

The second approach is to batch a set of delay-tolerant transmissions on the device. We used the same setup as previous, and instrumented the sender to generate packets at a rate of 1 Hz. However, the actual radio transmissions do not take place until the number of pending packets reaches the predefined batch size. Fig 3 shows that the energy reduction, as compared to the case without batching transmissions. The interesting observation is that the reduction can be significant even for small batches (e.g., 66% for a batch size of three). In addition, the energy reduction lowers as the batch size increases, which represents a diminishing return on amortizing a fixed amount of the wake-up cost.

### 5.2 Topology Robustness

For the star topology, the topology robustness is mainly determined by how stable the virtual link between app instances on two different devices is. In the context of mobile platforms, mobility is a major factor contributing to poor link quality and disconnections. Therefore, we evaluate the virtual link stability by looking at how Crossroads mitigates these problems.

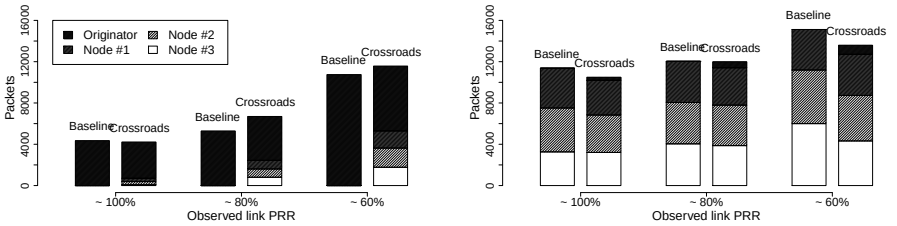
We experimented on two Nokia N800 smartphones connected via a virtual link over both Bluetooth and office Wi-Fi network. One device transmitted a 30-MB object over Bluetooth while we varied the inter-device distance at the (adult) walking speed. The distance increased from $\sim$ 1 m until the devices were beyond the Bluetooth range. Then, after 10 seconds, they were brought back to the initial position.

Fig 4 shows the switch from Bluetooth to Wi-Fi. As the two devices moved out of the Bluetooth range, the sender stopped receiving packet acknowledgments, and the receiver stopped receiving packets. After a time-out of 500 msec, both devices activated the Wi-Fi radio to reestablish the virtual link ($\sim$ 2581 msec). We note that the IP addresses were exchanged during the connection handshake. Finally, the first Wi-Fi packet arrived at the receiver after $\sim$ 266 msec.

Switching back from Wi-Fi to Bluetooth took a much shorter time, as the data transfer can continue on Wi-Fi while the sender probes for the receiver on Bluetooth. Then, after the Bluetooth link is available, the sender immediately stopped the transfer on Wi-Fi and then restarted it on Bluetooth. We observed a delay of $\sim$ 300 msec before the first Bluetooth packet arrived.

### 5.3 Group Dissemination

The evaluation of the group dissemination is based on the energy efficiency, which is derived from two metrics: (1) The number of packet transmissions and receptions on each node. (2) The group-wide dissemination completion time.




(a) Number of packet transmissions (b) Number of packet receptions

**Fig. 5.** The number of packet transmissions and receptions of each smartphone under different observed link qualities.

| | **Crossroads** | | **Baseline** | |
|---|---|---|---|---|
| **Group size** | Originator | Node average | Originator | Node average |
| **4** | 3516 | 306 | 3798 | 0 |
| **5** | 3633 | 255 | 4255 | 0 |
| **6** | 3477 | 165 | 7471 | 0 |

**Table 4.** The number of packet injected into the network as the group size changes.

| | **Total time (sec)** | | |
|---|---|---|---|
| **Observed link PRR** | Baseline | Crossroads | Speedup (%) |
| **∼ 100%** | 20 | 17 | 15 |
| **∼ 80%** | 37 | 32 | 13 |
| **∼ 60%** | 130 | 53 | 59 |

**Table 5.** Group dissemination completion time under different link quality.

We evaluate our group dissemination on a local Wi-Fi network, given the high bandwidth and native multicast support. All six Nokia N800 smartphones connected to a single TP-Link 802.11g access point (AP). During each experiment run, one node disseminated a 3-MB data object to the other five phones. Each experiment was repeated three times at different time of the day to capture any temporal variation.

The comparison baseline is the common originator-centric dissemination protocol. After each round of multicast flooding, all receivers report back their list of missing data packets. With this report from all receivers, the sender then starts another round of multicast flooding to retransmit only the missing packets.

**Node Transmission Counts** Fig 5(a) and 5(b) show the amount of network traffic generated by each node under different link quality. We controlled the link packet reception ratio (PRR) by injecting artificial packet losses following the Gilbert-Elliot model. As both packet transmission and reception consume a significant amount of energy, smaller values are desirable. There are two interesting observations from Fig 5. First, the Crossroads group collectively received less packets than the baseline. This difference is due to the fact that Crossroads switches to unicast if the number of intended receivers is one, which leverages the relatively higher reliability of unicast. Second, from the network point of view, Crossroads injects more packets than the baseline. In the worse case, this difference is about 20%. However, the break down in Fig 5(a) reveals that half of the originator's load shifts to other nodes in the network.

Building on the discussion of load shifting, Table 4 suggests that, in the case of Crossroads, the load of the originator decreases as the group size increases. This observation is related to the decreasing probability of a packet not being

received by any node in the group. On the other hand, in the case of baseline, the originator's load increases with the group size.

**Dissemination Completion Time** Table 5 examines the dissemination completion time under different link quality in a six-node group. When the observed link PRR is close to perfect, both the baseline and Crossroads finished very closely to each other. This is because multicast packets were rarely dropped, and the flooding can successfully deliver almost all of the data packets. As the network link quality became worse, the difference between finish times increased. In the case of 60% PRR, the difference is more than a factor of two. A closer investigation shows that, with multiple concurrent transmitters, Crossroads is able to more closely saturate the radio medium capacity. Since Crossroads nodes can successfully receive the data objects faster than the baseline, they can turn off the radio much earlier to save energy.

## 6 Related Work

**Proximity-based App Framework** Qualcomm's AllJoyn [18] is a cross-platform framework. Like Crossroads, AllJoyn also sits between the application layer and the physical radios, and provides APIs for data exchange among applications on devices in proximity. However, AllJoyn limits what traffic-specific properties that PSI applications can pass, and it does not have PSI-related optimizations, such as reliable group dissemination. Windows 8 ships with a Proximity API that provides an even more limited set of functionalities [14]. The problem of managing multiple radio interfaces on a node has been explored by several prior projects. However, in this problem space, our work explores PSI-specific issues and optimizations. Like Contact Networking [3], we try to provide the illusion of persistent links between applications on neighboring devices. And, we share the view of supplying hints of traffic semantics from applications to lower layers for optimization [11]. This design is absent in some related projects [24].

**Reliable Group Dissemination** Previous efforts on Bluetooth mostly focus on building a multicast tree with respect to some metrics, such as energy and latency [4, 5]. The lack of link-layer acknowledgment (ACK) is one source of low multicast delivery reliability on Wi-Fi. However, requiring ACKs from all receivers can cause an acknowledgment explosion in the network. Kuri et al. [12] proposed a leader-based protocol; RMAC [19] and 802.11MX [10] generate an out-of-band tone to signal positive and negative ACKs respectively. However, these work do not consider the energy constraint of smartphones, as the burden of packet retransmissions is still entirely on the sender.

## 7 Conclusion

Compared to solutions from the industry and the academic communities, Crossroads brings a set of expressive APIs and PSI-optimized services to balance PSI

app development overhead and developers' expressiveness. Our design was driven by first-hand PSI app development experience, and supported by the evaluation results. We are working on bringing Crossroads to more mobile platforms.

## References


1. C. Baber and O. Westmancott. Social Networks and Mobile Games: The Use Of Bluetooth For A Multiplayer Card Game. In *MobileHCI*, 2004.
2. N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy Consumption in Mobile Phones. In *IMC*, 2009.
3. C. Carter, R. Kravets, and J. Tourrilhes. Contact Networking: A Localized Mobility System. In *MobiSys*, 2003.
4. C.-T. Chang, C.-Y. Chang, and S.-W. Chang. Tmcp: Two-layer multicast communication protocol for bluetooth radio networks. *Computer Networks*, 52, 2008.
5. C.-Y. Chang, K.-P. Shih, H.-J. Chang, S.-C. Lee, and G.-J. Yu. Pamp: a power-aware multicast protocol for bluetooth radio systems. In *ICCCAS*, 2004.
6. Chatter, inc. Buzzmob- social media for real life. `http://www.buzzmob.com`.
7. Color Labs, Inc. Color - Broadcast Live. `http://color.com`.
8. R. C. Davis, J. A. Landay, V. Chen, J. Huang, R. B. Lee, F. C. Li, J. Lin, C. B. M. III, B. Schleimer, M. N. Price, and B. N. Schilit. Notepals: Lightweight note sharing by the group, for the group. In *CHI*, 1999.
9. Foursquare Labs, inc. Foursquare. `http://www.foursquare.com`.
10. S. K. S. Gupta, V. Shankar, and S. Lalwani. Reliable Multicast MAC Protocol for Wireless LANs. In *ICC*, 2003.
11. B. D. Higgins, A. Reda, T. Alperovich, J. Flinn, T. Giuli, B. Noble, and D. Watson. Intentional Networking: Opportunistic Exploitation of Mobile Network Diversity. In *MobiCom*, 2010.
12. J. Kuri and S. K. Kasera. Reliable multicast in multi-access wireless lans. In *Infocom*, 1999.
13. A. Le, L. Keller, C. Fragouli, and A. Markopoulou. MicroPlay: A Networking Framework for Local Multiplayer Games. In *MobiGames*, 2012.
14. Microsoft, inc. Windows.networking.proximity namespace. `http://msdn.microsoft.com/en-us/library/windows/apps/windows.networking.proximity`.
15. Monsoon Solutions, Inc. Power Monitor. `http://www.msoon.com/LabEquipment/PowerMonitor/`.
16. Nearverse, Inc. LoKast - Real-time Interactive Spaces. `http://www.lokast.com`.
17. J. J. Parsons and D. Oja. *New Perspectives on Computer Concepts 2012*. Cengage Learning, 14th edition, 2011.
18. Qualcomm Innovation Center, Inc. AllJoyn. `https://www.alljoyn.org`.
19. W. Si and C. Li. Rmac: A reliable multicast mac protocol for wireless ad hoc networks. In *ICPP*, 2004.
20. R. Spanek, P. Kovar, and P. Pirkl. The bluegame project: Ad-hoc multilayer mobile game with social dimension. In *CoNEXT*, 2007.
21. TechCrunch. Nearly 40% of facebook use is from mobile apps. `http://techcrunch.com/2011/12/29/nearly-40-of-facebook-use-is-from-mobile-apps/`.
22. Tencent, inc. Weixin. `http://weixin.qq.com`.
23. Z. Zhang, D. Chu, X. Chen, and T. Moscibroda. Swordfight: Enabling a new class of phone-to-phone action games on commodity phones. In *MobiSys*, 2012.
24. S. Zhuang, K. Lai, I. Stoica, R. Katz, and S. Shenker. Host Mobility Using An Internet Indirection Infrastructure. *Wireless Networks*, 11(6), Nov 2005.