Journal of Mobile Multimedia, Vol. 7, No.4 (2011) 239-255

SSN: A SEAMLESS SPONTANEOUS NETWORK DESIGN AROUND OPPORTUNISTIC CONTACTS

JIANXIONG YIN

Student Member, IEEE

Jianxiong.yin@ieee.org

MIN CHEN

Senior Member, IEEE

minchen@ieee.org

Received December 10, 2011

Revised April 7, 2012

Current mobile computing research has identified great potential of device-to-device local collaborations, e.g., crowd sourcing. The utilization of the free wireless radio channels has also been suggested for the benefit of infrastructure. On the other hand, both high transmission ratio and low security risks are considered fully necessary for mobile platform services. We identified such basic application scenario and system requirements, and present design principle of a device-to-device communication framework named Seamless Spontaneous Networks (SSNs). SSN links the applications layers of separated devices by raising APIs so that SSN could be leveraged for high-level data exchange with regards to privacy control, security and delivery to anonymous endpoints. We implemented SSN on Android-based off-the-shelf smartphones and carried out the assessment and analysis of its performance.

Key words: Delay Tolerant Applications, Spontaneous Networking, Content Sharing

1 Introduction

With the proliferation of mobile devices, spontaneous interaction between co-located devices that do not know each other a priori will become commonplace. Among mobile device, smartphones are rapidly

growing in number and in the same way the administration effort for handling the traffic generated by smartphones is also increasing.

Cellular network operators are deploying numerous Wi-Fi access points to shift traffic burden from

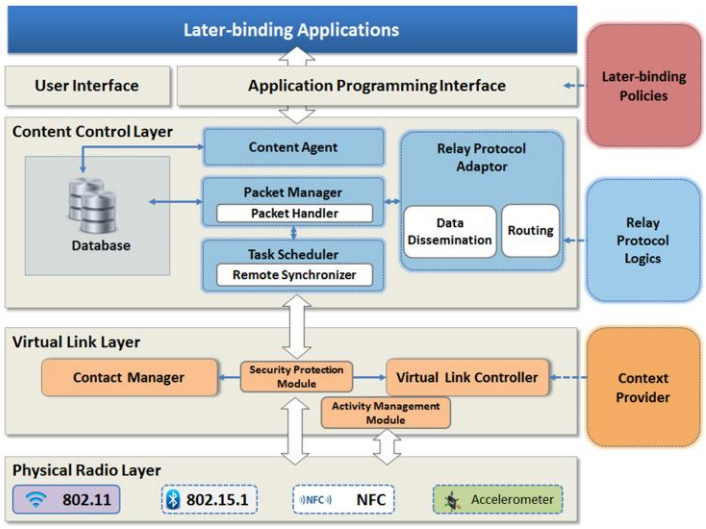


Fig. 1 SSN System Overview

the cellular network. For the sake of threatens from ambitious technology companies, e.g., Google, Apple, operators have to seek for innovative pattern for profit increase. One of the alternative solutions is using spontaneous opportunistic networking. Unlike traditional ad hoc networks, the spontaneous networking is an application oriented approach, hence has at least two major advantages: (1) it enables ad hoc contact between smartphones which enlarges chance of obtaining fresh data (2) it works in a plug and play manner minimizing the effort to integrate devices and services into network environments. Thus SSN is suitable for various networking patterns, such as pocket switched networks (PSN), delay or disruption tolerant networks (DTN), opportunistic networks (ON), Mobile Social Network and etc. In this paper, we assume existence of unique constraints such as availability of localization techniques, motion sensors and various radio interfaces.

Many research have proposed application that will benefit from the implementation and use of such networks. Andres et al. [2] present a number of applications. For instance, anonymous chatting, file sharing and bulk data transfer, and automatically synchronizing mobile and static devices.

A variety of previous works have focused on infrastructure free communication framework. In general, they are in 2 categories: special purpose implementations [2-4] and generic implementations [5-7] and revealed implementation principles for device-to-device frameworks. Some implementations [2, 4, 5] have showed good performance in networking and usability, they are still not welcomed by ordinary users. This is because either the systems are not instantly usable requiring modifications to device's software architecture or provided applications have only little incentive to users.

As stated in [2], we believe that, generally, following requirements should be met by ordinary systems. First of all, they must overcome technological constraints and should be instantly usable. Limitation of current smartphone's radio communication includes the connection oriented nature of Bluetooth radio, absence of Wi-Fi ad hoc authentication, limited battery capacity, and diversity of mobile devices. Specifically, devices that use Bluetooth radio should be pre-paired in addition to the small communication bandwidth. Although Wi-Fi ad hoc mode is not removed by many smartphones, it is not available unintrusively. Second, extensibility and later binding application should be supported. In addition, system must be energy and resource efficient, should protect user privacy.

In this paper, we present a set of design principles and implementation of a system of such kind. Our goal is providing good and instantly usable communication throughput as well as overcoming the aforementioned challenges.

The remainder of the paper is organized as follows: Section 2 first presents brief overview of the system, then describes system components in detail. We talk about implementation challenges, programming interfaces in Section 3. System evaluation in terms of complexity, networking efficiency is presented in Section 4. We discuss relevant works and limitations of our system in Section 5, and conclude the paper in Section 6.

2 Related Work

As we have mentioned, previous works have addressed spontaneous smartphone networking as pocket-switched networks, delay and disruption tolerant networks, and opportunistic networks. In this section, we cover known implementations of such network architectures. Among the implementations, DTN implementations are most prominent. DTN2 by the Delay Tolerant Networking Research Group (DTNRG) [5] is implemented to demonstrate basic functionality and therefore does not operate very efficiently. Interplanetary Overlay Network (ION) [17] is an implementation of the DTN Bundle Protocol, Licklider Transmission Protocol (LTP), and the CCSDS File Delivery Protocol (CFDP) and Asynchronous Message Service (AMS). The intent of the ION distribution is to support high-speed, small-footprint deployment of DTN in embedded systems such as robotic spacecraft, but it also runs well in more resource-rich environments such as local area networks [17].IBR-DTN [3] is light-weight implementation to run on embedded devices as well as on standard Linux systems. DTN architecture consists of an overlay, called the bundle layer, which operates above the transport layer. The goal is to deliver data units called bundles from a sender to a receiver in the presence of opportunistic connectivity using different transport protocols assuming that nodes store, carry and forward bundles to cope with link outages. Owing to the bundle layer DTN implementations support variety of delay-tolerant applications. However, they did not target for smartphones therefore did not consider constraints and advantages of smartphones.

In case of opportunistic network implementations, Haggle [6] is an architecture for mobile devices that facilitates the separation of application functionality from the underlying network technology to provide seamless operation despite disruptions in network connectivity. It is composed of a number of modular building blocks Connectivity is achieved based on late binding of network interfaces, protocols and names. Haggle is not a strict protocol architecture but rather proposes a node design that allows

nodes and applications to adapt to the network connectivity level. DASM [18] is an implementation of DTN family of protocols for Symbian-based mobile phones. DASM includes implementations of Bundle Protocol v4 [19], a TCP convergence layer, and Bluetooth convergence layer. However, DASM yet does not provide APIs for external applications.

To extend legacy applications, such as web and e-mail, to opportunistic environments, Moghadam[7] implements a modular platform that provides application developers with core underlying functionalities required for mobile disruption-tolerant communication. Since the platform is designed for general purpose applications, platform lacks optimizations and context-aware networking opportunities.

Cimbiosys [21] is a platform that provides content synchronization and replication through opportunistic peer-to-peer communication. It uses content based filters to specify which contents are synchronized and shared by the system. Blue Torrent [4] is an opportunistic file sharing application only for Bluetooth enabled devices.

Gunnar et al.[8] presents the original idea of a delay-tolerant broadcast system and evaluates its feasibility in an urban area while [9]introduces podcasting as an application for DTN. Yonekiet al.[20] presents a content-based routing scheme for publish/subscribe communications in a DTN environment.

In summary, we can state that previous implementations are either designed for special purpose applications or implemented generic DTN architecture. Implementations for special purpose applications are neither extensible nor context-aware. Although implementations based on generic DTN architecture, on the other hand, are extensible and provide context, they are not practically usable on smartphones. Because, some of them require active user intervention, e.g., pairing devices, while others require modification to software architecture. In contrast, our implementation is purely based on current software architecture of smartphones and available sensors. Therefore, it can be instantly used after.

3 System Overview

SSN is a spontaneous networking framework aims at efficient package exchange among smartphones without infrastructures. We implemented SSN on Android smartphone with official APIs. SSN works by 4 stages, and each stage is managed by corresponding modules, they are Content Control Layer (CCL), Virtual Link Layer (VLL), Sensor Physical Layer (SPL), Packet Manager and Protocol Adaptor.

We assume the bulk-data is usually exchanged in the places where smartphone users stay for the significant duration, we define these places as Landmarks. Consequently, the system is required to track the movement of the. The Activity Manager monitors the user activity and schedules other system tasks. The component continuously checks the variance of accelerometer signal and the phone state to identify to user activity. The user activity is classified as “moving” and “stationary.”

When a stationary activity is detected for a significant duration, SSN records the current attached Access Point and its BSSID and MAC is identified as landmark’s characteristic as fingerprint set.

Virtual Link Layer (VLL) discovers neighbouring smartphones and creates spontaneous network, the technical details will be described in the next sections. Collected neighbour information and

neighbour encounter information is then handed to the Contact Manager. The Contact Manager also monitors active neighbours and assists routing protocols in data exchange.

The Content Control Layer (CCL), Packet Manager and Relay Protocol Adaptor (RPA) are core components for data exchange. The Packet Manager and Content Agent receives application data packets and schedules for delivery. Also, messages arrived to routing protocol are handed to the Packet Manager, and delivered to applications. RPA contains a number of prebuilt opportunistic routing options, and based on current routing protocol this component makes decision on message forwarding. Additionally, RPA raises interfaces with customized relay protocol logics.

Finally, CCL manage storage such as aggregating raw data into refined data, removing expired message information. Applications Data Packages are transported between application layers of separated devices with consideration of security protection and content descriptions.

4 Virtual Link Layer

Virtual Link Layer is the abstract of establishing and controlling virtual communication links between nodes. Establishing communication links between moving nodes is neither energy-efficient nor sufficient for exchanging bulk-data. Since opportunistic contact instinctively doesn't support delay-guarantee data forwarding, our framework doesn't consider such circumstances. The Activity Manager triggers Link State Manager when a node keeps stationery status for a certain amount of time. Upon this, VLL searches for available Spontaneous Providers (SP). SPs are usually identified with previous Negotiated SSID. If the node finds existing SP, the node will be Spontaneous Client (SC) and connects to the SP using WSC method.

In case the node does not find SP, the node switch on software based AP (Soft-AP) functionality and serves as SP. In short, SSN utilize an approach which simulate a Wireless Local Area Network on SP using Soft-AP functionality, so that other nodes could connect to it. Soft-AP is a tethering mode of smartphone that is initially designed to support group mobile IP devices. Recently, it has got official support in mainstream smartphone operating system distributions such as Android [11] and iOS [12]. However, this feature's groundwork has as well been laid in the early stage of smartphone boom. For example, Broadcom's wireless module [13], which is adopted by Google Nexus One/S, HTC Desire series, iPhone 3GS/4, Samsung's Galaxy series, providently integrated hardware support for Soft-AP. When the node serves as SP, other nodes which arrive and stay near the SP will act as SC.

While node remains stationary, VLL stays idle until Activity Manager signals. Consequent to the movement detection, Link State Manager notifies Contact Manager link disconnection. Contact Manager broadcast a leave message and connection is terminated. When the leaving node is a SC, then node does nothing more than the basic action mentioned above. However, when SP is leaving, one of the remaining nodes should form new spontaneous network group. In order to avoid multi SSN coexistence, we use semi-randomized waiting time. Thus, a node k waits for the duration of $\Delta \mathbf{t}_k$ after receiving the leave message, performs Wi-Fi scan, and serves SP unless other SP is found. We compute $\Delta \mathbf{t}_k$ using following equation:

$$\Delta t_k = \psi\ (z\ mod\ x) + rand[0, \psi], \quad (1)$$

where ψ is average time required for configuration, z is the last byte of IP address obtained from previous SP, x is the maximum possible number of SCs per SP. Empirically, we have found out that ψ is 5 seconds and x is 8. Although last byte of node's IP address (z) is unique, modulating it by x is not absolutely unique. But, with additional random delay between 0 and ψ the conflict is minimized since each node performs scan before forming new group. Conflict in forming new group may also occur when two nodes arrive at landmark at similar time. To cover such cases, z is set to the mobile equipment identity number because nodes do not have IP address.

5 Neighbour Discovery and Privacy Security

Neighbour discovery process collects social context via detecting neighbours within the communication range. Neighbour discovery is a challenging issue for current smartphones since radio interface of Wi-Fi devices only listen to Link Layer probe packets from Access Points as power saving design. Therefore, we use passive dynamic neighbour discovery mechanism, and in this context we call it "blind date party" model. In this mode nodes can discover each other unless they are co-locating and temporarily linked to an active SSN.

The neighbour discovery is applied under assortative with contact management for the communication convenience. Virtual Link Layer is activated when node links to a SSN. Afterward, a

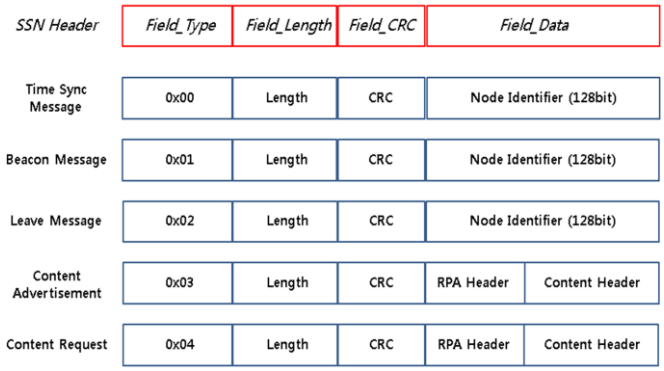


Fig. 2 Message Types and Formats. Type indicates message type: time-sync, beacon, leave, content advertisement and content request. Length is length of entire message. A cyclic redundancy check (CRC) is used to check the correctness of message content. Data contains actual message content.

node joins network group or forms a new one, Contact Manager broadcast periodical beacon messages which is carried by User Datagram Protocol (UDP) Broadcast method. At the same time, Contact

Manager runs a UDP listener thread, which keeps listening to SSN discovery port negotiated in advance. By receiving beacon messages Neighbour Manager creates new entry in neighbour table or updates old record. The neighbour table compromises neighbour identifier, connection status, local IPv4 address, and last contact time. In order uniquely to identify neighbours, we use International Mobile Equipment Identifier (IMEI) as neighbour identifier. Since IMEI is vulnerable to privacy security threats, it is hashed using MD5 [14] and 128-bit identifier, which is also adopted by IPv6 address. Connection status indicates if the node is currently connected to spontaneous network group, and it is set to "connected" upon receiving beacons. The last contact time is the time of beacon reception. While connection status is "connected", Packet Manager and Relay Protocol Adaptor deal with message exchange. Finally, when a node is leaving landmark, it broadcasts leave message. Other group members set corresponding update the flag as "disconnected". In case neighbour leave message might not be delivered properly e.g., unexpected broken link, we prepared following technique to detect lost neighbours. Nodes, as a rule, receive beacons from other nodes with a certain interval annotated as δ . For each neighbour, with a periodicity ϕ that is greater than δ , Contact Manager checks its lost neighbour by calculating the difference τ between current time and last contact time. If τ is greater than δ , it suggests the neighbour has left. Connection status of lost neighbours changes to "disconnected". Fig.4 shows the process of link state management and neighbour discovery.

The response to the question: "*Are most of the client needs captured before or after signing a contract for the project?*" is interesting. A notable percentage (38%) of the developers indicated that they often adopt a 2-step negotiation specifically because of client uncertainty. The 1st step involved a contract (usually based on an hourly rate or equivalent) for the development of a specification. The 2nd step would then involve another contract (usually fixed price) for the system build.

6 Time Synchronization

Networking among mobile devices is highly constrained by battery capacity. Therefore, an applicable framework shall not only have nice throughput but also power saving. UDP protocol is used for beacon message delivery in the past sections since it simplifies way of group communication, meanwhile, its default port listening consumes lots of resources. Following the stipulation of compatibility, we put our effort on the duty cycle scheduling criterion, so that SSN could works power efficiently regardless of the lower level differences.

Duty cycle scheduling in networking field is a long story since it's critical to every temporal and spatial re-use approaches. Most of the WCDMA smartphones sync time with base-stations in a way under layer in a millisecond level deviation, however, time quantum available from Android API has shown a deviation within range of [0, 15] seconds, which requires SSN has auto-time sync facilities ahead of any communication. In our implementation, SSN Clients syncs time with SSN Provider after links are initiated so that all the nodes act according to the time step of SSN Provider.

For simplicity, We set duty cycle periodicity to 15 seconds and divide it with a ratio of 1:2, which means 5 seconds active time and then 10 seconds sleep. During active period, UDP listener will keep

awake and TCP communication is available. When it comes to sleep status, SSN's related thread will fall asleep with low power consumption.

In case a node becomes SSN Provider, it runs with the aforementioned sleeping schedule. When a node joins SSN, after receiving neighbour beacon from local Provider node, it broadcast time sync message which contains its timestamp continuously until it receives time sync beacon from Provider node to itself. Provider node's Packet Manager will pick time sync message and response with time sync message which time deviation between Provider local time and Client's timestamp. When the previous time syncing Client received time deviation, it adjusts its local time with this deviation. Then it goes into real message handling phase.

7 Packet Handling and Synchronization

Packet Manager (PM) basically takes care of the contents from applications and delivering them to the destination. In general, MM dispatches, exchanges, and schedules messages.

Conventionally, we designed management message and content message for different communication periods. Basically, Management messages are distributed via UDP interface and content message flows by TCP.

First, we talk about general dispatching routines of these messages.

Management messages signals processing units within framework layer. There are five types of management messages in SSN: time sync message, neighbour beacon message, neighbour leave message, content advertisement message, and content request message. Fig. 2 shows message types and formats. MM dispatches these messages by types. Time sync messages synchronize communication

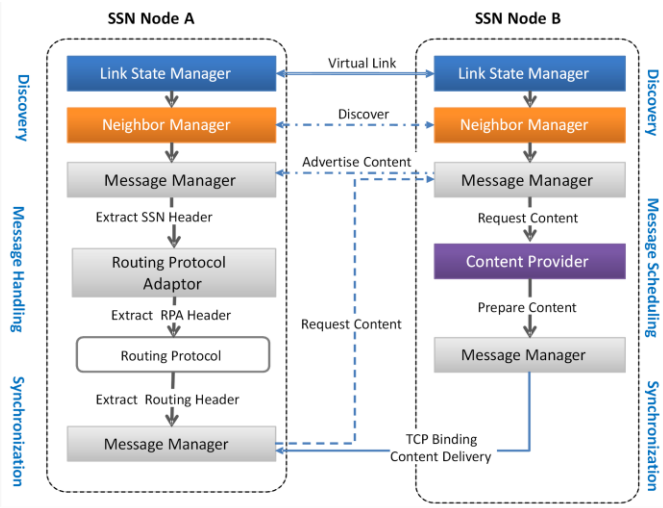
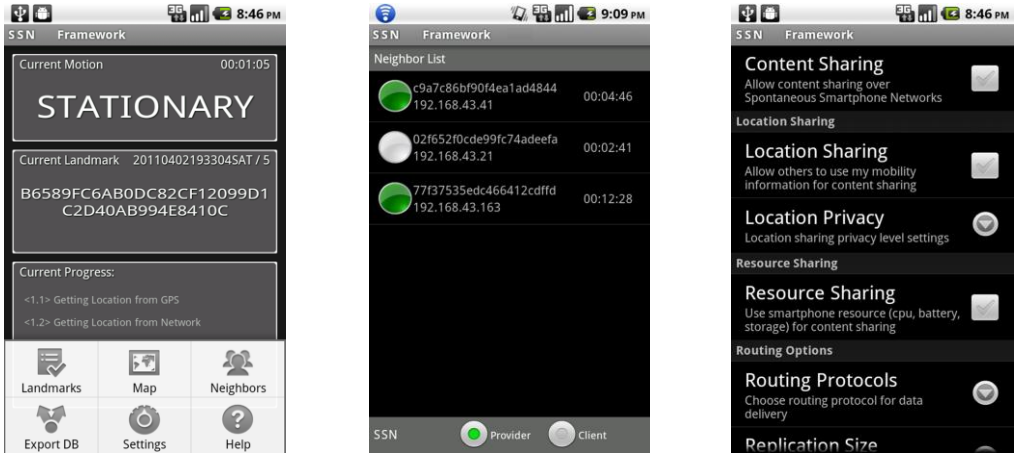


Fig. 3 Packet Handling and Synchronization. Dash-dotted lines are UDP broadcast, dashed lines are UDP unicast, and solid lines are TCP

steps of client nodes with provider's. Neighbour related messages –beacon and leave–are handed to CM and further processed there. Content related messages – content advertisement and content request – are forwarded to RPA and the header of RPA is extracted.

Secondly, the RPA header constitutes of routing protocol identifier and routing protocol header. Routing protocol identifier for both external and internal routing protocols is assigned by RPA at SSN initialization. According to the routing protocol identifier, the relevant routing protocol processes routing protocol header and makes decision upon discarding the advertisement message or requesting actual content for this advertisement. The decision is returned to MM and MM sends content request to content advertiser over UDP based on the decision. The content request contains content identifier extracted from advertisement and special TCP port to receive message. Content advertiser node then binds connection using TCP port and transfers actual content. Such data exchange method is important for anti-harassment meaning that malicious users can not directly flood data to others. Content is usually requested in two cases: (1) the receiving node is content destination, or (2) the receiving node can relay the content to the destination. Fig. 3 illustrates concise description of message handling and synchronization process.

Content message is either directly raised to application layer or queued for forward opportunities after being stored. Contents, which are not destined to the receiver, are stored in storage (e.g., SD card) and the message header for the content is inserted in message table. If the receiver is content destination, the content is raised to the application layer by checking the application port. If the receiver is not content's destination, CCL schedules this message and has interface to receive content messages from applications layer. When Content Agent receives a content message, it searches for the destination of the message in active neighbour list. If the destination node is found, PM binds TCP communication with the destination and immediately transfers the message. Otherwise, PM creates new entry in the content message table for this message. Content Message table is part of database and has following fields:



(a) Status and Progress View (b) Neighbor List View (c) Settings

Fig. 4 SSN user interface. User checks status of SSN (a), Active (green indicator) and inactive (gray indicator) neighbors are shown in (b). User configures SSN leveraging settings window (c).

message (content) identifier, application port, delivery deadline, and link to the actual data in storage. Elements of message header for contents are restricted to these fields of message table.

With a careful consideration of utilizing local storage, the Scheduler of PM periodically checks the message table to remove messages with expired deadline. When a node encounters other nodes, the scheduler prepares messages for delivery. The Scheduler is also linked with Storage Manager of Content Provider to apply message dropping algorithms.

8 Routing Adaption

Another key functionality of SSN is extensibility with routing protocols. SSN supports delay-tolerant routing protocols. Key functions of DTN routing protocol are to make decision on forwarding message and selecting next hop node. In SSN, each routing protocol registers three call-back functions – forward/receive decision maker, next hop definer, and header generator. Also, external routing protocols can use APIs to obtain other information.

In purpose of distinguishing routing protocols SSN assign unique identifier analogous to TCP/UDP port. Each routing protocols determines its own routing header, which is not limited to special format. RPA obtains routing headers using registered header generator call-back of routing protocol. Routing protocol identifier and routing header together comprises RPA header, which is mentioned in Section 2.3.4, that is attached to content advertisement and content request messages after SSN header. RPA headers are generated by only source of content or request generating node. Intermediate nodes that receive RPA headers extract routing header and forwards to decision maker call-back of relevant routing protocol. The decision maker function of routing protocol extracts elements of header and returns decision on receiving actual content. As we mentioned in previous section, contents are received either to delivery application or to relay other nodes. For instance, decision maker function of Epidemic routing would always return positive response on receiving actual content. However, CP checks if the same message is already available in the database and requests only missing messages.

Main function of Packet Manager (PM) is organizing content from applications and delivering them to the destination. In general, MM dispatches, exchanges, and schedules messages.

Management messages signals processing units within framework layer. There are five types of management messages in SSN: time sync message, neighbour beacon message, neighbour leave message, content advertisement message, and content request message. Fig. 2 shows message types and formats. MM dispatches these messages by types. Time sync messages synchronize communication steps of client nodes with provider's. Neighbour related messages –beacon and leave–are handed to CM and further processed there. Content related messages – content advertisement and content request – are forwarded to RPA and the header of RPA is extracted.

Next hop defines call-back of routing protocols receives destination address and active neighbour list, and decides which neighbour or neighbour set is best to carry the message. In fact, routing protocols use context information for better decision making.

SSN has three built-in routing protocols – Epidemic [15], Spray & Focus [16], and Utility-based Routing [1], which are created using RPA interfaces. The header of Epidemic routing contains only destination address and delivery deadline. Consequently, next hop definer of Epidemic returns all active neighbours as next hop. Spray & Focus assigns a replication number to a message, distributes message copies to a number of carrying nodes using history encounter times and encounter rates. Similarly, Utility-based routing protocol uses mobility history and future contact prediction to select message carrier. Routing header of these protocols compromises destination address, replication size, encounter rate, utilities, etc. The next hop definer of these routing protocols returns a single node or a set of nodes according to available context.

9 Implementation

We implemented SSN on the Google Android protocol stack and validated correct operation on a number of smartphones such as Nexus One, Nexus S and Galaxy Nexus. Software-based access point functionality is enabled on these devices and they are equipped with necessary sensor such as GPS, GSM, accelerometer, digital compass, etc. SSN is implemented as Android service that continuously runs in the background. Although SSN runs not in privileged user mode (i.e. rooted mode), which is required for

```
interface SSN_API{
     void       trigger_ssn();
     void       deliver_to_all(in byte[] _data, in long _deadline, in int _port);
     void       deliver_to(in byte[] _data, in long _deadline, in Neighbor _node, in int _port)
     Neighbor[] get_neighbor_list()
     int        get_encounter_rate(in Neighbor _node);
     long       get_next_encounter_time(in Neighbor _node);

     void       registerCallbak(SSN_Callback cb);
     void       unregisterCallback(SSN_Callback cb);
}

interface SSN_Callback{
       void      receive_data(int _port);
       void      neighbor_notification(Neighbor _node);
       boolean   forward_decision_maker(RPAHeader _header);
       Neighbor  next_hop_definer(Neighbor  _destination);
       RPAHeader generate_header(Neighbor _destination, long _dealine, int _replication);
}
```

Fig. 5 Application Programming Interfaces. These interfaces provide functions to obtain information, send and receive data, configure and extend the system.

some systems [5, 6, 22] implemented on Android has limitation due to System Fragmentation issue of Android system. For example, software-based access point functionality is enabled on some phones or control interface is not standardized. Also, accelerometer reading is disabled when screen is turned off due to the power saving. To identify these problems, SSN probes these functions at the initialization stage and operates accordingly. That is to say, if the Soft-AP functionality is not supported, SSN does not try to enable this function even if the other SSN provider is not found when a new landmark is discovered. Another issue in implementation is that when SSN enables Soft-AP mode, Wi-Fi communication with APs cannot be established. However, the problem can be solved with Wi-Fi Direct since it supports parallel connection with AP and other smartphones.

9.1. User Interface

In SSN, a console like User Interface (UI), which is shown in Fig. 4, is designed to assist intuitive configuration and fast check out of background status. User can have a Straight forward view of current or last visited landmark and current working progress in Fig. 4(a). All the encountered neighbours could be monitored via a neighbour list window Fig. 4(b) with the current contact duration and an online-indicator. Thus, connected neighbours are identified with green indicator. A configuration panel in Fig. 4(c) is used to tune the privacy sharing degrees and resource sharing strategies.

9.2. Application Programming Interfaces

Typical peer-to-peer data exchange system's API modules offer methods for communication. In addition, our system provides user context, which is geographical and social history of encountering. Also, SSN has a number of call-back functions to extend its functionalities such as adding new routing protocol and implementing neighbour discovery methods. We categorize SSN's API into three groups: SSN Transportation Interface (CTI) and Local Context Access Interfaces (LCAI), and Configuration and Control Interface (CCI).Fig. 5 shows list of mostly used APIs and call-backs and more APIs are available in SSN.

CTI implements the advertisement, delivery and download of data from other peers. Applications

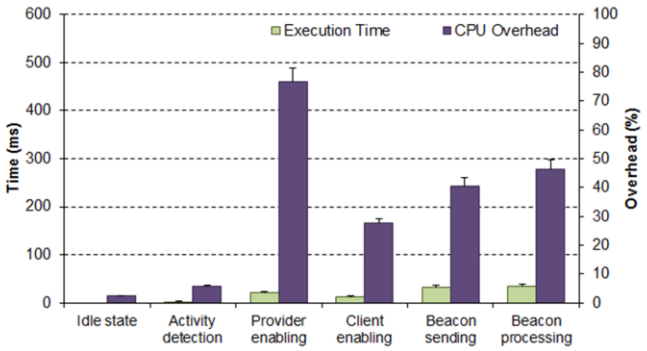


Fig. 6 CPU Overhead and Task Execution Time

may use delivery functions to share data with others. The application port must be specified to deliver

Table 1. Packet loss rate and Throughput

	Packet loss ratio	Through-put
PC to Mobile	6.5%	2779.48 Mbit/s
Mobile to Mobile (Ad hoc)	13.4%	822.65 Mbit/s
Mobile to Mobile (SSN)	3.4%	2695 Mbit/s
Mobile to Mobile (Bluetooth)	53.2%	237.24 Mbit/s

data to intended application. Applications should use unique port although duplicated ports are allowed for the purpose of delivering same data to multiple applications. Applications optionally specify deadline for delivery that may facilitate delivery and avoid unnecessary distribution of data. To receive data, applications register call-back for data receiving and specify a port. When MM receives data, it checks application port and forwards to applications with matched port number.

LCAI has components of social and geographical context access methods. Social context is basically neighbour based which could link to relevant temporal and spatial encounter histories. Geographical context organizes information corresponding to specific landmark, which includes visit duration and number records, transition and etc. These contexts can be used in both applications and external routing protocols. Applications use contexts for designing networking applications and user-centric applications. For example, using landmark information, transition between landmarks a "Digital Diary" or "Appointment Scheduler" application can be implemented.

In CCI, we provided access for control and configuration mechanisms on resource sharing, data exchange and location privacy availability degrees. APIs in this group are designed for routing protocols implementation, controlling behaviour of SSN on neighbour discovery and data exchange. CCI assists to set parameters of routing protocols. In order to reduce SSN's complexity, we placed global routing protocol strategy options in CCI, so that it will apply to all attached applications. In our future work, we plan to add more detailed settings, for instance, the social privacy accessible degree and the designation of routing options for each application or even each message and etc. Applications that run on SSN needs to import SSN-specific classes such as Neighbour, Landmark, Visit, and RPA-Header, and use Android Interface Definition Language (AIDL).

10 Experiments

As we mentioned in Section 1, spontaneous networking system smartphones should exhibit following characteristics to be pragmatic: lightweight in terms of CPU and memory overhead, efficient in resource utilization and energy consumption, rapid in transportation ratio. Specifically, task scheduling and process management should be carefully designed since the system runs continuously in the background. With the following experimental analysis our goal is to show the feasibility of spontaneous networking system for smartphones.

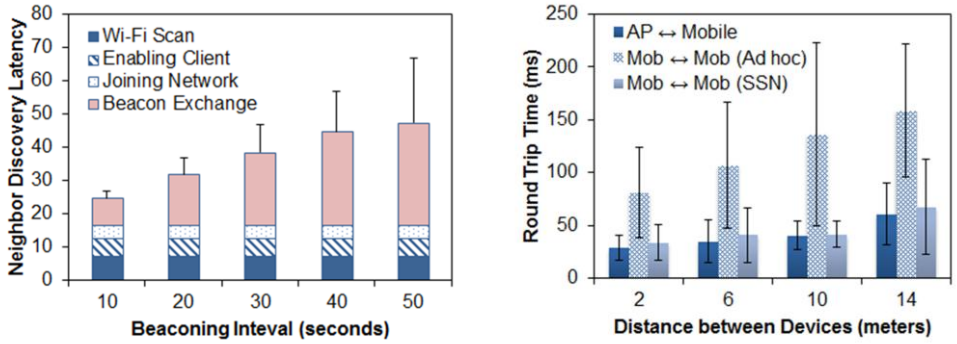
10.1. CPU Usage and Overhead

To perform context-aware networking, our system employs SSN service which is described in Section 3. The service continuously runs in the background to discover neighbours and exchange contents. Fig. 6 illustrates CPU overhead and CPU usage time of various tasks carried out by SSN. We have used Android tracing method to identify CPU usage time and overhead per task. In Fig. 6, the percentage of CPU usage for each task, which is average of 100 samples, is shown. Most of time, the SSN runs in the idle mode and it has only 2.5% CPU overhead. The accelerometer readings then processed by Activity Manager and type of user activity is detected, which has overhead of 5.7±0.8% and runs for the duration of 2.4 millisecond for each detection.

Four tasks that related to spontaneous networking – provider enabling, client enabling, beacon sending, beacon receiving – have very short runtime although they generate higher CPU overhead. Interestingly, enabling provider mode generates about 76% overhead, which is high, and runs for very short duration, i.e., 20 milliseconds It is because a number of operations to control hardware are done while enabling provider mode. On the other hand, enabling client mode generates about 28% CPU overhead during 12 milliseconds. In the client mode, a device attempts to connect to provider device. Tasks such as actual connection establishment and obtaining IP address from provider's DHCP server are not included here. The beacon sending and beacon receiving tasks are part of UDP packet processing. Since the size of beacon is small, both sending and receiving the beacon takes short time – 32~35 millisecond, and generates CPU overhead of 40.6% and 46.2% respectively. Other tasks of SSN such as TCP communication, Wi-Fi scanning, GPS reading significantly depend on user parameter settings.

10.2. Networking Efficiency

Networking efficiency of SSN depends on how efficiently SSN discovers neighbours and how efficiently transfers data between mobile devices. In this section we evaluate these two aspects of SSN. First, a neighbour discovery is an important task for routing protocols. Especially, for delay-tolerant applications an efficient neighbour discovery is key parameter to delivery contents on time. As discussed in Section 2.3.2, SSN uses Dynamic Neighbour Discover method to find neighbouring devices. Thus,



(a) Neighbor Discovery Latency (b) Round Trip Time

Fig. 7 Efficiency of Spontaneous Networking

after joining spontaneous network group, each node generates UDP beacon message to indicate its existence. We evaluate neighbour discovery efficiency by the discovering latency. Neighbour discovery latency of SSN is composed of latency on initiating or joint network and latency on receiving/processing neighbour beacons. Since nodes send neighbour beacons periodically, they arrival time at landmark largely influences on the discovery latency as well. In Fig. 7a, neighbour discovery latency is estimated according to varying beacon intervals using 100 samples. With increasing beacon intervals neighbour discovery interval increases due to the late reception of neighbour beacons. At the worst case beacon exchange latency is equal to beaconing interval. Note that beacon exchange includes waiting interval until next beacon after arriving at landmark in addition to delay of beacon processing. Here, the device is assumed as spontaneous client and joins spontaneous network. Therefore, scanning neighbouring APs, enabling client mode, and joining network are inevitable steps. Delay of these steps has very small variation, and it is not influenced by beaconing interval.

Next, we show networking efficiency of SSN in terms of Round Trip Time (RTT), packet loss, and throughput. We developed a ping application, and estimated RTT between devices using diverse communication types. We have used Nexus One as mobile device and 802.11n base station as an access point (AP). In this analysis, we have found out that the RTT varies in relation to distance between devices because signal attenuation is different. Our experiment carried out when devices were non-line of sight and there about 10 other APs that interfere communication. Distance between devices is 10 meters for the estimated packet loss ratio, which is ratio of lost packets to packets sent, and throughput. Results indicate that SSN has advantage over Wi-Fi ad hoc mode and Bluetooth communication both in terms of throughput and packet loss rate.

11 Conclusion

In this paper, we present the design of SSN, whose major component is spontaneous networking. Applications built on top of SSN use spontaneous networking to exchange data seamlesslySimilar works require intrusion of architecture permissions of smartphones required frustrating user intervention. Our goal was to avoid such tedious steps for users and implement practical system on commercial smartphones. We have implemented SSN on off the shelf Android smartphones and verified unsupervised operation correct. In our further work, we are trying to enhance SSN with richer communication context e.g., utilizing more sensors of smartphones for capability of location prediction [10].

References

1. Wu, J., Lu,M., and Li, F., Utility-based opportunistic routing in multi-hop wireless networks, in ICDCS ’08: Proceedings of the 28th International Conference on Distributed Computing Systems. Washington, DC, USA: IEEE Computer Society, 2008,pp. 470–477.

2. Lindgren, A., Hui, P., The quest for a killer app for opportunistic and delay tolerant networks: (invited paper), in CHANTS '09: Proceedings of the 4th ACM workshop on Challenged networks. New York, NY, USA, pp. 59-66.

3. Schildt, S., Morgenroth, J., Pöttner, W.B., and Wolf, L., IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation, in Electronic Communications of the EASST, vol. 37, pages 1-11, January 2011.

4. Jung, S., Lee, U., Chang, A., Cho, D.,and Gerla, M.,BlueTorrent: Cooperative content sharingfor Bluetooth users, in Proceedings of Pervasive Mobile Computing, White Plains, USA, March 2007, pp. 609-634.

5. Meroni, P., Pagani, E., Rossi, G. P., and Valerio, L., An opportunistic platform for Android-based mobile devices, in MobiOpp '10: Proceedings of the Second International Workshop on Mobile Opportunistic Networking. ACM, New York, NY, USA, pp. 191-193.

6. Su, J., Scott, J., Hui, P., Crowcroft, J., Lara, E.,Diot, C., Goel, A., Lim, M., and Upton, E., Haggle: Seamless networking for mobile applications, in Proceedings of 9th International Conference on Ubiquitous Computing, Springer, Innsburg, Austria, September 2007, pp. 391–408.

7. Moghadam, A., Srinivasan, S., and Schulzrinne, H., 7DS - A modular platform to develop mobile disruption-tolerant applications, in Proceedings of the Second IEEE International Conference on Next Generation Mobile Applications, Services, and Technologies (NGMAST), IEEE Computer Society, September 2008, Cardiff, Wales, UK.

8. Karlsson, G., Lenders, V., and May, M., Delay-tolerant broadcasting, in CHANTS'06: Proceedings of ACM SIGCOMM, Challenged Networks Workshop, Pisa, Italy, September 2006.

9. Lenders, V., May M., and Karlsson, G., Wireless ad hoc podcasting, in Proceedings of Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), San Diego, CA, June 2007.

10. Ma, Y., Hankins, R., and Racz, D., iLoc: A Framework for Incremental Location-State Acquisition and Prediction Based on Mobile Sensors, in Proceedings of 18th ACM Conf. Information and Knowledge Management (CIKM), ACM, 2009, pp. 1367–1376.

11. Google Inc. Google projects for android. http://www.android.com, access date: April, 2011.

12. Apple Inc., Apple iOS Developer Center. http://developer.apple.com, access date: April, 2011.

13. Broadcom Low-Power 802.11n with Bluetooth Chipset, BCM4329, http://www.broadcom.com/, access date: December 2011.

14. Rivest, R., The MD5 Message-Digest Algorithm, RFC1321, April 1992.

15. Vahdat, A., and Becker, D., Epidemic routing for partially connected ad hoc networks, Duke University, Durham, NC, Technical Report CS-200006, April 2000.

16. Vahdat, A., and Becker, D., Epidemic routing for partially connected ad hoc networks, Duke University, Durham, NC, Technical Report CS-200006, April 2000.

17. Interplanetary Overlay Network Project, https://ion.ocp.ohiou.edu/, access date: February 2011.

18. Hyyryläinen, T., Kärkkäinen, T., Luo, C.,Jaspertas, V., Karvo, J., Ott, J., Opportunistic Email Distribution and Access in Challenged Heterogeneous Environments, in Proceedings of ACM SIGCOMM Workshop on Challenged Networks, Montreal, September 2007.

19. ScottK., Burleigh, S., Bundle Protocol Specification, RFC5050, November 2007.

20. Yoneki, E., Hui, P., Chan, S., and Crowcroft, J., A socio-aware overlay for publish/subscribe communication in delay tolerant networks, in MSWiM '07:Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems, Crete Island,Greece, 2007.

21. Ramasubramanian V.,Rodeheffer, T., L.,Terry, D. B.,Walraed-Sullivan M., Wobber, T.,Marshall, C. C., and Vahdat, A., Cimbiosys: a platform for content-based partial replication, in Proceedings of USENIX Symposium on Networked Systems Design and Implementation, Boston, Massachusetts, 2009.

22. Helgason, O. R., Yavuz A. E., Kouyoumdjieva, T. S, Pajevic, L., and Karlsson, G., A mobile peer-to-peer system for opportunistic content-centric networking, in Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds. ACM, New York, NY, USA, pp. 21-26.