Decentralized Constraint Checking for Pervasive Computing

Chang Xu and S.C. Cheung
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{changxu, scc}@cse.ust.hk

Abstract

As computing becomes more pervasive, smart computing devices are increasingly connected. Applications provide pervasive services based on contexts of their host devices as well as other connected devices. However, the use of contexts commonly follows a centralized approach, i.e., copying all required contexts of other devices to one device that performs a computing task. This causes space overhead and privacy threats to resource-limited, personal devices. In this paper, we propose a constraint-based approach to systematically decentralizing constraint-expressed computing tasks to connected devices. This approach is expressive to support pervasive computing tasks such as situation evaluation to complete in a device-collaborating way. We show that the approach is effective for reducing space cost and protecting device privacy in pervasive computing.

1. Introduction

With the advance of technologies such as wireless communication and sensor networks, pervasive computing is gaining rapid growth. Applications in pervasive computing sense their environments based on the notion of *context*, and provide user-preferred services. Here, *context* refers to any information about features of a computing environment. For example, a user's personal digital assistant (PDA) may select the nearest printer in a library to print the user's document for easy access. This capability is known as *context-awareness*.

In recent years, various application frameworks and middleware infrastructures have been proposed for pervasive computing. Most of them assume that the host device for pervasive computing is a powerful machine, and commonly follow a centralized approach. For example, Henricksen and Indulska [1] presented a software engineering framework to manage contexts in a centralized database and interpret them through multiple layers. Roman et al. [6] proposed a centralized middleware infrastructure to organize and schedule pervasive devices and resources. Such an approach is straightforward for powerful machines but not suitable for portable computing devices (e.g., mobile phones and PDAs). These devices usually have limited computing resources (i.e., CPU, memory, and battery power) and cannot support large databases and CPU-intensive operations.

Some pieces of work tackle the context-awareness issue from a decentralized perspective. TinyDB[1] is a specially designed database for sensor networks. It resides at each sensor node to support SQL-alike queries to collect sensory data. Due to resource limitations, each TinyDB maintains only recent sensory data, and restrictions have to be enforced (e.g., no sub-queries, and only simple arithmetic and logical operators available) [4]. Middleware infrastructures EgoSpaces [3] and Lime [5] support portable computing devices. They use the notion of *tuple space* to manage contexts of different devices in a model transparent to applications. This model protects the detail of copying contexts from other devices, and performs computing tasks as if all necessary contexts were local. Although some decentralized operations (e.g., triggering remotely registered patterns) are supported (no need for copying contexts), the model's expressive power is too limited for sophisticated computing tasks.

Recent advance of pervasive computing has shown the need for sophisticated computing tasks such as *situation evaluation* [1][6] and *inconsistency detection* [2] [7]. These tasks cannot be adequately supported by existing decentralized models [3][4][5] as discussed. On the other hand, some projects attempt comprehensive models to support pervasive computing but follow a centralized approach [1][2][6][7]. This makes portable computing devices unsuitable for pervasive computing due to their resource limitations and privacy concerns. We envision a new computing model that supports sophisticated computing tasks as well as being able to decentralize them to connected devices for collaborative completion. This model does not have to explicitly copy contexts across devices, and thus reduces space overhead and alleviates privacy threats.

[1] http://telegraph.cs.berkeley.edu/tinydb/.

In the following Section 2, we use an example to show the inadequacy of existing approaches for a situation evaluation task. We present our constraint-based computing model in Section 3 and its discussions in Section 4. Finally, we conclude the paper in Section 5.

2. Running example

Consider a conference reception where each attendee carries a PDA. Every PDA has its user's friend list (social relation context). All PDAs form several ad hoc networks using Bluetooth technology. Thus every PDA also knows what attendees are around its user, i.e., in its connection range (people location context).

Peter and Essie are friends, and they are attending the reception. Peter would like to talk to any surrounding conference attendee who he does not know (i.e., not in his friend list) but is Essie's friend. How does his PDA work out whether such people exist?

Figure 1 illustrates an example scenario. Since each PDA maintains its own user's contexts, neither Peter's nor Essie's PDA can complete this computing task individually. Peter's PDA knows who are around him and whether they are Peter's friends, but only Essie's PDA can tell whether Essie knows them.

A centralized approach [1][6] would copy Essie's social relation contexts to Peter's PDA, or copy Peter's people location and social relation contexts to Essie's PDA. Then, either PDA has sufficient contexts to complete this task. However, this approach would cause large memory overhead when Essie has many friends or Peter is surrounded by many people. Besides, both of them may refuse to share own contexts to the other due to privacy concerns.

Existing decentralized approaches [3][4][5] cannot adequately support this computing task. TinyDB [4] propagates only queries that can be processed by single sensor nodes due to its restricted SQL syntax. EgoSpaces [3] and Lime [5] support remote pattern matching operations only, or still copy necessary contexts to one device in a way transparent to applications.

This is a simple example that shows the inadequacy of existing approaches for *a computing task whose required contexts are across multiple devices*. We note that such computation is common for pervasive computing applications [6][9], as nowadays applications are more capable for performing sophisticated computing tasks rather than simple context queries.

3. Constraint-based computing model

In this section we present our constraint-based computing model. We define our context model and constraint language, and show their expressiveness in constructing sophisticated computing tasks. We then discuss how to perform these constraint-based tasks in a decentralized, device-collaborative way.

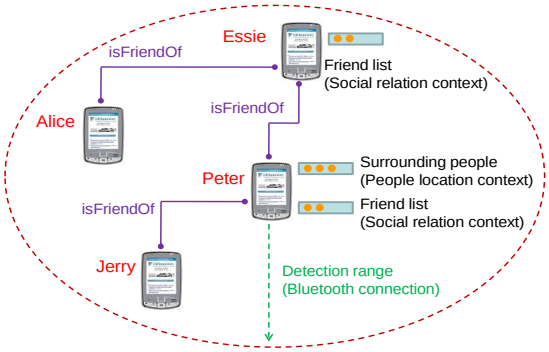

Figure 1. Peter would like to talk to Alice, who is not in his friend list but knows Essie

3.1 Context model and constraint language

A *context* is modeled as a tuple that contains multiple fields, and each field is a name-value pair <name, value>. For example, context c <<subject, "Alice">, <action, "near">, <object, "Peter">> represents that "Alice is near Peter".

A *pattern* is a tuple that contains conditions over fields of a context. For example, Pattern p <<action, "near">, <object, "Peter">> gives conditions for any context to have "near" as its action value and "Peter" as its object value.

A context c *matches* a pattern p (represented as $c \infty p$) if all conditions in p are satisfied in the fields of c.

We define a First-Order Logic (FOL) based constraint language as follows:

$f ::= \forall \gamma \infty p\ (f) \mid \exists \gamma \infty p\ (f) \mid (f) \wedge (f) \mid (f) \vee (f) \mid (f) \rightarrow (f) \mid \neg (f) \mid b(\gamma, \ldots, \gamma)$

f represents a recursively constructed formula. γ is a variable that represents any context matching pattern p. b is a user-defined function that accepts variables as input and returns a truth value (true $\top$ or false $\perp$).

For the running example in Section 2, let patterns
p_L: <<action, "near">, <object, "Peter">>,
p_{R1}: <<action, "isFriendOf">, <object, "Essie">>,
p_{R2}: <<action, "isFriendOf">, <object, "Peter">>.
Then, the following expression $\text{sit}_{\text{Peter}}$:

$\exists \gamma \infty p_L\ ((\exists \gamma_{r1} \infty p_{R1}\ (\text{sameSubject}(\gamma, \gamma_{r1}))) \wedge (\neg\ (\exists \gamma_{r2} \infty p_{R2}\ (\text{sameSubject}(\gamma, \gamma_{r2})))))$

represents the question: *Does any person around Peter, who is Essie's friend but Peter does not know, exist?* Function sameSubject tells whether two contexts refer to the same person (subject).

The constraint language has FOL-based expressive power to specify complex questions over multiple con-

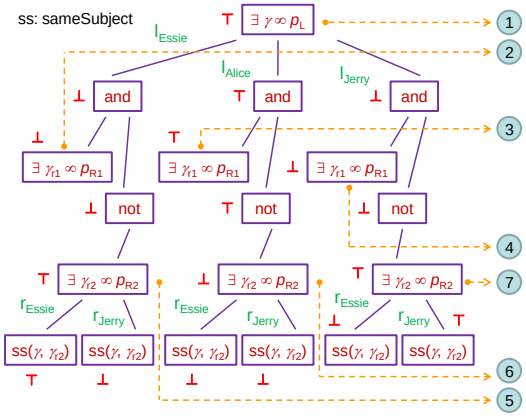

Figure 2. Peter's PDA evaluates $\mathsf{sit}_{\mathbf{Peter}}$, and broadcasts it for assistance of other PDAs

texts (via patterns) and their relationships (via *b* functions). These contexts can be resident at any devices.

3.2 Decentralized constraint checking

As contexts can be anywhere, answering a constraint-expressed question needs collaboration of all connected devices. Our approach does not copy all required contexts to one device. Instead, it decomposes a question and distributes its parts to other devices. Other devices may also do so if necessary. This approach is named *decentralized constraint checking* (DCC). We take an existential formula "$\exists\gamma \infty\ p\ (f)$" for example to explain our idea.

Let V be set of variables and C be set of contexts. α represents a *variable assignment* $\wp(V\times C)$ that contains mappings between variables and contexts. We define a *truth value function* $\mathcal{T}$ that accepts a formula and a variable assignment, and returns the formula's truth value under the given variable assignment. The following gives the truth value semantics of existential formulas:

$$\mathcal{T}[\exists\gamma\infty\ p\ (f)]_\alpha = \bot \vee \mathcal{T}[f]_{\alpha\cup\{(\gamma, c_1)\}} \vee \ldots \vee \mathcal{T}[f]_{\alpha\cup\{(\gamma, c_n)\}} \mid c_i \in \mathcal{M}[p]$$

In the semantics, $\mathcal{M}$ is the *match function* ($\mathcal{M}: P \rightarrow \wp(C)$) that finds all matched contexts for a given pattern ($\in P$ set of patterns). Since any device may not hold all matched contexts for a given pattern, DCC needs the assistance of other devices to evaluate the truth value for this formula.

DCC regards a formula as a *task*, and decomposes it into *subtasks* for distribution at other devices. Both tasks and subtasks are represented as a tuple formed by a formula and a variable assignment. DCC defines the following primitives:

deploy (t): broadcast task t to all connected devices;
collect (t): collect task t's results from other devices;
return (r): set r as the current task's result;

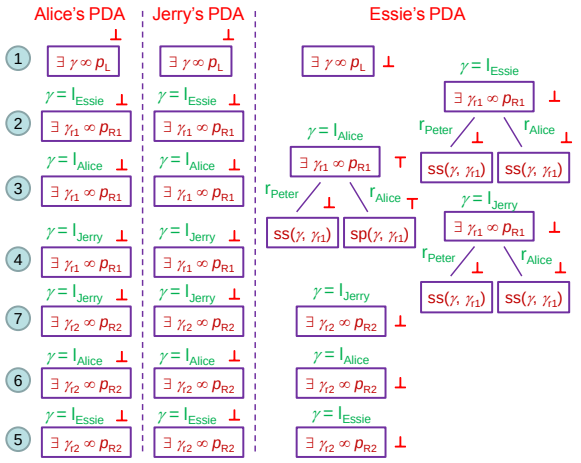

Figure 3. Alice's, Jerry's, and Essie's PDAs assist Peter's PDA for the evaluation of $\mathsf{sit}_{\mathbf{Peter}}$

reply (r): set r as the current task's result and reply it to the device that broadcasts this task;
local (p): find matched local contexts for pattern p.

The following gives the new truth value semantics in a decentralized manner:

$$\mathcal{T}_{Global}[\exists\ \gamma\infty\ p\ (f)]_\alpha = \{\ \mathsf{deploy}\ (\mathcal{T}_{Local}[\exists\ \gamma\infty\ p\ (f)]_\alpha);\ (r_1, \ldots, r_m) = \mathsf{collect}\ (\mathcal{T}_{Local}[\exists\ \gamma\infty\ p\ (f)]_\alpha);\ \mathsf{return}\ (r_1 \vee \ldots \vee r_m)\ \}$$

$$\mathcal{T}_{Local}[\exists\ \gamma\infty\ p\ (f)]_\alpha = \mathsf{reply}\ (\bot \vee \mathcal{T}_{Global}[f]_{\alpha\cup\{(\gamma, c_1)\}} \vee \ldots \vee \mathcal{T}_{Global}[f]_{\alpha\cup\{(\gamma, c_n)\}} \mid c_i \in \mathsf{local}(p))$$

As shown by the semantics, DCC evaluates a formula's truth value from a global perspective ($\mathcal{T}_{Global}$) and a local perspective ($\mathcal{T}_{Local}$). The former evaluates the truth value based on all contexts at connected devices, whereas the latter considers local contexts only at one device. We note that even a local evaluation may also decompose a task into subtasks for distribution at other devices if the associated formula's sub-formula refers to contexts outside the scope of the current device.

We omit the discussion of other formulas.

3.3 Illustration

Consider the running example. We assume that: (1) Essie, Alice, and Jerry are near Peter, (2) Peter and Alice are Essie's friends, and (3) Essie and Jerry are Peter's friends, as shown in Figure 1. These contexts are represented in our context model as follows:

$\mathcal{M}[p_L] = \{\mathsf{l}_{Essie}, \mathsf{l}_{Alice}, \mathsf{l}_{Jerry}\}$;
$\mathcal{M}[p_{R1}] = \{\mathsf{r}_{Peter}, \mathsf{r}_{Alice}\}$; $\mathcal{M}[p_{R2}] = \{\mathsf{r}_{Essie}, \mathsf{r}_{Jerry}\}$.

To learn whether there is any person near Peter, who is not Peter's friend but knows Essie, we evaluate the expression sit_{Peter} using DCC in Figure 2 and Figure 3.

Peter's PDA starts the evaluation of sit_{Peter} with its local contexts. It broadcasts to other devices in its con-

nection range when required contexts may be outside its scope. Alice's, Jerry's, and Essie's PDAs assist Peter's PDA for the evaluation with their contexts. As both $\mathcal{M}$ [p_L] and $\mathcal{M}$ [p_{R2}] are local to Peter's PDA, it undertakes most part of the evaluation. Essie's PDA also participates in the evaluation as it has contexts in $\mathcal{M}$ [p_{R1}]. Other devices (i.e., Alice's and Jerry's PDAs) do not have relevant contexts and thus just return.

4. Discussions

We discuss DCC's properties (soundness and distribution), complexities (space, time, and communication), and privacy protection capability in the following.

Soundness. We refer to the approach that evaluates constraint-based expressions with all local contexts as *centralized constraint checking* (CCC). DCC is sound in that it always returns the same results as CCC does at one device that holds all contexts. The soundness can be proved by induction. We omit the proof.

Distribution. DCC distributes the evaluation of constraint-based expressions across all connected devices without manual effort. For a device that holds any context for the evaluation, it participates in the evaluation automatically; for a device that does not hold any relevant context, it pays no effort.

Space and time complexities. DCC's distribution property implies that the space and time costs for evaluating an expression are distributed to all participating devices. DCC's space and time complexities are the same as CCC's ones. This can be proved by induction. Since each evaluation is completed by assistance of all connected devices, each device undertakes only part of total space and time costs. Then DCC is suitable for portable computing devices. Even if no device has sufficient memory for holding all contexts and space for the evaluation, it may be still possible for multiple devices to collaboratively complete the evaluation.

Communication complexity. Let m be the number of devices, s be the size of result of the match function, and l be the nested variable level of a given expression. The communication complexity for evaluating this expression is $O(s^{l-1} \cdot m)$ and each device undertakes $O(s^{l-1})$. This can be inferred by induction. As DCC does not copy contexts across devices, it trades space reduction and privacy protection with communication cost. We study the dominant factor l in 21 published applications from major pervasive computing and software engineering conferences and journals. Among 118 situations in these applications, 96 (81.4%) of them have l = 1, 20 (16.9%) have l = 2, and only 2 (1.7%) have l = 3. This suggests that in most cases (81.4%), the communication cost is small (only O(1) for each device).

Privacy protection. DCC protects each device's privacy by avoiding copying contexts across devices. Each deployed task at one device is with a variable assignment, but this does not violate the privacy. The reason is that the mappings in the variable assignment can come to different devices.

5. Conclusion

In this paper, we present a constraint-based computing model to support situation evaluations in a decentralized, device-collaborating way. The approach is expressive for modeling pervasive computing tasks, and contributes to space reduction and privacy protection for portable computing devices. The work is still in progress. We would continue to study it formally and evaluate it with simulations and real-life scenarios.

References

[1] K. Henricksen, and J. Indulska, "A Software Engineering Framework for Context-Aware Pervasive Computing", In *Proc. the 2nd IEEE Conference on Pervasive Computing and Communications*, Orlando, USA, Mar 2004, pp. 77-86.

[2] P. Insuk, D. Lee, and S.J. Hyun, "A Dynamic Context-Conflict Management Scheme for Group-Aware Ubiquitous Computing Environments", In *Proc. the 29th Annual International Computer Software and Applications Conference*, Edinburgh, UK, Jul 2005, pp. 359-364.

[3] C. Julien, and G.C. Roman, "EgoSpaces: Facilitating Rapid Development of Context-Aware Mobile Applications", *IEEE Trans. on Software Engineering 32*, 5 (May 2006), pp. 281-298.

[4] S. Madden, J. Hellerstein, and W. Hong, "TinyDB: In-Network Query Processing in TinyOS", http://telegraph.cs.berkeley.edu/tinydb/documentation.htm.

[5] A.L. Murphy, G.P. Picco, and G.C. Roman, "LIME: A Coordination Model and Middleware Supporting Mobility of Hosts and Agents", *ACM Trans. on Software Engineering and Methodology 15*, 3 (Jul 2006), pp. 279-328.

[6] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt, "A Middleware Infrastructure for Active Spaces", *IEEE Pervasive Computing 1*, 4 (Oct-Dec 2002), pp. 74-83.

[7] C. Xu, and S.C. Cheung, "Inconsistency Detection and Resolution for Context-Aware Middleware Support", In *Proc. the Joint 10th European Software Engineering Conference and 13th ACM SIGSOFT Symp. on the Foundations of Software Engineering*, Lisbon, Portugal, Sep 2005, pp. 336-345.

[8] C. Xu, S.C. Cheung, and W.K. Chan, "Incremental Consistency Checking for Pervasive Context", In *Proc. the 28th International Conference on Software Engineering*, Shanghai, China, May 2006, pp. 292-301.

[9] S.S. Yau, and F. Karim, "An Adaptive Middleware for Context-Sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments", *Real-Time Systems, 26*, 1 (2004), pp. 29-61.