Power Aware Management Middleware for Multiple Radio Interfaces

Roy Friedman and Alex Kogan

Department of Computer Science, Technion
{roy,sakogan}@cs.technion.ac.il

Abstract. Modern mobile phones and laptops are equipped with multiple wireless communication interfaces, such as WiFi and Bluetooth (BT), enabling the creation of ad-hoc networks. These interfaces significantly differ from one another in their power requirements, transmission range, bandwidth, etc. For example, BT is an order of magnitude more power efficient than WiFi, but its transmission range is an order of magnitude shorter. This paper introduces a management middleware that establishes a power efficient overlay for such ad-hoc networks, in which most devices can shut down their long range power hungry wireless interface (e.g., WiFi). Yet, the resulting overlay is fully connected, and for capacity and latency needs, no message ever travels more than $2k$ short range (e.g., BT) hops, where k is an arbitrary parameter. The paper describes the architecture of the solution and the management protocol, as well as a detailed simulations based performance study. The simulations largely validate the ability of the management infrastructure to obtain considerable power savings while keeping the network connected and maintaining reasonable latency. The performance study covers both static and mobile networks.

Keywords: Wireless Ad-hoc Networks, Power Aware Overlays, Multiple Radio Interfaces

1 Introduction

Mobile devices enabled with multiple wireless interfaces are becoming increasingly common. As evidence, most laptops, smartphones and PDAs are equipped with WiFi and BlueTooth (BT) radios. Future devices are expected to include even more interfaces, supporting currently emerging standards, such as WiMax and ZigBee. An important aspect of these technologies is their ability to create mobile ad-hoc networks (MANETs), which enable direct communication between devices in an infrastructure independent manner and offer fast and easy deployment in situations where it is not possible or not cost effective otherwise.

Yet, mobile devices are typically battery operated. Hence, any application or middleware for such ad-hoc networks must be energy efficient. Researchers have found that wireless communication is one of the main sources of power consumption in mobile devices [2,7,15]. Consequently, efficient power utilization by the wireless communication sub-system is crucial for the success of such networks.

All aforementioned technologies for wireless communication differ dramatically one from another in several parameters, e.g., maximum transmission range, energy requirements and available bandwidth [8, 15]. Hence, the choice of which wireless technology to use has great impact on the power consumption of the devices, but also on the network connectivity and capacity. For example, a simple approach of switching off WiFi and only keeping BT operational may result in a disconnected network. On the other hand, leaving all interfaces up all the time is too wasteful. This is because, especially in WiFi, the power consumed by an idle interface is only slightly lower than the power consumed while transmitting or receiving [3, 4, 7, 15].

Power utilization in mobile ad-hoc networks has received substantial attention in recent years. Most previous works, however, consider devices with a single wireless interface. The proposed solutions range from constructing an energy-efficient overlay of active nodes, e.g., [4, 19, 20], through adjusting transmission range of radios, e.g., [9, 11, 13], to exploiting interface-specific techniques, such as the power-saving mode (PSM) of WiFi, e.g., [1, 2, 12]. All approaches, however, share the problem of potentially lost connectivity and pose non-trivial assumptions on the underlying system, such as the availability of location information, radios with a varying transmission range, synchronized clocks, etc.

In this paper, we take a different, more integrated approach to the problem of power utilization for networks in which the devices own two wireless communication interfaces, e.g., WiFi and BT.[1] Specifically, we develop a middleware service, called *Overlay Construction and Maintenance (OCM)*, that controls which wireless interfaces should be used on each device in the following manner: OCM constructs an overlay of devices such that each device who is a member of the overlay keeps both its interfaces active. All other devices not in the overlay shut down their long range power hungry wireless interfaces (typically WiFi). In order to keep the network latency and capacity reasonable, no device is further than k short range hops from its nearest overlay member, where k is an arbitrary parameter. Moreover, the transitive network formed by the collection of short range radios and active long range radios must be fully connected. Also, devices elected by OCM to the overlay tend to be the ones with highest remaining battery power, and the overlay is constantly maintained in order to reflect both dynamic changes in the network topology as well as remaining battery power.

Each overlay member acts as a *cluster-head* for its non-overlay short-range k-hop neighborhood. OCM interacts with a slightly modified routing infrastructure at the IP layer such that messages are routed using a standard reactive routing protocol, such as DSR [10], to the cluster-head closest to the target node. From the cluster-head, the message is forwarded through table driven routing, also managed and dictated by OCM, along short range hops to the actual destination device.

Although at the abstract level our solution does not assume any specific technology, the most natural scenario for the application of OCM is a network composed from laptops and mobile phones equipped with BT and WiFi radios. Hence, we evaluate the performance of OCM with typical power consumption parameters for BT and WiFi network cards under several network conditions, such as density and mobility of nodes. For the latter, we use several mobility models, e.g., a static model with random place-

[1] Extending our solution to multiple interfaces is straight forward.

ment and random way-point [10]. Yet, one of the possible domains for the application of OCM is a university campus or a school. In these settings, the entire network can consist of hundreds of nodes. However, at any given time, there are sets of dozens of devices that remain in proximity for long periods of time, but may also move from one location to another. In order to capture the specific characteristics of this domain, we introduce and evaluate OCM under a novel mobility model. In this model nodes prefer to stay or move in the vicinity of one of the preselected hot-spots, corresponding to classrooms and gathering places (such as cafeterias, libraries, meeting rooms, etc.). Nodes mostly switch between these places at the same times, corresponding to beginnings and endings of classes and breaks.

Since measuring the performance for a network of a few hundreds of nodes was not feasible, we opted to present simulation results. Yet, the simulations were performed with the complete Java code of the implementation. Through extensive simulations, we show that OCM is able to save significant portions of energy, while keeping the latency under tolerable limits. As could be expected, the energy gain produced by OCM is especially high when the network becomes denser and mobility is low, which matches well campus and school environments.

The rest of the paper describes and evaluates the OCM middleware. The related work is surveyed in Section 2. Section 3 presents the architecture of OCM and Section 4 provides details on its management module. The integration of OCM with routing is discussed in Section 5. Section 6 presents an extensive simulation based evaluation of the performance of OCM. The technical details on extensions implemented in the simulator to enable our evaluation are given in Appendix A. We conclude in Section 7.

2 Related work

A few recent papers consider wireless networks consisting of devices equipped with multiple radio interfaces. The most related to our work is that of Bahl et al. [3], which outlines the advantages behind systems that use two radios in an integrated manner. They investigate multi-radio solutions to several problems in wireless computing, such as energy management and capacity enhancement, and show significant benefits for such solutions over single-radio ones. Although their work does not rely on any particular radio technology, it requires, however, that all radios of the sender should be able to communicate with all radios of the receiver, meaning effectively that devices should stand close enough or all radios should have the same transmission range.

Pering et al. [15] introduce a system called CoolSpot that enables traditional WiFi hot-spots with BT interfaces and with a policy for multi-radio management. The paper considers single-hop communication between a wireless mobile device and such a hot-spot, where a closely located device is able to switch automatically between two interfaces in order to reduce its power consumption. The authors also provide an empirical evaluation of several policies that effectively manage radio switching.

Energy efficiency in single-radio wireless networks is a well studied problem. Brevity precludes us from mentioning all of the works, so we will refer to exemplary ones, and explain the difference between their approach and ours.

The idea of constructing an overlay of active nodes responsible for routing while other nodes may turn their radios off was examined by several researchers. Xu et al. [20] propose the GAF algorithm, which partitions a wireless network into small virtual grids. Based on location information provided by a GPS or similar systems, GAF selects one node in each grid to remain active, while the rest of the nodes are put into sleep. Chen et al. [4] present SPAN, a probabilistic power-saving technique where a node decides to join an overlay if it discovers two neighboring nodes that cannot communicate directly or through another node in the overlay. Similar ideas are used by Wu et al. [19] in order to build an overlay consisting of nodes with high remaining energy level. Since these works consider a single wireless interface (SPAN is even designed for WiFi only), when a node turns its radio off, it is unable to receive incoming messages, eventually increasing the number of lost messages, or requiring overlay nodes to store messages for relatively long durations, increasing the latency and buffering requirements. Thus, this approach is inappropriate in systems with heavy communication patterns.

Another extensively studied power-saving technique is to control the topology by varying the transmission range of the radios, e.g., [9, 11, 13]. The main tradeoff in the topology control comes from the following fact: Decreasing the transmission range decreases the power consumption significantly and reduces the interference. On the other hand, it may hurt network connectivity and increase the diameter of the network (in terms of number of hops), which in turn may increase the communication latency and decrease the network capacity. In addition, power consumption in idle state still remains a problem.

The IEEE 802.11 standard defines a power-saving mode (PSM), which allows to switch the wireless card into sleep mode in case of no activity and wake it periodically to probe an access point for pending messages. While PSM can significantly reduce the power consumption during idle periods, it was found that in certain common interactive applications it may cause substantial increase in latency and even in power [1, 12]. In order to cope with the limitations of PSM, several enhancements were proposed, tailored mainly to application specific domains, such as web-browsing [2, 12] and distributed file systems [1]. Besides being developed solely for WiFi, though, PSM is orthogonal to the approach presented in this paper, and thus can be used in addition to it.

In order to evaluate the performance of OCM, we integrate it with an ad-hoc routing protocol in a way that uses a reactive routing on the long range interface between the overlay nodes, while the routing to/from non-overlay nodes is done proactively on the short range interface. The resulting scheme resembles several protocols proposed for hierarchical routing in single-radio networks (e.g., [14, 16]). The main benefit of such protocols is scalability, since only a small fraction of nodes, i.e., those in the overlay, are involved in the route discovery process.

3 Architecture of OCM

The OCM architecture is depicted in Figure 1. Its main part is a *management* module, accompanied by a couple of *heartbeat* modules, all running as independent processes within a MANET device equipped with two radio interfaces. The management module is responsible for the construction of the power aware overlay and its maintenance

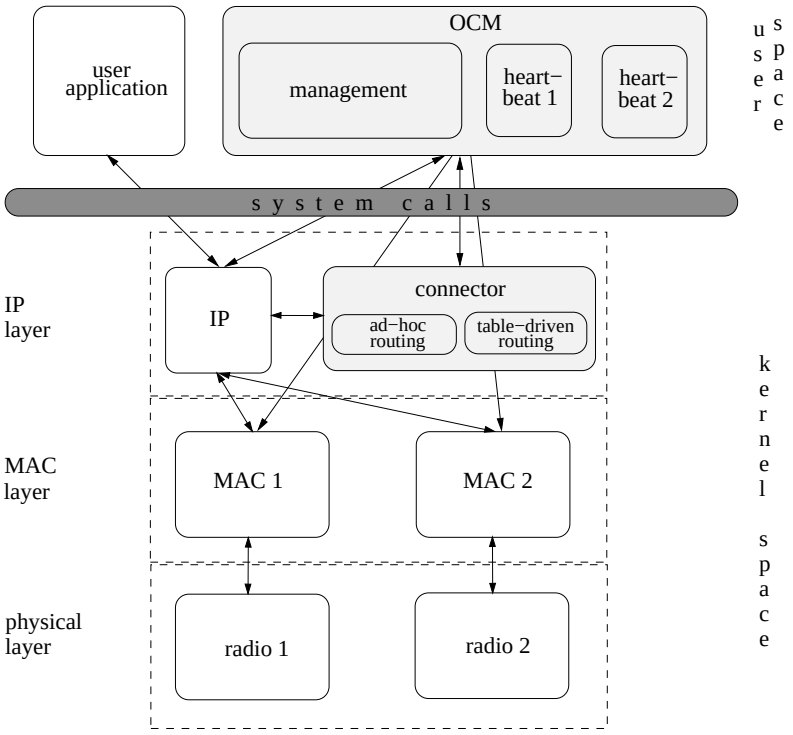

Fig. 1. Architecture overview.

according to dynamic changes in the topology and remaining battery power of the devices. Based on the internal logic described in the following section, the management module decides to switch radios on and off by executing system calls to corresponding MAC modules of the radios, as shown in Figure 1. By a slight abuse of terminology, in the sequel, when referring to OCM, we actually mean the management module.

The management module is assisted by heartbeat modules that utilize the common technique of periodically broadcasting heartbeat messages for discovery of new neighbors and notification on lost neighbors. When a heartbeat message from a previously unknown node is received, this node is considered as a new neighbor. When no heartbeat message is received from a neighbor during some predefined period of time, the link with this neighbor is considered to be lost. A clear trade-off exists between the accuracy of the information and the frequency of broadcasts.

The OCM middleware interacts with a *connector* module, which is a slightly modified routing infrastructure at the IP layer. The module gets its name from the capability to connect between two independent radio interfaces. It exposes a standard routing API for the IP module and encompasses two routing protocols: a reactive ad-hoc routing intended for the long range interface and a proactive table-driven routing intended for the short range interface. The routing table for the latter is managed by OCM through system calls. When the connector module is given a packet to be routed from the IP

module, it passes this packet to one of the two routing protocols depending on source and destination addresses of the packet. The integration of OCM with the connector module is detailed in Section 5.

4 The OCM management module

OCM is designed to manage the wireless communication interfaces by clustering the network. This section presents the assumed network model and data structures used by OCM, and gives high-level as well as elaborated details of the clustering process.

4.1 Network model and data structures

The system consists of a set of nodes communicating by exchanging messages over a wireless network. Each node is equipped with two wireless network interfaces A and B, with transmission ranges R and r, respectively, so that $R \gg r$. The nodes have unique identifiers and may move independently. Two nodes may communicate through an interface A (B) when the distance is less than or equal to R (r, respectively). Communication links of both types are unreliable, FIFO and mostly bidirectional. To cope with possible omissions, OCM employs local, unsynchronized clocks. Yet, the system as a whole is asynchronous.

We assume that the power consumed by B is lower than the power consumed by A. In other words, it is preferred to use interface B over A for communication whenever possible. Note that we do not make any assumptions on the technology type of A and B: they can be completely different (i.e., A is WiFi, while B is BT or TR100 low-power radio [3]) or can be identical (i.e., both A and B are WiFi's, operating at different predefined transmission ranges). At any given time, each interface may be active or turned off.

Each node p_i maintains up to three data structures, depending on the number of its active interfaces. They are: a list of the long range interface neighbors (for brevity, referred later as *long neighbors*), a list of the short range interface neighbors (referred later as *short neighbors*) and an intra-cluster routing table. As the name of the latter suggests, it is used for routing messages between nodes inside clusters created by OCM, and implemented as an adjacency matrix with a row and a column allocated for each node in the cluster to which the owner of the table belongs. Note that the total space required at each node is proportional to k and the maximal number of neighbors on each interface, but does not depend on the total number of nodes. This makes OCM very scalable for large systems.

4.2 High-level overview

The objective of the middleware protocol is to create an overlay consisting of nodes with both interfaces turned on. The overlay is required to be connected at the level of the long interface. Additionally, the protocol strives to minimize the value of the function $\sum_i \frac{1}{el(i)}$, where $el(i)$ is the energy level of a node i belonging to the overlay. The idea is to have an overlay, which is small in the number of nodes, and which consists of nodes

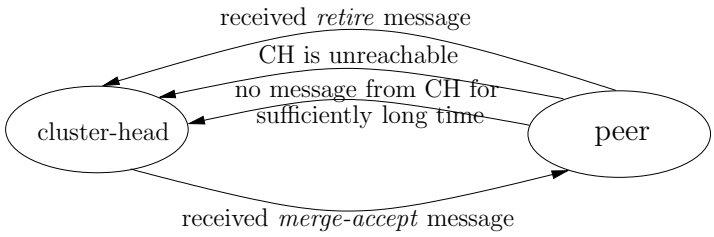

Fig. 2. State transitions in OCM. CH stands for the "cluster-head".

that have high remaining energy level. Another requirement from OCM is that all nodes which are not a part of the overlay are associated with some node in the overlay and are within k short range hops from it, where k is an arbitrary parameter of the protocol.

To achieve the above requirements, OCM employs the following scheme. A node p_i with an active long range interface periodically publishes information regarding its cluster to long neighbors. The neighbors check several conditions and decide whether they wish to merge with the cluster of p_i. If they decide positively, they send a corresponding message, and p_i chooses the best candidate p_j from those who agreed for the merge. Then p_i sends a message to that candidate, and if p_j is still ready for the merge, it becomes the head of the united cluster. The details are described below.

4.3 Clusters merging

During the run of the protocol, each node can be in one of two states: *cluster-head* or *peer*. A cluster-head is the node that has both interfaces enabled; this is the initial state of every node in the network. A node becomes a peer when it decides to turn its power-consuming long range interface off and associate itself with some cluster-head. A state transition diagram is shown in Figure 2.

A cluster-head p_j periodically broadcasts *merge-inquiry* messages (on its long interface), which include information on its energy level, current short and long neighbors and the intra-cluster routing table. When a cluster-head p_i receives such a message from p_j, it decides to respond with *merge-agree* if all four conditions hold: (1) p_i and p_j are connected by a path of short edges; (2) in the united routing table there is no path from any peer to p_i consisting of more than k short links; (3) the set of p_i's long neighbors contains a certain portion, β, of the long neighbors of p_j; (4) p_i's rank is higher than the rank of p_j. In our implementation, the ranking was based solely on the energy level of a node (with IDs used to break ties), although more sophisticated functions are possible (e.g., a ratio between the energy level and the number of nodes in the cluster).

Condition (3) is motivated by the request to create an overlay connected at the level of the long range interface. In order to ensure connectivity, one should set β to 1, i.e., p_i's set of long neighbors should include all long neighbors of p_j. This is a very strong limitation on the ability of clusters to merge, especially in dense networks. On the other hand, setting β close to 0 increases the chances of ending up with a disconnected overlay, especially in sparse networks. Although many heuristic approaches are possible,

including the ones that adapt dynamically to the sparsity of the network, in our implementation we simply use the value of 0.25. We show empirically that this value suffices to achieve a connected overlay with relatively large clusters (see Section 6).

After sending the *merge-inquiry* message, node p_j waits T_{mi} seconds (in our implementation, $T_{mi} = 1$) for *merge-agree* messages from its neighbors, which include information on their energy level. Then p_j selects the cluster-head with the highest energy level, p_k, as a candidate for the merge and sends a *merge-request* message to p_k. Node p_k responds with *merge-accept* if it does not wait for such a message from another cluster-head. Finally, if p_j receives the *merge-accept* message within T_{mr} seconds (in our implementation, $T_{mr} = 1$), it broadcasts a *new-cluster-head* message to all peers in its cluster, which notifies them on the change of their cluster-head, and switches the long range interface off to become a peer.

A node in the peer state remains with the long range interface turned off until one of the following three events occurs (cf. Figure 2): (1) it receives a *retire* message from its cluster-head, sent by the cluster-head when its energy level reduces below a predefined threshold (see details below); (2) it does not receive any message from its cluster-head for sufficiently long period of time; (3) it receives a notification on a failure of its adjacent short link, which makes the cluster-head unreachable by a path of short links between nodes in the cluster.

In order to implement the last event, as well as to keep the overlay structure current with the temporal network topology, the OCM protocol utilizes the common heartbeat technique. The technique is implemented in the corresponding heartbeat module discussed in Section 3. Note that OCM employs the heartbeat module on every active interface, i.e., a cluster-head runs two such modules, while a peer only issues heartbeats on a single interface.

To extend the lifetime of the network and distribute the load of the cluster-head duty on all nodes, OCM employs the following simple load balancing approach. The cluster-head p_i records its current energy level at the first time some peer joins the cluster. Node p_i continues to serve as the cluster-head for its peers until its energy level reduces below a predefined threshold compared to the recorded value. When it happens, p_i broadcasts a *retire* message to all its peers, indicating that they need to transit into the cluster-head mode. Hence, they start running the merge procedure. Since p_i wasted much more energy than those in the peer state and since the ranking of nodes relates directly to their remaining energy, p_i is unlikely to serve as a cluster-head for too long.

The detailed pseudo-code for the merge procedure is presented in Algorithm 1. Besides the el variable used to specify the energy level of each node, the pseudo-code uses RT, LN and SN variables to specify the intra-cluster routing table and the set of long and short neighbors, respectively.

5 Integration of OCM with routing

Although routing is purely a networking issue not related to the core operation of the OCM middleware, for the clarity of the whole picture we provide more details on the integration of OCM with the connector module, presented in Section 3. When the connector receives a packet to route, it looks-up the destination of the packet in the routing

Algorithm 1 Merge procedure, executed periodically by cluster-heads; code for node i

broadcast *merge-inquiry*(el_i, RT_i, LN_i, SN_i) to all long neighbors
wait for *merge-agree*(el_j) messages from all $j \in LN$ for T_{mi} seconds
$chosen \leftarrow$ choose j with *merge-agree*(el_j) s.t. el_j is maximal
if $chosen \neq nil$ **then** // there was some j that can be joined
    send *merge-request*(RT_i) to $chosen$
    wait for *merge-accept* from $chosen$ for T_{mr} seconds
    if received *merge-accept* **then**
        broadcast *new-cluster-head*($chosen$) to all peers in the cluster
        switch to *peer* state // and switch long interface off

when *merge-inquiry*(el_j, RT_j, LN_j, SN_j) is received from j
if $\langle el_j, id_j \rangle < \langle el_i, id_i \rangle \wedge$
  there is a path of short edges between i and $j \wedge$
  max distance between any peer and i in the joined $RT \leq k$ short edges $\wedge$
  $|LN_i \cap LN_j|/|LN_j| \geq \beta$ **then**
    send *merge-agree*(el_i) to j

when *merge-request*(RT_j) is received from j
if not waiting for *merge-accept* **then**
    send *merge-accept* to j
    update RT_i with RT_j
    if inside merge procedure **then**
        stop merge procedure

table provided by OCM. If the destination is in the routing table, the packet is given to the table-driven routing mechanism to be sent to its destination. If not, but the source of the packet is not a cluster-head, the table-driven routing is used again, this time to route the packet to the cluster-head of the source. Otherwise, the packet is given to the standard, off the shelf, ad-hoc routing mechanism, slightly modified as described below.

Consequently, the routing between nodes inside a cluster is managed solely by OCM, based on the intra-cluster routing table maintained at every node as part of the management module. A reactive ad-hoc routing protocol is not used for intra-cluster routing since the clusters are expected to be relatively small and dynamic, thus the overhead created by routing protocols on routes discovery and/or maintenance would not be cost-effective. A cluster-head is responsible for updating the routing table every time it absorbs another cluster (during the process of clusters merging) or when it gets a notification on failed links inside the cluster. Such a notification may arrive from the heartbeat module after a failure of an adjacent link or from some peer. The routing table is broadcasted periodically to all peers in the cluster. If a peer does not receive the update for sufficiently long time, it assumes that its cluster-head is unreachable and therefore transits into the cluster-head state, switching its long range interface on (cf. Figure 2).

As indicated above, inter-cluster routing is performed through an off the shelf ad-hoc routing protocol, which is run solely by cluster-heads. In principle, the OCM pro-

tocol can utilize any ad-hoc routing algorithm. The only required modification is that an instance of the routing protocol running on a cluster-head should act as a proxy for peers in its cluster. In practice, this means that when the ad-hoc routing algorithm running on a cluster-head inspects a message addressed to another node, it should query OCM whether the destination node is a peer in the cluster. If this is the case, the routing protocol should generate a response as expected from the actual destination. In our experiments, we evaluate OCM integrated with the well-known DSR [10] algorithm, modified according to the above.

Note that OCM decides to switch nodes to a peer state independently of the routing protocol. Thus, if a node is actively participating in routing, such transition may cause message losses, especially when the node serves as a cluster-head for a large cluster. In practise, however, the DSR algorithm succeeded to rediscover new routes and adapt to a new topology very fast, keeping the failure rate below 1% in most simulated scenarios. A possible optimization that may further reduce the failure rate is to inform the routing protocol about the planned change in the state of the node. Additionally, a cluster-head that merges its cluster into another cluster may update the new cluster-head regarding active routes.

6 Performance evaluation

We evaluate the performance of OCM in the Java-based SWANS simulator [6]. We would like to emphasize that the code used in the simulations is the full Java implementation of the protocol, including all modules specified in Section 3. To enable our evaluation, we have made two extensions to the simulator: monitoring the energy level of nodes and supporting two radio interfaces on the same device. The first extension is implemented by collecting information on the times each interface is in sending/receiving/idle/sleeping modes. The second extension required to synchronize the location of two radios and address interference issue. For more technical details, refer to Appendix A.

We assume a simulation area of size $500\text{x}500m^2$ with nodes placed at uniformly random locations. The number of nodes varies from 100 to 500 or from 100 to 1000 in steps of 100. The length of each simulation is 1000 (real time) seconds, where the first 100 seconds are considered as a warmup time and measurements are taken during the last 900 seconds. Each reported data point is produced by taking an average over 10 experiments.

In our experiments, each node is equipped with two types of radios: (1) WiFi with a transmission range of 100m and bandwidth of 11Mbps; (2) BT with a transmission range of 10m and bandwidth of 1Mbps. The numbers are taken according to the nominal transmission ranges and bandwidths of WiFi and BT technologies, as reported in [8]. We assume (and extend the simulator accordingly) that transmissions emitted by these radios do not interfere. This is in accordance with findings of several works that propose techniques to significantly mitigate the interference of BT and WiFi [5, 18].

Due to lack of support for a true BT MAC layer in the simulator, both radios use the same 802.11 MAC layer. We also do not explore the specific limitations existing in BT's network configuration, where the network is composed from *piconet* units, which

Common		
Signal Propagation model (PathLoss)	Free-space model	
Signal Interference model	Independent noise model	
Fading	None	
Radio-specific		
	WiFi	**BT**
Transmission range	100m	10m
Bandwidth	11Mbps	1Mbps
Power consumption:		
Transmitting	1346mW	81mW
Receiving	900mW	81mW
Idle	740mW	5.8mW
Sleeping	47mW	N/A
Simulation Scenarios		
Field size	500x500 m^2	
Message length	256 bytes + IP + MAC + PHY headers	
Nodes	100 to 500 or 100 to 1000 in steps of 100	
Mobility	1. Static 2. Random way-point with pause time of 60s 3. Random way-point with pause time of 180s 4. Two-phase with hot-spots, with pivot of 1 5. Two-phase with hot-spots, with pivot of 0.75	
Java pseudo random number generator, initialized with the current time in millis as seed		

Table 1. The summary of simulation parameters. The power consumed by BT in the sleeping mode is not stated since we do not use this mode in our simulations.

are built from up to eight devices working together [8]. (Nevertheless, due to the short range of BT, the simulated networks are not dense enough at the BT level. Hence, this does not pose a real limitation). All these issues are left for the future work. We note that we are not aware of any simulator providing a precise implementation of both BT and WiFi MAC layers.

The performance metrics we consider are the average latency of the sent messages and the average energy consumed by nodes during the simulation. Specifically, the consumed energy reflects the instantaneous consumed power multiplied by the duration of consumption. Simulation traffic was generated by choosing random source and destination nodes every second and sending a packet of 256 bytes. The latency is calculated only for packets received at the selected destinations. Unless otherwise specified, the failure rate of transmissions, i.e., the ratio of packets not received by the selected destination nodes, was below 1%. The simulation parameters are summarized in Table 1.

6.1 Energy model

The power consumption model of the WiFi interface is based on the measurements reported by Feeney and Nilsson [7] for a 802.11 wireless card operating at 11Mbps. They report costs of 1346mW for transmission, 900mW for receiving, 740mW for idle state and 47mW for sleeping. The numbers are similar to those reported by several other works for similar cards, e.g., [4, 20]. As for the BT interface, we use the measurements reported by Pering et. al. [15] for BlueCore3 BT radio operating at 1Mbps: 81mW

for transmission and 5.8mW for idle state. This work does not report on the power consumed during receiving, thus we conservatively assume that receiving requires as much as transmitting, i.e., 81mW.

These values for instantaneous power are summarized in Table 1. Note that in the simulation, we are not really turning off the WiFi radio, but put it into sleep mode. Although it still incurs a small cost in power (which is negligible compared to other WiFi modes), it allows to ignore the time required to switch from sleeping to active mode: as specified in [2], this time is insignificant.

To ensure that our measurements are fair, we conservatively assume that when the OCM middleware is disabled, nodes operate with BT radio turned off. Thus, in the following figures, when OCM is disabled, we report only on energy consumed by WiFi. Hence, the actual benefit from OCM might be even slightly better than reported if this is not the case. Note that our consumed energy measurements account for all messages transmitted, including messages generated by OCM and/or DSR as well as traffic generated for the purpose of the simulation.

6.2 Mobility models

We evaluate the performance of OCM under several mobility models. The first one is a static setting where nodes remain in their initial random locations during the entire simulation. The second model is the well-known random way-point model [10], where nodes alternate between pausing and moving to a randomly chosen position at a fixed speed. We consider speeds selected randomly from the range between 1 and $2m/s$ (thereby avoiding some of the pitfalls of the random way-point model), corresponding to walking speeds, and two pause times: 60 and 180 seconds. Note that the first (static) model can be considered as a private case of the second model with the pause time set to the length of the simulation.

In order to better capture the environment of a campus or a school, which appears to be an attractive domain for OCM implementation, we propose a new mobility model. In this new model, two sets of special locations, or *hot-spots*, are preselected and nodes alternate between the following two phases: In the first phase, the nodes are static and assigned random locations near one of the hot-spots from the first set, corresponding to classrooms. In the second phase, the nodes move to a new position, which is selected randomly (with low probability) or near one of the hot-spots from the second set (with high probability), corresponding to a gathering place in the campus, such as a cafeteria or a library. When a node reaches a position near the hot-spot, it prefers to remain nearby (i.e., move to a close location) with high probability, move to another hot-spot with lower probability, or choose a completely random location with low probability. The movement is done with speeds selected randomly from the range between 1 and $2m/s$ (walking speed).

We refer to the model presented above as a *two-phase model with hot-spots*. In addition to the already mentioned parameters, such as probabilities and speeds, the model is tuned by the number of iterations during the simulation, where each iteration consists of two phases, and a pivot controlling the ratio between duration of the first phase and duration of the whole iteration. We set the number of iterations to 3: this is a small value, meaning relatively long phases, yet larger than 1, allowing to capture the

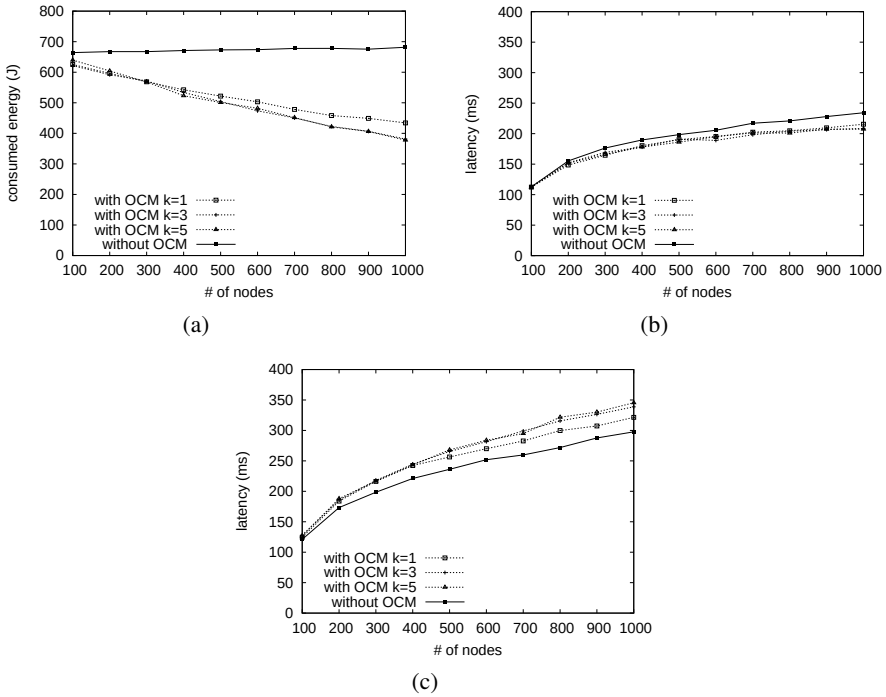

Fig. 3. Comparison of energy consumed by wireless communication (a) and latency ((b) and (c)) vs. the number of nodes in the static system, when OCM is run with various values of k and when OCM is disabled. The traffic in (a) and (b) consists of 256B messages and in (c) of 10KB messages.

effect of transitions between phases. We experiment with two pivots: 1, which means that the second phase is empty, and 0.75, which means the first phase is three times longer than the second, corresponding to an alternation between classes and breaks.

To summarize, we have experimented with five mobility models: static, random way-point with pause time of 60 and 180 seconds and the two-phase model with hot-spots with pivots of 1 and 0.75.

6.3 The impact of k on the performance of OCM

The first set of results presents the impact of the value of k, the maximal number of short range hops between a peer and its cluster-head, on the performance of OCM. In order to exploit the potential of larger clusters, nodes should remain close enough (i.e., in the transmission range of BT of up to 10 meters one from another) and stay that way as long as possible. Thus, we first evaluate the impact of k in the static system.

Figure 3(a) presents the measurements of the energy consumed by the wireless communication as a function of the number of nodes in the system. The average energy consumed without OCM is almost stable, increasing only slightly with the number of

nodes. The increase in the denser network occurs because of the *overhearing* effect observed in [17], due to which more nodes receive transmissions not intended for them. On the other hand, the power consumed when OCM is enabled reduces linearly with the number of nodes, since more effective clustering can be created, and thus more nodes may turn their WiFi radios off. The gain in power produced with $k = 1$ is lower than with larger k's, since when OCM is restricted to create small clusters, the result is larger overlays. There is no observable difference, however, between $k = 3$ and $k = 5$. This can be explained by the fact that even with 1000 nodes, the system is not dense enough at the BT level to allow for the creation of very large clusters. Thus, under the densities we experimented with in the static setting, further increase in k does not provide OCM with an opportunity to reduce the size of the overlay and save more energy.

Figure 3(b) compares the latency of transmitted messages. Here, surprisingly, we can see that although OCM uses slower BT hops, it performs better than communication without OCM. In addition, the latency exposed by OCM is insensitive to the value of k, even though larger k means that a message may traverse larger number of BT hops. The explanation to these phenomena is hidden in the relatively small size of generated messages (256 bytes), which cancels the superiority of faster WiFi links and, on the other hand, emphasizes the advantage of the hierarchical routing: the overhead of route discovery is eliminated when both source and destination belong to the same cluster or when the route is already cached by a cluster-head.

In order to validate this claim, we have measured the latency (and consumed energy) with traffic produced by larger messages of 10 kilobytes. The latency results, shown in Figure 3(c), fully comply with our hypothesis (the measurements of consumed energy exhibit similar behavior to the one shown in Figure 3(a) and thus are omitted). With larger messages, the higher bandwidth of WiFi plays a significant role. Thus, the latency produced with OCM is higher, and it increases for $k > 1$. The difference between $k = 3$ and $k = 5$ is almost not observable, due to the reasons explained above in the comparison of consumed energy.

To summarize, it seems that for the densities we experiment with, OCM achieves the best trade-off between consumed energy and latency with $k = 3$. Thus, the rest of the experiments are conducted with this value for k.

6.4 The effect of mobility on the performance of OCM

Figure 4 presents the behavior of the network under the random way-point mobility model with two values for pause times: 60 and 180 seconds. The power savings produced by OCM are expected to be less than in the static setting, since when nodes move, BT links are very unstable due to their relatively short range. As a result, according to Figure 2, peers transit frequently to the cluster-head state. This intuition is confirmed by Figure 4(a), which compares the energy consumed by nodes: OCM succeeds to save less energy than in the static setting and the savings are higher with longer pause time.

Figure 4(b) compares the latencies exhibited by the network. For the longer pause time, the latency does not change much due to the use of OCM; when the pause time is short, however, OCM introduces more considerable increase in latency. This happens, again, due to frequent changes in BT links, which break routes discovered by the DSR routing protocol. As explained in Section 5, a cluster-head responds to route discovery

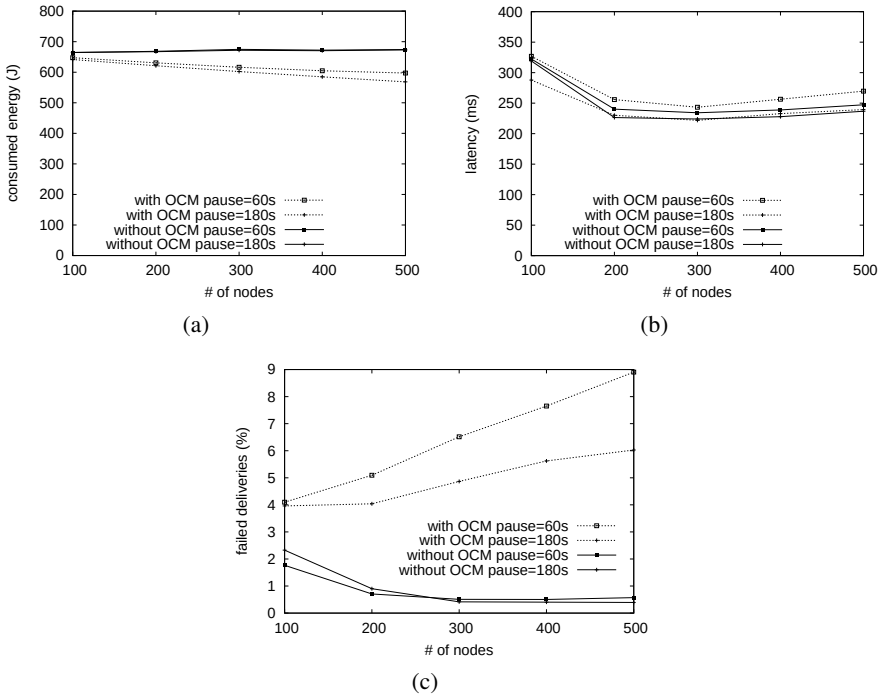

Fig. 4. Comparison of energy consumed by wireless communication (a), latency (b) and percentage of failed deliveries (c) vs. the number of nodes in the system. The nodes move according to the random way-point model with a pause time of $60s$ and $180s$.

messages intended for its peers. In the time interval until the actual message arrives, its peer may move far enough to break the path of BT links between them, thus the cluster-head would not be able to forward the message and would generate a route error. As a result, the sender of the message would retry to discover another route and then resend the message with a new route.

The process of route rediscovery may occur several times, until the routing protocol gives up and drops the message. As Figure 4(c) presents, this happens too often with OCM, especially when the pause time is short. Another source for failed deliveries, originating from the same issue of failing BT links, is intra-cluster routing. When a message is originated at a peer and destined to some node outside the cluster, it is relayed first to the cluster-head of the peer using the intra-cluster routing table (same for a message received by a cluster-head, but intended for its peer). Such relay is done by table-driven routing mechanism using point-to-point transmissions along the shortest path consisting of BT links. A failure of even one link along that path may cause to the omission of the message.

In order to cope with the problem of such intra-cluster omissions, we propose the following recovery mechanism: whenever a notification on a failed point-to-point transmission is received from the BT MAC layer, the connector module notifies OCM that

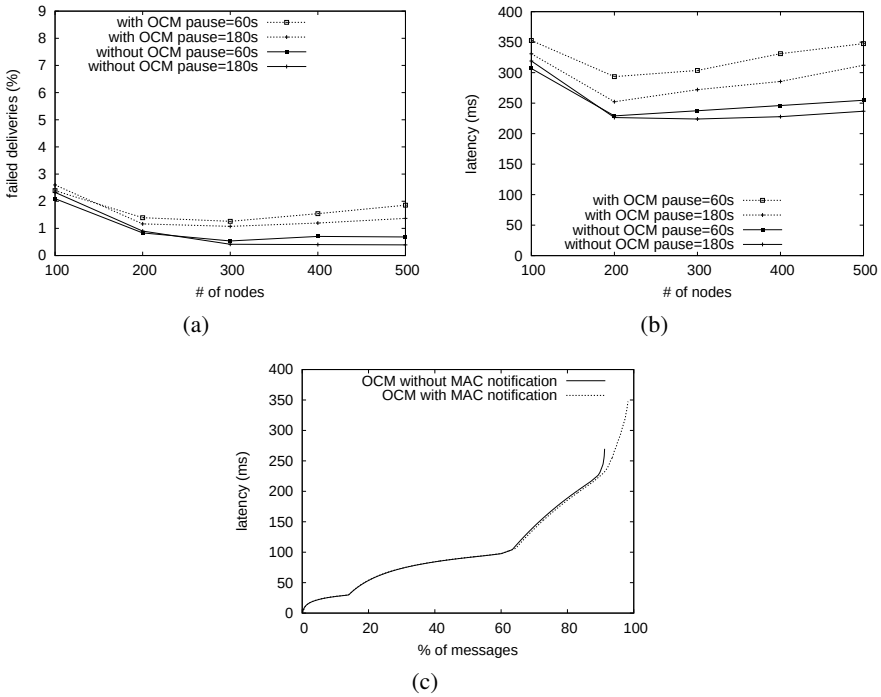

Fig. 5. Comparison of percentage of failed deliveries (a) and latency (b) when the MAC notification mechanism is enabled. Figure (c) compares average latency vs. the percentage of delivered messages sorted by latency (i.e., data point (x, y) shows the average latency (y) for $x\%$ of the fastest messages) in the system of 500 nodes with and without the MAC notification mechanism. The nodes move according to the random way-point model with a pause time of $60s$ and $180s$ in (a) and (b), and $60s$ in (c).

considers the link as lost (which may cause the peer to switch into the cluster-head state) and tries to resend the message. The retransmission may occur on another BT link or on a WiFi link, enabling the DSR routing with its own retrying mechanism mentioned above. Note that such a notification can be generated by any implementation of the MAC layer which provides reliable point-to-point transmissions and, in particular, by 802.11 MAC.

Figure 5 studies the impact of the recovery mechanism on the performance of OCM. As can be seen in Figure 5(a), the reliability of the protocol increases sharply: for example, for the short pause time and 500 nodes, the percentage of failed deliveries drops from 9% to less than 2%. The average latency, on the other hand, increases from 269 to 347 milliseconds (Figure 5(b)). This increase is introduced by messages that now succeed to reach their target using the recovery mechanism, but suffer from a delay caused by the first unsuccessful transmission on the BT link.

In order to validate that only recovered messages are delayed by the introduction of the recovery mechanism, we compile a graph showing the average latency of $x\%$

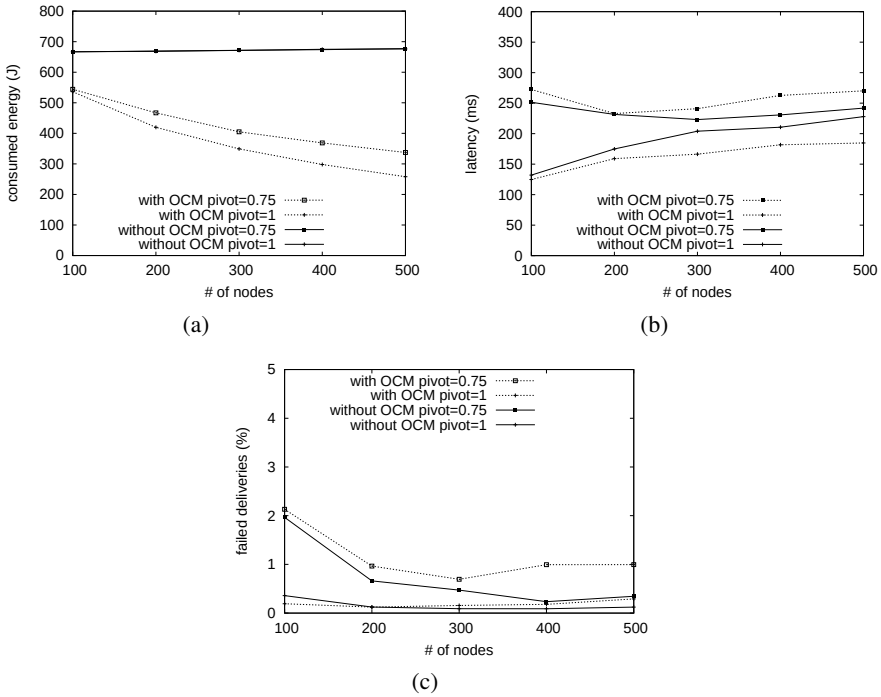

Fig. 6. Comparison of energy consumed by wireless communication (a), latency (b) and percentage of failed deliveries (c) vs. the number of nodes in the system. The nodes move according to the two-phase mobility model with hot-spots, with pivots of 0.75 and 1.

shortest deliveries in the setting of 500 nodes and the pause time of 60 seconds vs. x (Figure 5(c)). Here we can see that in both scenarios, i.e., with and without the recovery mechanism, approximately 90% of messages experience the similar average delay, confirming that the increase in the total latency occurs due to recovered messages. As a conclusion, the recovery mechanism appears to be a successful tool in coping with the reliability issue of OCM in mobile setting, while the latency of messages not requiring this tool remains unchanged.

6.5 The performance of OCM under two-phase mobility with hot-spots

The performance of OCM under the two-phase mobility model with hot-spots is shown in Figure 6. Due to similarity of phenomena to the ones discussed in the previous section, we present only the measurements performed with the recovery mechanism detailed above. The energy savings produced by OCM are impressive: in the static setting, i.e., when the pivot is set to 1, only 38.2% of the energy is consumed when OCM is enabled in the system of 500 nodes compared to the system without OCM (Figure 6(a)). When nodes move, i.e., when the pivot is set to 0.75, less than half of the energy is consumed. Such encouraging results are explained by the characteristics of the mobil-

ity model, where nodes concentrate around hot-spots, providing OCM with a potential to create smaller overlays.

The latency exhibits similar behavior to the ones measured under the static and random way-point models (Figure 6(b)). With the pivot set to 1, OCM achieves much better latency due to the hierarchical routing employed, while with the pivot set to 0.75, the latency of OCM is higher due to the recovery mechanism used (without it, OCM performs better even though nodes move). The percentage of failed messages is slightly higher for OCM, but still kept under 1% in most scenarios (Figure 6(c)).

7 Conclusions

This paper presents OCM, an efficient middleware that reduces power consumption in mobile ad-hoc networks composed of devices with multiple communication interfaces. OCM does not assume any specific communication technology; it requires only one interface with a larger transmission range and power consumption than the other. Moreover, it way be applied in addition to other power-saving techniques, such as PSM of WiFi [2].

OCM constructs an overlay of nodes connected at the level of long range interface, having the rest of nodes with their long range interface turned off and connected to some node in the overlay through an adjustably short path of short range links. The nodes are selected into the overlay based on their remaining energy level and the number of neighbors they can communicate with on both interfaces. OCM adapts quickly to topology changes by using local timers and turning long range interfaces on when a node suspects that it has lost a connection with its associative node in the overlay. Thus, wrong detection of link failures may hurt only the performance, but not the correctness of the protocol.

We evaluate the performance of OCM with typical parameters of BT and WiFi cards and show that the power savings produced by OCM are significant, while the latency and message loss are almost the same as with the standard ad-hoc routing algorithm. It is notable that in the static setting, OCM even achieves considerably better latency. In addition, OCM adapts well to the density of the network, exhibiting the behavior of "add more to improve service", similar to [20] and opposite to [4]. This feature is particularly important as the number of devices equipped with several communication interfaces continuously increases.

A possible direction for future research would be to evaluate the middleware with other technologies for wireless communication, e.g., WiMax and ZigBee, in addition to WiFi and BT or instead of them. The middleware can be easily extended to manage more than two wireless interfaces. For example, given a network consisting of devices that have also WiMax radio, OCM would construct an overlay with nodes having all radios active; this overlay would be connected at the level of WiMax. Another sub-overlay would be created out of nodes having WiMax turned off; every node in the sub-overlay would have a path of up to k WiFi links to some node in the overlay. Finally, the rest of nodes in the network would have both WiMax and WiFi turned off and have a path of up to k BT links to some node in the sub-overlay.

Another issue that warrants investigation is dealing with capacity enhancement. Although OCM keeps the number of short links in each path below the predefined threshold and thus preserves a significant portion of the original capacity, it may not be enough for certain applications. A possible direction is to adapt the middleware to local communication requirements on each node. That is, switch a peer into the cluster-head state when it generates or receives traffic above some threshold.

References

1. M. Anand, E. B. Nightingale, and J. Flinn. Self-tuning wireless network power management. *Wireless Networking*, 11(4):451–469, 2005.
2. G. Anastasi, M. Conti, E. Gregori, and A. Passarella. 802.11 power-saving mode for mobile computing in Wi-Fi hotspots: limitations, enhancements and open issues. *Wireless Networks*, 14(6):745–768, 2008.
3. P. Bahl, A. Adya, J. Padhye, and A. Walman. Reconsidering wireless systems with multiple radios. *ACM SIGCOMM Comput. Commun. Rev.*, 34(5):39–46, 2004.
4. B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *ACM Wireless Networks Journal*, pages 85–96, 2001.
5. C. F. Chiasserini and R. R. Rao. Coexistence mechanisms for interference mitigation between IEEE 802.11 WLANs and Bluetooth. In *Proc. IEEE INFOCOM*, volume 2, pages 590–598, 2002.
6. Cornell University. JiST/SWANS – Java in Simulation Time / Scalable Wireless Ad Hoc Network Simulator. Available at `http://jist.ece.cornell.edu`.
7. L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. IEEE INFOCOM*, pages 1548–1557, 2001.
8. E. Ferro and F. Potorti. Bluetooth and Wi-Fi wireless protocols: a survey and a comparison. *IEEE Wireless Communications*, 12(1):12–26, 2005.
9. J. Gomez and A. Campbell. Variable-range transmission power control in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 6(1):87–99, 2007.
10. D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.
11. L. M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc. Power consumption in packet radio networks. *Theoretical Computer Science*, 243(1-2):289–305, 2000.
12. R. Krashinsky and H. Balakrishnan. Minimizing energy for wireless web access with bounded slowdown. *Wireless Networking*, 11(1-2):135–148, 2005.
13. L. Li, J. Y. Halpern, P. Bahl, Y.-M. Wang, and R. Wattenhofer. Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks. In *Proc. 20th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 264–273, 2001.
14. G. Pei, M. Gerla, and X. Hong. LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proc. IEEE/ACM MobiHOC*, pages 11–18, 2000.
15. T. Pering, Y. Agarwal, R. Gupta, and C. Power. Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *Proc. ACM MOBISYS*, pages 220–232, 2006.
16. M. Q. Rieck and S. Dhar. Hierarchical routing in ad hoc networks using k-dominating sets. *SIGMOBILE Mob. Comput. Commun. Rev.*, 12(3):45–57, 2008.
17. S. Singh and C. S. Raghavendra. PAMAS: Power aware multi-access protocol with signalling for ad hoc networks. *ACM Computer Communication Review*, 28:5–26, 1998.

18. M. Song, S. Shetty, and D. Gopalpet. Coexistence of IEEE 802.11b and Bluetooth: An integrated performance analysis. *Mobile Networks and Applications*, 12(5):450–459, 2007.
19. J. Wu, F. Dai, M. Gao, and I. Stojmenovic. On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. *J. Communications and Networks*, pages 59–70, 2002.
20. Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proc. 7th Int. Conf. on Mobile Computing and Networking (MOBICOM)*, pages 70–84, 2001.

A Extensions to SWANS

This section provides technical details on the extensions introduced into the SWANS simulator [6] in order to enable the experiments with OCM. The sources can be downloaded from `http://www.cs.technion.ac.il/~sakogan/SWANS`.

Consumed energy measurements: Our energy model implies that a radio may be in one of the four following states: sleeping, idle, receiving or transmitting. These are also the modes used by SWANS for its radios. Thus, we intercept an event of the mode change in a radio, recording the time of the change and the new mode. At the subsequent mode change, we calculate the length of the interval for the current mode and, again, record the time of the change and the new mode. Thus, we accumulate the times during which the radio was in each of the four modes. During the simulation and at its end, we multiply these values by the power consumed at each mode (cf. Section 6.1) to receive the amount of energy consumed by wireless communication sub-system at each node up to that point.

Supporting multi-radio devices: The SWANS simulator features a limited support for nodes with multiple radios, which is summarized by an ability to add more than one radio and MAC module to the same node. These radios, however, operate independently from one another, which means that a movement of one does not imply the movement of another. In our extension, we eliminated this shortcoming by synchronizing the location of both radios, i.e., under any mobility model, both radios are always moved to a new location simultaneously.

Another issue we addressed in our support for multiple radios is interference. As opposite to the previous point, the handling of the interference depends tightly on the types of radios as well as on the system model assumptions. As mentioned above, in the base version of SWANS radios behave independently, which results in interference of a transmission by one radio with any other transmission in the range, no matter what the types of transmitting radios are and even if both transmissions are emitted from the same device. Moreover, the transmissions on one interface can be received by the other interface (in fact, two interfaces on the same device always receive transmissions one of the other). Choosing to experiment with common wireless technologies, namely BT and WiFi, and following the findings of several papers [5, 18], we assumed that transmissions by different types of radios do not interfere. Thus, we extended the way a packet is transmitted on a simulation field by filtering packets sent by one radio type from being received by another radio type.