Data Pre-Forwarding for Opportunistic Data Collection in Wireless Sensor Networks

XIUCHAO WU, KENNETH N. BROWN, and CORMAC J. SREENAN,
University College Cork, Ireland

Opportunistic data collection in wireless sensor networks uses passing smartphones to collect data from sensor nodes, thus avoiding the cost of multiple static sink nodes. Based on the observed mobility patterns of smartphone users, sensor data should be preforwarded to the nodes that are visited more frequently with the aim of improving network throughput. In this article, we construct a formal network model and an associated theoretical optimization problem to maximize the throughput subject to energy constraints of sensor nodes. Since a centralized controller is not available in opportunistic data collection, data pre-forwarding (DPF) must operate as a distributed mechanism in which each node decides when and where to forward data based on local information. Hence, we develop a simple distributed DPF mechanism with two heuristic algorithms, implement this proposal in Contiki-OS, and evaluate it thoroughly. We demonstrate empirically, in simulations, that our approach is close to the optimal solution obtained by a centralized algorithm. We also demonstrate that this approach performs well in scenarios based on real mobility traces of smartphone users. Finally, we evaluate our proposal on a small laboratory testbed, demonstrating that the distributed DPF mechanism with heuristic algorithms performs as predicted by simulations, and thus that it is a viable technique for opportunistic data collection through smartphones.

Categories and Subject Descriptors: C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network architecture and design*; C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Network protocols*

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless sensor network, opportunistic data collection, routing, data pre-forwarding, human mobility, smartphone

ACM Reference Format:
Xiuchao Wu, Kenneth N. Brown, and Cormac J. Sreenan. 2014. Data pre-forwarding for opportunistic data collection in wireless sensor networks. ACM Trans. Sensor Netw. 11, 1, Article 8 (June 2014), 33 pages.
DOI: http://dx.doi.org/10.1145/2629369

1. INTRODUCTION

As wireless sensor networks mature, we expect to see long-term deployments for applications such as environmental monitoring, domestic utility meter reading, and structural health monitoring. To achieve the expected long network life, these sensor nodes must be aggressively duty cycled. For example, millions of water meters will be installed across the Republic of Ireland in the near future, and many air quality monitoring

Preliminary results were presented at the 9th International Conference on Networked Sensing Systems (INSS 2012) [Wu et al. 2012].
This work was supported in part by the Irish HEA PRTLI-IV NEMBES Grant and the CTVR Grant (SFI 10/CE/I 1853) from Science Foundation Ireland.
Authors' addresses: X. Wu, K. N. Brown, and C. J. Sreenan, Department of Computer Science, University College Cork, Western Road, Cork City, Ireland; email: {xw2, k.brown, cjs}@cs.ucc.ie.

DOI: http://dx.doi.org/10.1145/2629369

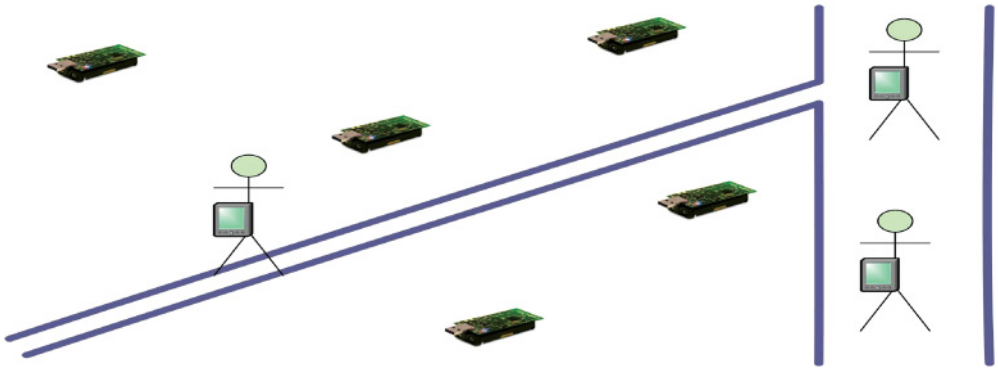


Fig. 1. Opportunistic data collection with smartphones.

systems will be deployed in large cities of Europe to satisfy EU regulations. These applications typically involve large numbers of (static) sensor nodes sparsely deployed in a large area, and their sensor reports are inherently delay tolerant, since the response (if any) requires human intervention over long time scales. For example, analysis of environmental monitoring data is rarely urgent, and meter readings for billing purposes can be delayed by weeks. Due to environmental constraints and/or cost issues, these sparse wireless sensor networks also tend to be partitioned, and deploying large numbers of static sink nodes for collecting sensor data[1] from these sensor nodes would incur prohibitive costs in terms of deployment, maintenance, and data back-haul.

In Li et al. [2011], Liu et al. [2009], Pasztor et al. [2007], Shah et al. [2003], Somasundara et al. [2006], Wang et al. [2005], and Zhao and Yang [2009], the use of resource-rich mobile nodes (mobile sinks or mobile relays) was proposed to move around in the deployed area and collect data from sensor nodes. Depending on the application, the mobile nodes can be either part of the external environment or part of the network, and their mobility can be either controlled or not. In this article, we assume that mobility is not controlled and thus sensor reports are collected opportunistically. Mobile nodes could be specific devices carried by scientists or animals who move around the deployed area for purposes other than data collection. More interestingly, as illustrated in Figure 1, they could also be the increasingly ubiquitous smartphones carried by people who happen to pass through the deployed area in their daily life.

Under this scenario, smartphones will gather data from sensor nodes automatically and incidentally (without any user intervention and route change). To participate in opportunistic data collection, a smartphone user just needs to run a background application on the phone. Here, we assume that the smartphone and sensor node can communicate through some low-power radios (Bluetooth, IEEE 802.15.4, etc.).[2] Consequently, many smartphone users could be motivated with a very low reward, and the total cost of data collection can be reduced significantly through exploiting their uncontrolled mobility. The feasibility, incentive, security, and privacy issues that arise in opportunistic data collection with smartphones have been discussed in Wu et al. [2011c, 2013], and these topics are beyond the scope of this article.

[1]Sensor report and sensor data are used interchangeably in this article.

[2]Bluetooth is distributed with almost all smartphones and is also adopted by many sensor nodes, such as IMote and BTnode. IEEE 802.15.4 is the most widely used radio on sensor nodes, and it has started to appear on smartphones. In Mobile World Congress 2012, TazTag released the first smartphone with both ZigBee (IEEE 802.15.4, protocol stack, etc.) and Near Field Communication (NFC) features (http://www.taztag.com/).

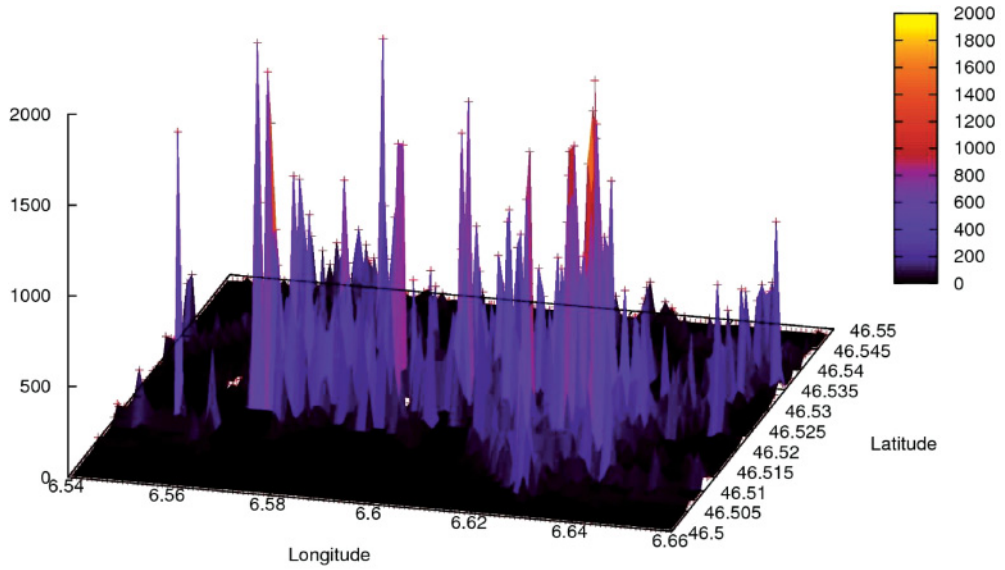


Fig. 2. Spatial distribution of smartphone users' mobility in Lausanne, Switzerland.

Apart from reducing the cost significantly, opportunistic data collection with smartphones could also have some of the benefits of adopting mobile sinks, such as the increased network reliability through removing the dependency on static sink nodes and the extended network lifetime through removing hotspots near the static sink nodes [Shah et al. 2003; Somasundara et al. 2006]. Hence, although data delivery latency could be long in opportunistic data collection with smartphones, there are many promising wireless sensor network applications that are delay tolerant, and it is worthwhile to improve the performance of opportunistic data collection.

However, to exploit smartphone users' mobility for opportunistic data collection, there are still many challenges to be solved, especially when sensor nodes need to be aggressively duty cycled for long life (e.g., the duty cycle of a sensor node is lower than 0.1%). The combination of deep duty-cycling and uncontrolled mobility makes contact probing a challenging task in opportunistic data collection with smartphones. In Wu et al. [2011a, 2011b], it has been reported that the duty-cycled sensor nodes should be responsible for broadcasting beacons, and the *temporal locality* of human mobility should be learned and exploited when determining the frequency of broadcasting beacons. In this article, we focus on the challenges brought by the strong *spatial locality* of human mobility, which has been identified in Gonzalez et al. [2008] and Kaltenbrunner et al. [2010]. Based on smartphone users' mobility traces from the Nokia Mobile Data Challenge [Laurila et al. 2012; Kiukkonen et al. 2010], we have further validated the strong spatial locality in a more appropriate spatial granularity. More specifically, the Lausanne urban area of Switzerland is first divided into small squares whose size is close to the communication range of IEEE 802.15.4 radio, and the spatial distribution of these smartphones' GPS readings is then plotted in Figure 2. This plot clearly indicates that smartphone users visit locations with different frequencies in their daily life—that is, the strong spatial locality exists in the mobility of smartphone users.

Due to the spatial locality of human mobility, sensor nodes deployed at different locations tend to be visited by people with different probabilities; thus, they have different *contact capacities*—the amount of data that could be collected from a sensor node directly by mobile nodes. In this article, a mobile node is used to refer to a smartphone

and its user. Furthermore, even in sparsely deployed wireless sensor networks, there may exist many connected components, and sensor reports could flow among nodes that belong to the same connected component, such as water meters deployed within the same city block. In addition, sensor nodes that are deployed in the same region by different owners could also communicate and collaborate with each other (assuming sufficient security safeguards) to improve the overall performance of opportunistic data collection through smartphones [Wu et al. 2011c]. Therefore, to improve network throughput (i.e., the total amount of sensor data collected by smartphones), sensor nodes that are seldom visited or are never visited should send their sensor reports to other nodes with higher contact capacity.

In this article, data pre-forwarding (DPF), in which the sensor reports are forwarded to sensor nodes with higher contact capacity even though they may not be currently being visited, is proposed for improving the performance of opportunistic data collection with smartphones. We discuss the motivation, present the network model, and formulate DPF as an optimization problem—that is, how to route sensor reports among sensor nodes for maximizing the total amount of collected data under the energy constraints of these sensor nodes. A centralized algorithm is then developed to produce the near-optimal DPF plan for all sensor nodes. Due to the infeasibility of the centralized algorithm in opportunistic data collection, a simple distributed DPF mechanism is proposed so that sensor nodes can rendezvous and communicate with each other energy efficiently. We have also designed two heuristic algorithms with which a sensor node produces its DPF plan based on the states of its direct neighbors, such as the buffer size, the current number of sensor reports in the buffer, the remaining energy resource, the rate of generating sensor data, the contact capacity, and so forth. The distributed DPF mechanism and these heuristic algorithms have been implemented in Contiki-OS [Dunkels et al. 2004] and evaluated with COOJA [Osterlind et al. 2007] under various scenarios. Simulation results indicate that the distributed DPF mechanism with these heuristic algorithms significantly improves on the default case (with no DPF), increasing the number of collected sensor reports by up to 155%. In most cases, these heuristic algorithms achieve at least 95% of the performance of the centralized algorithm. Furthermore, we evaluate these heuristic algorithms on simulated deployments based on GPS traces of smartphones from the Nokia Mobile Data Challenge, and the results indicate that our proposals do work well for these realistic mobility patterns. Finally, we validate our simulation results by studying these proposals on a small laboratory testbed, and we demonstrate similar performance gains.

The article is organized as follows. Section 2 discusses the motivation for DPF, describes the network model, formulates DPF as an optimization problem, and presents the centralized algorithm. Section 3 then gives the details of the distributed DPF mechanism and the heuristic algorithms. After that, several important issues are discussed in Section 4. Evaluation results are presented and analyzed in Section 5. Finally, Section 6 discusses related work, and Section 7 concludes this article.

2. MOTIVATION, PROBLEM FORMULATION, AND THE CENTRALIZED ALGORITHM

2.1. Motivation for Data Pre-Forwarding

To route data to sensor nodes with higher contact capacity, the obvious way is to let a mobile node (i.e., a smartphone and its user) collect data by establishing multihop communication when it is visiting the deployed area. When a sensor node is being visited and all of its sensor reports have been collected, the mobile node could then collect sensor reports from other nodes through this sensor node. However, it becomes unsuitable in the context of opportunistic data collection with smartphones, because mobile nodes are uncontrolled and sensor nodes must be aggressively duty cycled.

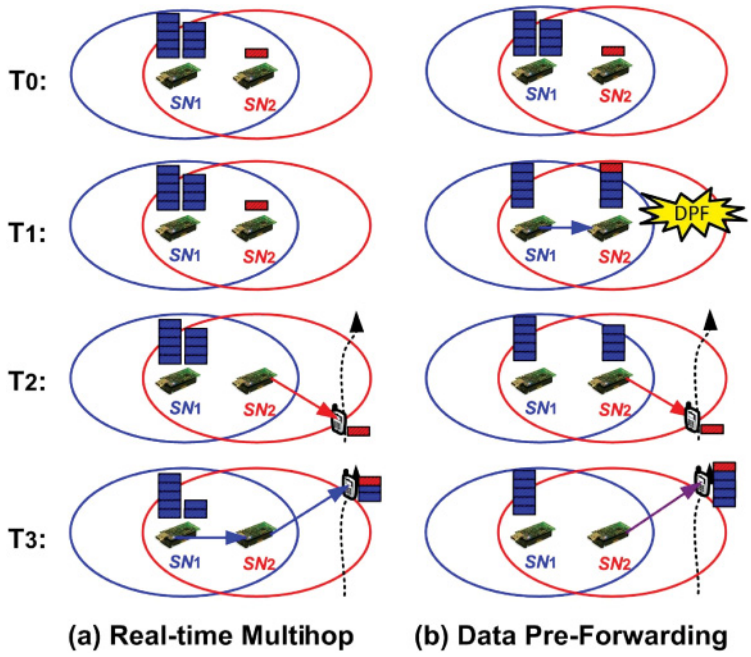


Fig. 3. Benefits of DPF.

First, due to the uncontrolled mobility of smartphone users, *contact length*—the period that a mobile node can communicate with a sensor node—could be as short as a few seconds. Considering that sensor nodes also need to be deeply duty cycled for longevity, during the period that a sensor node is visited, its neighbors may not wake up and their sensor reports cannot be collected through multihop communication.

Second, even when multihop communication could be used, the sensor node that is currently visited must also use the same wireless channel for both uploading sensor reports to the mobile node and receiving sensor reports from its neighbors. Here, we assume that multiple wireless channels cannot be used simultaneously by a sensor node with a simple radio. Thus, the precious opportunity of uploading data to the mobile node cannot be fully exploited. As illustrated in Figure 3, if sensor node 1 could forward sensor reports before sensor node 2 is visited, the amount of collected data could be increased significantly. Hence, DPF, in which the sensor reports are forwarded to sensor nodes with higher contact capacity even though they are not currently being visited, should be a promising scheme for improving the performance of opportunistic data collection with smartphones—that is, the total amount of sensor data collected by smartphones.

Third, DPF could decouple the data collection (between mobile node and sensor node) with data forwarding (among sensor nodes). Thus, in case energy harvesting is used by sensor nodes, data forwarding can be carried out at a time of day when energy supply is abundant and mobile nodes are absent.

2.2. Network Model

In the context of DPF for opportunistic data collection, we assume that the wireless sensor network is connected. This is reasonable because if the network is partitioned, we can treat each network partition separately as one independent wireless sensor network. We also assume that the expected applications of opportunistic data collection

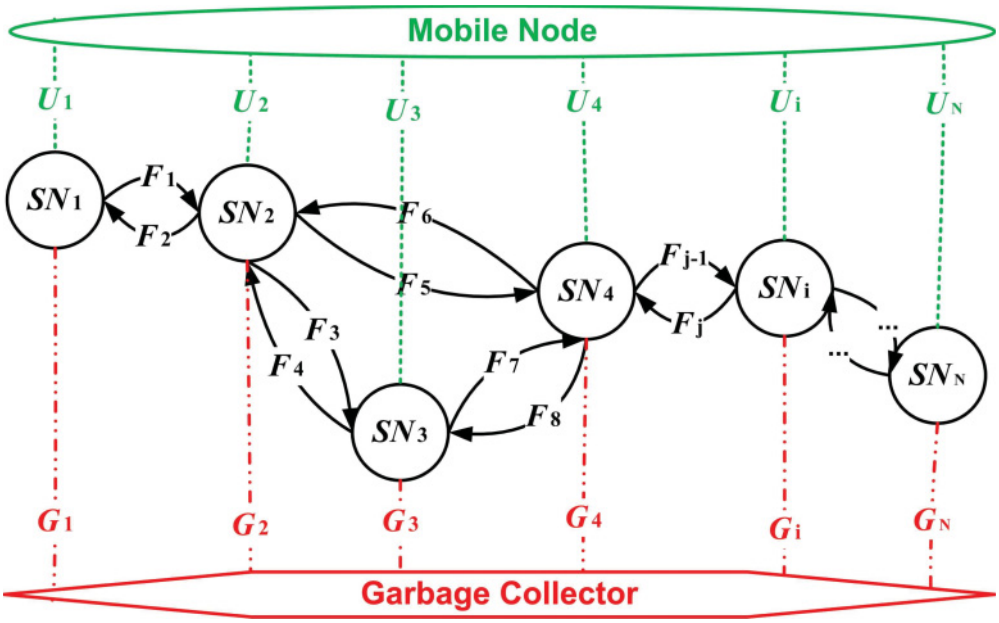


Fig. 4. Network model.

are low-data-rate applications. Otherwise, it would be better to deploy multiple static sink nodes for collecting the huge amount of sensor reports generated by sensor nodes continuously. For instance, one water meter reading per month is enough for billing purposes, and one reading per hour could be enough for time-based pricing schemes. Finally, we assume that sensor nodes carry out DPF periodically since human mobility tends to follow some repeated patterns [Gonzalez et al. 2008; Kaltenbrunner et al. 2010; Wu et al. 2013]. As for T_{dpf} (the interval between two consecutive runs of DPF), it should be a multiple of Γ (the epoch length of the repeated human mobility pattern) for avoiding unnecessary data exchange caused by the bursty contact arrivals within an epoch. Considering that Γ is normally measured in hours or days, we set T_{dpf} to Γ, and this value should be large enough to maintain a low overhead of DPF.

The network model is illustrated in Figure 4. We use one virtual mobile node to represent all mobile nodes, and we introduce one virtual garbage collector that receives the sensor reports dropped by all sensor nodes. For simplicity, we assume that the excess sensor reports of a sensor node are dropped when its buffer overflows. In real deployments, a sensor node may aggregate the excess sensor reports if it is allowed by the deployment scenario (sensor reports are not encrypted, sensor nodes belong to the same owner, etc.). The choice of which reports to drop or aggregate may be based on various criteria, such as the source, the creation time, and/or the value in a sensor report. These issues are beyond the scope of this article.

The wireless sensor network is modeled as a directed graph, $\widehat{G}(V, E)$. V is the set of N sensor nodes, and E is the set of M directed links that connect these sensor nodes. If node a can receive data from node b, there will be a link that starts from node b and ends at node a. Hence, the topology of a wireless sensor network can be described by the following $N * M$ matrix.

$$A_{ij} == \begin{cases} 1 & \text{node } i \text{ is the source of link } j, \\ -1 & \text{node } i \text{ is the destination of link } j, \\ 0 & \text{otherwise.} \end{cases}$$

Table I. Notations Used in the Network Model and the Optimization-Based DPF Problem

Notation	Comments
Γ	Epoch length of the repeated human mobility pattern.
T_{dpf}	Interval between two consecutive runs of DPF. It is set to Γ in this article.
R_i	Number of sensor reports generated by node i in an epoch.
ζ_i	Contact capacity of node i in an epoch. It is the maximal number of sensor reports that could be collected directly from this node. Note that it is determined by the location of node i and the mobility patterns of smartphones.
U_i	Number of sensor reports collected directly from node i in an epoch.
G_i	Number of sensor reports dropped by node i in an epoch.
F_j	Number of sensor reports that pass through link j in an epoch.
A_{ij}	Relationship between node i and link j.
Φ_i	Energy budget of node i in an epoch. It is the maximal number of sensor reports that this node can upload/receive/forward.

For each sensor node i, R_i denotes its data rate—that is, the number of sensor reports generated by this node in an epoch. ζ_i denotes its contact capacity—that is, the maximal number of sensor reports that could be collected directly from this node in an epoch by smartphones. ζ_i is determined by the location of sensor node i and the mobility patterns of smartphones. U_i denotes the number of sensor reports that are collected directly from sensor node i in an epoch, and G_i denotes the number of sensor reports dropped by this node in an epoch. For each link j, F_j denotes the number of sensor reports passed through link j in an epoch. Table I lists the notations used in the network model and in the following optimization-based DPF problem.

In this model, the bandwidth of a wireless link is not modeled explicitly based on the following observations. First, ζ_i, the contact capacity in this model, is the maximal number of sensor reports that could be collected directly from sensor node i during one epoch. Hence, it has modeled both the link bandwidth and the intermittent connections between sensor nodes and smartphones. Second, due to the low data rate of the expected applications and the small number of neighbors that each sensor node has in a sparsely deployed wireless sensor network, the amount of sensor reports to be exchanged between one sensor node and its neighbors is limited. Since sensor nodes are static, a sensor node is within the communication range of its neighbors in all time. Due to the long epoch of human mobility, we can assume that there is enough time for neighboring sensor nodes to exchange all necessary sensor reports. Thus, the link bandwidth is not modeled explicitly in this article for simplicity.

Furthermore, we assume that each sensor node has enough storage space to hold all sensor reports received from its neighbors during one epoch, because the mainstream sensor node platform is normally distributed with a large external flash memory [Polastre et al. 2005]. Without this assumption, we must model the constraint that the storage space imposes on the maximal amount of sensor data that a sensor node could receive from its neighbors. The effects of storage space also depend on the temporal order among the events of uploading to smartphones, forwarding to neighbors, and receiving from neighbors. It is very hard and complex to model this kind of constraints.

Note that the preceding simplifications are just used to control the complexity of the model and make sure that the following optimization problem can be solved. When designing and evaluating our proposals in Sections 3 and 5, both the link bandwidth and the storage space (buffer size) have been considered.

2.3. DPF Problem Formulation

With the network model stated previously, the task faced by DPF is to determine the value of F_j for each link j under necessary constraints to improve the performance of

opportunistic data collection. Since the expected applications are delay tolerant and sensor nodes have limited energy supply, we will model DPF as an optimization problem specifically to maximize the overall network throughput (the total number of sensor reports collected from all sensor nodes) under the energy constraints of these nodes. For each sensor node i, Φ_i is used here to denote the maximal amount of energy that could be consumed for data communication in an epoch. More specifically, the total number of sensor reports to be uploaded, received, or forwarded in an epoch should be less than Φ_i. We argue that this constraint is reasonable. For instance, when a sensor node is battery powered, this constraint could be used to ensure the minimal lifetime of the sensor node. In case a sensor node is installed with some energy-harvesting modules (e.g., solar panel), this constraint could be used to ensure that the consumed energy is less than the energy that could be harvested from the environment in one epoch.

In this optimization-based DPF problem, the function to be maximized is $\sum_{i=1}^{i=N} U_i$, and the following constraints must be satisfied:

$$\begin{aligned} &\text{for each link } j, && F_j \geq 0 \\ &\text{for each node } i, && U_i \geq 0 \text{ and } G_i \geq 0. \end{aligned}$$

These constraints state that all variables must be no less than 0. For each sensor node i, the following constraints must also be satisfied:

$$U_i \leq \zeta_i \tag{1}$$

$$U_i + G_i + \sum_{j=1}^{j=M} A_{ij} * F_j = R_i \tag{2}$$

$$U_i + \sum_{j=1}^{j=M} |A_{ij}| * F_j \leq \Phi_i. \tag{3}$$

More specifically, for each sensor node i, the sensor reports collected directly from it must be less than its contact capacity (Equation 1). The sum of the sensor reports generated by node i and the sensor reports received from its neighbors must be equal to the sum of the sensor reports uploaded to mobile node, the sensor reports dropped by itself, and the sensor reports forwarded to its neighbors (Equation 2). The total amount of data communications must also satisfy energy constraint (Equation 3). For simplicity, our model does not differentiate the energy cost for uploading/receiving/forwarding, because the low-power radio of the mainstream sensor node platforms consumes almost the same amount of energy in transmitting and receiving/listening modes [Polastre et al. 2005; Nordic Semiconductor 2007].

2.4. The Centralized Algorithm

In this subsection, a centralized algorithm is developed to produce the near-optimal DPF plan through solving the optimization-based DPF problem presented earlier. We assume that a centralized controller exists and that it knows the topology of the wireless sensor network (A_{N*M}) and the contact capacity of each sensor node (ζ_i). It also knows the number of sensor reports generated by each sensor node (R_i) and its energy budget for data communication (Φ_i).

Note that U_i, G_i, and F_j in the earlier DPF problem must be integers, considering that a sensor report cannot be transmitted partially. Hence, the DPF problem is an integer programming problem. However, the general integer programming problem is NP-hard, and we cannot hope to compute the optimal DPF plan efficiently. We solve instead the *linear program relaxation* of the DPF problem in Section 2.3. In other words, we allow U_i, G_i, and F_j to assume real values. The DPF problem now becomes

a linear programming problem, and there exist several efficient methods for solving it, such as the simplex algorithm that could produce the solution in polynomial time. In the centralized algorithm, we then apply the *randomized rounding* technique on the optimal solution of this linear program relaxation to obtain a feasible solution that does not drift too far from the optimal solution of the DPF problem [Motwani and Raghavan 1995].

Although this centralized algorithm could produce the near-optimal solution efficiently, it is not practical for opportunistic data collection with smartphones, as there is no good candidate to act as the centralized controller. First, a sensor node cannot act as the controller due to its limited resources. Sensor nodes do not have the necessary CPU and memory to solve the linear program relaxation of the DPF problem. It is also very slow and expensive to collect the necessary information from the whole wireless sensor network and distribute the generated DPF plan to all sensor nodes. Second, although a smartphone is much more powerful than a sensor node, it still cannot act as the controller due to the uncontrolled mobility of its user. The smartphone may not stay in the deployed area long enough to collect the information of the whole network, produce the solution, and distribute the near-optimal DPF plan to all sensor nodes. Although some special nodes may be dispatched to the deployed area to act as the controllers, this will increase the operational costs and contradict with the fundamentals of opportunistic data collection with smartphones.

Thus, the centralized algorithm cannot be applied in opportunistic data collection. The following section will focus on a distributed DPF mechanism with heuristic algorithms in which sensor nodes make local decisions based on different levels of knowledge about their neighbors. The results of the linear program relaxation of the DPF problem will be used to evaluate the performance of these heuristic algorithms.

3. THE DISTRIBUTED DPF MECHANISM AND HEURISTIC ALGORITHMS

To improve network throughput of opportunistic data collection through DPF, sensor nodes need not only upload sensor reports to mobile nodes but must also exchange states and sensor reports among each other. In this section, we first discuss the challenges to be solved for carrying out DPF energy efficiently. We then describe how a sensor node in this distributed DPF mechanism could rendezvous and exchange sensor reports with both mobile nodes and its neighbors energy efficiently.

3.1. Challenges and Overview of the Distributed DPF Mechanism

To carry out DPF, the straightforward way is to periodically wake up all sensor nodes so that they can exchange node states and sensor reports among each other. However, this scheme becomes infeasible in opportunistic data collection, because there is no static sink node and sensor nodes must be deeply duty cycled.

First, it is very challenging to wake up all sensor nodes energy efficiently. We may first synchronize the clock of all sensor nodes and let them wake up at the same time point. However, when there is no GPS on sensor nodes and there is no static sink node in the network, the protocols for multihop time synchronization could become quite complex and the overhead can be excessive [Maroti et al. 2004; Sommer and Wattenhofer 2009]. It is also not suitable to let mobile nodes provide time services, as some sensor nodes may not be visited at all. In addition, clock drift could be large between two consecutive runs of DPF, because T_{dpf} is normally measured in hours or days. Hence, sensor nodes need to turn on radios for a longer time to accommodate clock drift. Alternatively, we may wake up sensor nodes hop-by-hop through the scheme proposed in Koala [Musaloiu-E et al. 2008]. However, we need to determine which sensor node should act as the static sink node in Koala. Furthermore, since sensor nodes are deeply duty

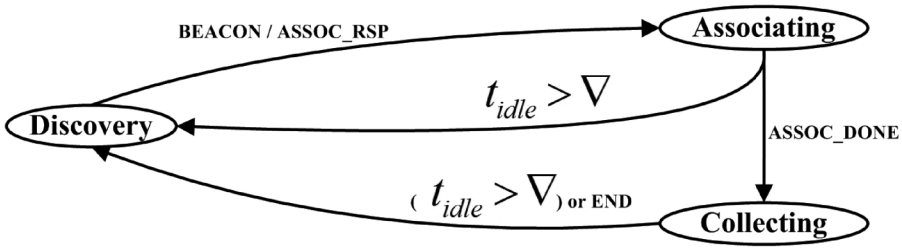


Fig. 5. State transition diagram of a mobile node.

cycled, sensor nodes must keep their radios on for a long time before all nodes have woken up, especially when network diameter is large.

Second, if all sensor nodes wake up simultaneously for exchanging node states and sensor reports, there will be a lot of concurrent communications. If the existing CSMA-based MAC protocols (B-MAC [Polastre et al. 2004], X-MAC [Buettner et al. 2006], Koala [Musaloiu-E et al. 2008], RI-MAC [Sun et al. 2008], etc.) are used, the hidden terminal problem could become severe and the existing solutions (RTS/CTS [Bharghavan et al. 1994], etc.) are not suitable due to their large overhead. Although TDMA-based MAC protocols can be used to avoid concurrent communications, it is not easy or cheap to allocate time slots among sensor nodes, especially when there is no static sink node [Rajendran et al. 2003; Rhee et al. 2004]. Furthermore, as stated previously, clock drift could be large between two consecutive runs of DPF. The guard time between time slots must be large, and sensor nodes must keep their radios on for a longer time.

In summary, if all sensor nodes wake up simultaneously for exchanging node states and sensor reports, it becomes impossible to carry out DPF energy efficiently. In this article, we propose to let each sensor node randomly and independently choose a time at which it initiates DPF for exchanging sensor reports with its direct neighbors. Since the wireless sensor network is sparsely deployed and the interval between two consecutive runs of DPF (T_{dpf}) is huge, it is unlikely that two nodes within a two-hop range will initiate DFP at the same time. Thus, a sensor node that initiates DPF could be in control of the wireless channel. The initiator can then turn on its radio for a long time (T_{rdv}) to receive the beacons broadcasted by all of its neighbors and instruct them to wake up at suitable times for exchanging sensor reports. Note that the beacons emanating from sensor nodes are necessary to energy-efficiently probe contacts with mobile nodes [Wu et al. 2011b], and the beacons can also be used to broadcast node states. Consequently, the initiator can coordinate with its neighbors for utilizing the wireless channel efficiently in the metrics of both channel throughput and sensor node energy consumption.

In opportunistic data collection with smartphones, a sensor node needs to communicate with both mobile nodes and the neighboring sensor nodes. We have studied the interaction between a sensor node and a mobile node in Wu et al. [2011a, 2011b], and the same protocols are used in this work to carry out contact probing and data uploading. More specifically, the duty-cycled sensor nodes will broadcast beacons, and the temporal locality of human mobility will be exploited when a sensor node determines the frequency of beacon broadcasting. The beacons broadcast by sensor nodes are exploited further in this work to rendezvous and exchange states among sensor nodes. Following the state transition diagram in Figures 5 and 6, we will present the details for mobile node and sensor node, respectively. At the end of this section, we will also introduce two heuristic algorithms designed for improving the total number of sensor reports collected from all sensor nodes through DPF. These heuristic algorithms enable a sensor node to decide the number of sensor reports exchanged with

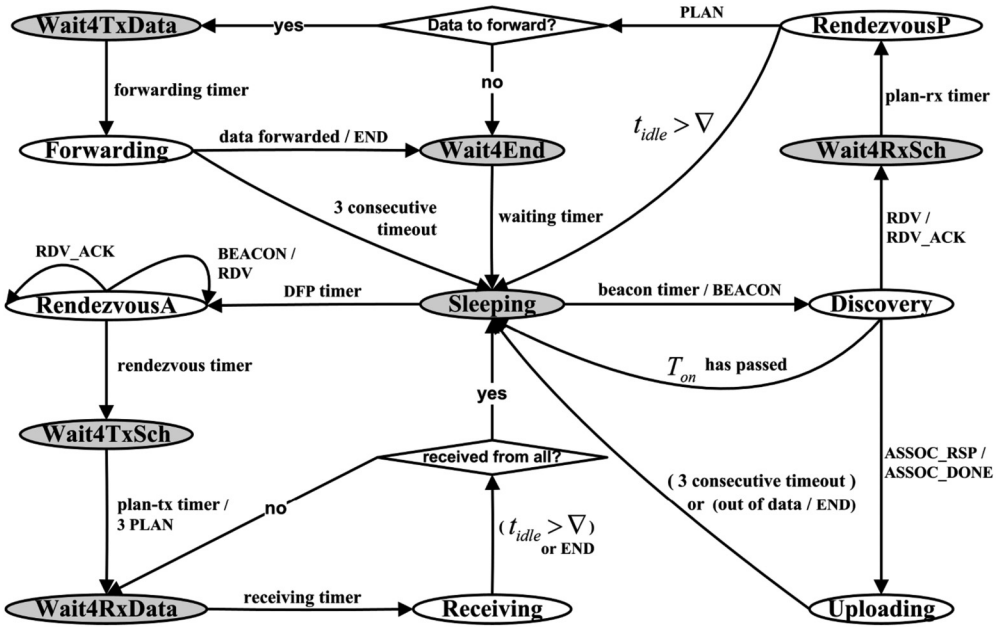


Fig. 6. Transition diagram of sensor node (radio is off in the shadowed states for saving energy).

each neighbor based on different levels of knowledge about its neighbors. Table II first lists the messages and notations used to describe the details of both mobile node and sensor node.

3.2. Mobile Node

Figure 5 shows the simple state transition diagram of a mobile node, which moves around in an uncontrolled manner; its radio is always turned on so that it can be discovered. This design is feasible, as a smartphone is normally equipped with relatively abundant and rechargeable power supply [Wu et al. 2011a].

When a mobile node is booting up, it first turns on its radio and enters into Discovery state. After receiving a BEACON from a sensor node, a mobile node will send back ASSOC_RSP and enter into Associating state. After receiving ASSOC_DONE from the sensor node, the association is completed. The mobile node will enter into Collecting state and start to collect data from the sensor node.

In Collecting state, the mobile node acts as the data receiver. When one data packet containing one or multiple sensor reports is received successfully, the mobile node will immediately send one acknowledgement to the sensor node. After all sensor reports have been received by the mobile node, data collection will be terminated by the sensor node through sending END to the mobile node. After receiving END, the mobile node will return back to Discovery state.

In both Associating and Collecting states, the mobile node also keeps monitoring whether it has moved away from the sensor node. When it finds that t_{idle} (the time that the channel is idle) is larger than a constant (∇), the mobile node returns back to Discovery state and is ready to be discovered again. Thus, ∇ should be long enough to avoid that the mobile node enters into Discovery state and terminates data collection unnecessarily. In this work, ∇ is currently set to 50ms, and this value is long enough to (re)transmit a packet for a few times through the wireless channel.

Table II. Messages and Notations Used in the Distributed DPF Mechanism

	Comments
BEACON	Broadcasted by a sensor node for contact probing and rendezvous with neighbors. It contains the current states of this node, such as buffer size, data rate, and so forth.
ASSOC_RSP	Sent by a mobile node after receiving a BEACON from a sensor node.
ASSOC_DONE	Sent by a sensor node after receiving ASSOC_RSP from mobile node. It indicates the completion of a successful contact probing.
END	Sent by a sensor node to notify a mobile node that all of its sensor reports have been uploaded to this mobile node.
RDV	Sent by the initiator, the sensor node initiating DPF, after receiving a BEACON from a neighbor. This message contains the time that PLAN will be broadcasted.
RDV_ACK	Sent by a neighbor after receiving a RDV from the initiator. It contains the signal-to-noise ratio (SNR) experienced by the RDV. Thus, the initiator can remove this neighbor if the link between them is too weak.
PLAN	Broadcast by the initiator after the DPF plan has been produced based on local information. For each neighbor, it specifies when and how many sensor reports this neighbor should forward to the initiator. It also contains the time that all neighbors should have completed data forwarding.
T_{on}	Period that the radio is turned on for broadcasting BEACON and receiving the potential response (ASSOC_RSP or RDV). It is set to 20ms in this article.
T_{off}	Period that the radio is turned off for maintaining a low duty cycle.
T_b	Interval between two consecutive beacons when a sensor node carries out contact probing. Note that $T_b = T_{on} + T_{off}$.
T_{rto}	Period used by the retransmission timer of a data sender for retransmitting the corrupted packets. It is set to 20ms in this article.
∇	Threshold used by a receiver to detect whether the communication opportunity has been terminated due to mobility, node death, and so forth. It is set to 50ms in this article.
T_{rdv}	Period that the initiator keeps its radio on for rendezvousing with all neighbors of this initiator. T_{rdv} should be larger than the maximal possible value of T_b.
T_{plan}	Interval between the beginning of DPF and the time to broadcast PLAN. Note that $T_{plan} > T_{rdv}$ and the difference should be large enough to produce a DPF plan based on some heuristic algorithms.

3.3. Sensor Node

Figure 6 shows the state transition diagram of a sensor node. This figure is quite complex, as a sensor node must be deeply duty cycled and also must carry out both contact probing and DPF. After a sensor node is booted up, its radio is off and it is in Sleeping state.

To carry out DPF periodically, after booting up, a sensor node needs to determine the time that it will initiate DPF. Thus, the DPF timer is started to initiate DPF after a period that is randomly selected in the range of [0, T_{dpf}]. Here, T_{dpf} is the large interval between two consecutive runs of DPF initiated by this sensor node. As discussed in Section 3.1, through letting sensor nodes randomly and independently select the time, it is unlikely that two sensor nodes within a two-hop range will initiate DPF at the same time. The sensor node that initiates DPF could be in control of the wireless channel, and sensor reports can be exchanged efficiently. Note that for exchanging data among sensor nodes through DPF, a sensor node needs to play two roles at different times. When its DPF timer expires and it is initiating DPF, the sensor node acts as the *initiator* for pulling sensor reports from its neighbors. When one of its neighbors is initiating DPF, it acts as one *neighbor* of that neighbor for sending sensor reports requested by that neighbor.

To probe mobile nodes, after booting up, the beacon timer is also started by a sensor node to broadcast a BEACON after a short period that is randomly selected in the range

of [0, T_b]. Here, T_b is the short interval between two consecutive beacons broadcast by this sensor node. In the following subsections, we will introduce the detailed behaviors of a sensor node for both contact probing and DPF.

3.3.1. Contact Probing. When the beacon timer expires, a sensor node will turn on its radio, send out a BEACON, and enter into Discovery state. To support DPF, the BEACON also contains the current states of this sensor node, such as the size of its buffer, the current number of sensor reports in the buffer, the remaining energy resource, the rate of generating sensor data, the contact capacity, and so forth.

If this node does not receive any response within a short period (T_{on}), it will turn off its radio, return to the Sleeping state, and start its beacon timer with a period (T_{off}) that is much longer than T_{on} for maintaining a low duty cycle. Note that T_{on} and T_{off} determine the interval between two consecutive beacons broadcast by a sensor node, $T_b = T_{on} + T_{off}$. Based on the current sensor node platforms, T_{on} is set to 20ms, which is enough for sending a BEACON and receiving a response from a mobile node or its neighbor. T_{off} may be adjusted to learn and exploit the temporal locality of human mobility [Wu et al. 2011a].

If ASSOC_RSP is received in Discovery state, it will send back ASSOC_DONE, enter into Uploading state, and start to transfer data to the associated mobile node. In Uploading state, the sensor node acts as the data sender. The simple Stop-and-Wait flow control protocol is used between the mobile node and the sensor node. A retransmission timer is also used by the sensor node for handling wireless transmission errors.

More specifically, after the sensor node sends one data packet to the mobile node,[3] it will start the retransmission timer and wait for the corresponding acknowledgement. Here, T_{rto} is the period used by the retransmission timer, and it is set to 20ms based on the current sensor node platforms. If the retransmission timer expires, the data packet will be retransmitted to handle wireless transmission errors. If the sensor node still cannot receive the acknowledgement after three consecutive retransmissions, it is more likely that the mobile node has moved away. Thus, the sensor node will turn off its radio, return back to Sleeping state, and start its beacon timer for the next probing.

If the acknowledgement arrives before the expiration of the retransmission timer, the sensor node will discard the sensor reports that have been uploaded successfully, restart its retransmission timer, and begin to transmit new sensor reports to the mobile node. If all data has been uploaded, the sensor node will send END to the mobile node for terminating this contact. The sensor node will then turn off its radio, return back to Sleeping state, and start its beacon timer for the next probing.

3.3.2. DPF: The Initiator. When a sensor node's DPF timer expires, it first turns on its radio and enters into RendezvousA state. The DPF timer is then restarted with a period (T_{dpf}) for scheduling the next run of DPF. This node will be referred to as the initiator in the following discussion.

The initiator will also start two timers: the rendezvous timer, whose period is T_{rdv}, and the plan-tx timer, whose period is T_{plan}. T_{rdv} is the period that the initiator keeps its radio on for rendezvousing with its neighbors. T_{rdv} should be longer than the largest possible value of T_b so that the initiator could receive a BEACON from each of its neighbors. T_{plan} is the interval between the beginning of this run of DPF and the time to broadcast PLAN, which instructs how the neighbors should forward data to the initiator. For each neighbor, PLAN specifies when and how many sensor reports this neighbor should forward to the initiator. It also contains the time that all neighbors should have completed data forwarding.

[3]Note that multiple sensor reports might be concatenated into one data packet for reducing the overhead of packet headers.

Obviously, T_{plan} must be larger than T_{rdv} so that the DPF plan could be produced before broadcasting PLAN. Since the following heuristic algorithms in Section 3.4 make decisions based on the states of a few direct neighbors of the initiator, the DPF plan can be produced quickly and the difference between T_{plan} and T_{rdv} can be set to a small value.

In RendezvousA state, if a BEACON is received from a neighbor, the initiator will send back a RDV that contains the time that the initiator will broadcast the PLAN. If a RDV_ACK, which contains the SNR experienced by the RDV, is then received from this neighbor, the initiator will decide whether this neighbor should be considered based on this SNR experienced by RDV_ACK and the content of this RDV_ACK. Thus, we can avoid using the asymmetric or weak links to exchange data among sensor nodes. When the rendezvous timer expires, the initiator should have collected the current node states from all of its neighbors, as T_{rdv} is longer than the largest possible value of T_b. It will turn off its radio, enter into Wait4TxSch state, and run one of the heuristic algorithms presented in Section 3.4 to produce the DPF plan.[4]

When the plan-tx timer expires, the initiator will broadcast the just-produced DPF plan to its neighbors through PLAN. This message will be broadcast for three times to be more robust to packet corruption. After that, the initiator will turn off its radio and enter into Wait4RxData state.

When it is the expected time to receive data from a neighbor (the receiving timer expires), the initiator will turn on the radio and enter into the Receiving state. Thus, based on PLAN, the initiator schedules its radio in a TDMA-like manner to receive sensor reports from its neighbors. As for data communication between the initiator and this neighbor, we adopt the same protocols used for data uploading between sensor node and mobile node. Here, the initiator acts as the data receiver, and this neighbor act as the data sender. When the initiator has completed the communications with all neighbors, it will enter into Sleeping state, turn off its radio, and start the beacon timer for carrying out contact probing.

3.3.3. DPF: The Neighbor. As stated previously, after a sensor node broadcasts a BEACON, it enters into Discovery state. If a RDV is received in this state, one of its neighbors must be initiating DPF, and this sensor node should act as one *neighbor* of that neighbor, who is now acting as the initiator. This sensor node will be referred to as the neighbor in the following discussion.

The neighbor will first send back an RDV_ACK, which contains the SNR experienced by the RDV. The neighbor will then turn off its radio and enter into the Wait4RxSch state. It also starts the plan-rx timer based on the information in this RDV—that is, the time that PLAN will be broadcast by the initiator. When the plan-rx timer expires in Wait4RxSch state, the neighbor will turn on its radio and enter into RendezvousP state. If a PLAN does not arrive as expected, the neighbor will turn off its radio, enter into Sleeping state, and start the beacon timer for contact probing.

If a PLAN is received, the neighbor will first check whether it need forward sensor reports to the initiator. If not, the neighbor will turn off its radio, enter into Wait4End state directly, and start a waiting timer based on the information in the PLAN—that is, the time that all neighbors should have completed data forwarding. Otherwise, the neighbor will turn off its radio and enter into Wait4TxData state. The forwarding timer is also started based on the expected time that the neighbor should forward sensor reports. When the forwarding timer expires, it will turn on its radio, enter into

[4]Note that when the rendezvous timer expires, the initiator may not have the states of all neighbors if some BEACONs are corrupted. In this case, the initiator will not wait, as it does not maintain a list of its neighbors. The initiator will make decisions immediately based on the partial knowledge collected so far.

Forwarding state, and start to send sensor reports to the initiator. After the required number of sensor reports have been forwarded, the neighbor will send END, turn off its radio, enter into Wait4End state, and start the waiting timer. When the wait timer expires—that is, all neighbors should have completed data forwarding—the neighbor will turn off its radio, enter into Sleeping state, and start the beacon timer for contact probing. If data forwarding cannot be carried out in rare cases (the energy of the initiator is totally depleted, the initiator is taken away, etc.), the neighbor will turn off its radio, enter into Sleeping state, and start the beacon timer for contact probing.

3.4. Heuristic Algorithms

Inspired by the good properties (the optimal network throughput, etc.) of backpressure-based routing for multihop wireless networks [Tassiulas and Ephremides 1992], two heuristic algorithms are designed here to determine the number of sensor reports that each neighbor should forward. The main philosophy of these algorithms is that if a sensor node is seldom visited, its excess sensor reports will accumulate in the buffer with the elapse of time. Through letting each sensor node pull sensor reports from its neighbors whose buffer has more data, sensor reports will flow to sensor nodes that are visited more frequently.

In the first algorithm (DPF-BUF), a sensor node only has knowledge of its buffer size, the current number of sensor reports in the buffer, and its remaining energy resource. For the second one (DPF-EST), we assume that a sensor node can also estimate its contact capacity and its data rate. The details of the two algorithms are presented next.

3.4.1. DPF-BUF Algorithm. When this algorithm is carried out for producing the DPF plan, the initiator has collected node states from all of its K neighbors through BEACON frames. For each neighbor i, the initiator gets to know B_i (buffer size of this neighbor), D_i (the current number of sensor reports in the buffer), and e_i (the remaining energy resource, or the number of sensor reports that this neighbor still can receive, forward, or upload in the current epoch).

The initiator first calculates its current available buffer space, $\psi' = B' - D'$. For each neighbor i, ψ_i is also calculated in the same way as ψ'. The average of the available buffer space across the initiator and the neighbors whose available buffer space is smaller than ψ' is then calculated as follows:

$$\overline{\psi} = \frac{\psi' + \sum_{\psi_i < \psi'} \psi_i}{1 + \sum_{\psi_i < \psi'} 1}. \tag{4}$$

For each neighbor i, it hopes that ψ_i could be no less than $\overline{\psi}$ after forwarding data to the initiator. The number of sensor reports that it could forward is also constrained by its energy resource. Thus, we can calculate the total number of sensor reports that these neighbors hope to forward ($\widehat{\tau}$) as follows:

$$\widehat{\tau} = \sum_{\psi_i < \overline{\psi}} \tau_i. \text{ Here, } \tau_i = \min((\overline{\psi} - \psi_i), e_i). \tag{5}$$

Following the same principles, the total number of sensor reports that the initiator can receive is $\widehat{\upsilon} = \max(\min(\psi' - \overline{\psi}, \widehat{\tau}, e'), 0)$. As for the number of sensor reports that the initiator will receive from neighbor i, υ_i is calculated as follows:

$$\upsilon_i = \begin{cases} 0 & \psi_i \geq \overline{\psi} \text{ or } \frac{\widehat{\upsilon} * \tau_i}{\widehat{\tau}} < \Delta \\ \frac{\widehat{\upsilon} * \tau_i}{\widehat{\tau}} & otherwise. \end{cases} \tag{6}$$

More specifically, when it is impossible to receive all sensor reports that the neighbors hope to forward, the initiator will receive from its neighbors proportionally. When υ_i is less than a threshold (Δ), neighbor i will not forward sensor reports to reduce the overhead of data transmission. Currently, Δ is set to 4 in the following evaluations based on TELOSB platform [Polastre et al. 2005]. It is the maximal number of sensor reports that a packet can hold when the size of a sensor report is 30 bytes.

3.4.2. DPF-EST Algorithm. Considering that the data rate is predictable in many applications (environmental monitoring, domestic utility meter reading, etc.) and the contact capacity of a sensor node could also be estimated, DPF-EST, another heuristic algorithm, is also designed here with the assumption that a sensor node has the knowledge of the data rate and its contact capacity. The details of DPF-EST are presented next.

When DPF-EST is carried out, the initiator has collected node states from all of its K neighbors through BEACON frames. For each neighbor i, the initiator can get not only B_i, D_i, and e_i but also ζ_i (contact capacity of this neighbor) and R_i (data rate of this neighbor). Instead of ψ, the DPF-EST algorithm works based on χ (the expected available buffer space when the next run of DPF is carried out). More specifically, the initiator first calculates its χ' as follows:

$$\chi' = B' - D' + \zeta' - R'. \tag{7}$$

Note that χ' could be less than 0 and could be larger than B'. For each neighbor i, χ_i is also calculated in the same way as χ'. Since data rate is considered in χ, the sensor nodes with higher data rate can reserve more buffer space for the sensor reports to be generated by itself. Thus, these sensor nodes can avoid unnecessary buffer overflow. Furthermore, the sensor nodes with higher contact capacity will advertise a larger χ since contact capacity is considered. Consequently, these sensor nodes could pull sensor data from their neighbors more quickly.

$\overline{\chi}$, the average of the expected available buffer space across the initiator and the neighbors whose expected available buffer space is smaller than χ', is then calculated in the same way as $\overline{\psi}$ in DPF-BUF algorithm (Equation 4), substituting χ for ψ. After that, we calculate $\widehat{\tau}$ in the same way as the DPF-BUF algorithm (Equation 5) except that $\tau_i = \min((\overline{\chi} - \chi_i), D_i, e_i)$. $\widehat{\upsilon}$ is then calculated as follows:

$$\widehat{\upsilon} = \min((\chi' - \overline{\chi}), (B' - D'), \widehat{\tau}, e'). \tag{8}$$

Finally, υ_i can be calculated in the same way as DPF-BUF algorithm (Equation 6), substituting χ for ψ.

After υ_i is determined for all neighbors through one of the two heuristic algorithms, the initiator can compose PLAN based on the bandwidth of the wireless channel. For each neighbor i, it will allocate enough time for forwarding υ_i sensor reports to the initiator. Hence, for each neighbor i that needs to forward sensor reports, the PLAN has a corresponding three-tuple: ⟨the node id of neighbor i, υ_i, the time to start to forward sensor reports⟩. It also contains the time that all neighbors should have completed data forwarding. Table III summarizes the notations used in heuristic algorithms.

4. DISCUSSION

In Section 3, we presented the distributed DPF mechanism and the heuristic algorithms designed for improving the performance of opportunistic data collection—that is, the total amount of sensor reports collected by smartphones. In this section, several important issues related to DPF will be discussed further.

Table III. Notations Used in Heuristic Algorithms

	Comments
B_i	Buffer size of node i in the unit of sensor report.
D_i	Current number of sensor reports in the buffer of node i.
e_i	Remaining energy of node i that can be used for communication during the current epoch.
ψ_i	Current available buffer space of node i. Note that $\psi_i = B_i - D_i$.
χ_i	Expected available buffer space of node i. Note that $\chi_i = B_i - D_i + \zeta_i - R_i$.
τ_i	Number of reports that node i hopes to forward.
υ_i	Number of reports that the initiator will receive from node i.
Δ	Threshold used to reduce the overhead of data forwarding. It is set to 4 in this work.

4.1. Time to Initiate DPF

In the distributed DPF mechanism, each sensor node will randomly and independently select the time to initiate DPF so that the nearby sensor nodes will not initiate at the same time. However, even though the expected wireless sensor networks are sparsely deployed and T_{dpf} could be measured in hours or days, the probability that two nodes within the two-hop range initiate DPF at the same time is still not negligible. For instance, we assume that T_{dpf} is 1 day (86,400s), a run of DPF can be completed in 20s, and there are 30 nodes within the collision domain of a given sensor node.[5] Approximately, it is similar to let 30 nodes independently and randomly select one number among 4,320 numbers. According to the Birthday Paradox, the collision probability is still near to 10%.

Hence, although it does not appear in Figure 6, we do let a sensor node detect the collision. If any unexpected frame (RDV originated from other sensor node, etc.) is received by an initiator, the collision is detected and this initiator will randomly select its time to carry out DPF again. We argue that the collision can be solved quickly since the probability of colliding again is very small (29/4,320, or 0.67%).

4.2. Dynamics of Sensor Nodes

Fundamentally, DPF is designed for static sensor nodes that form a steady network topology. It becomes much harder to decide the routes used by sensor reports if sensor nodes move around—that is, their contact capacity and the network topology change frequently. Although mobile sensor node is beyond the scope of this article, we still need to consider some infrequent node dynamics, such as the deployment of new sensor nodes and the death of sensor nodes due to energy depletion, hardware failure, or other reasons.

When a new sensor node is deployed, it will start to broadcast beacons very soon and its neighbors can find it easily when they initiate DPF. To decide the time of initiating DPF, the new sensor node can monitor the wireless channel for a while and pick up one free time slot. Thus, with the distributed DPF mechanism, a new sensor node can join the network smoothly without disturbing the existing sensor nodes. In case a sensor node is dead, sensor reports in this node will be lost unavoidably. However, its neighbors could continue to operate as normal, as the distributed DPF mechanism does not let sensor node maintain a list of its neighbors. Of course, the remaining sensor nodes will make decisions based on the new network topology.

[5]Note that the nodes within the two-hop range of a given sensor node are inside the collision domain of this sensor node. A collision domain with 30 nodes represents a quite dense wireless sensor network, as each node can have an average of five direct neighbors.

4.3. Wireless Transmission Errors

Since wireless transmission is prone to error, our distributed DPF mechanism has been carefully designed to solve the challenges brought by wireless transmission errors. To be more robust to packet corruption, PLAN—the message that contains the DPF plan—is broadcast for three times in the distributed DPF mechanism. When transmitting sensor reports among sensor nodes, a retransmission timer is used to detect packet corruption and reliable data transfer service is provided. The SNRs experienced by RDV and RDV_ACK are also exploited by the initiator when selecting the neighbors. Thus, we can avoid using a weak link, over which sensor nodes must consume more energy to transfer the same amount of sensor data.

Even though only the strong wireless links are used by sensor nodes to carry out data forwarding, beacons still may be corrupted over these links due to the fundamental lossy characteristic of wireless communication. Hence, the initiator must make decisions based on the partial knowledge of its neighbors, and the performance of DPF will be affected. This issue is studied in Section 5.3.3, and the simulation results indicate that our mechanism can perform well even when the beacon loss rate is as high as 40%.

4.4. Overhead of the Distributed DPF Mechanism

As shown in Figure 6, apart from the Forwarding and Receiving states used for exchanging data among sensor nodes, a sensor node mainly keeps its radio on in RendezvousA state for a period of T_{rdv}. To keep the distributed DPF mechanism simple and stateless, T_{rdv} must be longer than the largest possible value of T_b so that the initiator could receive a BEACON from each neighbor. Next, we will approximately calculate the lifetime of the busiest sensor node under a typical scenario.

Based on a human mobility pattern and the mainstream sensor node platform, we assume that T_{dpf} is 1 day (86,400s) and T_{on} is 20ms. T_b is set to 5s so that the duty cycle for contact probing is 0.4%. T_{rdv} can then be set to 6s. We also assume that the buffer size of a sensor node is 500KBytes and the wireless link bandwidth is 250Kbps. Hence, one node can transmit all data in a full buffer to another node within 20s. For the busiest sensor node, since it initiates DPF just once per epoch, it needs to spend at most 20s on receiving data from its neighbors and another 20s on forwarding data to its neighbors.

Under this scenario, for each epoch, a sensor node that does not implement the distributed DPF mechanism needs to turn on its radio for at most $86{,}400 * \frac{0.4}{100} = 345.6$ seconds to carry out contact probing. A sensor node that runs our DPF mechanism needs to turn on its radio for at most $86{,}400 * \frac{0.4}{100} + 6 + 2 * 20 = 391.6$ seconds to carry out both contact probing and DPF.

In case a sensor node spends 20s on uploading data to mobile nodes and its battery can keep the radio on for 1 week, the sensor node without DPF can live for $7 * \frac{86{,}400}{345.6+20} = 1{,}654$ days and the busiest sensor node with DPF can live for $7 * \frac{86{,}400}{391.6+20} = 1{,}470$ days. Thus, the overhead of the distributed DPF mechanism is quite small even for the busiest sensor node, and the network lifetime is not reduced significantly.

When T_b could be large for a lower duty cycle, the energy consumed for contact probing will be reduced. However, T_{rdv} must be increased, and the overhead of the distributed DPF mechanism may become excessive. In current work, we are developing methods to allow the sensor node to learn the times that its neighbors will initiate DPF. When a neighbor will initiate DPF soon, the sensor node can broadcast beacons with a much higher frequency for a short period. Hence, T_{rdv} could be quite short, and the overhead could be reduced significantly. Furthermore, T_{rdv} could also be set to a

multiple of this much shorter beacon interval so that the initiator could collect states from most of its neighbors even when some beacons might be corrupted.

4.5. Contact Capacity Estimation for DPF-EST

In DPF-EST, one of the two heuristic algorithms presented in Section 3.4.2, the data rate and contact capacity of a sensor node are exploited to improve the performance of DPF. Although the rate of generating sensor reports is predictable and/or estimable in many applications (environmental monitoring, meter reading, etc.), it is much more challenging to estimate the contact capacity of a sensor node considering that we are exploiting the uncontrollable mobility of smartphone users.

Fortunately, human mobility demonstrates strong repeated patterns, such as the diurnal pattern [Gonzalez et al. 2008; Becker et al. 2011; Wu et al. 2013]. Thus, a sensor node can learn its encounters with smartphones during each day and use an exponentially weighted moving average (EWMA) scheme to filter out the noise. Furthermore, it is reported that the probed contact capacity is linearly related with the duty cycle if it is low [Wu et al. 2011a]. Considering that low duty cycles are used by sensor nodes in opportunistic data collection, a sensor node can keep measuring the probed contact capacity on each day with a very low duty cycle and straightforwardly calculate the contact capacity that could be probed with a selected duty cycle.

Furthermore, DPF-EST is not sensitive to the estimation error, because sensor nodes make decisions based on the expected available buffer size, which is cumulative across the runs of DPF. For instance, when the estimation is larger than the actual contact capacity, a sensor node will first pull more data from its neighbors. However, the excessive sensor reports must stay in the buffer since they cannot be uploaded to smartphones or forwarded to other neighbors. Thus, during the following runs of DPF, its available buffer size will become smaller and the sensor node will not blindly keep pulling data from its neighbors.

In Section 5.3.1, we will study the effects of contact capacity estimation accuracy. The simulation results indicate that DPF-EST can perform very well even when the estimation error is quite large. In case it is too hard and/or complex to estimate contact capacity, we still can use DPF-BUF—another heuristic algorithm that could also improve the network throughput of opportunistic data collection significantly.

4.6. Data Rate and Human Mobility Pattern

DPF is proposed to exploit the spatial locality of human mobility. It works well when the amount of sensor data generated by a sensor node is lower than the contact capacity of sensor nodes that are visited frequently and higher than the contact capacity of sensor nodes that are visited seldom.

If the data rate is very high and a sensor node that is visited frequently can not upload its own data to smartphones, DPF cannot improve the total amount of collected data, although it still consumes the energy of sensor nodes. This is why we state that DPF is designed for low-data-rate wireless sensor network applications. In case the data rate is very low and a sensor node that is visited seldom could upload all data to smartphones, DPF also cannot improve the overall network throughput. However, DPF might be used to reduce the delay of data collection. This issue will be studied in the future.

5. EVALUATION RESULTS

To evaluate the preceding distributed DPF mechanism with two heuristic algorithms (DPF-BUF and DPF-EST), we have implemented them in Contiki-OS [Dunkels et al. 2004], because the same code can run on real sensor nodes and in the COOJA simulator [Osterlind et al. 2007]. In this article, they are evaluated mainly through extensive

simulation. For comparison, the naive algorithm (WO-DPF),[6] in which sensor reports do not flow among sensor nodes, is implemented in Contiki-OS and simulated with COOJA as well. In addition, the linear program relaxation of the DPF problem presented in Section 2 is referred to as DPF-OPT, and its results produced with MATLAB [MathWorks 2013] are also used here to evaluate the performance of these heuristic algorithms.[7] Hence, we will evaluate and compare four algorithms in this article: WO-DPF, DPF-BUF, DPF-EST, and DPF-OPT.

Since sensor nodes prefer high throughput during data communication (uploading data to mobile nodes and exchanging data among neighbors) and low duty cycle is preferred at other times, the existing low-power MAC protocols (X-MAC [Buettner et al. 2006], Koala [Musaloiu-E et al. 2008], etc.) cannot satisfy these requirements in both cases. Thus, NULLMAC of Contiki-OS[8] is used, and the radio is directly controlled by the DPF mechanism presented in Section 3. In the simulations with COOJA, unit disk graph medium (UDGM) is used as the radio medium and the transmission range is set to 50m, which is selected according to the communication range of the current sensor node platforms in urban environments. UDGM is used here so that we can control the network topology for comparing heuristic algorithms with DPF-OPT. To facilitate comparison with DPF-OPT, the overhead of packet headers, acknowledgement packets, and retransmission is not considered by other algorithms when calculating the energy consumed for exchanging sensor reports.

Since the mobility of smartphone users demonstrate a strong diurnal pattern [Wu et al. 2013], the epoch length (Γ) and the interval between two consecutive runs of DPF (T_{dpf}) should be a multiple of 24 hours. To get meaningful results, each simulation must also be run for a large number of epochs. However, simulations with COOJA are very slow, because a machine code instruction level emulator of the TELOSB node [Polastre et al. 2005] is used for each sensor node. Hence, in the following simulations, Γ and T_{dpf} will be set to 2 hours (i.e., 7,200s). Other parameters are also adjusted to reflect this short epoch, and the data rate and duty cycle of a simulated sensor node could become higher than the expected opportunistic data collection applications in the real world.

In this article, we are evaluating the DPF schemes. To control the problem size, we assume that the sensor nodes have already figured out the suitable contact probing scheme and have learned their contact capacity. Therefore, rather than simulate the movement of mobile nodes directly, we implement a process on each sensor node that simulates the contact arrivals and removes the corresponding number of sensor reports from the buffer. The parameters used to simulate contact arrivals are used by the sensor node directly to calculate its contact capacity. For simplicity, we assume that there are two kinds of sensor nodes: hot nodes and cold nodes. A hot node will communicate with mobile nodes 20 times per epoch, whereas a cold node only has the opportunity of communicating with mobile nodes once per epoch. We assume that the size of a sensor report is 30 bytes and that 400 sensor reports can be collected during each contact. This is close to the amount of data that the smartphone of a car driver can collect when it passes by a sensor node at a moderate speed. In the following simulations, contacts are distributed evenly in an epoch.

[6]When WO-DPF is used, sensor nodes do not carry out DPF. They just broadcast beacons periodically to probe mobile nodes and upload their own sensor reports to these mobile nodes. Thus, sensor nodes will not try to communicate with their neighbors, and sensor reports will not be exchanged among sensor nodes.

[7]Note that DPF-OPT gives an upper bound of the throughput of DPF. A heuristic algorithm whose performance is close to DPF-OPT must achieve almost the optimal result of the DPF problem.

[8]In Contiki-OS, NULLMAC does not schedule the radio for achieving low duty cycle. When a sensor node asks NULLMAC to transmit a packet, the simple carrier sense multiple access (CSMA) is used for accessing the wireless channel.

Table IV. Parameter Values

Parameter	T_{dpf}	T_b	T_{rdv}	T_{plan}
Value	7,200s	5s	6s	8s
Parameter	B (reports)	R (reports/epoch)	ζ_{hot} (reports/epoch)	ζ_{cold} (reports/epoch)
Value	7,200	1,440	8,000	400

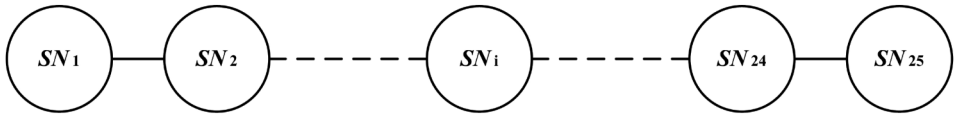


Fig. 7. A chain of 25 sensor nodes.

As for the data rate, each sensor node will generate one sensor report per 5s. Thus, a sensor node will generate $R = 7{,}200/5 = 1{,}440$ sensor reports in an epoch. T_b is also set to 5s. Accordingly, T_{rdv} and T_{plan} are set to 6 and 8 seconds, respectively. In opportunistic data collection, large flash memory should be installed on a sensor node for storing data, as the uncontrolled mobile nodes may not pass by for a long time.[9] Hence, the buffer size of a sensor node (B) is now set to 216Kbytes—that is, 7,200 sensor reports can be buffered at a sensor node. Table IV summarizes the parameter values used in these simulations.

In the following subsections, we will present experimental settings and simulation results under various scenarios, such as different network topologies and different energy budgets of a sensor node. For each experiment, the simulation will run for a long period (400 hours, or 200 epochs) to get meaningful results. Except Section 5.4, each simulation is run five times (with different random number seeds), and the average, minimal, and maximal results are plotted together. At the end of this section, we present a validation study on a small testbed.

5.1. Chain Topology

As illustrated in Figure 7, a chain of 25 sensor nodes is first used by us to evaluate WO-DPF, DPF-BUF, and DPF-EST under a simple environment.

The number of hot nodes (N_{hot}) is set to 1, 2, 3, 5, or 7, and these hot nodes spread evenly on the chain with equal distance. Through varying N_{hot}, we can evaluate these DPF algorithms under different levels of spatial locality. As for the energy budget of a sensor node, it is set according to R. More specifically, Φ will be set to $2 * R$, $4 * R$, $8 * R$, or $16 * R$ for evaluating the preceding DPF algorithms under different energy budgets of a sensor node.

Figure 8 shows the simulation results with the chain topology. The x-axis is the number of hot nodes, and the y-axis is the network throughput—that is, the total number of sensor reports collected from all sensor nodes. These plots indicate that network throughput will increase with the number of hot nodes. This is reasonable, because the overall contact capacity increases with the number of hot nodes. In addition, when there are more hot nodes, cold nodes are much closer to hot nodes and the energy consumed for forwarding their sensor reports to a hot node can be much lower. Thus, more sensor reports can be collected under the same energy budget. These plots also indicate that the two heuristic-based algorithms, DPF-BUF and DPF-EST, perform well. Under all cases, the two algorithms can collect many more sensor reports than WO-DPF, in which DPF is not carried out at all. Hence, DPF is an effective mechanism to improve

[9]Large flash memory is common on the mainstream sensor nodes. For instance, the TELOSB node has 1MB external flash memory for data logging.

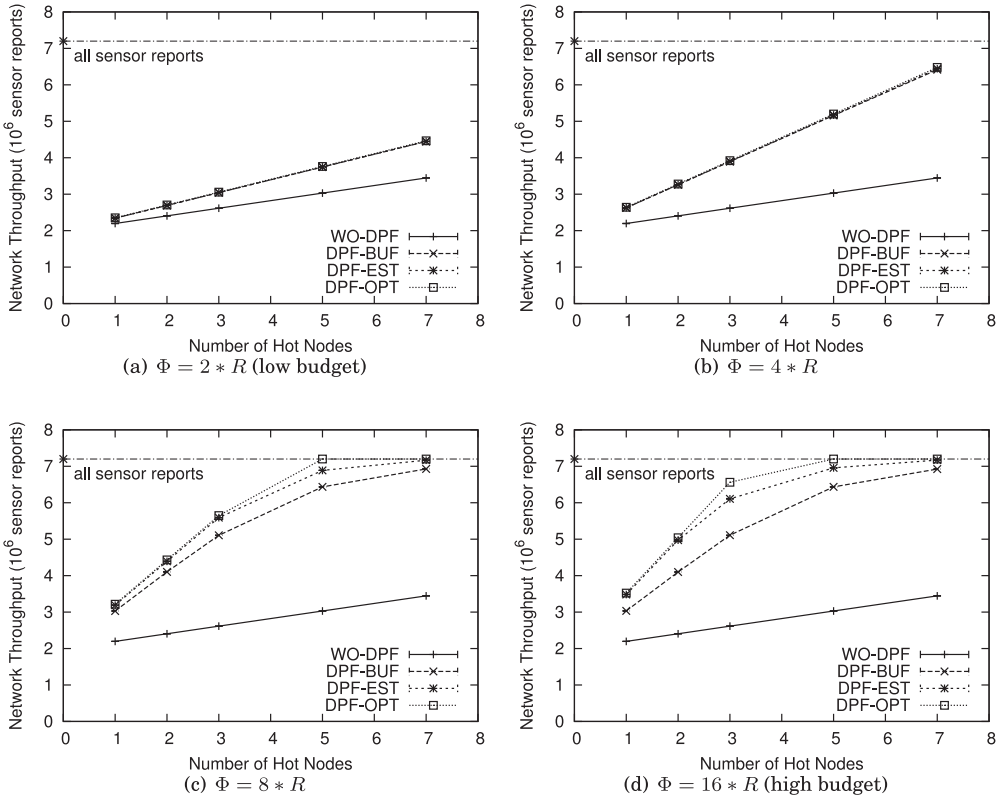


Fig. 8. Simulation results with chain topology.

the performance of opportunistic data collection even when very simple algorithms are used by a sensor node to make decisions based on local information—that is, the current states of its direct neighbors.

As indicated by Figure 8(a) and (b), DPF-BUF and DPF-EST perform the same as DPF-OPT (the optimal result) when the energy budget of the sensor node is low, because if the energy budget is low, the optimal DPF plan is also limited to one- or two-hop data forwarding. Hence, DPF-BUF and DPF-EST, which work based on one-hop information, can perform similar to DPF-OPT. When the energy budget is large, as shown in Figure 8(c) and (d), DPF-BUF and DPF-EST are poorer than DPF-OPT, as they cannot effectively exploit multihop data forwarding enabled by the large energy budget. However, the performance of DPF-EST is still very close to DPF-OPT, and it performs much better than DPF-BUF, because DPF-EST works based on the expected available buffer space, and hot nodes could pull data from other nodes more quickly.

5.2. Grid Topology

A 10×10 grid of 100 sensor nodes (Figure 9) is adopted in this group of experiments to evaluate WO-DPF, DPF-BUF, and DPF-EST under a more complex environment. For most of these sensor nodes, they will have four neighbors. N_{hot} will be set to 5, 10, 15, 20, or 25, and these hot nodes spread randomly and evenly in the grid. More specifically, the grid is divided into N_{hot} continuous blocks of the same size, and one node is randomly picked from each block as a hot node. The energy budget of a sensor node (Φ) is also set to $2 * R$, $4 * R$, $8 * R$, or $16 * R$.

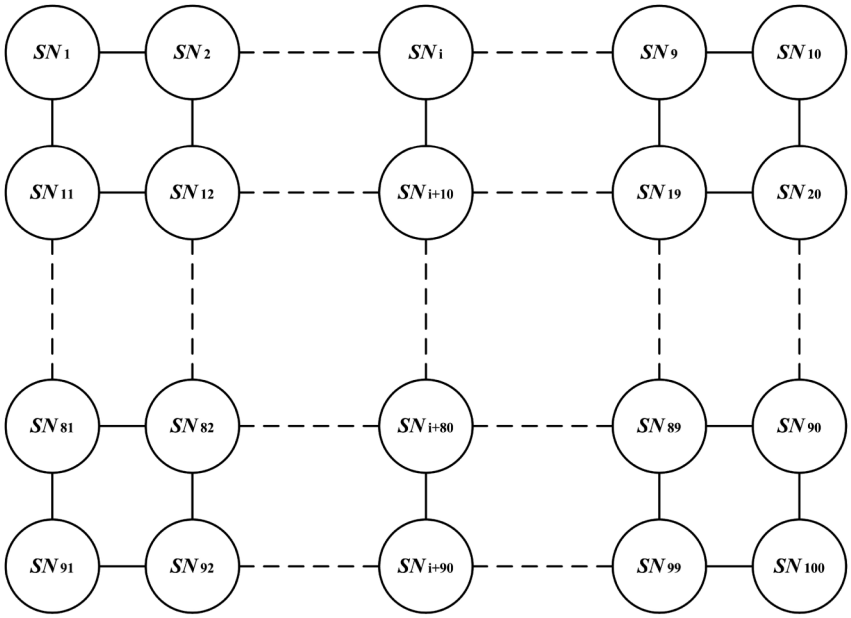


Fig. 9. A 10×10 grid of 100 sensor nodes.

Figure 10 shows the simulation results with the grid topology. These plots are very similar to the results with the chain topology shown in Figure 8. Hence, Figure 10 indicates that the arguments with the chain topology are also correct with the grid topology. The heuristic-based algorithms, especially DPF-EST, still can perform very well in this more complex topology.

5.3. Effects of Parameters

Although DPF-EST performs very well in all of the preceding cases, it assumes that a sensor node could estimate its contact capacity and data rate accurately. Although the rate of generating sensor data can be determined in many applications, it is not easy for a sensor node to learn its contact capacity accurately due to the uncontrolled mobility of smartphone users. In this subsection, we will evaluate the effects of contact capacity estimation accuracy on DPF-EST with the preceding chain and grid topologies. Furthermore, the buffer size of a sensor node will limit the number of sensor reports that a node can pull from its neighbors in an epoch. We will also evaluate its effects on DPF-BUF and DPF-EST. In addition, we will investigate the effects of wireless link dynamics. Since the corrupted data packets can be recovered easily through retransmission, we will focus on the effects of the corrupted beacons.

When studying the issues presented previously, N_{hot} is fixed to 5 and Φ is fixed to $8 * R$ in the chain topology. For the grid topology, N_{hot} is fixed to 15 and Φ is also fixed to $8 * R$.

5.3.1. The Accuracy of Contact Capacity Estimation. To investigate its effects on DPF-EST, the estimation error of contact capacity is set to 0, 10%, 20%, 30%, and 40% in different experiments. For instance, when the estimation error is 40%, the contact capacity used in the DPF-EST algorithm follows a uniform distribution between $0.6 * \zeta$ and $1.4 * \zeta$. Here, ζ is the real contact capacity of a sensor node. The data buffer size is fixed to 7,200 sensor reports.

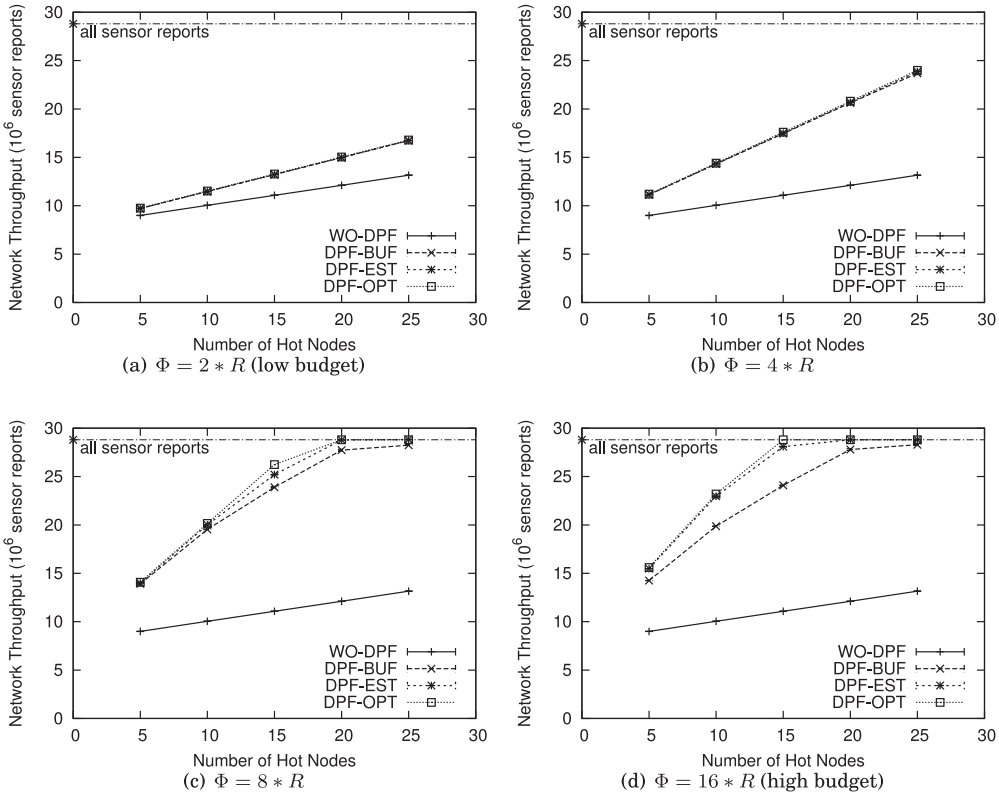


Fig. 10. Simulation results with grid topology.

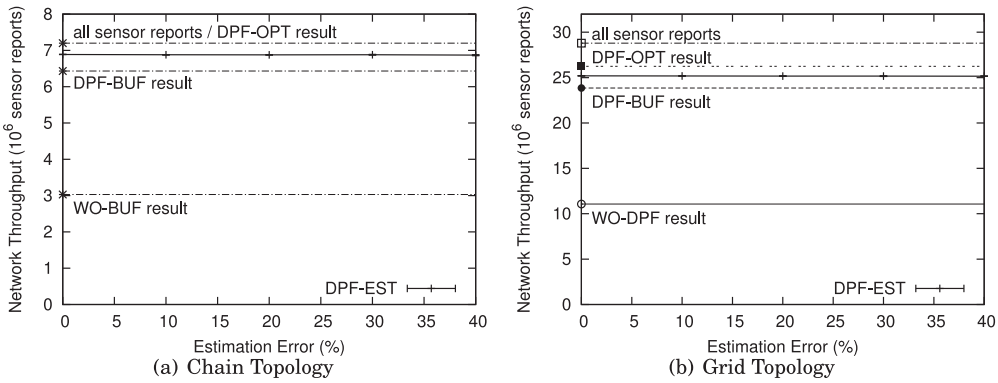
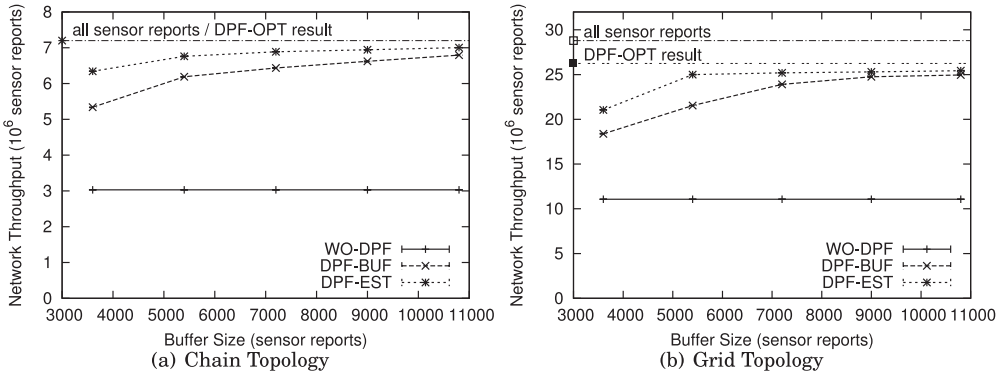


Fig. 11. Effects of contact capacity estimation accuracy.

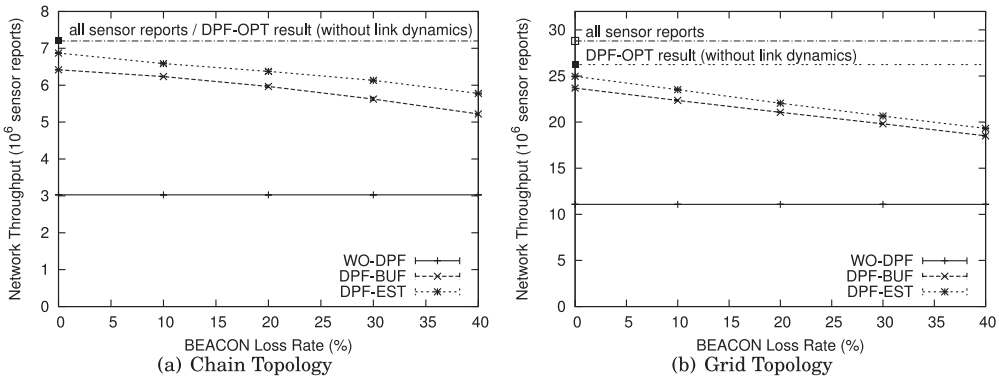
As shown in Figure 11, the performance of DPF-EST is not sensitive to the accuracy of contact capacity estimation. Even when the estimation error is as high as 40%, the throughput of DPF-EST is not obviously reduced.

5.3.2. The Size of Data Buffer. In this group of experiments, we still assume that a sensor node can estimate contact capacity accurately. The data buffer size of a sensor node is



(a) Chain Topology (b) Grid Topology

Fig. 12. Effects of buffer size.



(a) Chain Topology (b) Grid Topology

Fig. 13. Effects of link dynamics.

set to 3,600, 5,400, 7,200, 9,000, and 10,800 sensor reports in different experiments for studying the effects of buffer size.

Figure 12 illustrates that the throughput of DPF-BUF and DPF-EST increases with the buffer size, because when the buffer is large, a hot node can receive more sensor reports from its neighbors in an epoch. Considering that a large flash memory is normally used as the buffer in opportunistic data collection with smartphones, the good performance can be expected. Since contact capacity is exploited by DPF-EST, compared with a hot node with DPF-BUF, a hot node with DPF-EST can receive more sensor reports from its neighbors, and this difference becomes more obvious with the decrease of buffer size. This explains why DPF-EST performs better than DPF-BUF in all cases, especially when the buffer is small.

5.3.3. Link Dynamics. In this group of experiments, the buffer size is fixed to 7,200 sensor reports, and we assume that a sensor node can estimate contact capacity accurately. We will study the effects of link dynamics on beacons, as the corrupted beacons will affect the neighbors considered by a sensor node when it initiates DPF. In this group of experiments, the link dynamics are simulated through dropping beacons purposely. To study the performance of DPF-BUF and DPF-EST under different levels of link dynamics, the loss rate experienced by beacons is set to 0, 10%, 20%, 30%, and 40% in different experiments.

As shown in Figure 13, the throughput of DPF-BUF and DPF-EST decreases with the increase of link dynamics. However, DPF-BUF and DPF-EST still perform much

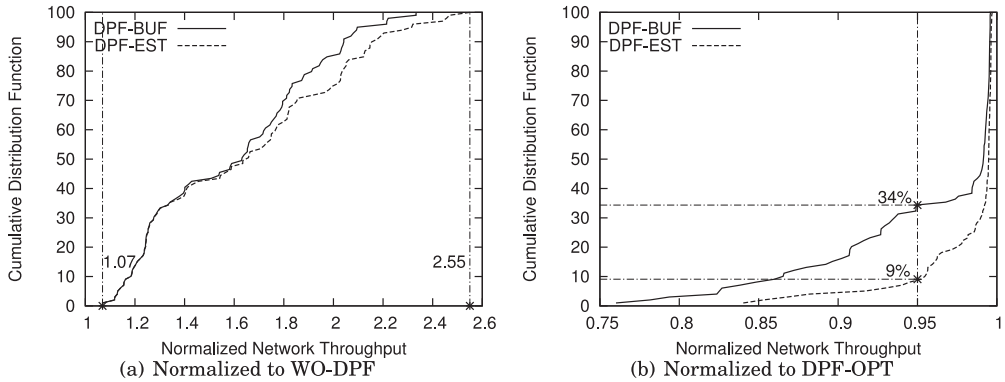


Fig. 14. CDFs of the normalized throughput.

better than WO-DPF even when the loss rate experienced by beacons is as high as 40%. The results indicate that these heuristic algorithms can work well in the dynamic and complex wireless environments.

5.4. Random Topology

To evaluate these algorithms extensively, we generate 100 random network topologies in this group of experiments. More specifically, 200 sensor nodes are deployed at random locations in one of the following areas: 600m*600m, 700*700m, 800m*800m, 900m*900m, and 1,000m*1,000m. For each area size, we repeat the preceding node placement operation 20 times. After that, for each of these 100 node placements, we extract its largest connected component based on the distances among these sensor nodes and use this component as the network topology to be evaluated. Among the 100 network topologies to be evaluated, the number of sensor nodes in a network varies from 9 to 194, and the number of neighbors of a sensor node varies from 1 to 13. In addition, for each network topology, we randomly select its percent of hot nodes in the range of [0.05, 0.25]. Each sensor node will then be marked as a hot node with the selected probability. The energy budget of all sensor nodes (Φ) is randomly selected from $2 * R$, $4 * R$, $8 * R$, and $16 * R$.

Figure 14(a) shows the cumulative distribution function of the network throughput normalized to WO-DPF. Figure 14(b) shows the cumulative distribution function of the network throughput normalized to DPF-OPT. These plots indicate that the heuristic algorithms perform well under various scenarios and that DPF-EST is better than DPF-BUF. As shown in Figure 14(a), DPF-EST can improve the number of collected sensor reports by 7% to 155%. Figure 14(b) indicates that DPF-EST achieves at least 95% of the optimal result in more than 90% of instances.

5.5. Spatial Correlation of Hot Nodes

In the preceding evaluations, hot nodes are evenly distributed in wireless sensor networks. Considering that spatial correlation does exist in the mobility of smartphone users [Wu et al. 2013], the neighbors of a hot node also tend to be hot nodes. Thus, in this group of experiments, we will study scenarios in which hot nodes have strong spatial correlation.

As shown in Figure 15(a), five hot nodes are all in the center of the chain topology. For the grid topology shown in Figure 15(b), 15 hot nodes form a cross in the middle of this grid. In this group of experiments, we assume that a sensor node can estimate contact capacity accurately and that there is no packet corruption. The data buffer size

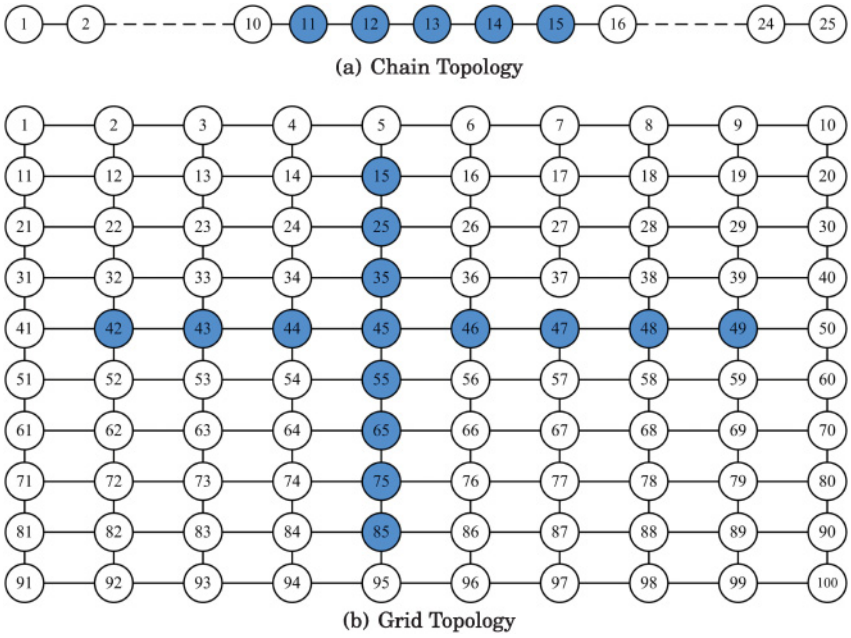


Fig. 15. Spatial distribution of hot nodes.

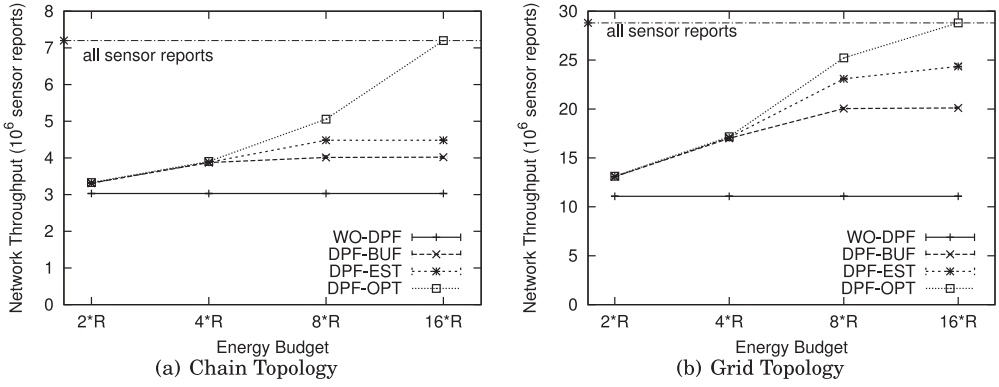


Fig. 16. Effects of spatial locality in hot nodes' distribution.

of a sensor node is always set to 7,200 sensor reports. However, the energy budget of a sensor node (Φ) is set to $2 * R$, $4 * R$, $8 * R$, or $16 * R$ in different experiments.

Figure 16 shows the simulation results of the preceding experiments. These plots indicate that although DPF-EST and DPF-BUF still perform much better than WO-DPF, their gaps to DPF-OPT are quite large. When the energy budget is increased from $8 * R$ to $16 * R$, there is also not much improvement on network throughput, especially for the chain topology. The fundamental reason is that DPF-OPT assumes that a sensor node has enough storage to hold sensor reports pulled from all neighbors during one epoch. However, the amount of sensor reports that a hot node can pull from its neighbors during one epoch is limited by its data buffer size, which is 7,200 sensor reports in our experiments. For instance, although there are five hot nodes in Figure 15(a), they can directly communicate with just two cold nodes. Thus, the amount

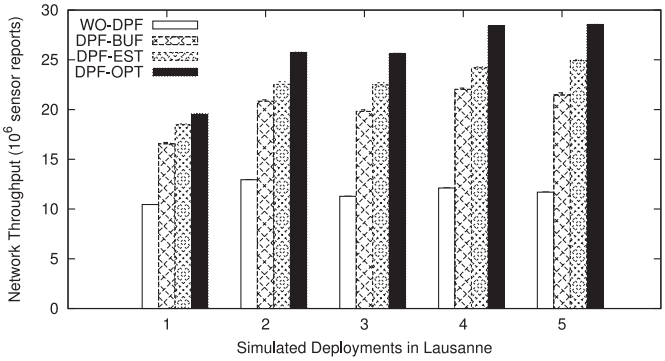


Fig. 17. Results with simulated deployments.

of sensor reports pulled from code nodes cannot exceed 14,400 even when the energy budget of a sensor node is unlimited.

Considering that a large flash memory is normally used as the buffer in opportunistic data collection with smartphones, the performance of DPF-EST and DPF-BUF should be quite good even when hot nodes are distributed unevenly. This issue will be studied further to alleviate the dependence on a huge data buffer.

5.6. Simulated Deployments in Lausanne, Switzerland

To evaluate these algorithms in a more realistic environment, five simulated deployments are studied here. For each deployment, we randomly pick up one area in Lausanne, Switzerland; 100 sensor nodes with a 10×10 grid topology are then deployed in this area; and each node is marked as *hot* or *cold* based on its location and the spatial distribution of smartphone users' mobility traces from the Nokia Mobile Data Challenge [Wu et al. 2013].

Figure 17 shows the simulation results when the energy budget of all sensor nodes (Φ) is $8 * R$. This plot indicates that our heuristic algorithms can perform well for these potential deployments in the real world and that DPF-EST is better than DPF-BUF. For these simulated deployments, DPF-EST reduced the optimality gap by around 50% over DPF-BUF and is always more than 85% of the optimal.

5.7. Testbed Evaluations

A chain of seven USN nodes (MTM-CM5000-MSP)[10] is also used by us to validate the simulation results. These seven nodes are put into a line, and the distance between the adjacent nodes is about 0.5m. The transmission power of their radios is carefully adjusted so that these nodes approximately form a network with a chain topology. In the following experiments, the first and fifth nodes act as hot nodes, and the other nodes act as cold nodes.

Figure 18 shows the testbed (21 nodes aligned in three lines) used to simultaneously evaluate WO-DPF, DPF-BUF, and DPF-EST in the same environment. Three different channels are used to avoid the interference among each other. The same code used in simulations is used in these sensor nodes. The only difference is that T_{dpf} is set to half an hour (1,800s) to get experimental results quickly. The contact arrival interval and the interval of generating sensor reports are also adjusted so that the parameter values listed in Table IV (except T_{dpf}) are also used in testbed evaluations. We argue

[10]The USN node is a clone of the TELOSB node, and it is produced by MAXFOR Technology Inc. (http://www.maxfor.co.kr/sub2_1_1_1.html).

Fig. 18. Testbed.

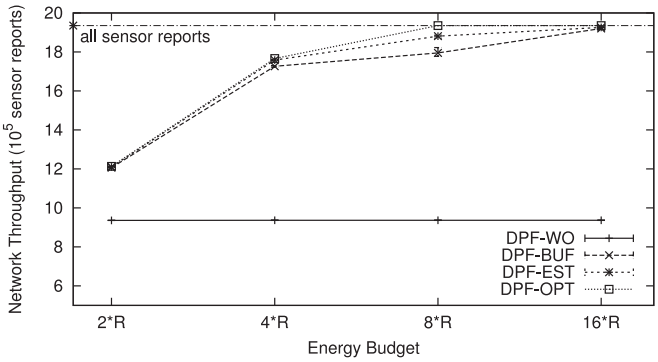


Fig. 19. Testbed evaluation results.

that this shorter T_{dpf} is acceptable, as a sensor node on this testbed only has a very few number of neighbors (one or two).

To evaluate these algorithms under different energy budgets, the energy budget of all sensor nodes (Φ) is also set to $2*R$, $4*R$, $8*R$, or $16*R$. For each combination of the algorithm and the energy budget, the experiment was run for 4 days (i.e., $4*24*2 = 192$ epochs) to get meaningful results. Each experiment was also run five times, and the average, minimal, and maximal results were plotted together.

Figure 19 shows the network throughput of these algorithms under different energy budgets. DPF-OPT results based on a chain topology are also plotted together. This figure indicates that the two heuristic algorithms can perform well on real hardware platforms and that DPF-EST is better than DPF-BUF. Both of them can collect many more sensor reports than WO-DPF, especially when the energy budget is high.

6. RELATED WORK

6.1. Data Collection

Due to the limited computing capability and storage size of sensor nodes, their sensor reports are normally sent to an application server for further processing. In the classical wireless sensor network, some dedicated static sink nodes are used to connect sensor nodes with the application server [Akyildiz et al. 2002; Gnawali et al. 2009]. Although sensor reports can be collected continuously, the cost of the static sink nodes can be prohibitive under the scenarios considered in this article.

In Li et al. [2011], Shah et al. [2003], Pasztor et al. [2007], Somasundara et al. [2006], Wang et al. [2005], and Zhao and Yang [2009], the use of mobile nodes was proposed to move around in the deployed area and collect data from sensor nodes. Depending on the applications, their mobility can be either controlled or not, and these mobile nodes may collect data from sensor nodes within the range of one or multiple hops. Li et al. [2011] studied how to maintain the collection tree when the sink is moving. However,

as stated in Section 2, the straightforward multihop data collection is not suitable for opportunistic data collection with smartphones, because the mobility of smartphone users is uncontrolled and sensor nodes must be deeply duty cycled. We instead use DPF to improve the performance of opportunistic data collection with smartphones.

In Kusy et al. [2009] and Truong et al. [2010], sensor nodes monitor and predict the location of a mobile sink so that their sensor data can be forwarded accordingly. However, even if the mobility of a smartphone user can be predicted, it is hard for the deeply duty cycled sensor nodes to exchange and exploit this information timely.

In Park and Heidemann [2011], the use of mobile phones is proposed to collect data from static sensor nodes purposely or opportunistically. However, the routing issues among sensor nodes for improving the throughput of data collection have not been studied.

In Dyo et al. [2010], spatial locality of the scientists' mobility has been leveraged. The domain experts assign priorities to sensor data, and sensor nodes are classified based on the frequency of being visited. The hot sensor nodes that are visited more frequently will then build up multiple collection trees, and other nodes will forward their important sensor data to the nearest hot sensor node to reduce the latency of data collection. However, a scientist will collect all data from a sensor node once it is visited. Hence, the mobility is not totally uncontrolled, and the problem studied in Dyo et al. [2010] is different from that of our article.

6.2. Node Rendezvous

In Wu et al. [2011a, 2011b], we studied contact probing in the context of opportunistic data collection. We found that the duty-cycled sensor nodes should be responsible for broadcasting beacons and that the temporal locality of human mobility should be learned and exploited when determining the frequency of broadcasting beacons. In this article, DPF is proposed to exploit the spatial locality of human mobility. These works are tightly related with this work, as the contact probing scheme determines the contact capacity of sensor nodes, which will affect how sensor reports flow among these nodes.

Rendezvous with neighbors has been well studied, and many MAC protocols have been proposed for wireless sensor networks, such as X-MAC [Buettner et al. 2006], Koala [Musaloiu-E et al. 2008], and TRAMA [Rajendran et al. 2003]. The distributed DPF mechanism is similar to Koala in the way of finding the neighbors—that is, a sensor node turns on its radio to listen for beacons from its potential neighbors. In PW-MAC [Tang et al. 2011], the authors propose to let a sensor node predict its neighbors' radio schedule for saving energy. In the future, we will adopt a similar approach to reduce the overhead of the distributed DPF mechanism.

6.3. Data Routing

As for data routing, our heuristic algorithms are similar to the approach proposed in Lin et al. [2008]. We establish contact capacity potentials that allow sensor reports to reach the sensor nodes with higher contact capacity through local greedy decisions, following information gradients. Instead of explicitly diffusing contact capacity across the network, the information is broadcast and composed through carrying out DPF hop-by-hop, and the gradients are reflected by the values of the available buffer space or the expected available buffer space.

Similar to our heuristic algorithms, many backpressure-based routing protocols have been designed for various wireless networks [Moeller et al. 2010; Ryu et al. 2010]. BCP [Moeller et al. 2010] is a routing protocol designed for wireless sensor network. When sensor reports are being collected by a mobile sink through multihop communication, a sensor node will choose the next hop for each packet based on queue backlog and link

quality of its neighbors. Many techniques have been proposed to make BCP a practical backpressure-based routing protocol at moderate duty cycles (15% to 25%). However, in opportunistic data collection with smartphones, we target wireless sensor networks with very low duty cycles ($\leq 0.1\%$), and sensor nodes exchange data when mobile nodes are absent. Hence, the problems and the challenges are totally different. For instance, BCP adopts *floating queue* to avoid that the small memory of a sensor node (which is far away from the sink) is used up to form the gradient needed by a backpressure-based routing protocol and the data stops to flow. In opportunistic data collection with smartphones, a sensor node uses a large flash memory to act as the buffer, and floating queue becomes irrelevant to DPF.

To the extent of our knowledge, our work is the first that studies how to let sensor nodes exploit the spatial locality of human mobility for improving the throughput of opportunistic data collection with smartphones.

7. CONCLUSION

In this article, we studied how to use DPF to improve the performance of opportunistic data collection with smartphones. We have modeled DPF as an optimization problem and proposed a simple distributed DPF mechanism with two heuristic algorithms (DPF-BUF and DPF-EST). These proposals have been implemented in Contiki-OS and evaluated with both COOJA simulations and testbed experiments. Evaluation results indicate that the distributed DPF mechanism with heuristic algorithm significantly outperforms the default approach (with no DPF) and is close to the performance of a centralized optimization algorithm, especially when sensor nodes can estimate its contact capacity and data rate.

In the future, we will study how to reduce the overhead of the distributed DPF mechanism through letting sensor nodes learn the times that their neighbors will initiate DPF. We will also handle contact probing and DPF together, considering joint optimization. We will carry out extensive simulations and large-scale testbed evaluations in which mobile nodes move around, sensor nodes carry out contact probing and learn contact capacity online, and sensor reports flow among sensor nodes for better performance. Another potential area is to consider the data reliability (in case sensor reports are replicated) and the value of a sensor report (the freshness and the type of the report, etc.) when carrying out DPF among sensor nodes.

ACKNOWLEDGMENTS

Portions of the research in this article used the MDC Database were made available by Idiap Research Institute, Switzerland, and owned by Nokia. The authors would also like to thank the Cork Constraint Computation Centre (4C) at University College Cork for the use of their computing cluster in simulations.

REFERENCES

Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. 2002. Wireless sensor networks: A survey. *Computer Networks* 38, 393–422.

Richard A. Becker, Ramon Caceres, Karrie Hanson, Ji Meng Loh, Simon Urbanek, Alexander Varshavsky, and Chris Volinsky. 2011. A tale of one city: Using cellular network data for urban planning. *IEEE Pervasive Computing* 10, 18–26.

Vaduvur Bharghavan, Alan J. Demers, Scott Shenker, and Lixia Zhang. 1994. MACAW: A media access protocol for wireless LANs. In *Proceedings of the ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications*. ACM, New York, NY. 212–225.

Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. 2006. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY. 307–320.

Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. 2004. Contiki—a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th IEEE International Conference on Local Computer Networks*. IEEE, Los Alamitos, CA. 455–462.

Vladimir Dyo, Stephen A. Ellwood, David W. Macdonald, Andrew Markham, Cecilia Mascolo, Bence Pasztor, Salvatore Scellato, Niki Trigoni, Ricklef Wohlers, and Kharsim Yousef. 2010. Evolution and sustainability of a wildlife monitoring sensor network. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY. 127–140.

Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. 2009. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY. 1–14.

Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. *Nature* 453, 779–782.

Andreas Kaltenbrunner, Rodrigo Meza, Jens Grivolla, Joan Codina, and Rafael Banchs. 2010. Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Pervasive and Mobile Computing* 6, 4, 455–466.

Niko Kiukkonen, Jan Blom, Olivier Dousse, Daniel Gatica-Perez, and Juha Laurila. 2010. Towards rich mobile phone datasets: Lausanne data collection campaign. In *Proceedings of the ACM International Conference on Pervasive Services (ICPS'10)*.

Branislav Kusy, HyungJune Lee, Martin Wicke, Nikola Milosavljevic, and Leonidas Guibas. 2009. Predictive QoS routing to mobile sinks in wireless sensor networks. In *Proceedings of the 8th International Conference on Information Processing in Sensor Networks*. IEEE, Los Alamitos, CA. 109–120.

Juha K. Laurila, Daniel Gatica-Perez, Imad Aad, Jan Blom, Olivier Bornet, Trinh-Minh-Tri Do, Olivier Dousse, Julien Eberle, and Markus Miettinen. 2012. The mobile data challenge: Big data for mobile computing research. In *Proceedings of the Mobile Data Challenge Workshop (MDC) in Conjunction with the International Conference on Pervasive Computing*.

Zhenjiang Li, Mo Li, Jiliang Wang, and Zhichao Cao. 2011. Ubiquitous data collection for mobile users in wireless sensor networks. In *Proceedings of the IEEE International Conference on Computer Communications*. IEEE, Los Alamitos, CA. 2246–2254.

Huijia Lin, Maohua Lu, Nikola Milosavljevic, Jie Gao, and Leonidas J. Guibas. 2008. Composable information gradients in wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*. IEEE, Los Alamitos, CA. 121–132.

Wang Liu, Jianping Wang, Guoliang Xing, and Liusheng Huang. 2009. Throughput capacity of mobility-assisted data collection in wireless sensor networks. In *Proceedings of the 6th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, Los Alamitos, CA. 70–79.

Miklos Maroti, Branislav Kusy, Gyula Simon, and Akos Ladeczi. 2004. The flooding time synchronization protocol. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. ACM, New York, NY. 39–49.

MathWorks. 2013. MATLAB: The Language of Technical Computing. Retrieved January 13, 2012, from http://www.mathworks.com/products/matlab/.

Scott Moeller, Avinash Sridharan, Bhaskar Krishnamachari, and Omprakash Gnawali. 2010. Routing without routes: The backpressure collection protocol. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, New York, NY. 279–290.

Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized Algorithms*. Cambridge University Press.

Razvan Musaloiu-E, Chieh-Jan Mike Liang, and Andreas Terzis. 2008. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*. IEEE, Los Alamitos, CA. 421–432.

Nordic Semiconductor. 2007. *Nordic Semiconductor Figures for nRF24AP1*. Nordic Semiconductor.

Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. 2007. Cross-level simulation in COOJA. In *Proceedings of the European Conference on Wireless Sensor Networks (Poster/Demo Session)*.

Unkyu Park and John Heidemann. 2011. Data muling with mobile phones for sensornets. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY. 162–175.

Bence Pasztor, Mirco Musolesi, and Cecilia Mascolo. 2007. Opportunistic mobile sensor data collection with scar. In *Proceedings of the IEEE International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, Los Alamitos, CA. 1–12.

Joseph Polastre, Jason Hill, and David Culler. 2004. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. ACM, New York, NY. 95–107.

Joseph Polastre, Robert Szewczyk, and David Culler. 2005. Telos: Enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*. 364–369.

Venkatesh Rajendran, Katia Obraczka, and J. J. Garcia-Luna-Aceves. 2003. Energy-efficient, collision-free medium access control for wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. ACM, New York, NY. 181–192.

Injong Rhee, Ajit C. Warrier, and Lisong Xu. 2004. *Randomized Dining Philosophers to TDMA Scheduling in Wireless Sensor Networks*. Technical Report. North Carolina State University.

Jung Ryu, Vidur Bhargava, Nick Paine, and Sanjay Shakkottai. 2010. Back-pressure routing and rate control for ICNs. In *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking*. ACM, New York, NY. 365–376.

Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. 2003. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*. IEEE, Los Alamitos, CA. 30–41.

Arun A. Somasundara, Aman Kansal, David D. Jea, Deborah Estrin, and Mani B. Srivastava. 2006. Controllably mobile infrastructure for low energy embedded networks. *IEEE Transactions on Mobile Computing* 5, 8, 958–973.

Philipp Sommer and Roger Wattenhofer. 2009. Gradient clock synchronization in wireless sensor networks. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*. IEEE, Los Alamitos, CA. 37–48.

Yanjun Sun, Omer Gurewitz, and David B. Johnson. 2008. RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*. ACM, New York, NY, 1–14.

Lei Tang, Yanjun Sun, Omer Gurewitz, and David B. Johnson. 2011. PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks. In *Proceedings of the 30th IEEE International Conference on Computer Communications*. IEEE, Los Alamitos, CA. 1305–1313.

Leandros Tassiulas and Anthony Ephremides. 1992. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control* 37, 12, 1936–1948.

Thuy T. Truong, Kenneth N. Brown, and Cormac J. Sreenan. 2010. Using mobile sinks in wireless sensor networks to improve building emergency response. In *Proceedings of Royal Irish Academy Research Colloquium on Wireless as an Enabling Technology*.

Wei Wang, Vikram Srinivasan, and Kee-Chaing Chua. 2005. Using mobile relays to prolong the lifetime of wireless sensor networks. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*. ACM, New York, NY. 270–283.

Xiuchao Wu, Kenneth N. Brown, and Cormac J. Sreenan. 2011a. Exploiting rush hours for energy-efficient contact probing in opportunistic data collection. In *Proceedings of the 31st International Conference on Distributed Computing Systems Workshops*. IEEE, Los Alamitos, CA. 240–247.

Xiuchao Wu, Kenneth N. Brown, and Cormac J. Sreenan. 2011b. SNIP: A sensor node-initiated probing mechanism for opportunistic data collection in sparse wireless sensor networks. In *Proceedings of the 1st International Workshop on Cyber-Physical Networking Systems*. IEEE, Los Alamitos, CA. 726–731.

Xiuchao Wu, Kenneth N. Brown, and Cormac J. Sreenan. 2012. Data pre-forwarding for opportunistic data collection in wireless sensor networks. In *Proceedings of the 9th International Conference on Networked Sensing Systems*. 1–8.

Xiuchao Wu, Kenneth N. Brown, and Cormac J. Sreenan. 2013. Analysis of smartphone user mobility traces for opportunistic data collection in wireless sensor networks. *Pervasive and Mobile Computing* 9, 6, 881–891. DOI:http://dx.doi.org/10.1016/j.pmcj.2013.07.003

Xiuchao Wu, Cormac J. Sreenan, and Kenneth N. Brown. 2011c. A shared opportunistic infrastructure for long-lived wireless sensor networks. In *Proceedings of the 2nd International ICST Conference on Mobile Lightweight Wireless Systems Workshops*. 330–337.

Miao Zhao and Yuanyuan Yang. 2009. Bounded relay hop mobile data gathering in wireless sensor networks. In *Proceedings of the 6th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, Los Alamitos, CA. 373–382.

Received April 2013; revised December 2013; accepted January 2014