Stacon: Self-Stabilizing Context Neighborhood for Mobile IoT Devices

Chenguang Liu, Jie Hua, Changyong Hu, Christine Julien
The University of Texas at Austin
{liuchg,mich94hj,colinhu9,c.julien}@utexas.edu

***Abstract*—Retrieving context information from on-board sensors is critical to many scenarios in the Internet of Things (IoT). The battery capacity and physical size of smart devices can restrict the on-board sensors that can be supported and thereby limit the potential of applications for multi-agent IoT systems. In this work, we propose a novel scheme for building a *context neighborhood* among nearby devices through infrastructure-less collaboration. A context neighborhood is an ad hoc grouping of sensing devices that can collaborate to sense physical attributes of the environment. In this way, not every device needs to directly sense every context attribute to be able to leverage the information in its applications. Our approach entails a distributed algorithm that dynamically adjusts a device's sensing and sharing strategy based on the heterogeneity of resources in the proximity. We develop a prototype system using off-the-shelf IoT sensor kits; our demonstration exploits the self-stabilization feature to show the benefits this system could bring to higher-layer applications.**

***Index Terms*—Internet of Things, Context awareness, Sensor systems and applications**

I. Introduction

In the past decade, sensor-equipped systems have been pervasively deployed in our world. These low-powered devices *sense* physical attributes of their surrounding environments and provide digital services accordingly as a form of *calm computing* [17]. Recently, there is an emerging sub-field that leverages coordination among *smart agents* hosted on co-located devices and explores the potential of building more flexible and sustainable systems using multi-agent collaboration. As a few examples, applications in a smart home can provide more accurate activity monitoring by using multiple sensors distributed around the home [6], while a travel assistant system can be more useful if it is able to access and collaborate with surrounding devices in a smart-city [10], [13].

We introduce Stacon, a framework for low-powered mobile IoT devices to share sensing capabilities. Stacon enables devices to automatically detect other nearby sensing resources and to self-select sensing tasks based on the device's own needs, the capabilities of the environment, and the needs of other nearby devices. The goal is to maximize the context fulfillment of the neighborhood as a whole. Here we use the term context neighborhood to refer a set of participating devices that are mutually reachable in one-hop network.

Fig. 1 shows a schematic of the generic framework. The figure shows five available sensing devices (numbered one through five), each with heterogeneous sensing capabilities

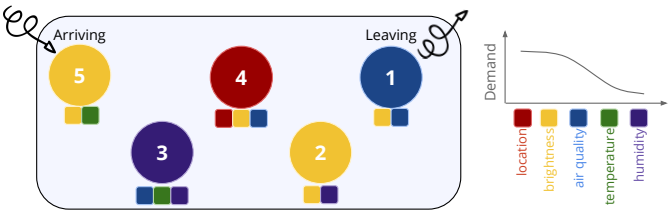

Fig. 1. Dynamics in a Context Neighborhood.

(depicted as colored squares underneath each numbered device). The neighborhood also has a *context demand model* [9], which characterizes the needs of the neighborhood for the various sensable attributes (the location attribute is the most important to devices in the neighborhood, while humidity is least important). The goal of Stacon is to enable each device to make a local decision about which attribute to sense (these choices are depicted as the colors of devices in the figure) based on information other devices share about their needs, capabilities, and selected tasks. Achieving a self-stabilizing behavior in this environment is plagued by challenges that are largely tied to network dynamics. When devices enter and leave the neighborhood, the neighborhood's needed context types, sensing capabilities, and selected tasks all change.

We will demonstrate the Stacon middleware with the complete software stack on real devices. Stacon builds on the BLEnd, a slotless neighbor discovery protocol [5]; in Stacon, BLEnd's neighbor discovery beacons carry descriptions about capabilities, needs, and tasks to aid in the self-stabilizing algorithm (Section III). The sensing tasks are performed on *off-the-shelf* IoT sensor kits [1] equipped with rich sensing equipment alongside a customized development board to demonstrate that the system can be easily extended (Section IV). The demonstration will show devices with heterogeneous sensing configurations, allow live and interactive network dynamics, and visualize the behavior and benefits of Stacon for cooperative context sensing in mobile IoT devices.

II. Background

Continuous resource discovery. Stacon relies on an existing continuous neighbor discovery protocol that uses broadcast to achieve reliable discovery across multiple devices. Some continuous neighbor discovery approaches [3], [12], [16] require fixed-length slots to ensure deterministic discovery, however this limit is incompatible with many wireless

[1] https://www.nordicsemi.com/eng/Products/Nordic-Thingy-52

978-1-5386-9151-9/19/$31.00 ©2019 IEEE

communication technologies, notably the Bluetooth Low Energy (BLE) technology found in many IoT devices. In contrast, recent work has designed slotless protocols that are compatible with technologies like BLE [5], [12]. Stacon relies on the BLEnd protocol [5]. In BLEnd, time is divided into epochs; each epoch repeats a schedule in which the device scans for beacons, sends beacons, and sleeps. The device starts each epoch by scanning for a fixed amount of time to discover other nearby devices' beacons. After scanning, the device sends beacons which are discoverable to other devices. A BLEnd schedule is carefully designed to provide a probabilistic discovery guarantee in terms of Λ, the *target discovery latency*, even considering the potential for beacons to collide and therefore not be received. BLEnd creates this schedule in an energy optimal way while meeting the communication and discovery constraints: the target discovery latency (Λ), the target probability of discovery (p_d), and a maximum number of expected devices (n).

Sensing capability sharing with nearby devices. Statistical properties of spatiotemporal correlations in sensed context have been well-studied in wireless sensor networks [11], [15], [20]. In recent years battery-operated devices have enabled context exchange via low-powered short-range wireless connectivity [1], [9], [14], [19]. These approaches require devices to *explicitly* express their context demands and share sensor readings using targeted packet exchange [4] or via temporary device-to-device links [2], [7]. More recent *implicit* approaches enable sensing devices to actively disseminate context information for data collection [1] or for building a context neighborhood [9]. Stacon is inspired by the latter, which uses randomized heuristics for sensor selection. In contrast, Stacon takes a deterministic approach and guarantees optimality.

III. Context Neighborhood with Stacon

We next formalize a self-stabilizing context neighborhood. We then describe Stacon in detail.

Problem (Self-stabilization of context neighborhood): *A network of sensing devices $D = \{d_1, \ldots, d_n\}$ collaboratively construct a context neighborhood via device-to-device communication. $C = \{c_1, \ldots, c_m\}$ context attributes can potentially be sensed, and each device d_i can sense $C_i \subseteq C$ depending on its on-board sensors. C is known* a priori*, and every c_j has an associated weight w_j that indicates its demand in the neighborhood. We define a device-context pair (i, j) to mean that d_i is sensing and sharing c_j. At time t, an assignment S_t is a set of pairs (i, j) such that $c_j \in C_i$ for all $(i, j) \in S_t$, and, for each d_i, there is at most one pair $(i, j) \in S_t$. The goal of Stacon is to stabilize the network neighborhood such that: (1) the neighborhood's context demands are fulfilled (i.e., $\max_{S_t} \sum_{j:\exists i,(i,j)\in S_t} w_j$), and (2) the self-stabilization process finishes within the discovery latency (Λ) with probability p_d.*

System overview. BLE requires fixed-length beacons, but continuous discovery information does not use the entire beacon payload. We therefore encode Stacon-specific information into the unused beacon space, as described in Section IV. As shown in Fig. 2, Stacon receives packets from the scan reports of the underlying BLEnd protocol. Every beacon b received during the BLEnd scan period will be stored into a beacon repository $\mathcal{B}$. At the end of each scan period, Stacon removes outdated beacons from $\mathcal{B}$ and computes the current sensing capabilities and demands of the neighborhood. Using this update, Stacon then *re-selects* a context type as its new task, queries the corresponding sensor if necessary, and encodes the new packet into the neighbor discovery beacon payload.

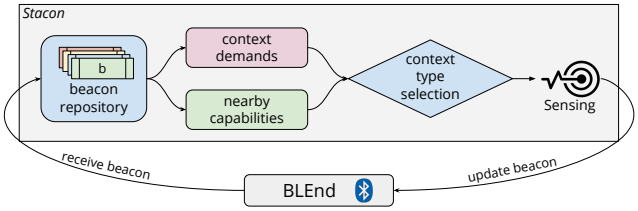

Fig. 2. Stacon overview.

Self-stabilization. The above self-stabilization problem can be mapped to a classic bipartite matching where the device is one party and the context sensing task is the other. Like bipartite matching, each device can select only one context type to sense in a round, and each context should only be sensed and shared by one device since repeated sensing in a neighborhood is not optimal; instead a device should conserve energy. In common IoT settings, a practical number of devices for slot-less symmetric discovery is typically under 50 [5], [12], [18]. For a battery-powered IoT device with limited computation resources (e.g., a Cortex M4 based system-on-chip runs at 64MHz), the worst case time complexity of a $O(n^3)$ algorithm for bipartite matching is acceptable.

Stacon's self-stabilization algorithm is shown in Algorithm 1. Each device calls the AfterScan function every epoch. The device uses information received in beacons to maintain a capability list for the other devices in the neighborhood; this list uses hashes to succinctly describe other devices' sensing capabilities. During self-stabilization, each device updates the list using the beacons it receives in this epoch. If there are changes in the received beacons relative to the previous epoch, i.e., a device has left, arrived, or changed its capabilities or task, the sensing capability of the neighborhood has changed and Stacon runs a matching algorithm similar to the *Hungarian* algorithm to solve the bipartite matching problem and get the optimal assignment $S_{optimal}$. Each device d_i chooses their context c_j as in the optimal assignment, i.e. $(i, j) \in S_{optimal}$. As we have mapped the problem to bipartite matching, the optimality of the assignment is guaranteed by the *Hungarian* algorithm assuming all devices have the same view of the neighborhood. Since BLEnd guarantees discovery within a target latency (Λ) and target probability (p_d), Stacon will stabilize with the same probability and the same latency, satisfying the second condition in the problem formulation.

IV. Implementation and Demo Plan

We implemented a prototype of Stacon on the Nordic Thingy52 IoT sensor kits. The device has 8 on-board sensors and is able to provide even more types of fine-grained context

Algorithm 1: Self-stabilization Algorithm

Function AFTERSCAN: *device* d_i, $\mathcal{B}$, $snap_{prev}$
  Compute capability list C_j s.t. $j \neq i$ from $\mathcal{B}$.
  Take a snapshot of local $< C_1, \cdots, C_n, D >$ as $snap_{new}$.
  if $snap_{new} \neq snap_{prev}$ **then**
    Compute $S_{optimal}$ using matching algorithm.
    Select c_j as its type of context where $(i, j) \in S_{optimal}$.
  $snap_{prev} \leftarrow snap_{new}$
end

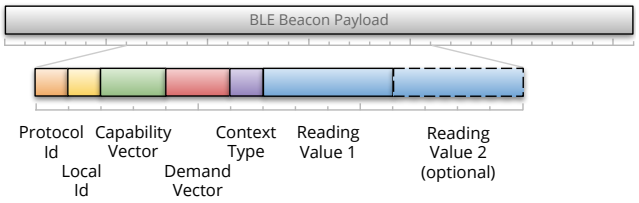

Fig. 3. Beacon structure in Stacon.

information about the environment (e.g. air quality, ambient noise level). It is equipped with BLE for wireless connectivity and a rechargeable lithium polymer battery.

We implemented Stacon on top of the BLEnd [5] scheduler[2]. BLEnd uses BLE to exchange wireless packets on 2.4GHz channels. The BLE beacon payload is 31 bytes; BLEnd uses only 5 bytes for neighbor discovery purposes. Therefore, we put information to support Stacon's self-stabilizing context neighborhood into the 26 unused bytes. We use one byte to identify the Stacon and one byte for the local id. Following that are two bit vectors of two bytes each, encoded from the sensors of the device and the context demand of its upper-layer applications. The shared sensor reading is placed in the next five to nine bytes. The first byte is used for a header and is followed by four bytes of the actual sensor reading. While four bytes are generally sufficient for most types of context, sometimes we need additional space to contain richer values (e.g., location); for those cases we allocate four more bytes as optional content. Fig. 3 shows the complete 15 bytes used in Stacon and its placement in the BLEnd beacon payload.

We plan to demonstrate Stacon with five Thingy52 sensor kits, one nRF52840 development kit and one Android tablet. The nRF52840 DK will provide location information using an Adafruit GPS module[3]. The Android tablet will be used as a BLE sniffer to allow visitors to visualize the wireless beacons that are exchanged and the *context fulfillment* of the network neighborhood compared to the optimal task allocation.

To simulate heterogeneous networks, each IoT sensor kit will have only a subset of sensors enabled. Context types are assigned with distinguishable RGB color values and we use the on-board LEDs to visualize the sensing task a device is currently executing. Together with user-triggered device arrival and departure events (triggered by power cycling the devices), this lightwell representation essentially shows Stacon's stablizing process. The instantaneous system status and estimated energy cost [8] will be visualized with the Android tablet.

[2]The code for this demo is at https://github.com/UT-MPC/BLEnd_Nordic.
[3]https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps.pdf

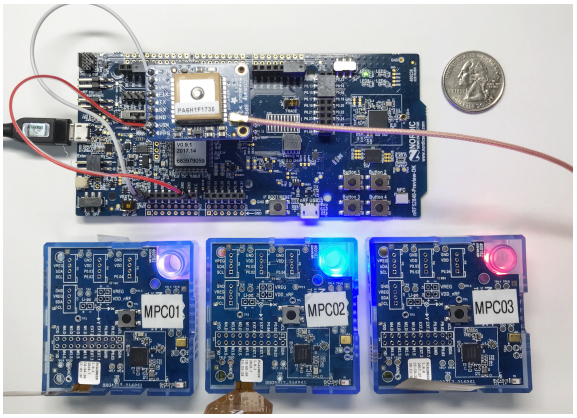

Fig. 4. Stacon demo devices

ACKNOWLEDGMENTS

This work was funded in part by the National Science Foundation, CNS-1703497.

REFERENCES

[1] J. Adkins et al. The signpost platform for city-scale sensing. In *Proc. of IPSN*, 2018.
[2] A. Amiri Sani et al. Rio: A system solution for sharing i/o between mobile systems. In *Proc. of ACM Mobisys*, pages 259–272, 2014.
[3] M. Bakht, M. Trower, and R. Kravets. Searchlight: Won't you be my neighbor? In *Proc. of ACM Mobicom*, pages 185–196, 2012.
[4] P. Costa et al. Programming wireless sensor networks with the teenylime middleware. In *Proc. of Middleware*, 2007.
[5] C. Julien, C. Liu, A. Murphy, and G. Picco. Blend: practical continuous neighbor discovery for bluetooth low energy. In *Proc. of IPSN*, 2017.
[6] S. Kang et al. Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In *Proc. of ACM Mobisys*, 2008.
[7] A. Levy et al. Beetle: Flexible communication for bluetooth low energy. In *Proc. of ACM MobiSys*, pages 111–122, 2016.
[8] C. Liu, J. Hua, and C. Julien. Scents: Collaborative sensing in proximity iot networks. In *Proc. of IEEE PerCom Workshops*, 2019.
[9] C. Liu, C. Julien, and A. Murphy. Pinch: Self-organized context neighborhoods for smart environments. In *Proc. of IEEE SASO*, 2018.
[10] C. Martella et al. Leveraging proximity sensing to mine the behavior of museum visitors. In *Proc. of IEEE PerCom*. IEEE, 2016.
[11] S. Nath. Ace: exploiting correlation for energy-efficient and continuous context sensing. In *Proc. of ACM Mobisys*, 2012.
[12] Y. Qiu et al. Talk more listen less: Energy-efficient neighbor discovery in wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2016.
[13] M. Saloni et al. Lasso: A device-to-device group monitoring service for smart cities. In *Proc. of IEEE ISC2*, pages 1–6, 2017.
[14] X. Sheng, J. Tang, and W. Zhang. Energy-efficient collaborative sensing with mobile phones. In *Proc. of IEEE INFOCOM*, 2012.
[15] M. Vuran et al. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, pages 245–259, 2004.
[16] L. Wei et al. Lightning: a high-efficient neighbor discovery protocol for low duty cycle wsns. *IEEE Comm. Letters*, pages 966–969, 2016.
[17] M. Weiser and J. Brown. The coming age of calm technology. In *Beyond calculation*, pages 75–85. Springer, 1997.
[18] R. Zheng, J. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *Proc. of ACM MobiHoc*, pages 35–45, 2003.
[19] X. Zheng, D. Perry, and C. Julien. BraceForce: A middleware to enable sensing integration in mobile applications for novice programmers. In *Proc. of IEEE/ACM MobileSoft*, pages 8–17, 2014.
[20] D. Zordan et al. Modeling and generation of space-time correlated signals for sensor network fields. In *Proc. of GLOBECOM*, 2011.