Managing Node Rendezvous in Opportunistic Sensor Networks

Shane B. Eisenman
Columbia University
New York, New York, USA
shane@ee.columbia.edu

Andrew T. Campbell
Dartmouth College
Hanover, New Hampshire, USA
campbell@cs.dartmouth.edu

***Abstract*— In an opportunistic sensor network (OSN), mobile sensor nodes are tasked by sensor access points (SAPs) on behalf of applications running remotely on back end infrastructure (e.g., Internet). Similarly, mobile sensor nodes upload sensed data to back end databases via this SAP tier. In a people-centric OSN, node mobility is uncontrolled and the architecture relies on opportunistic rendezvous between human-carried sensors and SAPs to provide tasking/uploading opportunities. However, in many reasonable scenarios application queries have a degree of time sensitivity such that the sensing target must be sampled and/or the resulting sensed data must be uploaded within a certain time window to be of greatest value. Halo efficiently, in terms of packet overhead and mobile node energy, provides improved delay performance in OSNs by: (i) managing tasking/uploading opportunity, (ii) using smart operation scheduling at the SAP, and (iii) exerting extra effort to complete ongoing tasking/uploading operations in a single SAP rendezvous.**

I. Introduction

The initial application focus of wireless sensor networking has been on *in situ* monitoring of ecological processes, or on industrial processes and equipment. Recently researchers in this field have begun to consider the urban domain with a focus on *people-centric* [3] [16] sensing and application development. Architectures in this new domain assume mobile smart phones and embedded sensing devices equipped with a short-range radio (e.g., ZigBee, Bluetooth) are carried by humans or mounted on vehicles, leading to a network of sensors with mobility uncontrolled by the sensing architecture. Such architectures often employ a multi-tiered hierarchical structure where sensor tasking (i.e., application query assignment) and data collection occur via mobility-enabled interactions between people-centric *mobile sensors (MSs)*, and edge wireless access nodes we call *sensor access points (SAPs)*. Initial responsibility for tasking of sensors resides at the SAPs and SAPs are also the collection point for sensed data.

Tasking and collection operations can occur when MSs enter the spheres of interaction of the SAPs. Generally, the term sphere of interaction refers to the region (i.e., the physical volume) within which services offered by a node is available to its neighbors. For the SAP case, to which we limit our discussion in this paper, these services include tasking and uploading. While applications that use opportunistic sensor networks should be delay tolerant, we draw a distinction between those that are delay-aware and those that are not. Delay-aware applications do not warrant time on a real time architecture, but may issue queries that have a degree of time sensitivity such that the sensing target must be sampled, and/or the resulting data must be uploaded, within a specified time window to be of greatest value. Examples are myriad, and include personal applications that seek to answer questions like, "Where can I find a quiet place to study for the next hour", and public utility applications that say, "Give me my local weather spotter data in time for the next newscast". On behalf of these delay-aware applications, we investigate a number of fundamental performance issues in opportunistic sensor networks, including the interplay between resource consumption and the timeliness of tasking and data collection.

To increase the frequency and duration of the sensors' travel through the sphere of interaction of a given SAP, the SAP might enlarge its sphere of interaction by increasing its transmission power and/or by using a multi-hop sphere of interaction. However, a multi-hop sphere of interaction requires increased signaling (e.g., to set up and maintain routes), requiring more energy expenditure. Multi-hop communication implies a higher packet loss probability; in a wireless environment with a link packet loss rate p the probability of success across n hops is $(1-p)^n$. Also, increasing the transmission power of the SAP implies an increased energy drain on the energy-limited MSs since these must match the higher transmission power of the SAP. Further, a larger SAP sphere of interaction disrupts local peer-to-peer sensor communication in a larger part of the field, a problem of increasing relevance given the recent interest in mobile peer-to-peer services using localized communication [9] [15] [11].

We design (Section II), implement and evaluate (Section III) Halo, a framework providing algorithmic and protocol support for managing rendezvous between SAPs and mobile sensors. Halo manages opportunities for tasking and uploading operations via a deadline-based SAP sphere of interaction. In addition, when multiple simultaneous operations are possible, Halo takes a snapshot of the system (i.e., the sensors available for tasking and uploading in its sphere of interaction, the pending tasking operations, and the applications waiting for data upload), and incorporates sensor-driven mobility prediction of the available MSs to generate a schedule of the tasking and uploading operations. This novel scheduling approach integrates a traditional shortest-job-first approach, and a mobility-based approach tailored for OSNs. Finally, with Halo SAPs attempt to complete ongoing operations in the face of mobility by expanding the SAP sphere of interaction, while avoiding unnecessary disruption to on-going communication in the region surrounding the SAP.

II. HALO DESIGN

With Halo, our goals are to simultaneously: reduce the energy spent by mobile sensors, reduce disruption to muling/peering communication ongoing in the area surrounding the SAPs, and increase the number of tasking and upload operations completed per unit time. Our three pronged approach is described in the following.

A. Managing Rendezvous Opportunity

There are competing pressures in managing tasking and collection opportunities. We wish to expand the SAP sphere of interaction to increase the number of MSs available to task, reduce tasking delay, increase the amount and utility of sensed data to delay-aware applications, reduce collection delay, and reduce the likelihood of mobile sensor storage overflow. On the other hand, we wish to contract sphere of interaction to reduce required energy expenditure of mobile sensors when transmitting to a SAP, reduce the risk of disrupting peering/muling communications ongoing in the vicinity of a SAP, and increase the security of the system by probabilistically reducing overhearing, and explicitly limiting the number of nodes offering (authenticated) proxy service on behalf of the SAP. We use a simple model of a single SAP is used to illustrate the impact of sphere of interaction radius on data transfer opportunity (e.g., for tasking, upload), required MS transmit energy, and the SAP interference area. Here, the radius of the "sphere" (our 2D analysis can directly be extended to 3D) is a real valued abstraction of the range extension due to power control only. The trigonometry is straightforward, but details are available in Appendix I for the interested reader. Figure 1(a) shows: (i) the data transfer opportunity (in abstract time units) of a single MS by plotting the average straight line trajectory length through the SAP sphere of interaction, assuming the MS maintains a constant unit velocity along the trajectory; (ii) the average transmit energy a MS must use to communicate with the SAP (assuming a symmetric link and a simplified Friis model with a loss exponent of 4) as it traverses the sphere of interaction along the average length chord; and (iii) the area disrupted by SAP-MS communications as the MS travels along the average length chord. The two curves show the disrupted area when the MS is closest to and furthest from the SAP, respectively, along its average chord walk. Data transfer opportunity grows linearly with the sphere of interaction radius, while energy cost and SAP interference experience super-quadratic growth. The tradeoff between data transfer opportunity, and MS energy and SAP interference impact, motivates a managed SAP sphere of interaction radius.

While there are a number of possible triggers for increasing or decreasing the SAP sphere of interaction, we believe the fundamental driver for sphere of interaction adaptation should be fulfilling application requests (i.e., tasking and collection operations) since this is the metric that mostly closely reflects the user experience. Generally, we wish to expand the sphere of interaction when application demands require it, and contract the sphere of interaction at all other times to reduce energy consumption and channel access contention in the vicinity of the SAP. In an OSN architecture, the baseline is the lazy approach where we passively wait on mobility to bring suitable sensors to task, or previously tasked sensors carrying back sensed data within the radio range of the SAP when the transmit power is fixed. However, if some sensed data is most valuable if sensed and/or delivered within a particular time window, users may require improving the performance over this lazy approach.

Assume we have an application query i with which to task the sensor network that requires data from a particular sensor type (e.g., CO_2 sensor). Suppose that the data must be sensed at time $t_{s_i}(min) \leq t \leq t_{s_i}(max)$ to capture the event of interest, and that a constant T_i exists that reflects the time it takes to travel from the tasking SAP (which is assumed to know its location) and the sensing target location defined in the query, using average case human mobility. When an application query is inserted into the SAP task queue at time t_i^0, we calculate the time until sphere expansion $\delta_{s_i}^0 = (t_{s(max)_i} - t_i^0) - T_i$. If a MS matching the task requirements is available for tasking within the current SAP sphere of interaction, then no sphere adjustment is necessary and the tasking operation can proceed. Otherwise, at any time t^j a SAP calculates its sensing-driven sphere adjustment multiplier ξ_s for query i as $\xi_{s_i} = (1 - \delta_{s_i}^j / \delta_{s_i}^0)$.

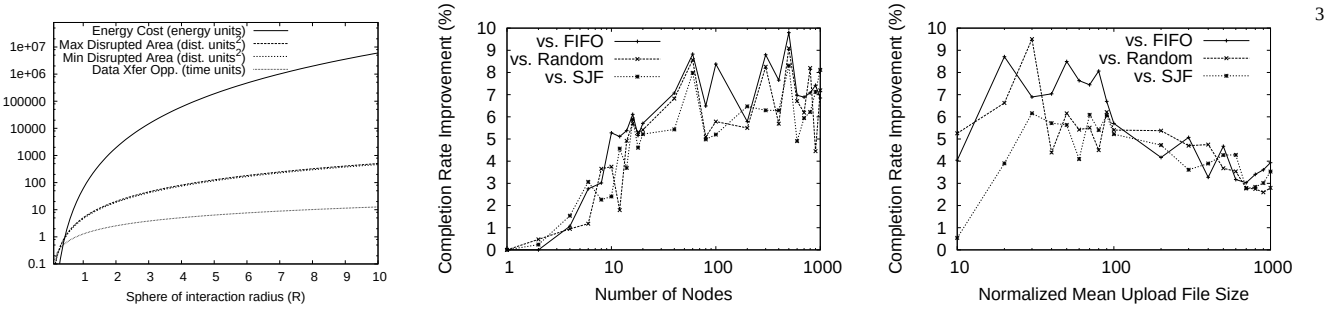
Similarly, assume for an application query i, an MS was previously tasked and was able to sample the requested target. Suppose the data must be delivered back to a SAP by time $t \leq t_u$ in order for the data to have the greatest utility, and T_i is defined as before. Then at any time t^j a SAP calculates its upload-driven sphere adjustment multiplier ξ_u for query i as $\xi_{u_i} = (1 - \delta_{u_i}^j / \delta_{u_i}^{s(max)})$, where $\delta_{u_i}^{s(max)} = (t_{u_i} - t_i^{s(max)}) - T_i$. For generality, we handle the case where queries do not specify sensing target locations by setting T_i=0. Similarly, for queries that do not have sensing or uploading deadlines, we set $t_s(max) = \infty$ and $t_u = \infty$, respectively.

We use a small set of power settings at both the MSs and SAPs, and limit the maximum number of hops of sphere expansion to keep the cost and complexity of interactions between mobile nodes low. The choice of the supported power levels can be arbitrary or based on historical information kept at the SAP about the number of MSs found at a given sphere radius. Let $P = \{p_1, ..., p_K\}$ be the supported power levels at each node and let M be the maximum allowed number of hops. Then there are $M \cdot K$ possible sphere extension settings, and we write the set of settings as $S = \{s_1, ..., s_{M \cdot K}\}$, where for s_j, the number of hops $m = 1 + \lfloor \frac{j-1}{K} \rfloor$ and the power setting of the last hop $k = j \ mod(K+1)$ (hops prior to the last hop are at power setting p_K).

For a set of tasks Q at a particular SAP, the sphere of interaction is set according to

$$s = \max \left(\left\lceil \max_Q(\xi_{s_i}) \cdot M \cdot K \right\rceil, \left\lceil \max_Q(\xi_{u_i}) \cdot M \cdot K \right\rceil \right) \quad (1)$$

Following this rule, the sphere of interaction setting adapts to the current set of pending deadlines. Taking tasking as an



(a) (b) (c)

Fig. 1. (a) Impact of SAP sphere of interaction radius. (b,c) Advantage of MB-SJF versus other common scheduling disciplines in simulation. MB-SJF consistently completes the upload tasks faster than FIFO, RAND and SJF, across all tested parameter values for node population and mean file size.

example, as the time t^j gets closer to $t_{s(max)_i} - T_i$, then $\delta^j_{s_i}$ goes to 0 and the sphere setting grows to $M \cdot K$. On the other hand, if all pending deadlines are far enough in the future, then $\delta^j_{s_i}$ is still close to $\delta^0_{s_i}$ and the sphere setting shrinks close to the minimum. While a number of variations on this scheme are possible, the rule in Equation 1 has the advantage of encouraging early submission of application queries to the system. Though beyond the scope of the current work, early submission might allow a system to do query load balancing among SAPs, or smart query assignment to particular SAP.

B. Scheduling Operations

In this work, we treat two aspects of scheduling that impact both the efficiency of communications between the SAPs and MSs and the average operation (i.e., tasking, uploading) turnaround time and throughput. First, we discuss reasons for serving a single MS until its operation is completed (or it leaves the SAP range) rather than switching between multiple MS sessions. Second we discuss how the SAP determines the order in which it will serve the MSs in its current sphere of interaction. To support scheduling, the SAP takes a snapshot of the system at particular points in time. Within this freeze frame, the SAP knows: the set of application tasks to complete, the set of MSs in the sphere of interaction of a SAP available for tasking, the set of MSs in the sphere of interaction of a SAP offering data to upload, the set of applications waiting for particular data, and an estimation of each node's proximity and mobility.

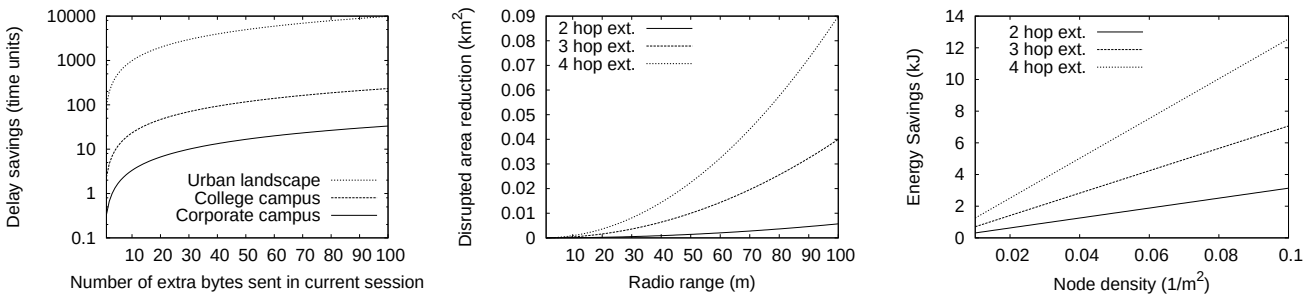
1) Atomic vs. Interleaved: Prior experimentation [3] with a testbed of Moteiv Tmote Invent motes (which use IEEE 802.15.4 radios) shows that at typical walking speeds and relatively low density of MSs, simultaneous uploading and tasking results in none of the operations being fully completed using state of the art sensor network transports. More recently, Miluzzo, et al. have characterized [10] the severe radio attenuation caused by the body that will be prevalent in human-centric networks, that will tend to limit the average contact time between SAP and MS even if higher data rate radios are used. Because of this limited contact time, an interleaved approach to operation scheduling (i.e., either preemptive or simultaneous uploads and downloads) may not be appropriate. Firstly, a preemptive approach leads to a longer average turnaround time than non-preemptive scheduling [13] for all operations. Also, simultaneous uploading and downloading requires more MAC layer overhead in terms of either backoffs or collisions in the case of contention-based MACs, or schedule maintenance and dissemination in the case of non-contention-based MACs. Instead, we take a non-interleaved or atomic approach with at most one active uploading or tasking session ongoing at each SAP.

2) Scheduling Discipline: To determine the order in-sphere MSs will be served, a simple approach is to not actively manage the order of operations at all and just serve MSs in the order they arrive (e.g., FIFO) at the SAP until they move out of range. However, schemes like FIFO or even random selection ignore important features such as the size (i.e., number of bytes) of tasking and uploading operations, and the MS dwell time in the SAP sphere of interaction. Thus, these naive approaches can lead to a lower operation throughput due to non-uniform MS inter-SAP-visitation times (i.e., orbits).

We use a hybrid mobility-based/shortest-job-first (MB-SJF) scheduling algorithm to decide the order in which MSs are atomically served. Let A denote the event that a tasking or uploading operation supported by the current set of MSs can be completed before the associated MS exits the maximum SAP sphere of interaction obtainable without multihop, since multihop extension is not always possible. Uploading and tasking operations are ordered by $Prob(A)$. This probability reflects the size of the operation to be completed (i.e., number of bytes b) and the estimated dwell time, t_{SAP}, of the associated MS in the sphere of interaction. We have

$$Prob(A) = Prob(t_{SAP} \geq \frac{b}{C} + \beta), \tag{2}$$

where C is the wireless channel rate in bytes per second, and $\beta = \overline{BO} \cdot \frac{b}{frame\_size}$, for a CSMA channel. Here, $\overline{BO}$ is the average MAC backoff interval across the packets needed to complete the operation, and $frame\_size$ is the maximum MAC frame size in bytes. It is worth nothing that with our atomic scheduling approach, the second term on the right side of the inequality can be driven to zero if the MAC parameters are tweaked such that a backoff window of zero is used during an upload/download session between a MS and the



(a) Possible average delay savings with extra-effort expansion of the SAP sphere of interaction.

(b) Disrupted area with targeted expansion is reduced quadratically versus omnidirectional expansion.

(c) Energy savings with targeted expansion scales linearly with density and quadratically with the number of hops.

Fig. 2.

SAP (the standard backoff window would still be used for communication between MSs and for the MS↔SAP session setup - c.f. Appendix II. Once $Prob(A)_i$ is calculated for each (operation, MS) pair i in the sphere of interaction, the operation schedule is set in descending order of the value $Prob(A)_i * \nu_i$. The optional priority factor ν_i can be used to prioritize certain event types (e.g., toxic spill) or users (e.g., those with long average orbits), but the exact meaning is left up to the system administrator and it may be dropped altogether if user/operation priority need not be supported.

To evaluate the performance of MB-SJF, we simulate a one-SAP/multi-MS scenario where all MSs are assumed to have data to upload. We compare MB-SJF with common scheduling disciplines such as first-in-first-out (FIFO), random selection (RAND), and shortest-remaining-job-first (SJF) in terms of the time it takes to upload the data from all of the MSs. In the simulation, each of the MSs is assigned a file to upload whose size is randomly chosen from an exponential distribution. MSs move between two states, at-SAP and not-at-SAP, where the dwell times (t_{SAP}) in each state are randomly drawn from different exponential distributions with means λ_{SAP} and $\lambda_{\overline{SAP}}$, respectively. To simulate a population of MSs with different mobility characteristics, each node is assigned a unique $\lambda_{SAP}, \lambda_{\overline{SAP}}$ pair, whose values are uniformly spread between 0 and $N\gamma$, where N is the number of MSs and γ is the spreading factor. The simulator updates MS states synchronously and both the file sizes and the location dwell times are normalized to its update period. For MB-SJF, it is assumed the MSs report their λ_{SAP} and $\lambda_{\overline{SAP}}$ values and remaining file size to upload (b) to the SAP upon entering the SAP's sphere of interaction (c.f., the *beacon reply* message in Appendix II. Neglecting MAC effects, from Equation 2 the SAP computes $Prob(A) = e^{-\lambda_{SAP} b}$ for each node in its sphere. $\nu = e^{-\lambda_{\overline{SAP}}}$ prioritizes nodes with long orbits.

In Figures 1(b) and 1(c), we plot the completion rate improvement MB-SJF gives versus FIFO, RAND, and SJF. Each point represents the average of 1000 trials, each with a different seed for the pseudo-random number generator driving upload file sizes and location dwell times.

Figure 1(b) shows the completion rate improvement versus the number of MSs in the simulation, with a fixed mean upload file size of 100. As the MS population grows, the advantage of MB-SJF steadily increases until settling around a 6-10% improvement after 60 nodes. MB-SJF, like plain SJF, is able to finish off small upload tasks quickly, but also takes advantage of mobility information to opportunely upload from nodes that visit the SAP relatively rarely. Figure 1(c) shows the completion rate improvement versus the mean upload file size when there are 20 MSs. As expected, for medium size files (implying medium aggregate upload times), MB-SFJ provides for a relatively constant improvement in completion rate. As file sizes get larger, the improvement begins to diminish. As file sizes tend to infinity, so does the completion time regardless of scheduling discipline, and therefore the possible improvement goes to 0. However, we believe the typical case for opportunistic wireless uploads from mobile consumer devices will be small to medium size files (e.g., a 1kB text file, a 1MB image file, a 10MB audio file). Based on these simulations, MB-SJF seems to be a good candidate for MS scheduling in the SAP sphere of interaction and we use MB-SJF for the full evaluation in Section III.

3) Estimating t_{SAP} in Practice: In the preceding simulation, the SAP's estimate for each MS's t_{SAP} is simply the mean of the exponential distribution from which the at-SAP dwell time is drawn. In practice, in lieu of a probability distribution, the notion in Equation 2 can be restated in terms of a *dwell score (DS)* computed based on a mean value estimate of t_{SAP}, easily tracked by the MS and shared with the SAP, and the remaining bytes, b, to upload.

$$DS = \overline{t}_{SAP}/(\frac{b}{C} + \beta), \tag{3}$$

where β is defined as before. Since we are considering the people-centric sensing domain, human activity inferred from on-board sensors can aid the MS in further refining its $\overline{t}_{SAP}$ estimate. In particular, samples from an accelerometer (embedded in many new mobile phone devices) can be processed to determine if a person is standing, walking, or running. In [9], the authors classify between these three states with an average accuracy of about 90%. Since average dwell times are likely to be highly correlated with human activity, we propose to keep a separate $\overline{t}_{SAP}$ for each classified activity and use this value in calculating Equation 3. The operation schedule is then set in descending order of the value $DS * \nu$.

4) Scheduling Epoch: MSs may enter a SAP's sphere of interaction during an ongoing schedule. In this case, the SAP could ignore all newcomers until its current schedule is complete and then come up with a new schedule that incorporates the newcomers, or it might create a new schedule upon the completion of every operation. In the former case, starvation is prevented, but new sensors that may rank higher are ignored. In the latter case, more energy is spent and time wasted by re-running the neighbor discovery after every operation. In Halo, we define a scheduling epoch of time length E to strike a middle ground. E is adaptive to the estimated mobility of the MSs involved in the current schedule. Let K_i represent this set of MSs for a given schedule; then $E = \max_{K_i}(t_{SAP})$. The next scheduling time is then defined as

$$t_{sched_{i+1}} = t_{sched_i} + min\left(\sum_{j=1}^{n-1} \ell_j^{sched_i}, \sum_{j=1}^{|K|} \ell_j^{sched_i}\right), \quad (4)$$

where $\ell_j^{sched_i}$ is the length of the jth operation in schedule i, and n is the ordinal of the first operation that makes the sum greater than E. MSs that depart the SAP's sphere of interaction before their schedule slot are skipped. This method of addressing starvation is more appropriate than standard aging techniques since MSs with a lower $P(A)$ (later in the schedule) are not likely to be around very long and would not be able to take advantage of the aging. Thus, with the schedule epoch, we focus on getting the higher $P(A)$ operations done rather than on fairness with respect to starvation.

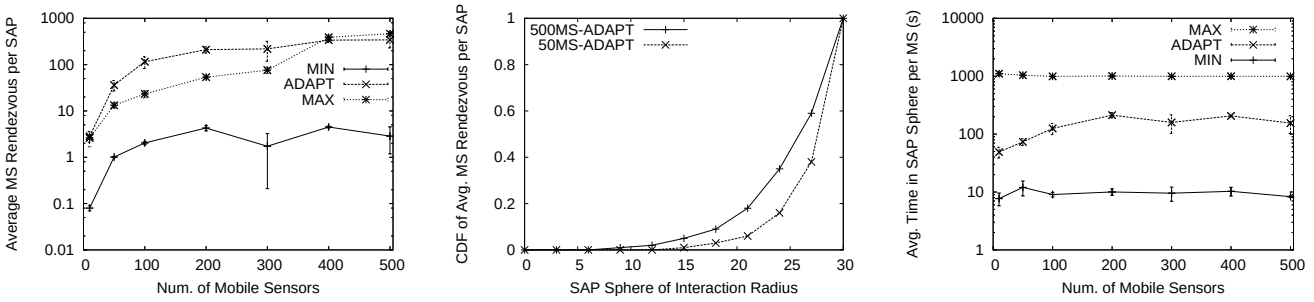
C. Extra-Effort Completion

Once the SAP scheduler has chosen which uploading or tasking operation to initiate, it is in the best interest of the system to complete that operation before the mobile sensor moves out of the sphere of interaction of the SAP. We choose this approach since in general we can not be sure of future tasking or uploading opportunities. With extra-effort completion, the SAP makes the best effort possible via sphere of interaction expansion to complete the tasking or uploading operation. Hereafter, we use the term *SAP last hop* to mean the SAP in the case of a single hop sphere of interaction and the SAP's last hop proxy in the case of a multi-hop sphere.

The SAP last hop adapts the sphere of interaction as necessary to maintain contact with the MS until the operation is completed, or further expansion is not possible. This leads to probabilistically earlier operation completion, compared to no expansion. To see this, consider a MS that associates with a SAP and needs to upload K bytes. Suppose that before it leaves the non-expanded SAP sphere of interaction it uploads k out of K bytes. Assume that MSs move with random mobility, SAPs are distributed in the total area (A_{tot}) uniformly at random and cover an aggregate area of A^*_{soi}, and the channel has a capacity of $C = 1$. Upon leaving the current SAP the long term average time the MS will take to complete its upload session is $k + \frac{A_{tot}}{A^*_{soi}}(K - k)$. On the other hand, if the SAP expands its sphere of interaction and allows the MS to transfer ℓ more bytes before leaving, then the average total upload time is $k + \ell + \frac{A_{tot}}{A^*_{soi}}(K - k - l)$. In Figure 2(a), we plot the delay savings Halo gains by expanding the sphere of interaction on the y-axis against the number of extra bytes the MS is able to transmit in its current session, i.e., ℓ in the expressions above, on the x-axis. Curves approximating a corporate campus - a controlled indoor environment ($\frac{A^*_{soi}}{A_{tot}}$=0.75), a college campus - an indoor/outdoor mix but still under a single administrative control ($\frac{A^*_{soi}}{A_{tot}}$=0.3), and an urban landscape - multiple points of control and a diverse environment ($\frac{A^*_{soi}}{A_{tot}}$=0.01) give a rough idea of the type of savings are possible. Even for a modest number of additional bytes transferred in the current session, the delay savings can be substantial.

Further, for extra-effort completion if a multi-hop sphere of interaction is required, we extend the sphere *only along a tunnel following the moving MS*. This leads to a higher operation completion probability compared to no expansion, while offering a reduced disruption probability to adjacent ongoing communications and a reduced energy consumption by the mobile sensors as compared with uniformly expanding the sphere of interaction in all directions, i.e., flooding. It is easy to see the advantage of this approach. The disrupted area for a straight multi-hop tunnel is $\pi r^2 + (n\bar{d})r$, where $\bar{d}$ is the average per hop range extension, n is the number of hops, and r is the radio range. The disrupted area for spherical expansion is lower bounded by $\pi(n\bar{d})^2 + \frac{\pi r^2}{2}$. Therefore, the relative reduction in disrupted area when using the tunnel grows as $Reduction = \pi(n\bar{d})^2 - (n\bar{d})r - \frac{\pi r^2}{2}$. Figure 2(b) shows how this reduction grows quadratically both with the number of hops extension n and the radio range r. Also, since we only forward along a tunnel after the MS, we save energy as well. Considering only the transmitters, for the tunnel forwarding the number of forwarders is $N_{tunnel} = n$, while flooding within a uniformly expanding sphere involves $N_{sphere} = \delta\pi(n\bar{d})^2$ transmitters, where δ is the average MS density in the area. Therefore, the energy savings of tunnel forwarding w.r.t. the flooding case grows as $Savings = \delta\pi(n\bar{d})^2 - n$. Figure 2(c) shows how the energy savings using the tunnel relative to the sphere scales linearly with density and quadratically with the number of hops extension. For Figures 2(b) and 2(c), assuming MSs are distributed uniformly at random, d is set to $r/2$. In summary, extending the sphere of interaction along a tunnel to the MS rather than omnidirectionally allows for quadratic reduction (w.r.t. the distance between the SAP and MS) in both the disrupted region and in the energy spent transmitting data between the SAP and MS.

To determine when the sphere of interaction should be expanded for extra-effort completion, we use a hybrid approach with both proactive (using RSSI) and reactive (using packet reception ratio) triggers. We want to proactively increase the sphere of interaction when the rate of RSSI decrease predicts that at the next time step the RSSI will be too low to support a session [14]. This method of adaptation is effective in adjusting to distance-based signal attenuation. Given the people-centric nature of the network, it is likely that human mobility will



(a) Impact of MS density on the number of MS/SAP rendezvous.

(b) Cumulative distribution of rendezvous vs. sphere radius for diffent MS densities (with ADAPT scheme).

(c) Impact of MS density on the average time each MS spends in the SAP's spheres of interaction.

Fig. 3.

sometimes cause abrupt changes in packet reception that can not be predicted by monitoring RSSI trends. For example, the authors of [10] show that when the human body is placed between an IEEE 802.14.5 transmitter and receiver, the achievable throughput (at fixed power) can drop dramatically. Therefore, if the packet reception ratio as measured during the operation at the SAP last hop falls below $Thresh_{PRR}$ then the sphere setting is reactively incremented, i.e. $s_j \rightarrow s_{j+1}$, in an attempt to maintain the connection.

III. Halo Evaluation

We base our evaluation of deadline-based sphere of interaction management on the comparison of three schemes: MIN, ADAPT, and MAX. We use ten SAP sphere of interaction settings $S = \{s_1, ...s_{10}\}$, with $M = 10$ and $K = 1$ (extra-effort completion is disabled), where setting s_j corresponds to a sphere radius of $j * 3$ distance units. In the MIN scheme SAPs always use s_1; in the MAX scheme SAPs always use s_{10}; in the ADAPT scheme each SAP independently varies its radius according to the sensing deadlines of tasks it is managing. Intuitively, the MIN scheme provides the lowest energy consumption and disrupted area; the MAX scheme provides the greatest opportunity for packet transfer between MSs and SAPs, and thus the highest operation completion rate; the ADAPT scheme attempts to hit the sweet spot of providing increased opportunities for packet transfer and operation completion when warranted by the sensing deadline, and otherwise shrinking the sphere of interaction to save energy and to be less disruptive to other transmissions. All schemes use MB-SJF scheduling.

A. Simulation Environment

We implement Halo algorithms and the required communications (see Appendix II) in nesC, and simulate several multi-SAP multi-MS scenarios using TOSSIM/Tython. TOSSIM [8] simulates Halo on the TinyOS platform, including packet exchange, timer events, etc. Tython [5] is a Python/Java front end used to manage node mobility and connectivity.

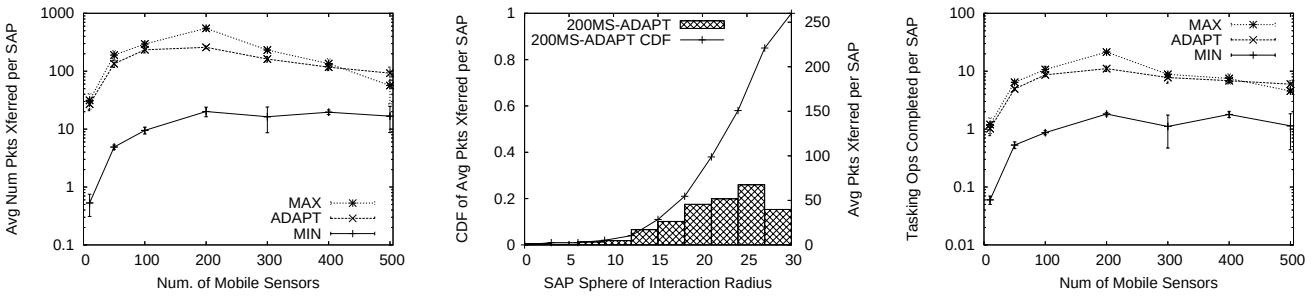
Each simulation trial is conducted on a 500×500 field. MSs are initially placed uniformly at random across the field and move according to a modified random walk. MSs choose an activity uniformly at random from {standing, walking, running} and continue with that activity for a period of time chosen uniformly at random between 1 and 1200 seconds. According to the chosen activity MSs move at a rate of, respectively, $\{0, 3, 15\}$ distance units per second in a direction chosen uniformly at random, between 1 and 360 degrees inclusive, at the same time the activity is chosen. MSs bounce off the field boundaries. 50 SAPs are placed uniformly across the field and remain stationary throughout the simulation.

MSs estimate their t_{SAP} and propagate this estimate to the SAP in the *beacon reply* to facilitate the MB-SJF schedule calculation (see Equation 3). We assume all MSs have an accelerometer that can be used for activity classification. We use empirical data from [9]; we reproduce the activity classifier confusion matrix in Table I for convenience. In the simulation, a MS may be, for example, "running" as dictated by the mobility model, but the MS believes it is running with only 90.9% probability. With 8.37% probability it thinks it is walking and tells the SAP the wrong information. Real world effects such as classification inaccuracy (or GPS error if a GPS system is used as a basis for a dwell time estimate) degrade the performance of the MB-SJF scheduler. Yet, even under worst case classification accuracy the scheduler just behaves as a random scheduler that does not consider mobility at all.

	Standing	Walking	Running
Standing	0.9844	0.0141	0.0014
Walking	0.0558	0.8603	0.0837
Running	0.0363	0.0545	0.9090

Table I. Confusion matrix for MS activity classifier. Based on actual activity classifer performance reported in [9].

Tasks arrive independently at each SAP with inter-arrival times drawn randomly from an exponential distribution with a mean of 10s. Task sizes are drawn randomly from an exponential distribution with a mean of 10 packets (packets are 128 bytes long). The sensing deadline ($t_{s(max)}$) is randomly chosen for each task from an exponential distribution with a mean of 1000s. The deadline threshold T should reflect the time it takes on average to travel the distance from the tasking SAP to the sensing target (see Section II-A). In our simulation, the T value for a given task is chosen uniformly at random in the interval from 1 to $t_{s(max)}$. This is equivalent to choosing a sensing target uniformly at random in the field and pre-filtering those tasks whose sensing deadlines do not allow enough travel time from SAP to target. A task whose sensing deadline passes



(a) ADAPT transfers nearly as many packets as MAX and about 10× more than MIN across all tested MS densities.

(b) Packets xferred generally increases with radius, but the largest radius shows a diminishing return.

(c) MIN completes an order of magnitude fewer tasking ops than ADAPT or MAX, giving poor service to apps.

Fig. 4.

before it is assigned to a MS is dropped from the SAP's task queue. The SAP's *beacon* interval (see Appendix II) is set to 5 seconds.

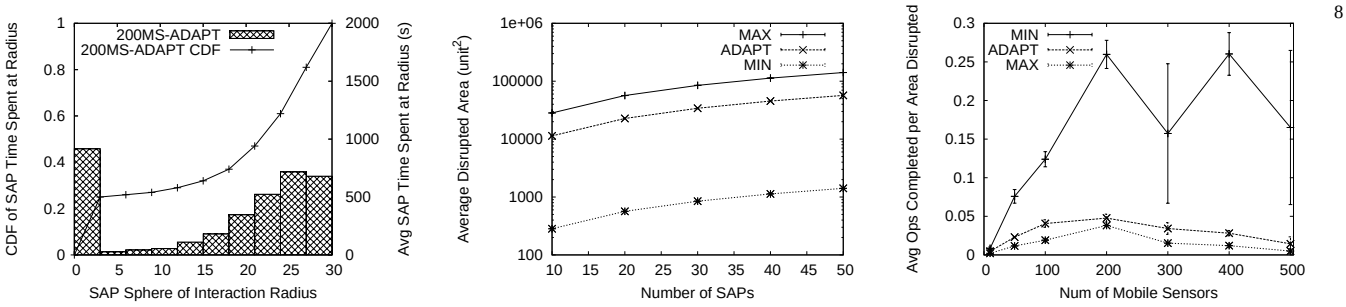
Each MS has a task queue of size 1, meaning it can only serve one application query at a time. If a task is partially transferred to a MS before a particular tasking session completes, the MS caches the state of the suspended session and resumes the session at the next met SAP. If the sensing deadline of a task that is partially transferred to a MS expires, the partial state is expunged from the MS. MSs that are fully tasked and then successfully sense the target generate a number of data packets to upload chosen randomly from an exponential distribution with a mean of 100 packets. In the simulations, approximately 20% of the fully tasked MSs successfully reach their respective target sensing regions before their sensing deadlines. We use an uploading deadline of infinity for all tasks; a MS with data to upload maintains the state of its upload session across how ever many SAP rendezvous it takes to complete the upload.

B. Impact on Tasking/Uploading Opportunity

To characterize the impact of sphere of interaction radius on the opportunity for MS/SAP rendezvous, we run simulations across a range of MS densities. Results are summarized in Figure 3, where each point represents the average of five trials (error bars indicate the 95% confidence interval), and each trial covers one hour of simulation time. In Figure 3(a), we quantify the impact of MS density on the number of MS/SAP rendezvous, plotting the average number of MS/SAP rendezvous per SAP (i.e., the total number of rendezvous divided by 50) versus the number of MSs. The y-axis is in log scale to better show the detail despite the wide spread between MIN and MAX. Unsurprisingly, the number of rendezvous generally increases with increasing MS density for MIN, ADAPT and MAX. Of interest, the ADAPT scheme actually results in more rendezvous for the intermediate MS densities tested. This is likely due to the dynamism of the sphere of interaction, resulting in "re-rendezvousing" for MSs that have moved little (e.g., standing) during the sphere adaptation time scale. The effect becomes negligible at the lowest densities (10 mobile sensors) since the overall probability of rendezvous shrinks dramatically. At high densities (i.e., above 400 MSs), this effect is overwhelmed by the sheer number of mobility-based rendezvous, and at the highest density (500 MSs) MAX yields more rendezvous than ADAPT since it always uses the largest sphere radius. Another way to see the effect of MSs density is to consider the sphere of interaction radius at which most rendezvous occur. In Figure 3(b), we show the cumulative distribution function (CDF) of the average number of MS rendezvous per SAP versus SAP sphere of interaction radius. Curves for 500 MSs and 50 MSs show how at lower densities the majority of rendezvous occur at higher values of sphere radius. For example, at a sphere radius of 27 (second largest), in the 50 MS scenario only 40% of the rendezvous have occurred, while in the 500 MS scenario already 60% of the rendezvous have occurred. While the number of rendezvous for ADAPT is sometimes greater than for MAX (see Figure 3(a)), Figure 3(c) shows that the average total time MSs spend in the SAP spheres does not exhibit the same effect. Rather, MIN<ADAPT<MAX across all tested MS densities. Here, the time values are normalized by the number of MSs, and the y-axis is plotted in log scale. While results for MIN and MAX are relatively constant across the tested densities, ADAPT has a higher variability due to its adaptive nature.

C. Impact on Bytes Transferred and Operations Completed

In Figure 4, we summarize the performance of the ADAPT scheme in terms of number of task and upload packets transferred between MSs and SAPs, and the number of tasking operations completed. As before, we run simulations across a range of MS densities, where each point represents the average of five trials (error bars indicate the 95% confidence interval), and each trial covers one hour of simulation time. In Figure 4(a), for MIN, ADAPT and MAX we report the average number of task and upload packets transferred per SAP across a range of MS densities. The y-axis is shown in log scale. Consistent with the transfer opportunity results shown in Figure 3(a), as the MS density increases the number of packets transferred increases up to the 200-MS point. After this point, the curve for MIN remains constant (as in Figure 3(a)), but the curves for ADAPT and MAX actually decrease. A closer look at our data log files reveals this is due to a combination of mobility (sessions are suspended as a MS moves out of the



(a) For ADAPT, SAPs spend a plurality of their time at the lowest setting, but the majority at the highest settings.

(b) Impact of SAP density on disrupted area. On avg., ADAPT disrupts $\approx \frac{2}{3}$ the area that MAX does.

(c) An efficiency metric for system tradeoffs shows ADAPT gives a 200% improvment over MAX.

Fig. 5.

current SAP sphere of interaction) and MAC layer collisions as the density in the contention region increases. Generally, we see that ADAPT stays close to the MAX scheme, especially at the tested density extremes, even though MAX maintains the largest sphere of interaction radius through the entirety of the simulation. In Figure 4(b), we provide insight into why the packet transfer performance of ADAPT is able to remain close to MAX. Figure 4(b) shows the distribution (on the right axis) and cumulative distribution function (on the left axis) of packets transferred across sphere of interaction radius for the median density scenario (200 MSs). We see that, in contrast to the rendezvous distribution shown in Figure 3(b), the packet transfer distribution does not monotonically increase with increasing sphere radius. Rather, the additional packets transferred at the maximum sphere radius is less than the penultimate radius, indicating a diminishing return for increasing the sphere radius. Figure 4(c) shows the average number of tasking operations completed per SAP plotted in log scale across a range of MS densities. We see that the behavior of adapting the sphere of interaction radius based on proximity to the sensing-deadline for a particular task leads to excellent comparative performance for ADAPT. While the MAX scheme completes somewhat more tasking operations, on average ADAPT completes 85.5% of the tasks MAX does and nearly 10 times as many as MIN does. As we show in the next section, the deadline threshold T (see Section II-A) offers a means to tune the proactivity of ADAPT to more closely approximate the performance of MAX in terms of packets transferred and operations completed by increasing the SAP sphere of interaction more aggressively. However, increasing the sphere of interaction also increases the area disrupted by the SAP, potentially interrupting communication among MSs in the field. Additionally, MS energy depletion increases since a multi-hop SAP proxy chain may be established and maintained, and the target MS may have to transmit at a higher power to match the transmission power of the SAP last hop.

D. Impact on Disrupted Area

In Figure 5, we characterize the extent to which an increased sphere radius impacts the disrupted area. Since MS energy depletion is similarly proportional to the sphere of interaction radius, we omit energy-related results here. As previously, each point represents the average of five trials (error bars, where shown, indicate the 95% confidence interval), and each trial covers one hour of simulation time. Figure 5(a) shows the distribution (and CDF) of the time that SAPs on average spend at each of the 10 sphere of interaction settings when using the ADAPT scheme. The data is from the median density 200 MS scenario, but time distributions for the other tested MS densities are similar. While SAPs spend a plurality of their time (about 25%) at the lowest sphere setting (radius of 3 distance units), implying the minimum possible disruption, in aggregate most of the time is spent at or above the eighth setting (radius of 24 distance units). In fact, the average sphere radius is 19.02 distance units, which is about two thirds of the sphere radius used by the MAX scheme. Figure 5(b) reflects this relationship and also indicates that the disturbed area in the field can quickly get very large as the SAP density increases (number of MSs is fixed at 200). The MIN scheme disrupts a much smaller area, but as we see in Figure 4 MIN also transfers and completes tasking operations at a rate an order of magnitude lower rate than ADAPT. To get a sense of how these pros and cons compare, we define the average number of operations completed per unit area disrupted as an efficiency metric $\eta = Ops/Area$. Figure 5(c) shows this metric plotted for the MIN, ADAPT and MAX schemes across a range of MS densities. On average $\eta_{\text{ADAPT}}/\eta_{\text{MIN}} = 0.27$ and $\eta_{\text{ADAPT}}/\eta_{\text{MAX}} = 2.18$; ADAPT gives a 200% improvement over MAX., while facilitating a nearly $10\times$ improvement over MIN in terms of completed operations. The MIN scheme can exhibit high variability due to the small number of total tasking and uploading operations that are completed. While η provides a notion of efficiency, as formulated it is meant to reflect the tradeoff between resource conservation, i.e., disrupted area and MS energy, and a coarse-grained quality of service in terms of system responsiveness to application queries, rather than defining the optimal operating point.

In Figure 6, we illustrate how adjusting the deadline threshold T can be used to move the operating point of the ADAPT scheme from more resource conserving to offering a lower average completion delay to application queries. For the 200-MS scenario, Figure 6 shows the CDF of the time that SAPs on average spend at each of the 10 sphere of interaction settings when using the ADAPT scheme for different val-

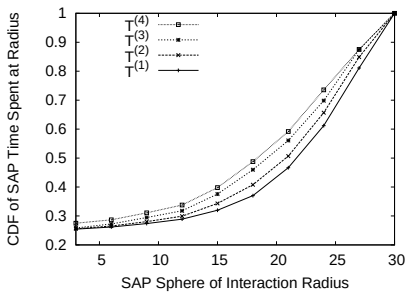


Fig. 6. Impact of the deadline threshold T.

ues of the average deadline threshold, $\overline{T}$. In the previous simulations, T is chosen uniformly between 1 and $t_{s(max)}$, so $\overline{T} = t_{s(max)}/2$. In Figure 6, we adopt the following shorthand: $T^{(i)} = t_{s(max)}/2 \cdot i$. As $\overline{T}$ increases, SAPs spend proportionally more time at higher sphere settings, resulting in more packets transferred and better delay service to the applications but consuming more resources of the MS cloud (i.e., energy and peer-to-peer communications opportunities).

IV. Related Work

Managing the SAP sphere of interaction is related to adaptive clustering. Work from the MANET community proposes various clustering techniques, but invariably to increase routing efficiency and/or reduce routing protocol overhead (e.g., [7] [6] [4] [12]). Zone Routing Protocol [7] sets a zone boundary between proactive and reactive routing to reduce the number of route request packets while providing good route acquisition delay. However, a method of determining an appropriate zone radius is not specified. A study of adaptive clustering with the end of maximizing operation completion rate does not exist. The relationship between node density, transmission power, and neighbor set cardinality has been studied in the context of wireless graph connectivity [1] [17]. The effects of the relationships between transmission power, node density, and node mobility patterns on the operation completion rate have not been reported. A number of existing scheduling policies may be appropriate for SAP operation scheduling, but none have been evaluated with the unique combination of constraints present in a large scale mobile sensor network.

The modulation of the SAP's sphere of interaction to manage collection and tasking opportunities is analogous to the "cell breathing" approach used in cellular telephony and proposed [2] for 802.11 access nodes for system load balancing. In our case, we wish not to balance load among the SAPs, but to adapt to backend application demands and mobile sensor mobility patterns. Power control in cell phones is not new, but the focus there is primarily to save handset energy and secondarily to reduce adjacent cell interference. While we share the first concern, cellular mechanisms are too complex, and use a separate control channel which is not generally available on embedded sensing platforms.

V. Conclusion

We have shown how by adapting SAPs' spheres of interaction Halo can manage the opportunity for interaction between mobile sensors and sensor access points, while striking a balance between resource consumption and operation completion. We have proposed extra-effort completion to reduce delay by lowering the average number of SAP sessions a mobile sensor requires to complete tasking and uploading operations. Halo uses a new scheduling discipline (MB-SJF), based on mobility statistics and sensor-based inputs, that is tailored for the typical characteristics of the people-centric sensing domain. MB-SJF is shown to provide up to a 10% increase in operation completion rate compared to FIFO, random selection, and shortest-job-first scheduling, independent of the gains achievable from SAP sphere of interaction management. Together, the Halo framework provides improved support for the needs of delay-aware applications in the people-centric sensing context.

References

[1] A. Agarwal and P. R. Kumar. Capacity bounds for ad-hoc and hybrid wireless networks. In *ACM SIGCOMM CCR, Special Issue on Science of Networking Design*, pp. 71-81, Vol. 34, No. 3, July 2004.

[2] P. Bahl, M. Hajiaghayi, K. Jain, V. Mirrokni, L. Qiu, and A. Saberi. Cell Breathing in Wireless LANs: Algorithms and Evaluation. In *IEEE Trans. on Mobile Computing*, Vol. 6, No. 2, Feb 2007.

[3] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, and R. Peterson. People-Centric Urban Sensing. In *Proc. WICON'06*, Boston, Aug 2006.

[4] M. Chatterjee, S. K. Das, and D. Targut. WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks. In *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, Vol. 5, Apr 2002.

[5] M. Demmer, P. Levis, A. Joki, E. Brewer and D. Culler. Tython: a Dynamic Simulation Environment for Sensor Networks. EECS Dept., UCal Berkeley Tech. Report No. UCB/CSD-05-1372, 2005.

[6] S. Du, et al. Self-Organizing Hierarchical Routing for Scalable Ad Hoc Networking. In *ACM Ad Hoc Networks*, Vol 6, Iss 4, Jun 2008.

[7] Z. J. Haas. A New Routing Protocol for the Reconfigurable Wireless Networks. In *Proc. ICUPC'97*, San Diego, Oct 1997.

[8] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proc. SENSYS'03*, pp. 126-137, Los Angeles, Nov 5-7, 2003.

[9] E. Miluzzo, N. D. Lane, S. B. Eisenman, and A. T. Campbell. CenceMe - Injecting Sensing Presence into Social Networking Applications. In *Proc. EUROSSC'07*, pp. 1–28, 2007.

[10] E. Miluzzo, X. Zheng, K. Fodor and A. T Campbell. Radio Characterization of 802.15.4 and its Impact on the Design of Mobile Sensor Networks. In *Proc. EWSN'08*, Bologna, Jan 30/31 - Feb 1, 2008.

[11] N. Ravi, P. Stern, N. Desai, L. Iftode. Accessing ubiquitous services using smart phones. In *Proc. PERVASIVE'05*, Mar 8-12, 2005.

[12] J. Sharony. A Mobile Radio Network Architecture with Dynamically Changing Topology Using Virtual Subnets. In *Proc. ICC'96*, pp. 807–812, Vol. 2, Jun 1996.

[13] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating Systems Concepts, Seventh Edition*, John Wiley & Sons, Inc., 2004.

[14] K. Srinivasan and P. Levis. RSSI is Under Appreciated. In *Proc. EMNETS'06*, Cambridge, MA, 2006.

[15] S. Srinivasan, A. Moghadam, S. G. Hong, H. G. Schulzrinne 7DS - Node Cooperation and Information Exchange in Mostly Disconnected Networks. In *Proc. ICC'07*, Jun 1, 2007.

[16] M. Srivastava, et al. Wireless Urban Sensing. In *CENS Tech. Report #65*, Apr 2006.

[17] X. Zhang and N. F. Maxemchuk. The Effects of the Number of Neighbors in Multihop Wireless Networks. To appear in *Int'l Journal of Wireless and Mobile Computing, Special Issue on Group Communications in Ad hoc Networks*.

APPENDIX I: SAP MODEL FOR FIGURE 1(A)

In the following, we provide the trigonometry formulations used in the calculation of the curves in Figure 1(a).

A. Average chord length through a circle of radius r

To estimate how the data transfer opportunity between a SAP and a MS walking through the SAP's sphere of interaction scales with the radius of the sphere of interaction, we place a SAP at the center of a circle with radius r and calculate the average chord length (representing possible MS trajectories) through the circle. The transfer opportunity in seconds is simply this average chord length multiplied by the MS speed. Let A and B be two points on the circumference of a circle centered at point O. A and B represent the entry and exit points, respectively, of the MS's trajectory through the SAP's sphere of interaction. Let c be the length of the chord connecting A and B, and let θ be the acute angle between the rays OA and OB. Then

$$c = (2r^2 - 2r^2 cos\theta)^{\frac{1}{2}} = 2r \cdot sin(\frac{\theta}{2}).$$

The average chord length $\overline{c}$ can then be found by integrating across all values of θ. The problem is symmetric so we only need to consider $0 \leq \theta \leq \pi$.

$$\overline{c} = \frac{1}{\pi}\int_0^{\pi} c \cdot d\theta = \frac{4r}{\pi}$$

B. Average distance from the average length chord to the origin

To estimate how the radio transmit energy cost to the MS scales with the radius of the SAP's sphere of interaction, we place a SAP at the center of a circle with radius r and begin by calculating the average distance from the average length chord to the SAP. Let A and B be the two endpoints of the average length chord of a circle centered at point O. From the previous calculation, we have the length of AB as $\frac{4r}{\pi}$. Let C be the midpoint of AB, and let α be the angle between ray OA and segment OC; let b represent the length of segment OC. Then $\alpha = arcsin(2/\pi)$, and $b = r \cdot cos(\alpha)$.

At any point K along the chord AB, the distance to the origin is $d = b/cos(\phi)$, where ϕ is the angle AOK. The average distance $\overline{d}$ can then be found by integrating across the values ϕ takes as the MS traverses the chord from A to B. The problem is symmetric so we only need to consider $0 \leq \phi \leq \alpha$.

$$\overline{d} = \frac{1}{\alpha}\int_0^{\alpha} d \cdot d\phi = \frac{b}{\alpha}\left[ln\left|tan(\frac{\alpha}{2} + \frac{\pi}{4})\right|\right]$$

With the average distance from the MS to the SAP along the average length chord, we can now estimate the energy cost E by multiplying the average transmit power by the time the MS spends traversing the average length chord. We use a simplified model and take the average transmit power to be $\overline{d}^{\,n}$, where n is the Friis path loss exponent. The traversal time is simply $\overline{c} \cdot s$, where s is the average traversal speed.

$$E = \overline{d}^{\,n} \cdot \overline{c} \cdot s$$

C. Disrupted area as MS traverses the average length chord

To estimate how the average area disrupted by MS/SAP communications scales with the radius of the SAP's sphere of interaction, we place a SAP at the center of a circle i_{SAP} with radius r, with points A, B, C, and O defined as before. We place the MS at the center of its own circle i_{MS} of radius r. As the MS traverses the average length chord, the minimum separation between SAP and MS of b occurs when the MS reaches point C. The maximum separation of r occurs at points A and B. We calculate bounds on the disrupted area (union of circles i_{SAP} and i_{MS}) where points of minimum and maximum MS/SAP separation along the average length chord respectively correspond to the minimum and maximum areas disrupted by MS/SAP communication.

To calculate both the minimum disrupted area A_{min} and the maximum disrupted area A_{max} we use well known methods to calculate the circle intersection and subtract it from the sum of circle areas.

$$A_{min} = 2\pi r^2 - 2(r^2 \cdot arccos(\frac{b/2}{r}) - \frac{r \cdot b}{2} sin(arccos(\frac{b/2}{r})))$$

$$A_{max} = 2\pi r^2 - 2(\frac{\pi r^2}{3} - \frac{r^2}{2}\frac{\sqrt{3}}{2})$$

APPENDIX II: COMMUNICATION PROTOCOL DESCRIPTION

In the following, we describe an implementation of the signaling required to support the Halo mechanisms.

A. Sphere Population Discovery

In order for the SAP to determine what nodes are within its current sphere of interaction, when the SAP's operation queue is non-empty it periodically broadcasts a *beacon* packet except during ongoing uploading and tasking operations. This packet contains two fields, the SAP address and the transmit power with which the SAP transmitted the beacon packet. After an optional mutual authentication exchange (e.g., using public key cryptography) that minimally establishes a globally unique session ID, recipients of this beacon unicast a *beacon reply* packet to the SAP address and at the same transmit power as indicated in the *beacon*. However, MS nodes that are not available for tasking (e.g., a full task queue) and do not have anything to upload do not reply to beacons. The *beacon reply* packet contains the following information: respondent address, the RSSI of the received *beacon*, equipped sensors of the respondent, metadata about sensed data to be uploaded (i.e., query ID, and the number of bytes), metadata about any previously unfinished tasking sessions (query ID and bytes left to transfer), and t_{SAP} and $t_{\overline{SAP}}$ estimates. From the information obtained by the received *beacon reply* packets, the SAP generates a schedule in accordance with current system data collection and tasking priorities.

B. Operation Sessions

To initiate the scheduled uploading or tasking operation, the SAP unicasts a *start* packet to the associated MS. The *start* packet contains the session ID established during the authentication exchange address, the query identifier, and the number of bytes to be transferred. A session heart beat interval is also included, whose usage is described later. The *start* packet is acknowledged by the target MS to verify that it has not been already tasked by another SAP in the interim, in the case where SAP spheres are overlapping.

For tasking operations, the SAP unicasts *task* packets to the target MS. Each task packet includes the SAP transmit power, the session identifier previously sent in the *start* packet, the query identifier of the query being tasked, and the number of tasking bytes contained in the payload. Since operation scheduling is atomic, it is important to identify when a MS has left the SAP's sphere of interaction. For this purpose, the target MS unicasts periodic *session heartbeat* packets[1] to the SAP at the interval specified in the start packet throughout the tasking session. Each session heartbeat packet contains the session identifier and the RSSI value of the last received task packet. The tasking session times out on the MS side if task packets have not been received for a given time. On the SAP side, the session times out if session heartbeat packets are not received as often as expected. Halo uses a selective negative acknowledgments to provide reliability to the tasking transfer. Omitting the periodic session heartbeats, the message exchange for a tasking session is shown in Figure 7.

[1]Link layer acknowledgments can be used instead, if available.

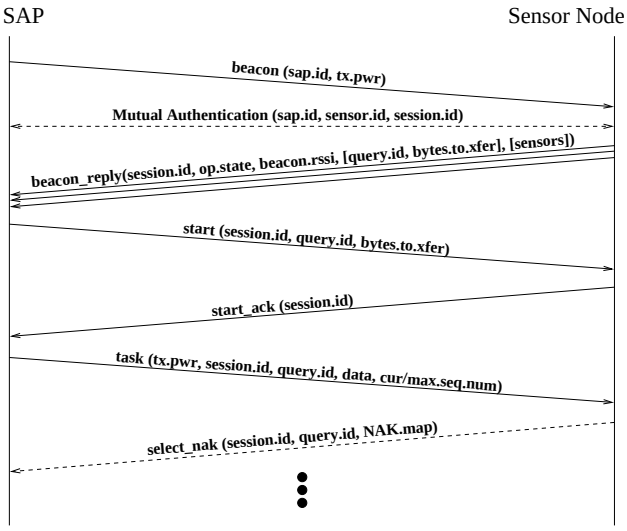


Fig. 7. Message exchange diagram for Halo tasking operations.

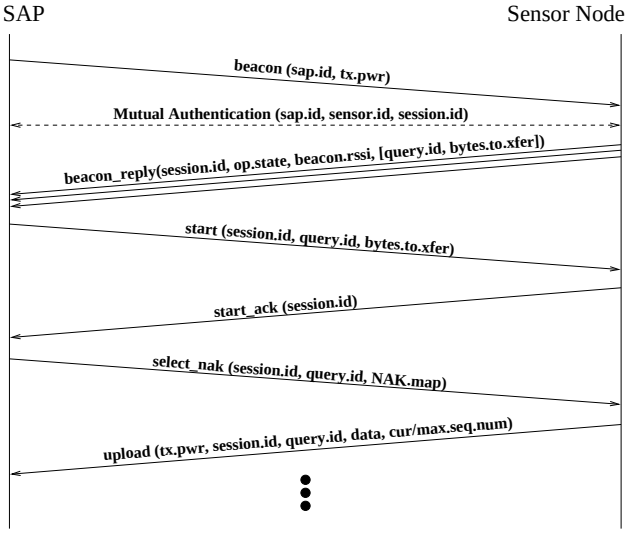


Fig. 8. Message exchange diagram for Halo uploading operations.

For uploading operations, the MS unicasts *upload* packets to the SAP. Each upload packet includes the session identifier, the query identifier of the query associated with the uploaded data, and the number of sensed data bytes in the payload. To avoid the MS sending upload packets when it has moved out of the sphere of interaction (energy waste, bandwidth waste), throughout the uploading session the SAP unicasts periodic session heartbeat packets to the MS at the interval specified in the *start* packet. Each session heartbeat packet contains the session identifier and the RSSI value of the last received *upload* packet. Additionally, it contains the SAP transmit power, which is used for sphere adaptation as discussed in the next section. The upload session timeout conditions are analogous to the tasking case. Selective negative acknowledgements are used to provide a reliable upload packet transfer. Omitting the periodic session heartbeats, the message exchange for an

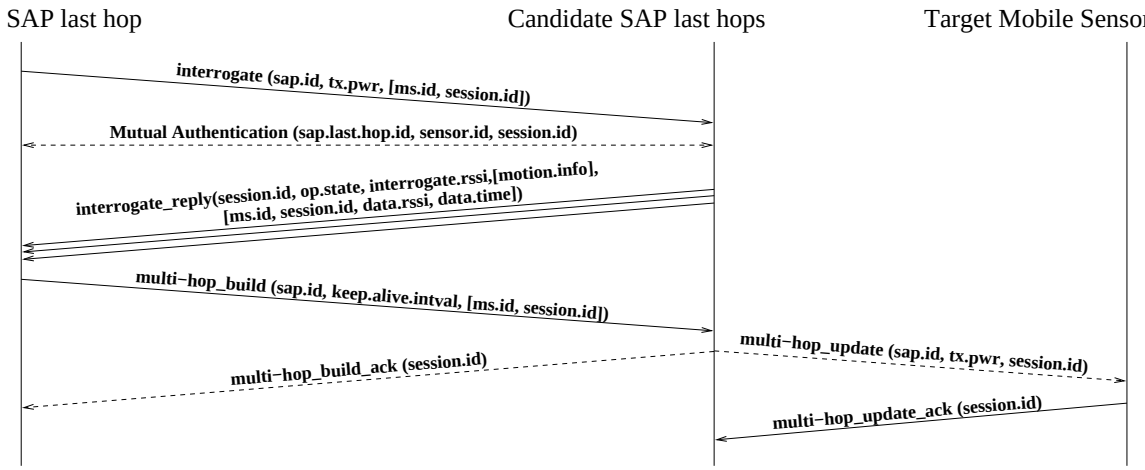


Fig. 9. Message exchange diagram for Halo SAP last hop selection.

uploading session is shown in Figure 8.

SAPs and MSs combine the trend of the RSSI values of received *session heartbeat* packets, with the expected heart beat interval to determine when the MS has drifted out of the SAP sphere of interaction during the course of a tasking or uploading session. In case the MS leaves the SAP sphere of interaction before an operation session is complete, the session identifier and current transfer state can be kept by the MS for use at the next encountered SAP, while the SAP can store this "session cookie" in a central server accessible by all SAPs.

C. Sphere of Interaction Adaptation

During the course of Halo's extra-effort completion, the SAP last hop makes decisions on when to expand the current sphere of interaction based on the RSSI of packets received from the target MS. During a tasking session, the RSSI of incoming *session heartbeat* packets (combined with their frequency) is used. During an uploading session, the RSSI of incoming *upload* packets is used. Outside of any ongoing uploading or tasking operation, the SAP may wish to expand the sphere of interaction based on application demands (c.f. Section II-A). Depending on the sphere extension setting a transmit power increase and/or a multi-hop extension may be required.

Transmit power control is always applied to the SAP last hop, and is applied to the MS target during an uploading or tasking operation. The former case is a trivial matter of updating a local radio parameter, while the latter case requires downstream signaling. In the case of a tasking session, the transmit power that the target MS should use for its communication (e.g., *session heartbeat* packets) with the SAP last hop is included in the *task* packet. For the case of an uploading session, the power information is read from the periodic *session heartbeat* packets sent to the MS by the SAP last hop.

Multi-hop extension of the sphere of interaction is a two stage process. In the first stage, the SAP last hop identifies the best MS within the current sphere of interaction to act as the new last hop SAP proxy. In the second stage the SAP proxy chain is updated and (in the case of an ongoing uploading or tasking operation) the session with the target MS is reestablished. The message exchange diagram for multi-hop extension is shown in Figure 9.

1) Last Hop SAP Proxy Selection: In order for the current SAP last hop to identify the best node to extend the SAP proxy chain, the SAP broadcasts an *interrogate* packet. This packet contains the SAP last hop address, and the transmit power with which the SAP last hop transmitted the *interrogate* packet. In the case where multi-hop extension occurs in the context of extra-effort operation completion, the *interrogate* packet additionally contains the address of the target MS and the session identifier. Recipients of this interrogation unicast an *interrogate reply* packet to the SAP last hop and at the same transmit power as indicated in the *interrogate* packet. The *interrogate reply* packet contains the following information: respondent address, the RSSI of the received *interrogate* packet, and motion information (e.g., position and velocity) if available. In the case where multi-hop extension occurs in the context of extra-effort operation completion, the *interrogate reply* packet additionally contains the target MS address, operation session identifier, the RSSI of the most recent packet overheard from the session in question, and the time that packet was received. With this information obtained from the *interrogate reply* packets, the SAP last hop chooses the new SAP last hop based on a combination of an estimate of which candidate is least mobile, and which candidate most recently had the strongest signal from the target MS.

2) SAP Proxy Chain Setup and Maintenance: Once the "new" SAP last hop has been chosen from the available candidates, the current SAP last hop unicasts a *multi-hop build* packet to the new SAP last hop. The *multi-hop build* packet contains the SAP address and a keep alive interval whose uses is described later; in the case of extra-effort operation completion the target MS address and the session identifier are additionally contained. The "new" SAP last hop then becomes the "current" SAP last hop. If the expansion is in the context of extra-effort completion, the SAP last hop then unicasts a *multi-hop update* packet to the target MS. The *multi-*

hop update packet contains the SAP last hop address, the session identifier, and the transmit power at which the packet is sent. Upon reception of the *multi-hop update* packet the target MS sends an acknowledgment to confirm the reestablishment of the session. The *multi-hop update* packet also acts as an implicit acknowledgment to the previous SAP last hop that the transfer of "current" SAP last hop status has been successfully passed. Without receiving this acknowledgment, the previous last hop again takes the "current" status and tries again to extend the chain. At this point the operation data transfer continues as before along the SAP proxy chain between the MS and the root SAP. It is possible that even after a multi-hop extension, the SAP last hop does not immediately reestablish connection with the target MS. In that case, the SAP last hop will iteratively initiate a further sphere of interaction expansion through transmit power increase and multi-hop extension up to the multi-hop cap M and the highest supported transmit power. If the session can still not be reestablished, the SAP last hop at that time initiates a chain tear down by sending a *terminate* packet back up the SAP proxy chain.

All uploading and tasking data transfer, as well as the *session heartbeat* packets that would normally pass directly between the target MS and the SAP in the single hop case travel along the hops of the SAP proxy chain. This session traffic allows Halo to verify the integrity of the proxy chain, since the SAP proxies themselves may also move, breaking the chain. In case the SAP proxy chain breaks, the last remaining link of the chain still connected to the SAP will attempt to rebuild the chain by selecting a new proxy and thereby reestablishing connectivity with the target MS. Members of the isolated portion of the chain will realize they are no longer receiving *session heartbeat* and will terminate the session. Outside of an operation session, there is no traffic along the chain. Therefore, a *keepalive* packet is sent and echoed along the chain at the rate indicated in the *multi-hop build* packet, where the *keepalive* packet plays the same roll in chain integrity verification and maintenance as the *session heartbeat* packet.