Grapevine: Efficient Situational Awareness in Pervasive Computing Environments

Evan Grim, Chien-Liang Fok, and Christine Julien
The Center for Advanced Research in Software Engineering
{evangrim, liangfok, c.julien}@mail.utexas.edu

***Abstract*—Many pervasive computing applications demand expressive *situational awareness*, which entails an entity learning detailed information about its immediate and surrounding context. Previous work has largely focused on individual entities' context, in this paper we present Grapevine, a framework for efficiently sharing context information in a localized region of a pervasive computing network, using that information to dynamically form *groups* defined by their shared situations, and assessing the aggregate context of that group. We provide an implementation of Grapevine and benchmark its performance in a live pervasive computing network deployment.**

***Keywords*- context-awareness, network context, Bloom filters, context coordination**

I. Introduction

In pervasive computing, an entity's ability to acquire expressive *situational awareness* requires myriad sensing and communication activities. An entity must assess its local environment, including information about its own capabilities, status, location, and environmental conditions. It must also coordinate with other entities to assess shared local conditions including the state of the network, availability of data and resources, and social network connections. From a user's perspective, the availability of this detailed *context* information is essential for applications to integrate themselves into the environment, adapting their behavior to suit the current situation and take advantage of currently available resources.

There are many instances in which knowledge about the shared situation of a group is invaluable, particularly in emerging social scenarios. Consider an opportunistic network of mobile devices in a public park where collective context information identifies a group of people interested in a pick-up game of football and having similar skill levels that could support *ad hoc* team formation. A device on an automobile may generate an individualized group containing nearby automobiles that can collide with it. Knowledge about connection qualities in a neighborhood of a dynamic mobile network can support selecting the best routing protocol for a region of the network [8]. All of these groups are defined by their context, and the group itself exhibits an aggregate context that can in turn affect its component entities.

Our motivation differs from previous work in that we explicitly investigate the interplay between *context* and *groups*. Specifically, we tackle the practicalities surrounding two interrelated challenges in dynamic pervasive computing environments: (1) how to expressively identify *groups* of entities based on predicates over their contexts and (2) how to define, assess, and share *context* of individual entities and groups of entities. These challenges are intertwined because the groups may be defined by members' contexts, which must be continuously assessed in dynamic environments. To make the problem tractable, we untangle these two challenges into three concrete objectives. First, we define a mechanism and architecture for efficiently and effectively sharing context in a local region of a network. We then formalize the notion of a *group* and extend our architecture to compute, monitor and maintain group membership information. Finally, we demonstrate how our mechanism and architecture seamlessly support the assessment and sharing of the context of a group of entities.

The work described below is part of an ongoing research effort to synthesize a general purpose framework for coordinating context information in pervasive computing called Grapevine. In our final section we evaluate our progress to date and explore opportunities for further research.

II. Summarizing Context and Defining Groups

Our primary goal is the efficient representation of both individual and group context information to permit transmission in environments where connectivity is limited by bandwidth and energy. We minimize the transmission overhead as described in [7] by leveraging Bloomier filters, a space efficient probabilistic associative array data structure that extends the hashing constructs employed by the more commonly encountered Bloom filter data structure [2]. Bloomier filters allow for values to be associated with keys and stored with space complexity $O(rn)$, where n is the number of elements and r is the number of bits needed to represent an element's value. This compares favorably with the requirements of $O(rn \log N)$ for enumerating every context value (where N is the number of possible context items) [4] and $O(n(r+l))$ for enumerating pairs of context item labels and values (where l is the length of the label). This increased space efficiency comes at the cost of a false positive rate of $\epsilon \propto 2^{-r}$. We employ the algebraic techniques from [4] to optimize our implementation; different constructions make varying tradeoffs in space and time complexity [3], [9].

As explored in our previous work [7], individual context summaries can be aggregated into additional summaries that capture the *context of groups*. These group summaries are defined by criteria that describe group membership and a

978-1-4673-0907-3/12/$31.00 ©2012 IEEE

prescription for aggregating the individual context information into the group's context summary.

III. Grapevine: An Architecture for Context Sharing

We reduced the size of entities' and groups' contexts to reduce the overhead of sharing them. Grapevine's architecture, shown in Fig. 1, piggybacks these summaries on existing packets via the *Context Handler* and the *Context Shim*.

The *Context Handler* maintains and processes all context information generated by its host or received from the network. It computes the Bloomier Filter context summaries on demand, and provides the application updated local and remote information. It also uses received summaries to compute groups. The application notifies the *Context Handler* of relevant group definitions, and the *Context Handler* dynamically computes group membership based on the context data it receives.

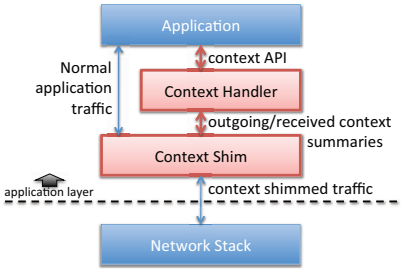

Fig. 1. Grapevine Architecture

The *Context Shim* is an *interceptor* that appends context information to every outgoing packet; we could also attach context summaries to only some subset of packets. Upon receiving a packet from the application, the *Context Shim* retrieves an up-to-date Bloomier filter context summary from the *Context Handler*, attaches it to the packet, and passes the packet down the network stack. On the receiving end, the *Context Shim* does the reverse, removing context summaries from incoming packets, sending context information to the *Context Handler*, and passing the remainder to the application.

An important design concern in Grapevine is a parameter τ, which is the number of network hops context information may propagate. Every entity in the network piggybacks its own context on its outgoing packets. It can also piggyback context received from other entities. τ designates how widely context information is disseminated: if $\tau = 1$ then an entity's context is shared only with its directly connected neighbors, i.e., nodes do not forward received context information at all. If $\tau > 1$, context is shared more widely. In our current implementation, τ is a fixed setting; dynamically adjusting τ based on application demands and network conditions is an area of future work. The obvious tradeoff τ addresses is the cost of disseminating more context (in terms of the amount of data piggybacked on packets) versus the amount of shared information. However, because nodes also share group context summaries, neighbors can learn about each other in aggregate instead of individually; this is a significant added benefit of our joint approach to sharing context and forming groups. Many group contexts can be learned this way but network partitioning may cause incomplete group awareness as described in [7]. The importance is subtle and stems from the fact that sharing group context summaries enables groups to be built incrementally by combining a (partial) group summary and individual context summaries [7]. Coupled with our succinct representation of context, this is a significant benefit of Grapevine: *an entity can construct a group and compute its context without directly collecting the context of each individual group member.*

IV. Grapevine: The Implementation

We implemented Grapevine in Java, in the application-layer, which has the advantages of ease of implementation, ease of use and maintenance, and disjointness from the complexities of the network stack in the operating system. On the other hand, we can only attach context information on outgoing *application* packets. Integrating Grapevine with the operating system's network layer would decrease portability, but it would give us finer grained control over how and when context is disseminated; understanding these detailed tradeoffs is an important piece of future work. Fig. 2 shows the information flow within Grapevine's *Context Handler* and *Context Shim*.

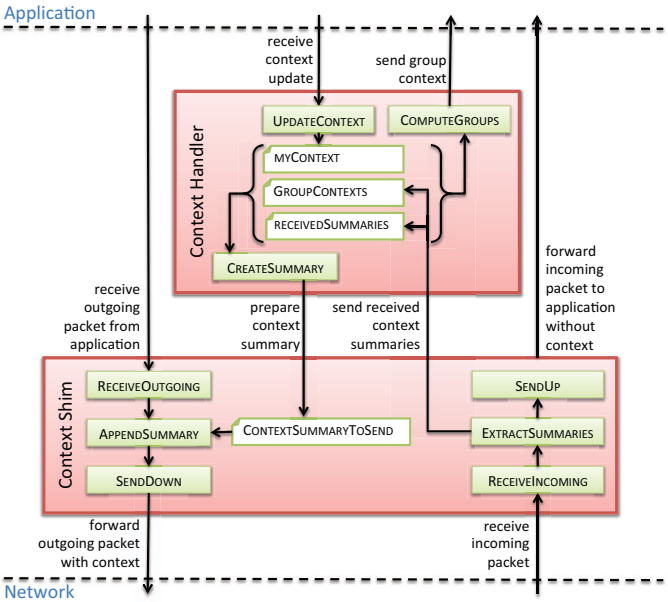

Fig. 2. Information Flow in the Context Shim and Context Handler

A. Context Handler

All context information flows through the *Context Handler*, whose interfaces to the application and the *Context Shim* are shown in Fig. 2. We do not introduce new metrics for context nor any new approaches for assessing individuals' contexts. We assume these tasks are handled by an application-level thread that shares collected context with the *Context Handler*; the *Context Handler* stores this information as MyContext, a simple map of context labels to values.

The *Context Handler* stores received (Bloomier filter) summaries of other entities' contexts as ReceivedSummaries and provides an API for the application to access them. The application also sets the value of τ, which determines which received summaries are included in outgoing packets.

The final interactions between the application and the *Context Handler* surround groups. The application declares groups, and the *Context Handler* monitors stored context information and assesses group membership and group contexts. Group summaries are stored in the *Context Handler* as simple maps of labels to values (GROUPCONTEXTS in Fig. 2). To define groups, applications define a function in a subclass of the `GroupDefinition` class that is evaluated over one or more summaries. The format for this function depends on the type of group. Applications can also register to be notified of changes to groups or their contexts.

B. Context Shim

The *Context Shim* piggybacks context on outgoing packets and extracts it from incoming packets. When a packet is about to be sent, the *Context Shim* retrieves up-to-date context from the *Context Handler* and adds it to the packet. It reverses the process for received packets. We constructed a drop-in replacement for Java's datagram socket that automatically injects and extracts context information to and from datagram packets that pass through. As previously described, when the shim can be more explicitly integrated with the network stack in the operating system, this interception of packets will be migrated into the operating system's networking components.

V. EVALUATION

In this section we evaluate our current implementation of Grapevine to explore how its ability to reduce the overhead of sharing context information can improve the quality of context and group knowledge in real distributed pervasive computing networks. Our evaluations are divided into two categories. First, we describe benchmarking experiments for our implementation in simulation. Then we use a real network deployment to evaluate the performance of our architecture on real hardware, with real network links, and amidst real application traffic.

A. Simulation Results

We created a simple simulated network of 100 nodes in an equidistant 10 by 10 grid. A node could communicate only with its nearest neighbor nodes in each cardinal direction. We did not carefully simulate real-world network effects in our simulation's network stack; this evaluation focuses instead on the relative sizes and ideal propagations of the different context dissemination options. Our second set of evaluations incorporates a real network.

First, we demonstrate the savings (in terms of number of bytes appended to outgoing packets) of our approach in comparison with the labeled context approach used in existing systems. We vary the number of context attributes a node senses (i.e., n) from 0 to 100, and we vary the number of hops over which this context is forwarded (i.e., τ) from 1 to 3. As Fig. 3 shows, Bloomier-based context distribution has significant space savings—an average of 46% in our results.

This reduction in size is especially important in real networks when this context is appended to real packets. Any additional data transmitted can interfere with application-level traffic to the point of preventing the application from functioning, which has traditionally prevented sharing of context in pervasive computing environments.

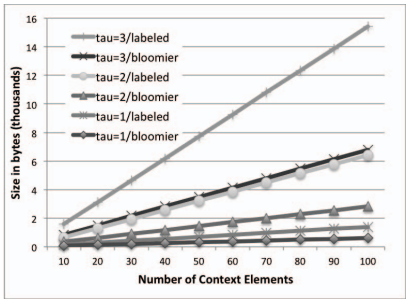

Fig. 3. Size of context information piggybacked for labeled and bloomier context representations

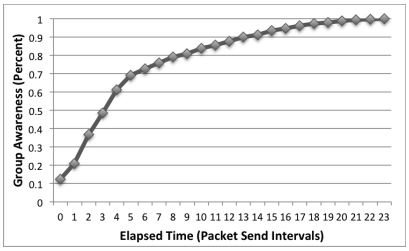

Fig. 4. Labeled group membership propagation

We also evaluated Grapevine's ability to use shared context to form groups. We compared the groups that Grapevine discovered to those that *should* have been discovered (given our omniscient view as the network controllers), and plotted how the group awareness of the collective nodes progressed over time. We again simulated 100 nodes, this time assigning a third of them to a labeled group. As Fig. 4 shows, group membership awareness reached 50% after only three packet beacon intervals. This means that on average each node knew half the group members (network wide) after receiving just three packets from each of its direct neighbors. Group awareness increases to 80% after nine packet intervals and full group awareness is achieved for every node in the network after 23 intervals[1].

B. Evaluation in the Pharos Testbed

Simulation is useful in benchmarking Grapevine, especially in comparison to the other context representation options. However it does not provide an in-depth sense of how Grapevine will perform in real networks. Indeed, an exploration of both the negative impacts on application-level communication and positive impacts on applications' perspectives of their environments is necessary. This is especially important with Grapevine, which, by adding context information to packets, can cause packet fragmentation or otherwise interfere with application-level traffic. Therefore, we performed experiments in a real pervasive computing network deployment in two pieces: first we use a static scenario to quantify the impact of our context summaries on application traffic. We then we use our approach to support a real application, multi-robot patrol, demonstrating the potential gains of using Grapevine. Our deployment environment is a testbed for mobile and pervasive

[1]Altering τ for this experiment had negligible impact since nodes forward aggregated membership information to each of their neighbors

computing systems; we integrate Grapevine with the testbed both for evaluating it and in support of other applications on the testbed that need context and group information.

We use the Pharos testbed [6], which consists of highly modular Proteus mobile robots, configured to form a wireless ad hoc network among the Proteus nodes in an experiment. We first report experiments with a static topology. This allows us to better examine the impact of real network links on context dissemination given different context sizes and the impact of the context summaries on other traffic. It also allows us to more carefully manipulate the network topology, ensuring that the width of the network is always more than one hop. We arranged ten Proteus nodes in a two by five grid providing a multi-hop network in which context must be shared through intermediate nodes.

We first measured the practical impact of adding context information to these packets in terms of the number of lost packets when we appended varying amounts of context. To ensure that context information is widely disseminated, we set τ to three hops.

We measured the percentage of packets dropped, normalized to when no context information is sent, for both Bloomier Filter context summaries and labeled context summaries. As Fig. 5 shows, increases in packet size naturally lead to reduced network reliability. However, in comparison to sending labeled context information, the current state of the art, our more succinct context distribution mechanism increases packet reception by 25% on average.

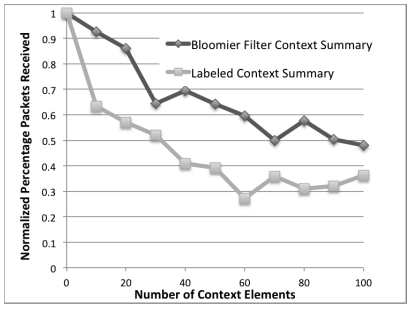

Fig. 5. Normalized percentage of dropped packets for both Bloomier Filter context summaries and labeled context summaries

Finally, we evaluate Grapevine efficacy in solving real world applications by modifying a previous Pharos Testbed experiment that investigated multi-robot patrol [1]. These experiments involved the robots coordinating patrol of a perimeter by communicating their position and arrival times for fixed waypoints as seen in Fig. 6[2]. Collectively, the team forms a group context that captures whether they are patrolling in an ideal manner in terms of achieving equal spacing among the route [5] and minimizing the time spent waiting at waypoints. We significantly simplified the previous implementation by replacing its custom-built coordination software with Grapevine while maintaining the same (or better) level of performance.

Fig. 6. The multi-robot patrol testbed.

[2]See http://bit.ly/grapevine_mrp for more detailed experiment information

VI. Conclusions and Future Work

Pervasive computing applications demand greater expressiveness in gathering and sharing local and distributed context. We presented Grapevine, a practical framework with strong theoretical underpinnings that supports sharing the context of individuals and groups. Grapevine provides a succinct representation of context, efficiently shares context in the immediate network, and uses that shared information to create a view of a situation shared among groups of entities. While we have demonstrated the feasibility of our approach in real pervasive computing networks, we have also opened many interesting future directions. We plan to investigate ways we can mitigate false positives by incorporating context specific heuristics to further lessen their probability. Also, sending large amounts of context can tax resource-constrained networks and we see several avenues for further improving our space efficiency using traditional compression and additional Bloomier related encoding. Furthermore, intelligently adapting the piggybacking strategy to context priorities or the current communication environment, and avoiding packet fragmentation can improve Grapevine's performance. In addition, as we apply Grapevine to additional applications we find that context information is often the bulk of the information a pervasive computing application communicates. This hints that refinements that handle context prioritization and network tuning automatically will greatly increase its value and may eventually supplant the need for many other communications altogether. Lastly, we believe Grapevine's built-in support for groups is an important step forward and will focus future research on addressing the scalability of such computations through heuristic group algorithms with reduced costs.

Ultimately, the vision of pervasive computing demands coordination and collaboration among entities in shared physical spaces. This work is foundational in providing practical support for defining and assessing this shared situational awareness and facilitates building such applications with significantly reduced effort and improved performance.

References

[1] N. Agmon, C. Fok, Y. Emaliah, P. Strone, C. Julien, and S. Vishwanath. On coordination in practical multi-robot patrol. In *ICRA*, 2012.

[2] B. Bloom. Space/time tradeoffs in hash coding with allowable errors. *Comm. of the ACM*, 13(7), 1970.

[3] D. Charles and K. Chellapilla. Bloomier filters: A second look. In *ESA*, 2008.

[4] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The bloomier filter: An efficient data structure for static support lookup tables. In *SIAM*, 2004.

[5] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *IAT*, 2004.

[6] C. Fok, A. Petz, D. Stovall, N. Paine, C. Julien, and S. Vishwanath. Pharos: A testbed for mobile cyber-physical systems. Technical Report TR-ARiSE-2011-001, Univ. of Texas at Austin, 2011.

[7] C. Julien. The context of coordinating groups in dynamic mobile environments. In *Coordination*, 2011.

[8] T. Jun and C. Julien. Automated routing protocol selection in mobile ad hoc networks. In *SAC*, 2007.

[9] E. Porat. An optimal bloom filter replacement based on matrix solving. In *CSR*, 2009.