CoMon+: A Cooperative Context Monitoring System for Multi-Device Personal Sensing Environments

Youngki Lee, Seungwoo Kang, Chulhong Min, Younghyun Ju, Inseok Hwang, and Junehwa Song

Abstract—Continuous mobile sensing applications are emerging. Despite their usefulness, their real-world adoption has been slow. Many users are turned away by the drastic battery drain caused by continuous sensing and processing. In this paper, we propose CoMon+, a novel cooperative context monitoring system, which addresses the energy problem through opportunistic cooperation among nearby users. For effective cooperation, we develop a benefit-aware negotiation method to maximize the energy benefit of context sharing. CoMon+ employs heuristics to detect cooperators who are likely to remain in the vicinity for a long period of time, and the negotiation method automatically devises a cooperation plan that provides mutual benefit to cooperators, while considering running applications, available devices, and user policies. Especially, CoMon+ improves the negotiation method proposed in our earlier work, CoMon [30], to exploit multiple processing plans enabled by various personal sensing devices; each plan can be alternatively used for cooperation, which in turn will maximize overall power saving. We implement a CoMon+ prototype and show that it provides significant benefit for mobile sensing applications, e.g., saving 27-71 percent of smartphone power consumption depending on cooperation cases. Also, our deployment study shows that CoMon+ saves an average 19.7 percent of battery under daily use of a prototype application compared to the case without CoMon+ running.

Index Terms—Cooperation, context sensing, peer discovery, negotiation, energy, personal sensing device

1 Introduction

Continuous mobile sensing applications have been increasingly emerging, for instance, trajectory logging [44], dust level monitor [25], interaction monitor [17], [31], group-aware ads and resource planning [41], and calorie monitor [33]. These applications provide useful services to mobile users while running in the background, not requiring any explicit user intervention. However, many users are still reluctant to run such applications; these applications incur significant energy consumption and take up computational resources, potentially disrupting other common uses of the smartphones.

We approach the problem from a novel perspective, by utilizing *in-situ cooperation* of mobile users. We note that, people's daily lives are highly social; they spend a significant time with others, e.g., family members, friends, or even some strangers. According to our study, a user is co-located with acquaintances about 8.5 hours out of 15 active hours of a day, and even more, when accounting for co-location with strangers. In addition, 65 percent of meetings last for more than 30 minutes, allowing opportunities for stable cooperation in continuous sensing. Moreover, collocated mobile users often share common interests in many situational contexts related to *ambience* such as locations and atmosphere. These contexts can potentially be shared by nearby users, e.g., friends in a social gathering. Thus, users can avoid repetitive sensing and processing redundantly performed by individual users that consume precious energy. This sharing becomes more practical due to the probable cost savings of the sharing. The power consumption for sensing and processing often exceeds the overhead to obtain context data from nearby users; for instance, performing location sensing every 10 seconds consumes 410 mW on a Nexus One phone while it consumes only 34 mW to receive the same data from others through Bluetooth communication.

To realize the approach, we propose CoMon+, a novel cooperative context monitoring system. CoMon+ automatically finds cooperators in situ and initiates the cooperation in a way that enhances its energy capacity or extends its sensing modalities. Applications simply delegate their monitoring requests to CoMon+ and fully exploit its own and cooperators' resources if available. By employing cooperation, CoMon+ significantly mitigates the quick battery depletion of devices, or overcomes the absence of specific sensing modalities.

The design of CoMon+ involves a number of challenges to address. A key challenge is how to construct cooperation groups and build network channels for continuous cooperation. We employ a continuity-aware cooperator

- *Y. Lee is with School of Information Systems at Singapore Management University, Singapore. E-mail: youngkilee@smu.edu.sg.*
- *S. Kang is with School of Computer Science and Engineering at Korea University of Technology and Education, Republic of Korea. E-mail: swkang@koreatech.ac.kr.*
- *C. Min and J. Song are with School of Computing at KAIST, Republic of Korea. E-mail: {chulhong, junesong}@nclab.kaist.ac.kr.*
- *Y. Ju is with NAVER LABS, Republic of Korea. E-mail: younghyun.ju@navercorp.com.*
- *I. Hwang is with IBM Research Austin, Austin, TX 78758. E-mail: ihwang@us.ibm.com.*

Manuscript received 16 Aug. 2014; revised 24 May 2015; accepted 14 June 2015. Date of publication 25 Sept. 2015; date of current version 29 June 2016. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2015.2452900

detection method, which enables CoMon+ to maintain stable cooperation channels and reduce the complexity in cooperation network management.

Another important challenge is to provide incentives to cooperating participants. Without benefits, a mobile user would be reluctant to actively participate in cooperation and share her resources. However, it is not a straightforward problem to guarantee mutual benefits to all cooperators. Cooperators often run different sets of applications, and possess different sensing devices. Also, they have their own preferences and policies in the use of energy of their devices. Such differences complicate the negotiation to guarantee fair and mutual benefit for cooperators. We propose a benefit-aware negotiation mechanism, which addresses the challenges and builds a mutually beneficial cooperation contract.

In this paper, we especially delve into the negotiation mechanism, by extending CoMon proposed in our earlier work [30]. Recently, mobile users start to carry multiple personal devices, e.g., a smart-watch and a smart-glass. Such devices provide multiple alternatives to sense and infer a context, which extends opportunities for cooperation while complicating the negotiation. To take such alternatives into account and maximize cooperation benefit, we devise a *local-plan-aware negotiation* mechanism. It updates processing alternatives (namely *local plans*) on-the-fly, reflecting the remaining battery and context supportability of available personal devices. Then, upon negotiation, the benefit for each local plan is evaluated in terms of a holistic battery use policy, and the plan to maximize the benefit is selected. The negotiation is periodically re-conducted as the expected benefit becomes obsolete due to the battery depletion of the devices to run the selected plans.

CoMon+ opens a new dimension to address the energy problem for continuous mobile context sensing. Many research efforts have been made to reduce energy consumption for context processing [24], [37], [39], taking an *intra-device optimization* approach, e.g., deactivating a sensor based on mobility patterns [39], applying an early-stage filter [37], sharing resources among processing pipelines [22], [32], [34]. Our *cooperation* approach complements such intra-device optimization techniques, providing further reduction in energy consumption. This additional dimension of benefit is significant, considering continuous and background operation of concurrent mobile sensing applications.

The contributions of this paper are as follows. First, we propose a novel cooperative context monitoring system, CoMon+; it significantly improves energy efficiency of smartphones and newly adopts unavailable sensing modalities. Second, we support the practicality of our cooperation approach through motivational studies on ATUS data [1] and Bluetooth-based encounter data [10]. Third, as core techniques, we develop continuity-aware cooperator detection and benefit-aware negotiation mechanisms, which enable CoMon+ to obtain resource benefits from inter-user cooperation. Especially, we extend the mechanism to incorporate multiple sensing alternatives enabled by various personal devices to best exploit their resources for cooperation. Finally, we perform extensive experiments based on our prototype implemented over Android phones and custom-designed sensor motes, with diverse sensing capabilities. We show the resource benefits and overheads for diverse cooperation scenarios. Moreover, we conduct extensive simulation study to understand the benefit of local-plan-aware mechanism.

TABLE 1
Context Examples and Their Categories

Context Category		Context Types
Ambience Context	*Spatial Context*	location, ambient sound, place, temperature, humidity, UV, dust-level, noise-level, mood, pollution (CO_2, O_3, ...), crowdedness, ...
	Social Context	discussion, meeting, conversation, lecture, group exercise, ...
Personal Context		activity (walking, standing, ...), gesture, health (heartbeat, gait, ...), emotion, ...

In the rest of the paper, we first motivate CoMon+ in Section 2 with studies on opportunities for cooperation. Section 3 describes the model of cooperation benefits and the CoMon+ systm architecture. Section 4 describes the basic cooperation planning mechanism, and Section 5 introduces the advanced local-plan-aware negotiation mechanism. In Section 6, we show experimental results with our prototype implementation, and Section 7 presents in-depth performance study on the advanced negotiation mechanism with extensive simulation. In Section 8, we discuss other issues for CoMon+ and Section 9 discusses related work. We conclude the paper in Section 10.

2 Opportunity for Cooperation

Mobile sensing applications have high potential to leverage cooperation between nearby people [20], [30], [41]. As they become popular, many of them will run concurrently, actively using diverse user contexts. Table 1 shows example contexts used by emerging applications [25], [37], [40]. A number of *ambience contexts* including *spatial* and *social contexts* would be shareable with nearby users.

Understanding that there will be many sharable contexts, two key questions are raised: (1) Does the cooperation result in actual energy benefits for context monitoring? (2) Are there enough cooperation opportunities in the everyday life of mobile users?

We first demonstrate an interesting scenario showing the expected power savings in Section 2.1. Note that the energy-related figures used in the scenario are presented based on actual measurements (See Section 6 for detailed setting). We then show that cooperation opportunities are actually prevalent in everyday life through our analysis of human activity and mobility datasets in Section 2.2.

2.1 Scenarios on Expected Energy Saving

Chandler, Ross, and Joey are friends in Manhattan. On Saturday, Chandler plans to meet Ross for shopping in the SoHo area. Chandler always runs two apps, *PollutionAlarm* and *LifeLogger* as in Fig. 1. He runs *PollutionAlarm* to avoid exposure to air pollution such as dust and exhaust fumes, and *LifeLogger* to record his route (using GPS) and optionally ambient sound contexts (using the microphone to

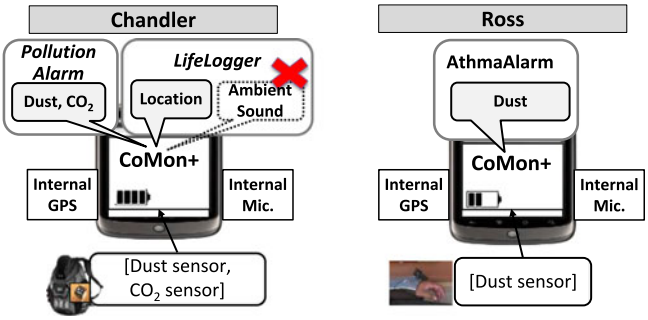


Fig. 1. An example cooperation scenario.

record music genres, meetings, etc.) [37]. Today, he turns off the ambient sound monitoring to extend the phone's battery life. Ross runs *AsthmaAlarm* due to his asthma problem. It monitors the dust levels in the air, a major allergen for asthmatics. While Ross is on his way to SoHo, he discovers that his dust sensor blinks notifying him that there are 'fewer than 3 hours of battery remaining'. Ross gets anxious, regretting that he forgot to recharge the sensor last night.

When Ross meets Chandler, Ross's CoMon+ starts cooperation with Chandler's to monitor the dust level in turn. This reduces the net power-on duration of each sensor by half; the average power consumption by Ross's dust sensor decreases to almost half, from 848 to 487 mW, and the estimated sensor lifetime increases from 3 to 5.2 hours. Note that Ross's smartphone requires a slight additional power expenditure of 25 mW to send and receive the dust level to and from Chandler's phone during the cooperation.

Joey was walking in a park near SoHo for his daily exercise; he runs the *CalorieMonitor* application which uses his movement speeds for calorimetry calculation. He also uses *LifeLogger*. On his way home, Joey happens to meet Chandler and Ross and they decide to go to a café. Detecting Joey's devices, Chandler's CoMon+ system entrusts sound monitoring to Joey's CoMon+ while supporting location monitoring for Joey instead. This cooperation enables Chandler's *LifeLogger* to again be fully functional by reactivating the disabled sound monitoring. Now Joey's phone turns off energy-intensive GPS sensing which has consumed 410 mW; instead, it needs only 34 mW to receive location context from Chandler. The additional cost to Joey's device to provide the ambient sound context every 10 seconds is insignificant (51 mW), since he has been monitoring this context for his own purpose. Through the cooperation, the total power consumption of Joey's phone is reduced from 570 to 365 mW, increasing its lifetime by about 56 percent.

2.2 Study on Cooperation Opportunity

To study cooperation opportunities in the daily life of mobile users, we analyze two public datasets on human activity and mobility behaviors. Table 2 shows their summaries.

ATUS. The American Time Use Survey (ATUS) dataset [1] includes the list of all activities of American participants over a 24 hour period and the acquaintances who were present during each activity. We use the dataset collected in 2010 from 13,258 interviewees over wide age, sex, and occupation distributions. We analyze this data to find the cooperation opportunities in everyday activities, especially in terms of the acquaintances being together.

TABLE 2
Summary Statistics of Datasets

Dataset	ATUS	MIT/BT
Data source	American time use study (interview)	Bluetooth scanning trace (period: 5 min)
Participants	13,258	100
Start time	01/01/2010	09/08/2004
Duration	1 year	3 months
# of events	257,193 activities	285,512 encounters

MIT/BT. The MIT/BT dataset is the mobility dataset collected from 100 mobile phones of MIT students and staffs [10]. It is collected by Bluetooth scanning performed every 5 minutes. We analyze the encounters between the phones, i.e., the encounters with nearby people including strangers as well as acquaintances. We use the three-month dataset from the fall semester of 2004.

2.2.1 How Many Opportunities for Cooperation?

The longer people are together with others, the more opportunities for cooperative context monitoring we can exploit. To quantify the amount of such time in everyday life, we analyze the ATUS dataset. We do not use the MIT/BT dataset here since it is limited to the devices only discoverable by Bluetooth scanning.

Fig. 2 shows the daily amount of time in terms of the presence of acquaintances for every participant. The average time with one or more acquaintances is 8.5 hours. We can confirm that people have lots of cooperation opportunities with acquaintances, i.e., more than one-third of a day. Specifically, 78 percent of the participants have more than 4 hours of co-located time with others, and 50 percent have more than 9.3 hours.

We further elaborate on with whom and how long participants spent time with acquaintances, i.e., family (average 5.9 hours), work-related people (1.7 hours), friends (0.6 hours), etc. Also, we analyze the number of acquaintances a user is together with; it gives an intuition on the number of cooperator candidates at a time. For 42 percent of the time, people are with more than one acquaintance, giving more chances of cooperation, i.e., two (23 percent), three (12 percent), four or more (7 percent). Note that the opportunities for cooperation are not limited to those with acquaintances but can include those with strangers, e.g., a

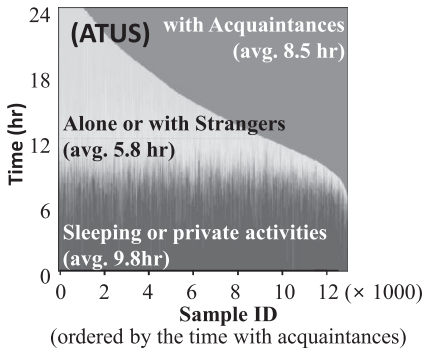


Fig. 2. Distribution of the time together.

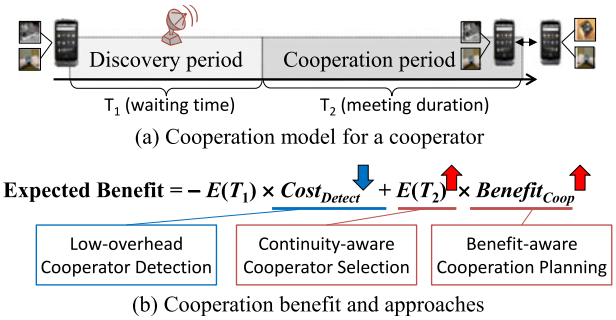


Fig. 3. Cooperation benefit model.

user riding a bus can cooperate to monitor the route of the bus with other passengers. However, the ATUS dataset does not contain encounters with strangers.

We also investigate the continuity of meetings, i.e. how long people are together during an encounter. Once cooperative monitoring has been established when they are together, this monitoring could continue as long as they remain together. Long-lasting cooperation would enable prolonged support for applications. ATUS dataset shows that 65 percent of meetings with acquaintances last for more than 30 minutes and 47 percent last for more than one hour. In case of MIT/BT data, for people who meet once a week or more, 36 percent of the meetings between them last for more than 30 minutes while only 13 percent of the meetings with the others do. From these results, we can obtain typical durations of cooperative monitoring between acquaintances. More details can be found in [30]. Also, a useful related study on grouping behavior of university students in Singapore Management University can be found in [20].

Expected energy saving in practice. At this point, a question on 'whether or not CoMon+ can achieve energy savings in practice' can naturally be raised. As discussed, actual energy saving varies depending on two key factors: instant cooperation benefit and possible cooperation time. We note that continuous monitoring of a context often require significant power consumption (e.g., 491 mW for location, 297 mW for ambient sound) which is far beyond the overheads of CoMon+ (20 mW for discovery, 164 mW to share context at 1 Hz). Thus, even a short cooperation time can lead to considerable energy saving – we present the power consumption for context monitoring and exchange for diverse contexts and parameters in Section 6.3. We also show energy saving of CoMon+ in practice through a small-scale deployment study (See Section 6.4), and more studies are to be done in the future work.

3 CoMon+ Design

3.1 Benefit-Aware Cooperation Approach

A key goal of CoMon+ is to maximize the energy benefit from the opportunistic collaboration with nearby users. To achieve this goal, our approach fully exploits the opportunity of in-situ cooperation as well as the resources of multiple personal devices.

We first model the energy benefits obtainable from cooperation as shown in Fig. 3. We divide the operation time into two periods, i.e., discovery period and cooperation period. In the discovery period, the system attempts to detect nearby cooperator candidates. This incurs a cost, which can be represented as $Cost_{Detect} \times E(T_1)$; $Cost_{Detect}$ is the average discovery cost per unit time, T_1 is the random variable of the waiting time until meeting a cooperator, and $E(T_1)$ is the expected waiting time. Once the cooperation starts with a cooperator, it can produce a benefit. We model the benefit for the cooperation period as $Benefit_{Coop} \times E(T_2)$, where $Benefit_{Coop}$ is the average benefit from the cooperation per unit time and $E(T_2)$ is the expected duration of the cooperation. Taking all the cost and the benefits into account, the expected total benefit can be evaluated as follows:

$$Expected\,Benefit = Benefit_{Coop} \times E(T_2) - Cost_{Detect} \times E(T_1).$$

To increase the expected energy benefit, we devise the process of our cooperation mechanism as below.

Cooperator detection. The first step is to detect potential cooperators periodically. Here, the interval should be carefully chosen; a long interval reduces $Cost_{Detect}$, but might decrease the potential cooperation duration, $E(T_2)$.

Cooperator selection. Increasing the cooperation period T_2 is crucial for a higher benefit. Our system predicts the expected meeting durations upon the discovery of cooperator candidates. The negotiation for cooperation starts only with the candidates who might stay together long enough to obtain benefits. We develop the continuity-aware cooperator selection method [30].

Cooperation planning. To increase the benefit per unit time, it is important to carefully determine a cooperation plan, e.g., selection of contexts to share and distribution of tasks to different devices. The planning could significantly influence the benefit from the cooperation. We develop a planning method, which makes the cooperation plan to maximize $Benefit_{Coop}$ by considering the costs of different options to use local resources. It also ensures mutual benefits to both cooperators.

Cooperation adaptation. In addition, CoMon+ handles the dynamics of local resources. The availability of local plans can vary depending on available devices and their resource status. To keep the cooperation beneficial, it is important to adapt cooperation plans to such dynamics.

3.2 Architecture Overview

We carefully design the architecture of CoMon+ applying the benefit-aware cooperation approach, as shown in Fig. 4. It runs as a middleware on top of a smartphone OS and external sensor OSes [29], [32]. CoMon+ provides mobile sensing applications with intuitive APIs, allowing them to specify the contexts of interest (e.g. location, activity) in a declarative query [23], [24]. Consider a pollution monitor that wants to monitor CO_2 level with 90 percent of accuracy every 30 seconds. Then, it specifies the query as follows:

CONTEXT CO_2 level	**ACCURACY** 90 percent
PERIOD 30 Seconds	**DURATION** Always

CoMon+ processes registered queries by leveraging cooperation opportunities with nearby users. It takes charge of all the underlying tasks for the cooperation, keeping it transparent to applications. In terms of applications, the quality of service (QoS) provided by CoMon+ might vary due to the heterogeneity of devices or dynamic system situations. We believe that the slight QoS difference caused by

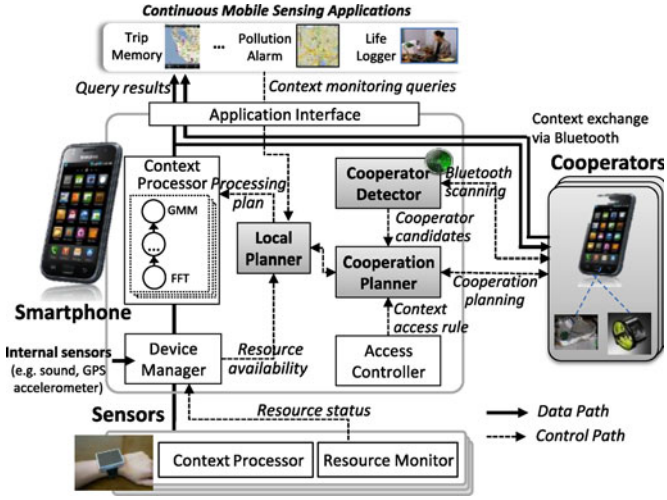


Fig. 4. CoMon+ architecture.

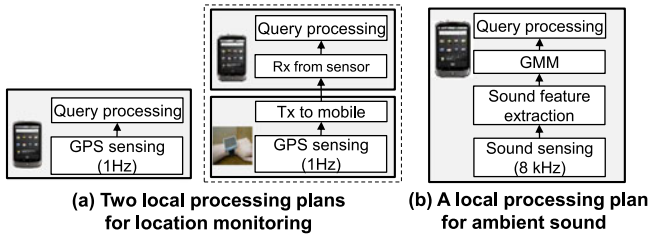


Fig. 5. Example of processing plans (simplified).

cooperation does not cause severe problems for many applications; many sensing APIs of current mobile OSes such as Android and iOS also do not guarantee fine-granule QoS. For the applications that have hard QoS requirements, CoMon+ may not initiate cooperation or can check QoS condition while planning. Note that we assume available cooperation cases satisfy the minimum accuracy requirement of relevant queries for the discussion in the later sections.

The benefit-aware cooperation approach is realized by three key components: *cooperator detector*, *cooperation planner*, and *local planner*. The *cooperator detector* dynamically discovers nearby devices by periodic Bluetooth scans at a small overhead, and selects candidates that will potentially stay in the vicinity for a long period. The *cooperation planner* negotiates with the selected one, and then decides the best cooperation plan while closely working together with the *local planner*. When there are no available cooperators any more, the *local planner* instantly updates its plans to process the contexts only with its own resources.

According to the cooperation planning, the *context processors* on the smartphone and sensors continuously process the requests and deliver the processing results to the applications and cooperators (via Bluetooth in current implementation). It incorporates a variety of modules for sensing, feature extraction, and context classification to support diverse types of contexts. The processing of a context is represented as a graph of tasks, denoted as a *processing plan*. A plan consists of a set of utilized devices and processing tasks allocated to each device. CoMon+ prepares multiple plans for a context if available and selectively utilizes them. Fig. 5 shows example plans for location monitoring and ambient sound monitoring.

The *device manager* provides the cooperation planner with up-to-date energy information, required to make a proper plan. As a basic support of privacy, CoMon+ employs *access controller*, which restricts unauthorized accesses to certain contexts. CoMon+ allows users to specify the access rules about what context information can be shared with whom.

We employ a smartphone-centered architecture; the external sensors of a user are exposed to cooperators only through a smartphone. This architecture is reasonable because many external sensors are hard to work as an independent participant for cooperation due to their lack of multi-user supports and limited resources.

In this paper, we delve into resource planning across cooperative users. Refer to ealier version of this paper [30] for the details of cooperator detection and selection.

3.3 Design Considerations and Choices

We present the key considerations in our system design.

Long-term cooperation. Dynamic changes of cooperators could incur high overheads for frequent discovery, negotiation, and connection management. To minimize such overheads, CoMon+ targets the cooperation only with long stayers. Even when a user walks around in crowded places, CoMon+ selects the cooperators only among acquaintances doing the activity together, or familiar strangers who would stay together for more than a certain amount of time. We find that cooperation opportunities are sufficient even with long-term cooperations only.

Pair-wise negotiation. When there are multiple cooperators, it is important to determine how to organize the group for cooperation planning and execution. Our key idea is to localize the effect of membership changes. CoMon+ performs the cooperation in the unit of a pair to localize the effect within some pairs. It negotiates with the cooperator candidates one at a time and incrementally continues the negotiation. An alternative approach would consider the whole group as a single cooperation unit, and perform a group-wide negotiation at once. Although this approach would lead to the group-wide resource optimum, it is relatively vulnerable to the mobility of users. Whenever a single cooperator joins or leaves, all the other cooperators should re-perform the negotiation process. Also, the group-wide negotiation significantly escalates the complexity of cooperation planning.

Context-level service as cooperation interface. For negotiation, an important design choice is the appropriate abstraction level in exposing the user's resources to cooperators. CoMon+ exposes the underlying resources of a user as context-level services. The context-level service hides the heterogeneity and dynamics of other users' resources. Also, context-level exchanges could greatly save energy which might be high if high-rate raw data are exchanged. We assume that there would be consensus on a common context model as in [46], which could help extend the scope of the cooperation. Based on such model, different applications running over heterogeneous devices can share context information. Even with the common consensus on a context, different applications may require different level of accuracies and monitoring intervals. CoMon+ can evaluate such condition in the planning process but we do not handle such cases here for simplicity.

4 Basic Cooperation Planning

In this section, we present basic cooperation planning method in detail. To simplify explanation, we first assume that a user has a single processing plan for a context. In Section 5, we extend the planning to cover multiple processing plans enabled by various sensing devices of the user.

Upon the detection of a candidate, CoMon+ conducts cooperation planning to decide which contexts to share and trade. In the planning, each user's goal is to maximize her benefit. At the same time, the system tries to ensure mutual and fair benefits for cooperators.

Providing such maximized and mutual benefits, however, is not a simple problem. A naïve solution is that cooperators take turns to monitor the common, energy-hungry contexts. The system just needs to identify the cooperators' common requests and compare the energy demands to process it locally with the ones in case of cooperation.

Such a solution works in simple cases, but it needs to be further improved to deal with complex system environments. A key challenge results from the complexity in benefit estimation. The cooperation benefit cannot be statically determined in advance; even for the same context, the benefit could vary depending on resource availability, running applications, and user policies. First, cooperators may have different policies on the energy use. For example, a user who will be outside quite a while would want to save energy as much as possible, but one who will soon go home would not mind consuming energy if he can benefit from new contexts. Moreover, the energy demand to monitor a context might be different depending on other concurrently monitored contexts. CoMon+ processes multiple contexts in a shared way; it figures out the overlapping tasks among contexts and eliminates redundancy. Thus, the cooperation benefit for a context needs to be evaluated, taking such shared evaluation into account.

The planning becomes more challenging when a user carries multiple wearable sensors with a smartphone. In this case, monitor-able contexts among users vary quite much, and sharing the common contexts only provides limited benefits. Also, the user policies could be more complex depending on the in-situ availability of sensing devices and their remaining energy. For example, a user can obtain significant benefit from location sharing, saving the battery of his smartphone. However, the benefit would be small if he has a full-charged external GPS.

4.1 Cooperation Planning Problem

To understand the problem in depth, we first clarify the problem. According to our context-level sharing principle, we describe a cooperator, u, as follows:

Definition 1. *A cooperator,* u, *is specified as:* $u = <D, S, P>$,

- D *is a set of demanding contexts,* $\{ctx_d\}$, *by applications; the set is obtained from the registered queries.*
- S *is a set of supply-able contexts,* $\{ctx_s\}$, *which the user can provide to other cooperators. CoMon+ identifies the set based on the current resource availability.*
- P *is a policy that denotes the desirable benefit from the cooperation. It is specified by the user based on his preference or resource situation. The policy is substantialized as a cost function,* $cost_P$, *within the system. If* $cost_P$ *is reduced as a result of cooperation, the cooperation is considered beneficial.*

Input: $\{ctx_d\}$, a set of contexts to monitor
Output: cost to execute $\{ctx_d\}$

1. ***EDVector*** ← ***GetEDVector***($\{ctx_d\}$)
2. *totalEC* ← 0 // *init total energy consumption*
3. for $\forall d_j$, where d_j is a device in *EDVector*
   totalEC ← *totalEC* + $weight_j \cdot$ ***EDVector***(d_j)
4. Return *totalEC*

Fig. 6. A cost function for a policy 2.

Now, given two cooperators $u_1 = <D_1, S_1, P_1>$, and $u_2 = <D_2, S_2, P_2>$, the cooperation planning problem is to find *a cooperation schedule*, *CS*, as its output for the estimated cooperation duration, where

- CS = {(ctx_c, u_i, t) | ctx_c is a context to cooperatively monitor, u_i is a cooperator in charge, either u_1 or u_2, t is a time duration to take charge},

such that $cost_{P1}$ and $cost_{P2}$ should decrease by applying CS.

4.2 Cooperation Benefits and Policies

A user can apply diverse policies to describe his preferential benefits from cooperation. We introduce useful example policies described in terms of device resources and application supportability.

Policy 1. A basic policy is to save the battery consumption of a smartphone for context monitoring. Since a phone is a generic personal computing platform utilized for diverse applications, it would be a good default policy.

Policy 2. When a smartphone works together with external sensing devices, a user might want to consider the battery status of other devices as well. According to the relative importance of each device, a policy can be defined to minimize the weighted sum of the power consumptions over distributed sensing devices.

Policy 3. In terms of application supportability, a policy can be defined as to increase the number of supported queries. CoMon+ may not continue to support some requests due to shutdown or low battery level of the corresponding devices. This policy attempts to resume the support for such requests through cooperation.

Policy 4. Sometimes, it is expected that a user will recharge the devices after a certain time, T, e.g., 3 hours. In this case, a policy could be specified to increase the running time up to 3 hours for all applications.

CoMon+ provides several system functions to allow easy specification of diverse policies as cost functions. The key primitives are *getEDVector*({*ctx*}) and *getEAVector*(). *getEAVector()* returns the remaining energy of all sensing devices. Given a set of contexts to monitor, {*ctx*}, *getEDVector*({*ctx*}) returns the expected power consumptions on relevant sensing devices. For example, *getEDVector*({*Dust*}) returns an energy demand vector, (28.5, 720.7 mW), where the elements represent the energy demands on the smartphone and dust sensor, respectively. Fig. 6 shows an example cost function, $Cost_{P2}$, realizing policy 2 based on the primitives.

To compute the energy demands, CoMon+ manages energy use profiles for the sensing, processing and

communication tasks required to monitor contexts; currently, the energy use are profiled offline while on-line profiling can be applied further. CoMon+ estimates the energy demand to execute a plan by adding the energy demands for all tasks constituting the plan. Note that CoMon+ properly reflects the effect of the shared processing; the energy demands for redundant tasks between multiple contexts are accounted only once.

We build the system functions extending our previous systems [29], [32]; they leverage such information for the coordination of multiple applications' resource use over personal devices.

4.3 Benefit-Aware Negotiation Mechanism

Careful cooperation planning is essential to ensure mutual benefits for cooperators under complications in running applications, resource availabilities, and different user policies. To address such challenges, we develop a benefit-aware negotiation mechanism. As a key idea, the mechanism pursues the fairness of opportunity to make beneficial cooperation decisions by themselves, rather than guaranteeing mutually identical benefits to the cooperators; the identical benefit is not even possible due to participants' different policies and energy availability. In this principle, the mechanism utilizes one-to-one context exchange as a first-stage negotiation unit. Each cooperator has a chance to weigh up each unit by its own cost function. For each unit, it estimates the benefit reflecting in-situ resource availability and concurrent requests. Then, the benefit is cross-validated by each cooperator to ensure mutual benefits. The cases beneficial to only one side are excluded in advance, so that the planning results ensure the mutual benefit. Finally, the mechanism allows the cooperators to take turns to select the unit of exchange, providing each one with fair opportunities to maximize its benefit.

In more detail, the mechanism introduces a *cooperation case* as an atomic unit of cooperation planning. We identify two representative types of cooperation cases as follows. Note that cooperation cases are built on a context level, hiding the low-level resource details. We describe the mechanism in perspective of a cooperator, u_1

- **Exchange of two contexts, ctx_{out} and ctx_{in},** denoted as ***case_ex*(ctx_{out}, ctx_{in})**, is a case that u_1 obtains a context ctx_{in} from u_2 in exchange of providing ctx_{out}. This case enables the participants to save the energy by delegating the costly monitoring of a context or obtain an unavailable context.
- **Co-monitoring of a context, ctx_{co}, *case_co*(ctx_{co})**, is a case that u_1 and u_2 monitor ctx_{co} in rotation. This case enables the participants to save the energy by halving the monitoring duration of the context.

Based on the cooperation cases, our planning method is performed in the following three steps.

Step 1. Cooperation case generation: First, participants generate applicable cooperation cases by exchanging their demanding and supply-able contexts, i.e., D and S. The generated cases include a set of exchange cases, EX, and a set of co-monitoring cases, CM, where

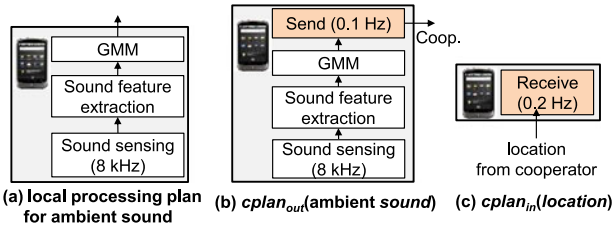


Fig. 7. (a) A local plan for ambient sound context (simplified). (b) and (c) cplans for case_ex(sound, location).

- $EX = \{case\_ex(ctx_{out}, ctx_{in}) | ctx_{out} \in (S_1 \cap D_2), ctx_{in} \in (D_1 \cap S_2), ctx_{out} \neq ctx_{in}\}$, and
- $CM = \{case\_co(ctx_{co}) | ctx_{co} \in (S_1 \cap D_1 \cap S_2 \cap D_2)\}$.

For an exchange case, ctx_{out} is the one that u_1 provides and u_2 demands. ctx_{in} is vice versa. A co-monitoring case is generated for a context that u_1 and u_2 both can provide and demand at the same time. If a cooperator has been already cooperating with another one u_3, it excludes the contexts involved in the cooperation with u_3 from its S and D for the case generation, following our pair-localized negotiation design.

Step 2. In-situ benefit estimation and cross-validation: The second step is to estimate the benefit of each generated cooperation case and exclude the cases that provide only one-side benefit. Since the benefit of a case can be different depending on each participant's policy and energy availability, the benefit estimation is separately done by each participant based on its *cost* function. Based on the estimated benefit, each participant excludes the cases that are not beneficial to the participant. Then, they exchange the list of the cases to exclude the cases that are not beneficial to the other participant as well. The cross-validation results in only mutually beneficial cases.

Details on benefit estimation. Specifically, the benefit of a cooperation case is calculated in two sub-steps: 1) introducing *a cooperation plan*, and 2) policy-based benefit calculation applying the new plan.

A cooperation case introduces a new processing plan to monitor the corresponding context. We denote such newly introduced plan as *a cooperation plan*, *cplan*, while denoting the original local plan as *lplan*. For an exchange case, *case_ex* (ctx_{out}, ctx_{in}), a new $cplan_{in}(ctx_{in})$ is created for ctx_{in}. The $cplan_{in}(ctx_{in})$ simply consists of a task to receive the results for ctx_{in} from the cooperator. For ctx_{out}, a new $cplan_{out}(ctx_{out})$ is built by inserting a task to send results at the end of its original local processing plan. Figs. 7b and 7c show example *cplans* created by the *case_ex*(*sound*, *location*). For a co-monitoring case, *case_co*(ctx_{co}), a cooperation plan $cplan_{in}(ctx_{co})$ is used for every first half of rotation epoch to receive ctx_{co} and $cplan_{out}(ctx_{co})$ is used for the second half to provide ctx_{co}.

With the new *cplans*, CoMon+ computes a new cost by using *GetEDVector*() and *GetEAVector*(). Then, the benefit is calculated by subtracting the new cost from the one before applying the *cplans*, i.e., only with the local plans.

Step 3. Turn-by-turn case selection: The final step is to select the validated cooperation cases one-by-one in turn. A participant who has a turn selects the case of the maximum estimated benefit and notifies it to the other. After the selection, the participants delete the cases associated with the contexts in the selected case. For example, if a participant selects the co-monitoring case of location context, both participants delete the cases that exchange the location context with

another context. The selection process continues until there is no case to select. Such turn-based selection provides each cooperator with fair opportunities to maximize its benefit. After the selection, the cooperation planner applies and executes the cooperation plan for the selected cases.

5 Local-Plan-Aware Negotiation

In this section, we introduce a local-plan-aware negotiation mechanism to extend the basic cooperation planning method explained in Section 4. The key purpose of extension is to maximize cooperation benefit in an upcoming multi-device personal sensing environment.

We first make several assumptions on the upcoming personal sensing environments to design the mechanism. Note that our assumptions and design considerations are firmly based on our prior system, Orchestrator [23], [32], a distributed system to coordinate resource use of multiple personal sensing devices. First, mobile users will carry multiple personal devices such as a tablet, a smart watch, and a smart glass; many mobile users already carry tablets and phones together while various smartwatches are available in the market and smartglass prototypes have been released. Second, personal devices will share sensing and processing capabilities to maximize efficiency in resource use for context monitroing. For example, GPS sensing and subsequent activity analysis can be flexibly performed either on a smartphone or a tablet, depending on their remaining battery level; such sharing capabilities were demonstrated in earlier systems such as Orchestrator [23], [32].

In such multi-device environments, the basic negotiation mechanism may not create the best possible cooperation plan. The limitation mainly results from that it considers one possible way of collaborating for a given context. For a simple example, the basic method might determine that co-monitoring location is beneficial, assuming that location sensing is designated to a smartphone, and co-monitoring location could reduce its power use almost to half. However, such cooperation might not be preferable when the user has a fully-charged tablet, which can solely take care of the location monitoring. The negotiation needs to be extended to maximize benefit even under these situations, and the problem becomes more complicated when more devices are available and more contexts need to be monitored.

5.1 Extended Cooperation Problem

The key improvement of the local-plan-aware negotiation is to incorporate multiple processing alternatives (enabled by various personal devices) for a context into the process of cooperation planning. To take multiple local plans into account, we first extend the cooperation planning problem in Section 4.1. For the purpose, we redefine a cooperator, u, specified in Section 4.1 as follows.

Definition. 2. *A cooperator,* u, *is specified as:* u = <D, S, P, LPlan>, *where*

- D, S, and P *are defined as in the basic planning (see Def intion 1).*
- $LPlan = \{lp_{i,j} | lp_{i,j}$ *is the jth local processing plan for a context* ctx_i, *where* $ctx_i \in S\}$.

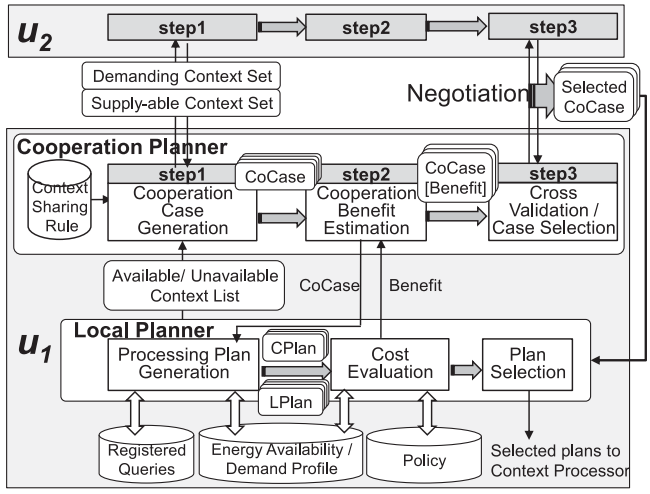


Fig. 8. Cooperation planning process incorporating local planning.

The definition is extended to have LPlan, a set of local processing plans for S. The key difference is that a context can be processed by multiple processing alternatives, namely *local processing plans (lp),* for a given context *(ctx);* this is to consider multi-device environments, where many processing alternatives might exist and influence cooperation benefit.

Now, given two cooperators $u_1 = <D_1, S_1, P_1, LPlan_1>$, and $u_2 = <D_2, S_2, P_2, LPlan_2>$, the cooperation planning problem is to find *a Cooperation Schedule, CS,* as its output for the estimated cooperation duration, where

- CS = {(ctx_c, $lp_{j,c}$, u_i, t) | ctx_c is a context to cooperatively monitor, $lp_{j,c}$ is a local plan to apply, u_i is a cooperator in charge, and t is a time duration to take charge},
  such that $cost_{P1}$ and $cost_{P2}$ should decrease by applying CS. In addition to CS, the method needs to determine a *Local Schedule, LS,* a set of contexts to locally monitor without cooperation.
- LS = {(ctx_l, $lp_{j,l}$, t) | ctx_l is a context to locally monitor without cooperation, $ctx_l \in D - \{ctx_c\}$, $lp_{j,l}$ is the local plan for ctx_l, and t is a time duration to apply the plan}.

5.2 Local-Plan-Aware Negotiation Mechanism

To address the problem, we employ a two-layered planning approach, which separates the *local planning* from the *cooperation planning*. This approach enables to isolate additional complexity introduced by multiple local plans while achieving increased cooperation benefits. Specifically, the cooperation planner negotiates with a cooperator in a context level without concerning lower-level local plans; as it is in the basic planner. Instead, the local planner prepares multiple available local plans, and informs the cooperation planner of which cooperation case would maximize benefit when considering the multiple local plans.

A key to realize this approach is to devise the local planner. We design and develop the local planner based on Orchestrator [23], [32], a system to coordinate concurrent context monitoring requests for a single user by effectively harnessing alternative local plans. Note that the cooperation planner just needs to be changed to delegate the benefit evaluation to the local planner. The overall process of this two-layered planning is depicted in Fig. 8.

Local planner. Prior to cooperation, the local planner prepares a variety of applicable processing plans for a context, exploiting diverse sensing modalities and context recognition methods. The diverse plans utilize different combination of distributed resources, and provide opportunities for the system to maximize benefits. Applicable plans are dynamically updated based on available sensors and their sensing and processing capabilities at runtime. When there is no cooperator, the planner selects and applies the best combination of plans among all the possible local plans, depending on the system policy.

Interaction between local and cooperation planner. In the step 2, the cooperation planner requests the local planner to calculate the estimated benefit for each cooperation case based on the available local plans. Given a cooperation case, the local planner first generates *CPlan*, a set of cooperation plans for the case. Different from the basic planning, multiple *cooperation plans* can be generated for each cooperation case based on alternative local plans. For example, given an exchange case, $case\_ex(ctx_{out}, ctx_{in})$, multiple $cplan_{out}(ctx_{out})$ are generated from the corresponding local plans for ctx_{out} while there is one $cplan_{in}(ctx_{in})$. Then, the local planner determines a set of plans to execute, $Plan_e$ (a subset of $CPlan \cup LPlan$), which minimizes $Cost_p(Plan_e)$.

5.3 Adaptation

CoMon+ dynamically adapts its cooperation plan to obtain continuous benefit from cooperation. There are multiple triggers to initiate adaptation. First, it is obvious to find a new plan when a new cooperator is detected or an existing cooperator disappears. In addition, given our definition of a cooperator, $u = <D, S, P, LPlan>$, planning needs to be re-performed when there are changes in the set of demanding contexts, D, the set of supply-able contexts, S, the applied policy, P, and changes in LPlan. Any of these changes may invalidate the cooperation benefit expected from the previous negotiation.

Upon the detection of such changes, CoMon+ incrementally adapts its cooperation plan. Such incremental adaptation prevents severe negotiation overheads and delays to regenerate and redeploy the whole plans. Here, we briefly describe different cases of the adaptation.

First, upon the discovery of a new cooperator, new cooperation cases are generated only for the non-cooperating contexts. When an existing cooperator disappears, CoMon+ instantly performs local planning for the contexts that had been provided by the cooperator. If there are other remaining cooperators, a negotiation can be newly initiated. Second, upon the changes in the registered queries and available sensors, CoMon+ performs cooperation planning only for the contexts affected by the changes. If a local plan becomes no longer applicable for a cooperating context, the plan is replaced with another available local plan providing cooperation benefit. If there is no replaceable plan, it additionally performs cooperation planning with an existing cooperator only regarding the context. If a co-monitoring context is deregistered, the execution of the corresponding plan is stopped and the case is invalidated. Accordingly, the relevant cooperator is notified of it.

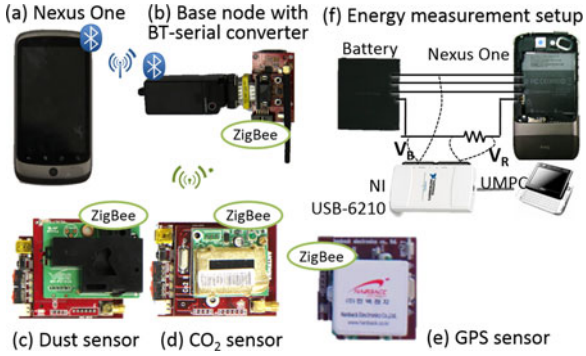


Fig. 9. Hardware and energy measurement setup.

6 Experiments

We prototyped CoMon+ on Android phones and various types of sensor devices. Fig. 9 shows our hardware setup. We used Google Nexus One with 1 GHz CPU, 512 MB RAM. We connect a base sensor node to Nexus One via Bluetooth-to-serial converter to support ZigBee communication between Nexus One and sensor devices. We used commercially available ZigbeX sensor motes running TinyOS 1.1.11. They are equipped with Atmega 128L MCU, CC2420 RF transceiver supporting ZigBee protocols, and an additional extension board of dust and CO_2 sensors. We developed mobile-side CoMon+ architecture as a background service on the Android platform. On the sensors, we implemented the sensor-side architecture in NesC.

To demonstrate the effectiveness of CoMon+, we evaluate the system based on the aforementioned prototype. First, we present the energy benefit achieved in diverse cooperation cases. Second, we show that our cooperation planning method effectively provides mutual benefit. Third, we investigate the overhead for the cooperation. Fourth, we examine the end-to-end energy saving by the CoMon+ platform through a small-scale real deployment experiment. Lastly, we further evaluate CoMon+ incorporating local planning through simulation-based study to extensively investigate the effect of varying system parameters. For the power measurements, we used a data acquisition tool, NI USB-6210, as shown in Fig. 9f.

6.1 Energy Benefits of Cooperation

We evaluate the energy benefits achieved by cooperative context monitoring. We measure and analyze the energy saving of the smartphone and the sensor devices for basic cooperation cases, i.e., co-monitoring cases and exchange cases. For the experiments, we configured two Nexus One phones to cooperate each other. We used a phone-embedded GPS device for location monitoring. For dust and CO_2 monitoring, each phone was connected with two external sensor devices, i.e., a dust sensor and a CO_2 sensor. We compare the power consumption after applying the cooperation cases against the non-cooperative, standalone setting.

For the detailed analysis, we break down the power consumption into three parts; *base*, *monitoring*, and *transmission*. The *base* represents the power consumption for the primitive operations of the smartphones and the sensor devices, i.e., the power consumed by CPU of the idle state. The *monitoring* includes the power consumed by sensors and CPU

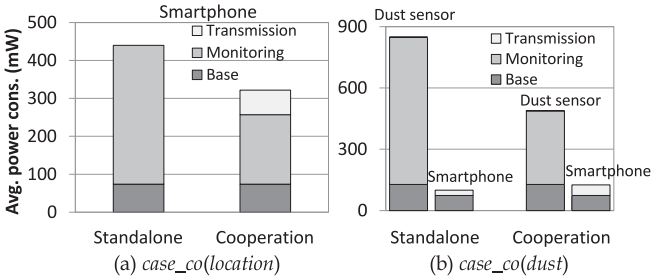


Fig. 10. Power consumptions for co-monitoring case.

for sensing and data processing. For example, in case of location monitoring, it represents the power consumed mainly by the phone-embedded GPS device. The *transmission* represents the power to send and receive the data. This includes both cases to communicate with a person's own external sensor devices and the cooperators; both communications are done via Bluetooth.

Co-monitoring cases. Figs. 10a and 10b show the average power consumption for two co-monitoring cases, *case_co* (*location*) and *case_co*(*dust*), respectively. Fig. 10a shows that CoMon+ achieves 27 percent of power saving on the smartphone through the co-monitoring of the location context, i.e., from 440 to 321 mW. We expect that the energy saving extends the lifetime of the smartphone by 37 percent. The major contribution comes from halving the total duration of GPS activation; the power consumed by GPS for the *monitoring* decreases from 366 to 183 mW. The amount of power saving is less than the exact half due to the base consumption and the Bluetooth transmission for context exchange. For cooperation, the smartphone additionally consumes 65 mW for the Bluetooth transmission.

Fig. 10b shows the results for *case_co*(*dust*), which employs an external dust sensor. CoMon+ reduces the power consumption of the dust sensor by 43 percent (from 848 to 487 mW), since it is turned off for a half of the monitoring duration and thus the average power consumption for the *monitoring* by the dust sensor decreases from 720 to 360 mW. On the other hand, the power consumption of the smartphone slightly increases by 26 mW as it transmits the monitoring results to the cooperator during its monitoring turn. This overhead is marginal in most cases; this is because, even in the standalone setting, the smartphone consumes the power for the *transmission* by Bluetooth, to receive the data from the external sensor device. Taking such overheads or not is governed by the user's policy.

Exchange cases. Fig. 11a shows the average power consumption for two exchange cases: when the user takes charge of CO_2 in return of location (*case_ex*(CO_2, *location*)), and vice versa (*case_ex*(*location*, CO_2)). For *case_ex*(CO_2, *location*), CoMon+ significantly reduces the power consumption of the smartphone (492 to 142 mW) by deactivating its GPS; the additional cost to deliver its CO_2 context is insignificant, i.e., 7 mW. The consumption of the CO_2 sensor remains the same at 251 mW. In contrast, for *case_ex*(*location*, CO_2), the power consumption of CO_2 sensor is largely reduced from 251 to 129 mW, whereas the smartphone slightly consumes 9 mW of more power to transmit the location context. Fig. 11b shows the exchange cases of CO_2 and *dust* contexts. These cases provide similar energy benefits as shown in Fig. 11a.

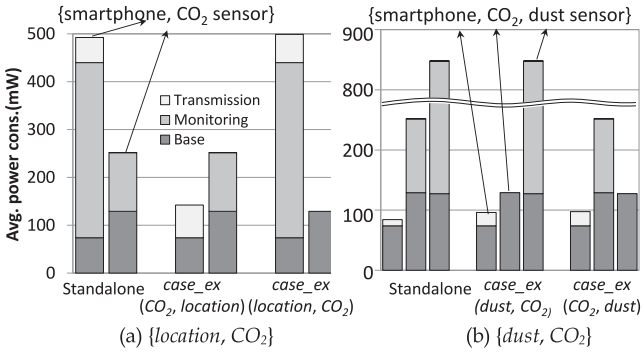


Fig. 11. Power consumptions for exchange cases.

6.2 Cooperation Planning for Mutual Benefit

We validate our cooperation planning mechanism and its effectiveness in terms of mutual benefits. We conducted an experiment with three users, u_A, u_B, and u_C, each having different devices and monitoring queries (see Fig. 12). We investigate how cooperation planning is performed when the users come across, stay with, and leave each other. Fig. 13 depicts five phases separated by the users' meeting and parting events as well as the event of local resource status change. We first show u_A's viewpoint in details and verify the actual energy benefits. Then, we briefly present the benefit from the viewpoint of u_B and u_C. We set different cooperation policies for the users: u_A wants to maximize the energy saving only for her smartphone, whereas u_B wants to increase the number of activated queries and u_C wants to maximize the total energy saving of the smartphone and the sensor devices.

Phase 1. u_A registers her location and ambient sound monitoring queries. As there is no cooperator, all those queries are processed by u_A's own resources. She has two local plans for location monitoring which use her smartphone and external GPS sensor. According to her policy to maximize the energy saving of the phone, location monitoring is performed with the external GPS sensor. Fig. 13c shows the power consumption of u_A's smartphone and two sensors in Phase 1, i.e., 330, 397, and 127 mW, respectively. That of dust sensor is the base power consumption.

Phase 2. Phase 2 begins when u_A meets u_B. Upon meeting each other, their CoMon+ systems start the cooperation planning process. By exchanging their demanding and suppliable contexts, both of them generate four cooperation cases, *case_co*(*location*), *case_ex*(u_A:*dust*, u_B:*ambient*), *case_ex* (u_A: *dust*, u_B:*location*), *case_ex*(u_A:*location*, u_B:*ambient*). They estimate the expected benefit of each case, considering all available local plans. As mentioned, u_A has two local plans for location monitoring. u_A's CoMon+ determines that *case_ex*(u_A:*dust*, u_B:*ambient*) is beneficial according to her policy to maximize energy saving for smartphone. u_A's CoMon+ reduces the power consumption of its smartphone

	u_A	u_B	u_C
Queries	Location, Ambient Sound	Location, Dust	Location, CO_2 Ambient Sound
Devices	Smartphone, GPS, Dust	Smartphone	Smartphone, CO_2

Fig. 12. Experimental setup.

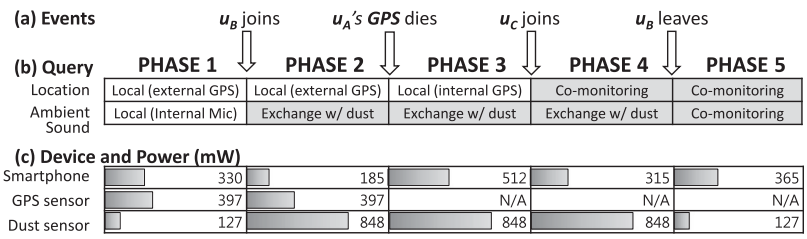


Fig. 13. Experiment results from the viewpoint of u_A.

from 330 to 185mW at the cost of its dust sensor; that of dust sensor increases from 127 to 848 mW.

Phase 3. While u_A and u_B cooperate with each other, u_A's external GPS sensor runs out of battery. Its location monitoring plan currently executed becomes unavailable. u_A's CoMon+ adapts to this event. It replaces the plan with an available plan using the smartphone's built-in GPS sensor. Since location monitoring is not involved in the cooperation case for u_B, u_A's CoMon+ does not perform additional negotiation. The power consumption of u_A's smartphone increases to 512 mW.

Phase 4. While u_A is being together with u_B, u_C comes across them. u_A's CoMon+ starts the cooperation planning with u_C as well, generating one cooperation case, i.e., *case_co*(*location*). Note that cases regarding the ambient sound context are not generated as it is already under the cooperation with u_B. u_A's CoMon+ begins additional cooperation with u_C by applying the case; the power consumption of u_A's phone reduces from 512 to 315 mW.

Phase 5. u_B has just left u_A. Detecting the event, u_A's CoMon+ promptly adapts to the situation; it stops the dust monitoring and starts the ambient sound monitoring with its local plan using the smarthpone's mic. Accordingly, the power consumption of u_A's smartphone increases to 570 mW. u_A begins additional planning with u_C on the ambient sound which u_A has cooperatively monitored with u_B. They generate and select a cooperation case, *case_co*(*ambient*), which is mutually beneficial. This new cooperation reduces the power consumption of u_A's smartphone from 570 to 365 mW.

The benefit of u_B and u_C. In Phase 2, u_B's CoMon+ applies *case_ex*(u_A:*dust*, u_B:*ambient*) through the cooperation with u_A and makes the dust query activated by obtaining the dust data from u_A, increasing the number of activated queries. In case of u_C, *case_co*(*location*) and *case_co*(*ambient*) are applied through the cooperation with u_A in Phase 4 and 5, respectively. The power consumption of u_C's smartphone decreases from 735 to 572 and 365 mW in succession.

6.3 Cooperation Overhead

We examine the energy overhead for cooperative monitoring. We observe two major causes of overheads: to discover nearby cooperator candidates and to exchange the monitoring results. Our measurement shows that those are insignificant compared to the expected benefits.

Discovery overhead. CoMon+ conducts periodic Bluetooth scans for discovery, consuming additional energy. The overhead in terms of average power consumption is 20 mW in our default interval of 5 minutes. This is relatively small compared to the expected benefits of many cooperation cases in Section 6.1. For example, if CoMon+ has been looking for cooperators for 60 minutes, it just needs 6 minutes to break even after starting the cooperation of *case_ex*(*ambient sound*, *location*).

Context exchange overhead. We measure the smartphone's power consumptions for Bluetooth message exchanges as shown in Fig. 14. To figure out the relative amount of the overhead, we also plot the smartphone's energy cost for monitoring several example contexts. For the contexts requiring power-hungry sensing or heavy computation, the overhead to exchange a context is much smaller than the cost to monitor the context in terms of average power consumption. For instance, receiving the location context from a cooperator consumes only 60 mW of the smartphone when the monitoring interval is 30 seconds, whereas monitoring the context using phone-embedded GPS costs 393 mW. In the case of *dust* and CO_2 contexts, the smartphone does not benefit from the cooperation, but the sensor devices significantly save their energy as in Section 6.1. Other than the contexts above mainly relying on power-hungry sensing, we can also expect the cooperation benefit for some contexts involving long CPU wakelock due to its high power cost. On Nexus S, the CPU wakelock consumes 252 mW of power even without any processing workload. More power would be consumed if complex processing logics are executed, e.g., HMM, and GMM. Nonetheless, in some cases the cooperative monitoring might not provide the energy benefit, i.e., when the monitoring cost is less than the exchange overhead. Such cases are excluded from cooperation options in the cooperation planning process.

Cooperation on Nexus 5. We investigate the cooperation overheads and context monitoring costs on a more recent mobile device and OS, Nexus 5 with Android 5.1.0. The discovery overhead is 9 mW in the interval of 5 minutes. Fig. 15 shows the power consumption for Bluetooth

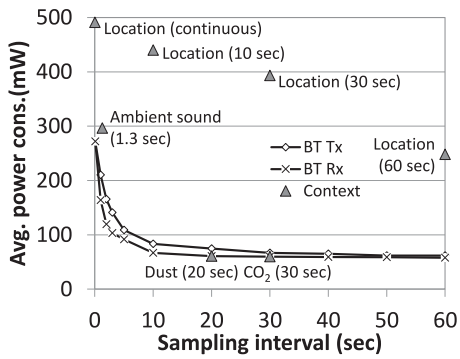


Fig. 14. Exchange overheads and monitoring costs on Nexus One in terms of avg. power consumption; the dust and the CO2 sensor consumes 848 and 252 mW, respectively.

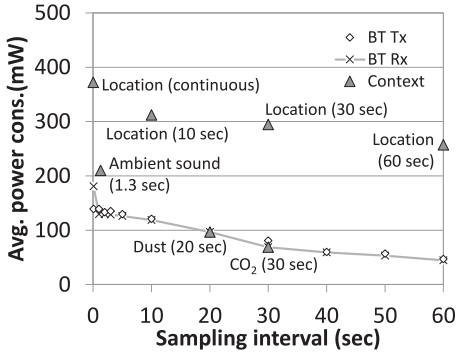


Fig. 15. Exchange overheads and monitoring costs on Nexus 5 in terms of avg. power consumption.

message exchanges and context monitoring. While the power consumption by Nexus 5 is generally less than that by Nexus One, it is expected that the cooperation is still considerably beneficial. Similar to Nexus One, the overhead to exchange a context is much smaller than the cost to monitor the context on Nexus 5.

Cooperative monitoring delay. We briefly discuss potential delay incurred by cooperative monitoring and its implication. The cooperation can add delay of context monitoring since it takes time to obtain context data from the cooperator. Currently, CoMon+ relies on Bluetooth communication for data transmission between cooperators. Thus, we measure cooperative monitoring delay by taking half of round-trip time of Bluetooth message exchange. According to our measurement, the average delay was 43ms. Considering that the monitoring interval of several to tens of seconds is likely to suffice for many ambience contexts shown in Table 1 (e.g., location and temperature), this delay would be acceptable for many of ambience monitoring applications. If an application requires a quite short monitoring interval for a certain context, tens of milliseconds might not be negligible. CoMon+ can filter out such a case from potential cooperation cases via simple constraint check on registered queries.

6.4 End-to-End Energy Saving

To investigate the end-to-end energy saving by the CoMon+ platform, we prototyped a proof-of-concept application, TripMemory. It is an Android application that tracks the user's travelling path and logs her surrounding events extracted from ambient sound. Upon the start of TripMemory, it registers either one or both of the following queries to CoMon+ requesting for user preferences.

CONTEXT location	**CONTEXT** ambient sound
PERIOD 5 Seconds	**PERIOD** 10 Seconds
DURATION Always	**DURATION** Always

The query registration is performed once a day only. CoMon+ notifies the application of monitoring results through the Android service interface.

We recruited 12 participants consisting of six pairs of friends via the bulletin board of KAIST. Each participant was given a Nexus One phone with the CoMon+ platform and TripMemory installed. For comparison, each was given another phone with the same setting but deactivating the cooperation functionality of CoMon+ (named non-CoMon). For fair comparison of the energy consumption, we used brand-new batteries. The phones' battery levels are logged using Android library. We had the participants fully charge every night and not run no application but TripMemory. They roamed freely for a week.

According to our data analysis, each participant runs TripMemory 6.2 days on average and 8.6 hours per day; some forgot to run for a day. On average, a pair cooperated 5.9 hours per day across 6.8 times of meetings. The average meeting duration is longer than we expected; we guess that this is because the participants are mostly close friends who are roommates or attending classes together. CoMon+'s average battery consumption is 19.7 percent less than those of non-CoMon phones; this means that about 19.7 percent battery remains for a CoMon+ phone at the moment that the corresponding non-CoMon phone runs out of battery. Looking into the data, the cooperation benefits vary largely depending on the cooperation patterns. Only accounting for when a user turns on location monitoring, the benefits are 31.1 percent on average. When both users turn on location and ambient sound monitoring, the benefits differ for the ones providing locations and the sound contexts. For the location providers, the average benefit is 6.9 percent only while the average benefit is 22 percent for the sound providers; note that location monitoring consumes a lot more energy. We expect that the benefit of CoMon+ will increase as CoMon+ is deployed by more people and more energy-intensive context processing is performed. We plan to perform extensive experiments to understand potential benefit at scale in a large-scale mobile testbed [3].

TABLE 3
System Parameters

Parameter	Default	Range
Number of processing plans per context (N_p)	3	1-5
Number of cooperators (N_c)	3	1-5
Number of sensor nodes per cooperator (N_s)	8	4-12
Number of queries per cooperator (N_q)	10	6-14
Number of context types	15	-
Number of types of sensor nodes	15	-
Number of tasks per a processing plan	1-2	-
Number of task types per sensor node	4	-
Base power consumption of sensor node	40 mW	-
Energy demand per task	20-300 mW	-

7 Effect of Local-Plan-Aware Negotiation

We further evaluate CoMon+ with the local-plan-aware mechanism through extensive simulations. It enables fast benefit assessments with various system parameters, i.e., the numbers of cooperators and available local plans.

7.1 Parameter Setup

Table 3 summarizes the parameters with their default values. We control the parameter ranges carefully considering realistic system environments. In particular, we set the energy-related parameters reflecting the energy profiling results in Section 6.1. By default, there are three cooperators and each has 10 registered queries and eight sensor devices. The number of local processing plans per context is three. The context types of the queries and the types of sensor devices are randomly selected among 15 context types and

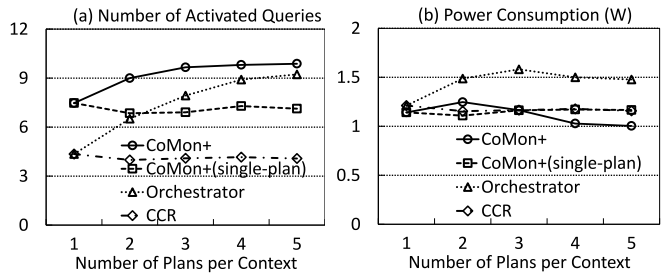


Fig. 16. Effect of number of local plans (N_p).

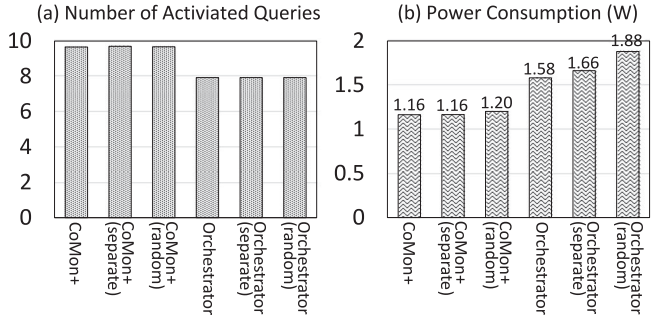


Fig. 17. Effectiveness of local planning (# of local plans: 3).

15 device types, respectively. The cooperation is sequentially done between all pairs of the cooperators. We repeated experiments for 50 times, under the same parameter setting, and report the average result.

We measure the effectiveness of CoMon+ in terms of the context richness and energy efficiency. Firstly, we measure the level of the context richness through the average number of activated queries of cooperators (NAQ). Also, as the metric of energy efficiency, we use the average power consumption of cooperators (PC). The power consumption is computed as the total sum of the power consumption of its mobile and sensor devices.

We make comparison with several alternatives to clarify the effectiveness of CoMon+ incorporating multiple local plans. The alternatives include CoMon+ using a single plan for a context, i.e., CoMon+ (single-plan), a conventional context recognizer (CCR), and Orchestrator [32]. CCR represents conventional context-aware systems that process a query with a single and fixed processing plan using local resources only. In contrast, Orchestrator incorporates and utilizes diverse processing plans for each query. Its resource use, however, is still restricted within a single user. We can consider Orchestrator as a non-cooperation version of CoMon+ and CCR as that of CoMon+ (single-plan). We use a policy that maximizes the number of activated queries (NAQ). If NAQ is the same, it tries to minimize the total power consumption of devices.

7.2 Evaluation Results

Firstly, we examine the effect of the number of local processing plans per context, N_p; it is varied from 1 to 5. Fig. 16a shows the number of activitated queries as a function of N_p. As N_p increases, CoMon+ activates more number of queries, whereas CoMon+ (single-plan) and CCR activates the same number of queries regardless of N_p. In case of Orchestrator, the number of activated queries increases, but it is smaller than CoMon+. As N_p increases, CoMon+ has more opportunity to activate queries by effectively utilizing its local plans in addition to cooperation with other users. Even if there is one local plan for a context, the both versions of CoMon+ can process more number of queries through cooperation. The context richness of Orchestrator and CCR, however, is limited by the capability of its local resources. When there are more than two local plans for a context, the NAQ of Orchestrator becomes larger than that of CoMon+ (single-plan). It is because Orchestrator can utilize multiple options to process queries only with its local resources. However, this improvement is achieved at the cost of power consumption as shown in Fig. 16b. To activate more queries, Orchestrator uses more sensor nodes, which increases overall power consumption. Interestingly, as N_p increases, the PC of CoMon+ decreases while the number of activated queries increases. CoMon+ uses the multiple options for effective cooperation to obtain more benefit, thereby decreasing PC.

We also analyze the effectiveness of our local-plan-aware cooperation. We compare CoMon+ and Orchestrator with their two variants. One is to randomly select a plan to monitor a context (random). The other is to select a plan with the smallest power consumption among multiple plans to monitor a context. The selection is done separately for each context (separate). Fig. 17 shows the results when there are three local plans for a context. The NAQ is hardly affected by the local plan selection for both CoMon+ and Orchestrator while CoMon+ outperforms Orchestrator. However, the PC varies largely. In case of Orchestrator, the PC is reduced by 300 mW than Orchestrator (random) and 80 mW that Orchestrator (separate). CoMon+ saves 40 mW compared to CoMon+ (random) with no noticeable difference between CoMon+ and CoMon+ (separate). This shows a significant benefit of local planning obtainable when there is no other available cooperators. In case of CoMon+, the effect of local plan selection decreases. It is because CoMon+ (separate) and CoMon+ (random) also benefit from the cooperation, which to some extent countervails the loss due to random selection.

Secondly, we investigate the effect of the number of cooperators, N_c, varying from 1 to 5. Note that the number of local plans per context is three. We omit the result of CCR since it is similar to the previous one. Fig. 18a shows the number of activated queries as a function of N_c. As N_c increases, CoMon+ and CoMon+ (single-plan) activate more number of queries, since they have more opportunities to activate queries through cooperation. In contrast, Orchestrator activates the same number of queries

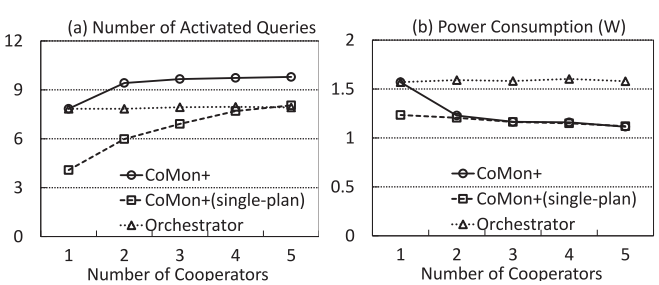


Fig. 18. Effect of number of cooperators (N_c).

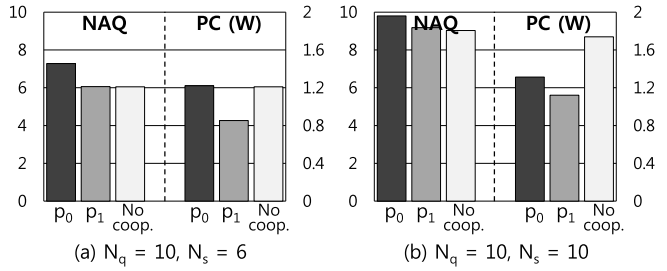


Fig. 19. Effect of different policies.

regardless of N_c. Basically, when there are multiple local plans for a context, more number of queries can be activated compared to the single plan cases; two times more number of queries are activated, when there is no other cooperator. CoMon+ can further activate more number of queries by utilizing the sensor nodes of other cooperators. CoMon+ activates 1.2 times more number of queries than Orchestrator when there are three cooperators. With more than three, almost all queries are activiated by CoMon+. The NAQ of CoMon+ (single-plan) is comparable to Orchestrator when the number of cooperators is 5.

Fig. 18b shows the average power consumption of cooperators (PC) as a function of N_c. As N_c increases, the PC of CoMon+ decreases while the number of activated queries increases. Through cooperation, CoMon+ finds opportunities to activate more queries. At the same time, CoMon+ minimizes the increase of PC to activate queries by effectively utilizing multiple options of local plans. In CoMon+ (single-plan), the decrease of the PC is marginal because the energy overhead for cooperation increases as the number of cooperators increases for higher NAQ. The PC of Orchestartor are not affected by N_c.

7.2.1 Effect of Differenct Policies

To show that CoMon+ effectively supports different cooperation policies of different users, we consider two cooperators, p_0 and p_1. They have different policies; p_0 applies the maximum NAQ policy and p_1 does the minimum PC policy which minimizes the total power consumption for energy saving. We measure the NAQ and PC of p_0 and p_1, and compare them with the no cooperation case. Figs. 19a and 19b show the results when available sensors are relatively limited ($N_s = 6$) and sufficient ($N_s = 10$), respectively. The other parameters are set to their default values. The results show that CoMon+ effectively meets the operational goals of each cooperator. When N_s is 6, p_0 activates 20 percent more queries and p_1 reduces 29 percent of PC compared to no cooperation case. When available sensors become sufficient, CoMon+ actively utilizes the sensors, increasing the resource benefit through cooperation. When N_s is 10, p_0 activates 8.5 percent more queries with 75 percent of PC compared to the no cooperation case. p_1 further reduces power consumption, 64.5 percent of the no cooperation case.

8 Discussion

Coverage of context sharing. In the current design, we simply assume the range of Bluetooth (< 10 m) as the coverage of context sharing. This works quite well in our deployment, where a pair of cooperators stays closeby during most of their meeting time. However, simply being within Bluetooth range does not ensure that two users have common contexts. For example, a user may detect another in the next room but may not have many common contexts. We believe this issue can be addressed in several ways. Exploiting Bluetooth RSSI [35] may deliver fine-grained clues on the inter-user proximity or the presence of obstacles separating them. Exchanging some contextual signature prior to cooperation may help determine if they are in the same place. Place detection techniques, e.g., SurroundSense [2] could be adopted for this purpose.

Privacy. Letting others know my context inherently raises privacy concerns. To be optimistic, we believe that such concerns might be relatively mitigated in the target environments of CoMon+, where the users are physically in the same contexts. A study on location sharing supports that people are less conscious of sharing their locations when they are closeby [7]. A study on phone sharing shows that sharing is more acceptable with those in close social relations such as families or friends [36].

To be conservative, privacy concerns largely depend on users and how the sensed contexts are to be used [26]. A study implicates that people would be highly selective during their private time depending on their context and activities [5]. In this light, CoMon+ aims to provide users with the controllability and visibility on the sharing of their contexts. First, CoMon+ allows users to specify their sharing policies, i.e., the rules governing the access to their contexts from other cooperators. Second, CoMon+ provides simple UI showing the currently shared contexts and cooperator information. We understand that, the rules and UI address only basic concerns on privacy; it is an open research question requiring in-depth studies.

Security. There might be some security issues by malicious users during cooperation. Malicious attacks might cause applications not to work properly due to the wrong data transferred by cooperators. For example, the DustAlarm application might fire a false alarm due to incorrect data about ambient dust level and perform unnecessary action, which would eventually annoy users.

CoMon+ mainly relies on acquaintances to ensure potentially long cooperation which results in desirable cooperation benefits. While such an approach can reduce the chance of malicious attacks compared to the cooperation with total strangers, we may still need to be cautious about potential malicious attacks. As a potential approach to prevent malicious users, we consider adopting reputation systems, which have been extensively studied in peer-to-peer networks and computing areas to evaluate the trustworthiness of peer users and to prevent the selfish and malicious peer behaviors [15], [45]. More specifically, CoMon+ can provide an interface to allow users to assess the validity or credit of cooperators of previous cooperation. For example, when a user performs co-monitoring, CoMon+ provides the comparison between monitored context data by cooperators and by itself. If the data from cooperator deviates from its own data too much, the user would doubt the validity of the data and mark low reputation score. Note that it is still an open problem to be addressed in the future work.

9 Related Work

Collaborative applications and techniques. Opportunistic collaboration among smartphones has drawn attention in many domains, e.g., video playback and recording [4], [42], finding emergent group activities [13], [18], [19], and context inference [38]. CoMon+ takes collaboration opportunities for different purposes, e.g., saving energy for continuous context monitoring or obtaining new sensing modalities. Also, CoMon+ is the first to incorporate personal sensing devices into cooperation beyond phones.

Collaborative sensing techniques has been proposed to incorporate new sensing modalities and enhance data fidelity [11], [27]. They share a high-level goal with CoMon+ aiming to increase the capability of individual users through the collaboration. CoMon+ conducts its in-depth study on the cooperation opportunity and resource benefits of cooperation for continuous context monitoring.

Participatory and crowd sensing. The concept of participatory sensing has been proposed to exploit the widely distributed mobile devices for urban-scale sensing applications. It has been adopted by many applications, e.g., pothole patrol [12], and has evolved into common platforms, e.g., PRISM [9]. These applications extend the spatio-temporal sensing coverage of a mobile user. Different from such works, CoMon+ aims to reduce the monitoring redundancies among the users in close proximity to save resources. CoMon+ is not a competing technology with participatory sensing but complements each other. CoMon+ can serve as a client of participatory sensing, providing the contexts in greater energy efficiencies. In the other way, CoMon+ could utilize participatory sensing to extend its spatial context coverage.

The participatory sensing concept has been extended to crowd sensing, combined with crowdsourcing. There has been active research including diverse application cases, e.g., finding a missing child [43], automatic place characterization [6], and energy efficient crowd sensing framework, e.g., PCS [28]. Similar to participatory sensing, CoMon+ and crowd sensing systems complement each other.

Energy optimization. There have been huge research efforts to reduce energy consumption for continuous sensing and data processing [37], [39]. They focus on optimizing energy use within a single device whereas CoMon+ newly attempts to optimize resource use in consideration of multiple users and devices. CoMon+ complements such techniques by further improving resource efficiency through active cooperation with nearby users.

Task offloading as in MAUI [8], Odessa [40], Gabriel [16], and Tango [14] reduces resource consumption of smartphones; heavy back-end tasks in a processing pipeline are offloaded to servers. However, CoMon+ takes cooperation approach distributing tasks over nearby devices, having benefits not provided by server-side offloading. Many sensing tasks are not transferrable to servers as the sensing itself can be performed only where the context exists. Even for processing tasks, the overhead to transfer high-rate data often overwhelms the benefit from offloading.

Our previous works provide a common underlying platform for mobile context monitoring applications [21], [22], [23], [24], [32]. They, however, focus on coordination and optimization of mobile and sensor devices from the perspective of an individual user. CoMon+ significantly extends the scope of platform to harness opportunistic cooperation between users. Also, it addresses important issues such as continuity and benefit awareness to build an effective cooperative context monitoring platform.

10 Conclusion

We present the design and implementation of CoMon+, a novel cooperative context monitoring system. We built CoMon+ by exploiting the prevailing cooperation opportunities among mobile users. CoMon+ allows every participant to take benefits from cooperation, through the continuity-aware cooperator selection and benefit-aware negotiation. Also, it employs a local-plan-aware negotiation mechanism to extend the basic cooperation between peers considering an upcoming multi-device personal sensing environment. The mechanism maximizes cooperation benefit by incorporating multiple alternatives to sense and infer a context, which extends opportunities for cooperation. We built CoMon+ prototype on off-the-shelf smartphones and diverse sensor devices and showed that it significantly improves resource efficiency for continuous mobile sensing and processing. It also extends the available contexts beyond those from one's own devices.

Acknowledgments

This work was partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2011-0018120). This work was also partially supported by the Singapore Ministry of Education Academic Research Fund Tier 2 under the research grant MOE2014-T2–1–063. An earlier version of this paper was presented at MobiSys 2012 [30]. Seungwoo Kang is the corresponding author.

References

[1] American Time Use Survey [Online]. Available: http://www.bls.gov/tus

[2] M. Azizyan, I. Constandache, and R. R. Choudhury, "SurroundSense: Mobile phone localization via ambience fingerprinting," in *Proc. 15th Annu. Int. Conf. Mobile Comput. Networking*, 2009, pp. 261–272.

[3] R. K. Balan, A. Misra, and Y. Lee, "LiveLabs: Building an in-situ real-time mobile experimentation testbed," in *Proc. 15th Workshop Mobile Comput. Syst. Appl.*, 2014, article 14.

[4] X. Bao and R. R. Choudhury, "MoVi: Mobile phone based video highlights via collaborative sensing," in *Proc. 8th Int. Conf. Mobile Syst., Appl. Services*, 2010, pp. 357–370.

[5] E. K. Choe, S. Consolvo, J. Jung, B. Harrison, and J. A. Kientz, "Living in a glass house: A survey of private moments in the home," in *Proc. 13th Int. Conf. Ubiquitous Comput.*, 2011, pp. 41–44.

[6] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, "Automatically characterizing places with opportunistic crowdsensing using smartphones," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 481–490.

[7] S. Consolvo, I. E. Smith, T. Matthews, A. LaMarch, J. Tabert, and P. Powledge, "Location disclosure to social relations: Why, when, & what people want to share," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2005, pp. 81–90.

[8] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl. Services*, 2010, pp. 49–62.

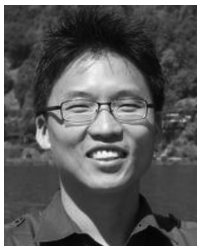
[9] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: Platform for remote sensing using smartphones," in *Proc. 8th Int. Conf. Mobile Syst., Appl. Services*, 2010, pp. 63–76.
[10] N. Eagle and A. S. Pentland, "Reality mining: Sensing complex social systems," *J. Pers. Ubiquitous Comput.*, vol. 10, no. 4, pp. 255–268, Mar. 2006.
[11] S. B. Eisenman, "People-centric mobile sensing networks," Ph.D. dissertation, Columbia Univ., New York, NY, USA, 2008.
[12] J. Eriksson, L. Girod, and B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proc. 6th Int. Conf. Mobile Syst., Appl. Services* , 2008, pp. 29–39.
[13] D. Gordon, M. Scholz, and M. Beigl, "Group activity recognition using belief propagation for wearable devices," in *Proc. ISWC*, 2014, pp. 3–10.
[14] M. S. Gordon, D. K. Hong, P. M. Chen, J. Flinn, S. Mahlke, and Z. M. Mao, "Accelerating mobile applications through flip-flop replication," in *Proc. Int. Conf. Mobile Syst., Appl. Services*, 2015, pp. 137–150.
[15] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," in *Proc. 3th Int. Workshop Netw. Operating Syst. Support Digital Audio Video*, 2003, pp. 144–152.
[16] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assitatnce," in *Proc. Int. Conf. Mobile Syst., Appl. Services*, 2014, pp. 68–81.
[17] I. Hwang, C. Yoo, C. Hwang, D. Yim, Y. Lee, C. Min, J. Kim, and J. Song, "TalkBetter: Family-driven mobile intervention care for children with language delay," in *Proc. 17th ACM Conf. Comput. Supported Cooperative Work Social Comput.*, 2014, pp. 1283–1293.
[18] I. Hwang, H. Jang, L. Nachman, and J. Song, "Exploring interchild behavioral relativity in a shared social environment: A field study in a kindergarten," in *Proc. 12th ACM Int. Conf. Ubiquitous Comput.*, 2010, pp. 271–280.
[19] I. Hwang, H. Jang, T. Park, A. Choi, Y. Lee, C. Hwang, Y. Choi, L. Nachman, and J. Song, "Leveraging children's behavioral distribution and singularities in new interactive environments: Study in kindergarten field trips," in *Proc. 10th Int. Conf. Pervasive Comput.*, 2012, pp. 39–56.
[20] K. Jayarajah, Y. Lee, A. Misra, and R. K. Balan, "Need accurate user behaviour? pay attention to groups!," To appear in *Proc. UbiComp*, Sep. 2015.
[21] Y. Ju, C. Min, Y. Lee, J. Yu, and J. Song, "An efficient dataflow execution method for mobile context monitoring applications," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2012, pp. 116–121.
[22] Y. Ju, Y. Lee, J. Yu, C. Min, I. Shin, and J. Song, "SymPhoney: A coordinated sensing flow execution engine for concurrent mobile sensing applications," in *Proc. SenSys*, 2012, pp. 211–224.
[23] S. Kang, Y. Lee, C. Min, Y. Ju, T. Park, J. Lee, Y. Rhee, and J. Song, "Orchestrator: An active resource orchestration framework for mobile context monitoring in sensor-rich mobile environments," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2010, pp. 135–144.
[24] S. Kang, J. Lee, H. Jang, Y. Lee, S. Park, and J. Song, "A scalable and energy-efficient context monitoring framework for mobile personal sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 5, pp. 686–702, May 2010.
[25] S. Kim and E. Paulos, "inAir: Sharing indoor air quality measurements and visualizations," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2010, pp. 1861–1870.
[26] P. Klasnja, S. Consolvo, T. Choudhury, R. Beckwith, and J. Hightower, "Exploring privacy concerns about personal sensing," in *Proc. IEEE 7th Int. Conf. Pervasive Comput.* , 2009, pp. 176–183.
[27] N. D. Lane, H. Lu, S. B. Eisenman, and A. T. Campbell, "Cooperative techniques supporting sensor-based people-centric inferencing," in *Proc. IEEE 6th Int. Conf. Pervasive Comput.*, 2008, pp. 75–92.
[28] N.D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha, "Piggyback crowdsensing (PCS): Energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities," in *Proc. 11th ACM Conf. Embedded Netw. Sensor Syst.*, 2013, article 7.
[29] Y. Lee, S. S. Iyengar, C. Min, Y. Ju, S. Kang, T. Park, J. Lee, Y. Rhee, and J. Song, "MobiCon: A Mobile context-monitoring platform," *Commun. ACM*, vol. 55, pp. 54–65, 2012.
[30] Y. Lee, Y. Ju, C. Min, S. Kang, I. Hwang, and J. Song, "CoMon: Cooperative ambience monitoring platform with continuity and benefit awareness," in *Proc. 10th Int. Conf. Mobile Syst., Appl. Services*, 2012, pp. 43–56.
[31] Y. Lee, et al., "SocioPhone: everyday face-to-face interaction monitoring platform using multi-phone sensor fusion," in *Proc. 11th Int. Conf. Mobile Syst., Appl. Services*, 2013, pp. 375–388.
[32] Y. Lee, C. Min, Y. Ju, S. Kang, Y. Rhee, and J. Song, "An active resource orchestration framework for PAN-scale, sensor-rich environments," *IEEE Trans. Mobile Comput.*, vol. 13, no. 3, pp. 596–610, Mar. 2014.
[33] J. Lester, C. Hartung, L. Pina, R. Libby, G. Borriello, and G. Duncan, "Validated caloric expenditure estimation using a single body-worn sensor," in *Proc. 11th Int. Conf. Ubiquitous Comput.*, 2009, pp. 225–234.
[34] R. LiKamWa and L. Zhong, "Starfish: Efficient concurrency support for computer vision applications," in *Proc. 13th Annu. Int. Conf. Mobile Syst., Appl. Services*, 2015, pp. 213–226.
[35] S. Liu and A. Striegel, "Accurate extraction of face-to-face proximity using smartphones and Bluetooth," in *Proc. 20th Int. Conf. Comput. Commun. Netw.*, 2011, pp. 1–5.
[36] Y. Liu, A. Rahmati, Y. Huang, H. Jang, L. Zhong, Y. Zhang, and S. Zhang, "xShare: Supporting impromptu sharing of mobile phones," in *Proc. 7th Int. Conf. Mobile Syst., Appl. Services*, 2009, pp. 15–28.
[37] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The Jigsaw continuous sensing engine for mobile phone applications," in *Proc. 8th ACM Conf. Embedded Netw. Sensor Syst.*, 2010, pp. 71–84.
[38] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell, "Darwin phones: The evolution of sensing and inference on mobile phones," in *Proc. 8th Int. Conf. Mobile Syst., Appl. Services*, 2010, pp. 5–20.
[39] J. Paek, J. Kim, and R. Govindan, "Energy-efficient rate-adaptive GPS-based positioning for smartphones," in *Proc. 8th Int. Conf. Mobile Syst., Appl. Services*, 2010, pp. 299–314.
[40] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proc. 9th Int. Conf. Mobile Syst., Appl. Services*, 2011, pp. 43–56.
[41] R. Sen, Y. Lee, K. Jayarajah, A. Misra, and R. K. Balan, "GruMon: Fast and accurate group monitoring for heterogeneous urban spaces," in *Proc. 12th ACM Conf. Embedded Netw. Sensor Syst.*, 2014, pp. 46–60.
[42] G. Shen, Y. Li, and Y. Zhang, "MobiUS: Enable together-viewing video experience across two mobile devices," in *Proc. 5th Int. Conf. Mobile Syst., Appl. Services*, 2007, pp. 30–42.
[43] H. Shin, T. Park, S. Kang, B. Lee, J. Song, Y. Chon, and H. Cha, "CosMiC: Designing a mobile crowd-sourced collaborative application to find a missing child in situ," in *Proc. 6th Int. Conf. Human-Comput. Interaction Mobile Dev. Services*, 2014, pp. 389–398.
[44] Walkmeter [Online]. Available: http://itunes.apple.com/us/app/walkmeter-gps-walking-stopwatch/id330594424?mt = 8
[45] R. Zhou and K. Hwang, "PowerTrust: A robust and scalable reputation system for trusted peer-to-peer computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 4, pp. 460–473, Apr. 2007.
[46] T. Gu, H. K. Pung, and D. Q. Zhang, "A service-oriented middleware for building context-aware services," *J. Netw. Comput. Applications*, vol. 28, no. 1, pp. 1–18, Jan. 2005.

Youngki Lee received the PhD degree in computer science from KAIST. He has been an assistant professor at Singapore Management University since March 2013. He has broad research interests in building experimental and creative software systems, which covers a wide design spectrum across operating systems, applications, and users. More specifically, his research interest lies in building underlying mobile and sensor platforms to enable always-available and highly enriched awareness on human behavior and contexts. He is also interested in building innovative lifeimmersive mobile applications in various domains such as daily well-being, childcare, and advertisement in collaboration with domain experts.

Seungwoo Kang received the PhD degree in computer science from KAIST. He is an assistant professor in the School of Computer Science and Engineering, Korea University of Technology and Education (KOREATECH). His research interests include mobile and ubiquitous computing, mobile sensing systems, mobile system support for healthcare, mobile social computing systems and applications, and urban-scale context computing.

Chulhong Min is currently working toward the PhD degree in the School of Computing, KAIST. His research interests include mobile and pervasive computing systems, ubiquitous services, mobile and sensor systems, and social and culture computing.

Younghyun Ju received the PhD degree in computer science from KAIST. He is currently a researcher at NAVER LABS. His research interests include mobile and pervasive computing, platform support for context-aware services, and large-scale distributed systems.



Inseok Hwang received the PhD degree in computer science from KAIST. He has been a research staff member at IBM Research - Austin since July 2014. His research interests primarily lie in the intersection of innovative user experiences and ubiquitous context awareness technologies. Accordingly, his research agenda ranges over a number of interdisciplinary verticals including next-generation welfare, healthy social life, and human augmentation along with supporting technologies such as mobile systems, sensing and inference, and user interfaces.

Junehwa Song received the PhD degree in computer science from the University of Maryland at College Park. He is a professor in the School of Computing, KAIST. His research interests include mobile and ubiquitous systems, Internet technologies, and multimedia systems.