A Switch Agent for Wireless Sensor Nodes with Dual Interfaces: Implementation and Evaluation

Tao Zheng, Sridhar Radhakrishnan
School of Computer Science
University of Oklahoma
Norman, Oklahoma 73019–6151
Email: tao@ou.edu, sridhar@ou.edu

Venkatesh Sarangan
Computer Science Department
Oklahoma State University
Stillwater, Oklahoma 74078
Email: saranga@cs.okstate.edu

***Abstract*—Data generation in wireless sensor networks could be bursty as it is dictated by the presence or absence of events of interest that generate these data. While conventional sensor nodes possessed only one radio interface, next generation sensor nodes are expected to have two (possibly more) radio interfaces, each with different range, capacity, and power consumption. Equipping sensor nodes with dual radios has its own benefits and can be quite useful in handling bursty traffic while at the same time satisfying the application's delivery requirements. In this paper, we propose an adaptive interface switch agent that intelligently selects the interface to be used for data transmission at a sensor node based on the data burst length while taking into consideration power consumption, throughput, and end-to-end delay. The proposed work generalizes earlier works in this area to enable both the source nodes and intermediate data forwarding nodes to initiate the activation of high power radios so that they can be utilized to a higher degree for convergecast communication. We have performed extensive simulations with sensor nodes containing both IEEE 802.15.4 and IEEE 802.11 compatible radios. Our simulation results indicate that: (i) the end-to-end delay and throughput achieved by the proposed interface switch agent are comparable to those achieved in a network of sensor nodes equipped only with IEEE 802.11 radios, (ii) the energy consumed in the network using our interface switch agent is a fraction of that consumed in a network of the IEEE 802.11 sensor nodes and is comparable to that of sensors using only IEEE 802.15.4 radios.**

I. Introduction

Wireless sensor networks (WSNs) are multiple-hop ad-hoc networks consisting of a large number of sensor nodes communicating through wireless medium. The sensor nodes usually have limited amount of resources like memory, channel bandwidth, and battery capacity.

The study of wireless sensor network has become a hotspot in networking area due to its broad range of potential applications. Specifically, wireless sensor networks have been reported to be used in environmental monitoring, habitat monitoring [1], military surveillance, inventory tracking, smart buildings, homeland security, etc. Several of these applications are characterized by bursty traffic. That is, they have little or low data rate for most of the time and when an event occurs, the data rate increases drastically.

There have been a number of approaches to deal with the bursty traffic in wireless sensor networks with sensor nodes bearing a single interface. These approaches have been studied on medium access control (MAC) and routing layers. Adaptive MAC protocols such as SMAC [2], TMAC [3] and PMAC [4] adopt adaptive duty cycles based on the traffic load. The radio is woken up when the traffic load is heavy and is put into sleep when the traffic load is light. The protocols assume the physical layer has a large enough bandwidth to handle the peak traffic.

Pering [5] proposed the CoolSpots model which focuses on using Bluetooth and WiFi radios in a single hop mode. A few switching policies have been proposed to make the switching decisions. Although reference [5] claims that the CoolSpots model can be used for adhoc peer-to-peer configuration, it does not address the complexities of the routing layer that arise as a result of different transmission ranges of the two radios. Yarvis et al. [6] proposed a network architecture consisting of a set of high-bandwidth nodes and low bandwidth nodes. The low bandwidth nodes connect to the high-bandwidth nodes thereby reduce the total number of transmissions to reach the destination. This increases the lifetime of low-bandwidth nodes. The high bandwidth nodes are assumed to be connected to a power source and hence do not have any energy constraints.

A recent work by Stathopoulos et. al [7] introduces ideas similar to the ones introduced by us in this paper. They proposed a network architecture containing devices that have a single low-bandwidth radio and devices with both low-bandwidth and high-bandwidth radios. The low-bandwidth radios are always turned on and the high-bandwidth radios are always turned off. However, the high power radio at a specific node called the topology controller is always kept on. When a low-bandwidth node has data to send to a particular destination, it sends a path request to the topology controller. The topology controller selectively wakes up the high-bandwidth radios along the path from the source to the destination. The data travels along low-bandwidth nodes to high-bandwidth node, passes through a sequence of high-bandwidth nodes, and finally goes from the high-bandwidth node to the destination along low-bandwidth nodes.

While the work done in [7] is interesting, it has a few drawbacks. It uses a centralized controller (with no energy constraints) which partially negates the advantages and philosophy of a truly distributed sensor network. Also, only the

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7294
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7294

source nodes are allowed to make the decision on using the high-bandwidth nodes for transmission which could decrease the utility of the high power radios. Our proposed approach is completely decentralized in that each node makes a local decision to wake up the appropriate high-bandwidth nodes for transferring the data to the destination. We leverage on the existing distributed routing protocols such as AODV [8] to determine appropriate high-bandwidth paths to the destination. The use of protocols such as AODV also helps to find new paths when sensor nodes lose their battery power. Further, our proposed solution permits any node along a path (both source and intermediate) to activate the high power radios in downstream nodes which can be quite useful in convergecast data dissemination scenarios.

The main contributions of this paper are as follows: (i) For the network model wherein *all* the nodes are equipped with dual radios (one low power and one high power), we have presented a distributed framework for activating high power radios at appropriate nodes. The activation mechanism is sensitive to the energy efficiency of the radios and can adapt automatically to the network traffic pattern and the application's data delivery requirements. The proposed scheme is tolerant to node failures as well. (ii) We have also proposed supporting enhancements (such as schemes for maintaining the routing cache) that can result in additional energy savings at network nodes. (iii) We have provided extensive simulation results to test the performance of our proposed interface activation scheme using ns-2 which required us to design, implement, and test new protocol agents. Our simulation results indicate that: (i) the end-to-end delay and throughput achieved by the proposed distributed interface switch framework are comparable to those achieved in a network of sensor nodes equipped only with IEEE 802.11 radios, (ii) the energy consumed in the network using our interface switch framework is a fraction of that consumed in a network of the IEEE 802.11 sensor nodes and is quite comparable to that of sensors using only IEEE 802.15.4 radios.

This paper is organized as follows. In Section II, we describe the concept of the switch agent and describe its components, respectively. In section III, we give the protocol details of the switch agent, and Section IV shows the simulation results using IEEE 802.15.4 and IEEE 802.11 dual interfaces. We state our future work and conclude in section V.

II. Overview of the Switch Agent

We will call the interface with the lower bandwidth and shorter transmission range as Interface-I and the one with the higher bandwidth and longer transmission range as Interface-II. A switch agent is a software component that distributes traffic between these two interfaces. By default, Interface-II is powered-off and is woken up by sending appropriate control message along Interface-I. We will assume that Interface-I is always active (with appropriate duty cycle) and that the network is connected when all the nodes activate their Interface-I. A distributed protocol such as AODV is executed to populate the routing tables at each node. This routing table

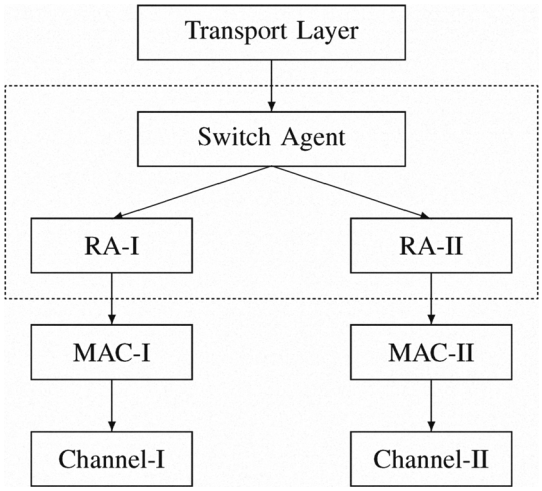


Fig. 1. Positioning of the switch agent

will provide next-hop information along different routes in the network based on Interface-I. Since the reachability of Interface-II is higher than Interface-I, at each sensor node we need to find a new set of routing table based on Interface-II. If the reachability of Interface-II is a multiple of the reachability of Interface-I, then with appropriate modification to the AODV protocol, it may be possible to approximate the routing table related to Interface-II from the Interface-I routing tables, thus saving energy at nodes. In general, this may not be possible and hence we are required to separately execute the routing protocols for each interface. Running the routing protocols on Interface-II will likely find a shorter higher-bandwidth path than constructing one from the Interface-I routing tables. This saves more energy for heavier traffic compared to the energy overhead incurred by broadcasting wake-up message.

The switch agent has to monitor the traffic flow and use the appropriate interface to send control information and data packets. In order to facilitate this, the switch agent is placed above the routing agents. The protocol stack is illustrated in Fig. 1, where RA-I and RA-II are the routing agents used by Interface-I and Interface-II, respectively.

A. Components of the Switch Agent

Every switch agent has the following components: interface queue monitor, sleep-wakeup unit, route cache unit, and timers.

1) Interface Queue Monitor: The duty of the switch agent is to switch Interface-II on and forward the traffic to Interface-II in order to meet the application demands and/or when the traffic rate at a node becomes high, necessitating the use of Interface-II. In order to make this happen, we need a component to monitor the packet transmission queue at every node. In this paper, we have chosen to monitor the length of the interface queue in between RA-I and MAC-I, since the queue length reflects the cumulative effect of both

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7294
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7294

incoming traffic rate and transmission rate. We will use a predefined threshold called THRESHOLD_HT. Whenever the queue length exceeds THRESHOLD_HT and if the Interface-II is on, then the incoming traffic is diverted to Interface-II. We have used the queue length to demonstrate how traffic is routed to the second interface, we could also have done so based on other application requirements such as desirable end-to-end delay.

2) Route Cache: On-demand routing agents like AODV, establish routes whenever there are data packets bound for a certain destination. We will use two route caches, one for each interface. We will denote these route caches (the routing tables) as RC-I and RC-II corresponding to Interface-I and Interface-II, respectively. When the network initially starts up, only Interface-I is on. If there is a data packet originated at some node, RA-I at that node will broadcast a route request. A route will be established after receiving a route reply from either the destination node or a node knowing how to reach the destination node. The routing agent like AODV can reestablish a route whenever a route is expired or repair a route when the nexthop neighbor is dead. All the Interface-IIs will be turned on for the initial switching since no cached routes yet. RA-II will do the same to establish a route and the route will be cached for the subsequent switching.

A RC-I entry is made to expire if messages along Interface-I are not delivered to the next-hop specified in that entry. In this case, the entry is purged and the routing agent at the node is activated to recalculate the route. An RC-II cache entry (corresponding to Interface-II) will expire if it has not routed any data packets on it for a period of time. Apart from caching all the route information to destinations at relating to Interface-I, the number of hops (with Interface-I) required to reach each destination can also be stored and updated from time to time. The caching of hop counts will help to make a ringcast instead of broadcast for sending wake-up control messages. Ringcast is a broadcast but restricting the TTL (Time-To-Live) of the broadcasted packet to a limited number of hops. This can save energy and improve the scalability of the protocol.

3) Sleep-Wakeup Unit: The Sleep-Wakeup unit at a node is responsible for (i) turning on Interface-II at the node and (ii) sending control messages along the node's Interface-I to turn on Interface-II at other nodes. If an entry for a particular destination has to be populated in RC-II, we have resort to a broadcast/ringcast to determine the path (made up of Interface-II) to the destination. These broadcast/ringcast control messages are sent by the sleep-wakeup unit along Interface-I. Let us consider a scenario when a RC-I entry is available and the RC-II entry has expired at node x for destination d. The cache RC-I at x will give us the next hop node (say y) on the path to d using Interface-I. Waking up node y's Interface-II may not be prudent since we can possibly bypass node y owing to the increased range of Interface-II at x. For this reason, we have to let node x send a broadcast/ringcast message to wake up Interface-II on all the nodes that receive the message. After this, routing protocol is run on Interface-II to create an entry for d in RC-II after which x starts forwarding data along this path. The nodes that turned on their Interface-II will turn it off in case they do not receive any data for forwarding along Interface-II for a period of time.

Now consider the scenario that both RC-I and RC-II entries are available at node x for destination d. Having a RC-II entry for d does not imply that the route is active, since the intermediate nodes along that path could have turned off their Interface-II due to inactivity. Let z be the next hop neighbor of x on RC-II towards d. Further let y' be the next hop node in RC-I to the destination z. Now a unicast wake-up control message is sent to node y' using Interface-I at node x with the destination as node z.

A wake up registry is maintained at node x to indicate that node z is woken up to route data packets to destination d. This will avoid unnecessary transmission of wake-up messages. The entries in the registry are purged from time to time.

4) Timers: As part of our protocol we will maintain three timers: IDLE_TIMER, CACHE_TIMER and REGISTRY_TIMER. The IDLE_TIMER keeps track of the traffic that is seen by Interface-II. In the absence of any traffic for a duration of time, defined by IDLE_INTERVAL, Interface-II will be turned off. CACHE_TIMER is fired periodically to purge old paths and determine new ones. The REGISTRY_TIMER is maintained to purge the entries in the registry. Each registry entry will be of the form $[d, t]$, where d is the destination and t is the timestamp indicating the latest time the node has seen a data traffic to destination d through its Interface-II. When the REGISTRY_TIMER is fired, all the entries that satisfy the relation $curTime - t >$ REG_TTL will be removed, where $curTime$ is the current wall clock time and REG_TTL is a predefined threshold.

III. Protocol Details

As stated earlier, the switch agent sits at the routing layer and on top of two routing agents, each of which is for two different interfaces. In our distributed protocol nodes receive three types of packets:

- wake-up packets - control packets generated from the switch agent to wake up the Interface-II. Wake-up message could be either a unicast message or a broadcast/ringcast message. All control packets communicate through the Interface-I;
- routing packets - control packets generated by the routing agents to establish or update a route;
- data packets - data packets originated from the sensors.

In each of the intermediate nodes along a route, every packet must go through all the protocol layers until the switch agent. When the data packet reaches its destination node, it will be passed to the upper layers above the routing layer.

We tag all packet headers with a channel ID. The data packet generated by the application or the control packet generated by the switch agent at the node of origin will have a channel ID 0. The switch agent, after determining the channel to use, will use an ID 1 and 2 for the packets going through Interface-I and Interface-II, respectively. We will describe the protocol details based on the types of the packet the switch agent receives.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7294
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7294

A. Receiving a Routing Control Packet

When the switch agent receives a routing control packet, it will first check the header of the packet. If the header is tagged with a channel ID 1 (resp. 2), the routing packet will be forwarded to RA-I (resp. RA-II).

B. Receiving a Wake-up Packet

A wake-up packet sent by a node could be either a broadcast/ringcast message or a unicast depending upon the availability of caching information.

1) Receiving a broadcast/ringcast message: If the switch agent receives a broadcast/ringcast message, it will drop the packet when TTL is 0. If the TTL is greater than zero, the node's sequence number will be used to eliminate broadcasting of the old packet. In response to the new wake-up packet, Interface-II will be started. If this particular node does not observe data flowing through its interface for a period of time IDLE_INTERVAL after it has been switched on, then its Interface-II will be put into sleep. An IDLE_TIMER will be used to countdown this interval of time.
2) Receiving a unicast message: If the switch agent receives a unicast wake-up message through Interface-I and it is the next-hop identified, then the Interface-II will be switched on and the IDLE_TIMER will be rescheduled. Now it will determine if a unicast or broadcast/ringcast message has to be sent to wake up the nodes along the path to the destination. This decision is based on information available at the cache RC-II. In addition, before sending a unicast message is the wakeup registry searched to see if the next hop node is already active on Interface-II. If it is active on Interface-II, then the wakeup message along Interface-I is avoided and the data packets are sent via Interface-II to the next hop node directly.
   If the node receiving the wake-up message is not the next-hop that is identified, then the message will propagate towards the next-hop node through Interface-I. If the current node is the destination node of the original data flow, the message will be dropped and no further unicast of the message will be sent.

C. Receiving a Data Packet

The data packets received by a switch agent can be either a data packet tagged with channel ID 1 (pass-through traffic through Interface-I) or a data packet tagged with ID 0, which is a data packet originated from the current node. If there is no registry entry for the destination of the data packet, the switch agent will forward it to RA-1 and the packet will be tagged with channel ID 1. If the interface queue length exceeds the predefined threshold after receiving the packet, the switch agent will turn on the Interface-II if it is not currently on. The interface queue will be scanned and the destination with the maximum number of packets will be found. A registry entry will be added for that destination.

```
recvWakeup(pkt){
if pkt is a broadcast message then
   ttl ← the Time-To-Live of pkt
   if ttl < 0 then
      drop pkt
   else
      rq_src ← source where pkt is originated
      rq_bid ← broadcast id given by the source
      fresh ← bid_lookup(rq_src, rq_bid)
      if fresh = FALSE then
         drop pkt // obsolete wake-up message
      else
         bid_insert(rq_src, rq_bid) // store the latest broadcast id
         if registry is empty then
            resched_idletimer(IDLE_INTERVAL)
         end if
         wake up Interface-II
         ttl-- // decrement ttl by 1
         broadcast pkt
      end if
   end if
else if pkt is a unicast message then
   rq_dst ← destination which the wake-up request is bound to
   ip_dst ← destination address of pkt
   if registry is empty then
      resched_idletimer(IDLE_INTERVAL)
   end if
   if rq_dst = address of the receiving node then
      wake up Interface-II
      drop pkt // no need to propagate the wake-up message any more
   else if ip_dst = address of the receiving node then
      wake up Interface-II
      if there exists a cached route to rq_dst then
         nexthop ← cache_lookup(rq_dst)
         call sendWakeup(rq_dst, nexthop)
      else
         call bcastWakeup( )
      end if
      drop pkt
   else
      continue to propagate pkt
   end if
end if
}

fire_idletimer( ){
put Interface-II into sleep
}
```

Fig. 2. Pseudocode of the switch agent

A broadcast/ringcast or a unicast wake-up message need to be sent to wake up downstream nodes to that destination. First look up the route cache to see if any route has been cached for the destination. If the answer is yes, construct a unicast wake-up packet and set its destination to be the next hop of the cached route and send the wake-up packet through Interface-I. If the answer is no, construct a broadcast wake-up packet and broadcast it through Interface-I.

The data packets received by a switch agent could also be tagged with ID 2. That indicates the Interface-II must be awake already. Otherwise, no packet tagged with ID 2 can be received. The packet will be sent back down through the Interface-II to the next hop. A registry entry will be added for the destination of the packet, if it does not exist yet. The expiration time of the entry will be REG_TTL. If such an

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7294
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7294

```
RECEIVE(pkt) {
// PT_RT: routing message from the routing agents
// PT_WK: wake-up message from the switch agent
// PT_DATA: data packets from the application
// RA-I: routing agent 1; RA-II: routing agent 2
// q: the interface queue between RA-I and MAC-I
ptype ← packet type of pkt
cid ← channel ID tagged in the header of pkt
dst ← destination address of pkt
if ptype = PT_RT then
   if cid = 1 then
      hand pkt over to RA-I
   end if
   if cid = 2 then
      hand pkt over to RA-II
   end if
else if ptype = PT_WK then
   call recvWakeup(pkt)
else
   found ← registry_lookup(dst)
   if found = TRUE then
      setRegistryExpireTime(dst, REG_TTL)
      call sendThroughChannel2(pkt)
      return
   else if cid = 2 then
      cancel_idletimer( ) // cancel the idle_timer if it is active
      registry_insert(dst) // insert a registry entry for dst
      setRegistryExpireTime(dst, REG_TTL)
      call sendThroughChannel2(pkt)
      return
   else
      tag the header of pkt with cid = 1
      send pkt down through Interface-I
      if q.length >= THRESHOLD_HT then
         if Interface-II is sleep then
            wake up Interface-II
         end if
         if registry table is empty then
            cancel_idletimer( ) // cancel the idle_timer if it is active
            dst_max ← findMax(q)
            found ← registry_lookup(dst_max)
            if found = TRUE then
               call wakeupChannel2()
               registry_insert(dst_max) // add a registry entry for dst_max
               setRegistryExpireTime(dst, REG_TTL)
            end if
         end if
      end if
   end if
end if}
```

Fig. 3. Pseudocode of the switch agent

entry already exists it will be re-timestamped.

Some MAC protocols (like IEEE 802.11 and IEEE 802.15.4) and the routing agents (like AODV) support callback functions in case of transmission errors. Those callback functions can be leveraged to make sure the wake-up message can reach the downstream nodes. If any callback occurs on the unicast wake-up message due to collisions or poor link quality, an alternative path might be used or a broadcast/ringcast wake-up message will be sent out instead.

The pseudocode is listed in Fig. 2 and Fig. 3.

IV. Performance Evaluation

A. Simulation Setup

We have carried out the simulations based on the ns-2 [9] implementation of IEEE 802.15.4 PHY/MAC [10] (the interface with a lower data rate, shorter-range and lower power consumption) and IEEE 802.11 PHY/MAC [11] (the interface with a higher data rate and power consumption). The IEEE 802.11 MAC in ns-2 only implements DCF function, without the complexity of association.

IEEE 802.15.4 standard has been developed to address the unique needs of low cost and low power of wireless sensor networks. IEEE 802.15.4 and IEEE 802.11 both operate within the 2.4G ISM band. Interferences can occur when both types of devices coexist within a close region [12], [13]. However, two clear channels (25 and 26) exist outside the 802.11 spectrum and can be used as the primary 802.15.4 channels for interference-free deployment [14]. Given the above, in our simulations, we have assumed that the two interfaces operate in different channels with no interference between them.

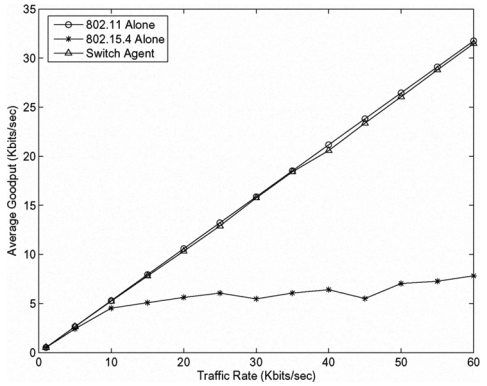
We have used on-demand routing protocol — AODV as the routing agents for both interfaces. We have simulated on a random topology as in Fig. 4(f). The random topology contains 100 nodes randomly generated with three source and destination pairs which share paths. All the nodes have at least one neighboring node within the transmission range of IEEE 802.15.4. We have also simulated on two other different topologies, one with a single joint node, the other with a single joint path. The simulation results turned out to have the same trends as the random topology. Due to space limitations, we only present the results for the random topology in this paper. Bursty traffic was generated at each source node. The bursty traffic alternates between idle time period and burst time period. During the idle time period, no data was sent. During the burst time period, data were sent at a constant bit rate. The time span of the idle time period and the burst time period both follow a poisson distribution with a certain average. The data packet size is 70 bytes and the simulation time is set for 30 minutes. Some of the other parameters used in the simulations are listed in Table I.

TABLE I
PARAMETERS USED IN THE SIMULATIONS [15]

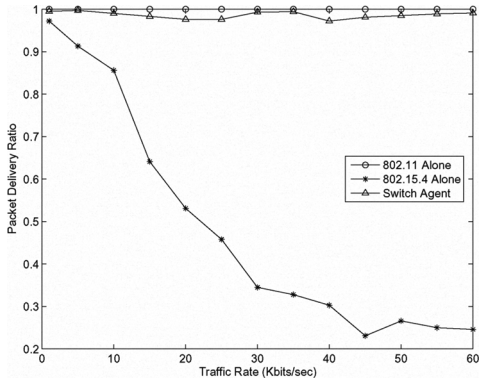
Parameter	IEEE 802.15.4	IEEE 802.11
transmission power	28.1 mW	660 mW
receiving power	62.1 mW	395 mW
idle power	1.4 mW	35 mW
data rate	250 kbps	2 Mbps
range	15 m	250 m

In the following discussions, the term "switch agent" refers to the scenario where sensor nodes having both IEEE 802.15.4 and IEEE 802.11 interfaces are used in conjunction with the switch agent proposed in this paper. The term "802.15.4 alone" refers to the scenario where sensor nodes with just IEEE 802.15.4 interface alone are used. The term "802.11 alone" refers to the scenario where sensor nodes with just IEEE 802.11 interface alone are used.

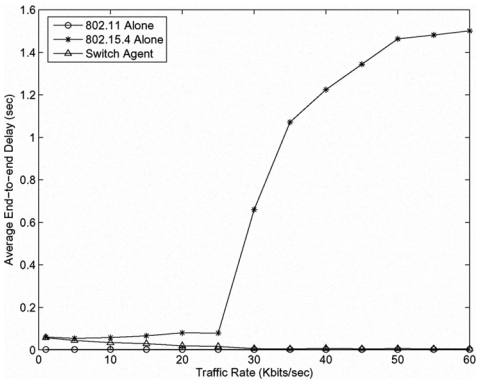
Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7294
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7294



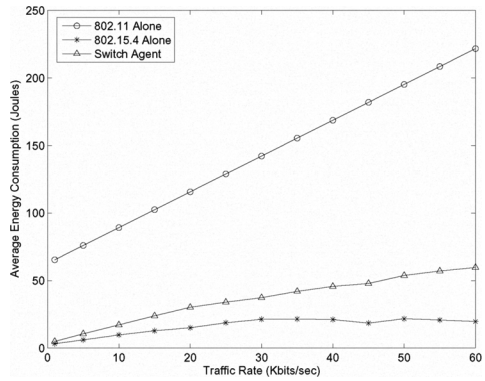
(a) The comparison of goodput



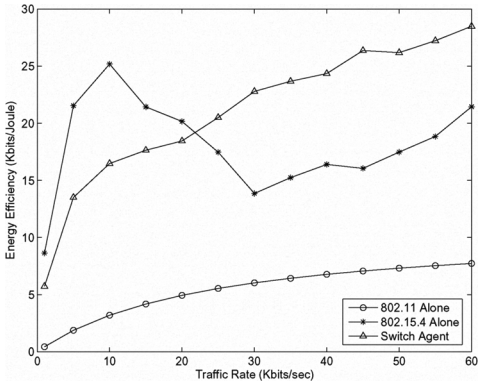
(b) The comparison of packet delivery ratio



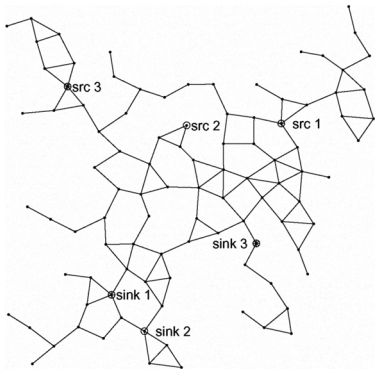
(c) The comparison of average end-to-end delay.



(d) The comparison of energy consumption



(e) The comparison of energy efficiency.



(f) Random topology with 100 nodes. Lines indicate 802.15.4 linkages.

Fig. 4. Comparison among 802.11 alone, 802.15.4 alone and the switch agent with different traffic load

B. Performance Metrics

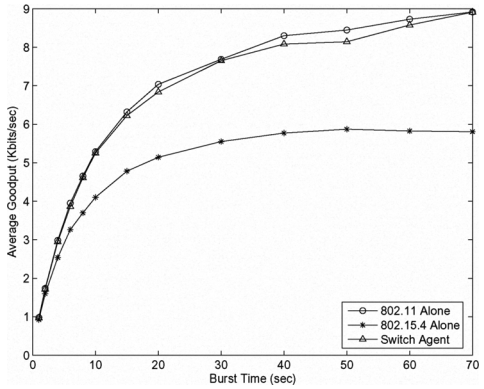
The following metrics have been used to evaluate the performance of the switch agent.

1) Average Goodput: the average number of bits (data packets only) received at a sink node within a unit of time;
2) Packet Delivery Ratio: the ratio of the number of data packets received over the number of data packets sent out;
3) Average End-to-End Delay: the average end-to-end delay between transmitting a data packet and receiving it at its destination;
4) Average Energy Consumption: the energy consumption of a single node on average;
5) Energy Efficiency: the number of bits received at the sink nodes versus the total energy consumed.

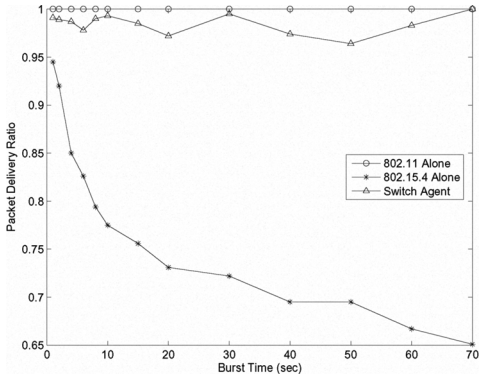
C. Bursty Traffic with Varying Rates

Fig. 4 shows the simulation results for bursty traffic with varying rates. In this simulation, we have kept the average

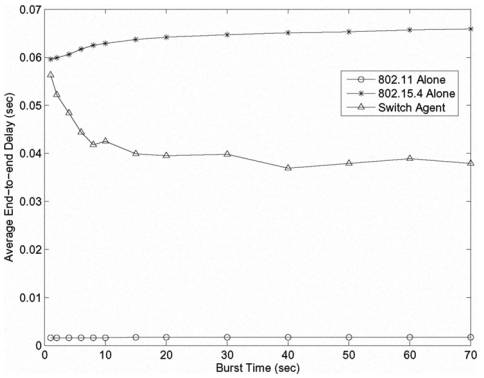
Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7294
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7294



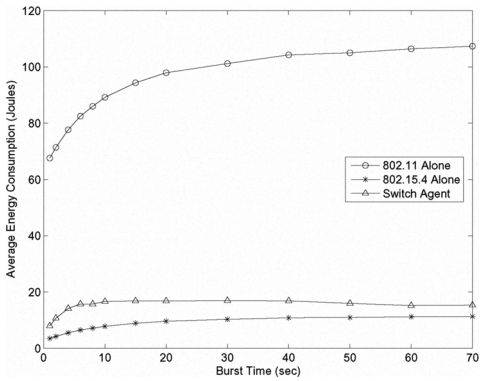
(a) The comparison of goodput



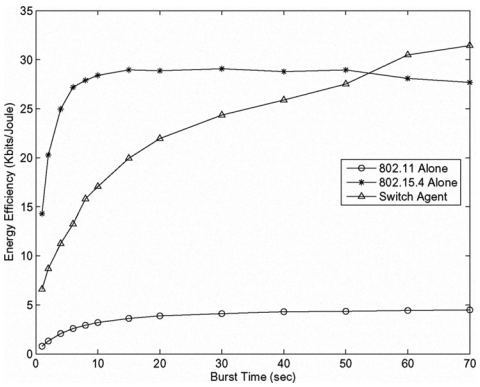
(b) The comparison of packet delivery ratio



(c) The comparison of average end-to-end delay.



(d) The comparison of energy consumption



(e) The comparison of energy efficiency.

Fig. 5. Comparison among 802.11 alone, 802.15.4 alone and the switch agent with different burst time

idle time and the average burst time to be fixed (both are 10 seconds), and investigated the performance under various traffic rates during the burst time period.

Fig. 4(a) shows that all the three scenarios yield the same average goodputs when the data rate is lower than 10Kbits/sec. For the data rate greater than 10Kbits/sec, the traffic load exceeds the data rate limit of IEEE 802.15.4. Packets start dropping for the 802.15.4 alone. For the switch agent, the goodput remains close to that of the 802.11 alone, since the IEEE 802.11 interface is turned on by the switch agent.

Fig. 4(b) shows the packet delivery ratio drops abruptly for the 802.15.4 alone when the data rate limit of IEEE 802.15.4 is reached, while the switch agent has as good delivery ratio as the 802.11 alone.

Fig. 4(c) shows a transition occurs on the end-to-end delay of the switch agent when the traffic load becomes heavier. It is the same as that of the 802.15.4 alone when the traffic load is light, while it gets close to that of the 802.11 alone when the traffic load increases. The end-to-end delay of the 802.15.4 alone increases dramatically when traffic load becomes heavy due to the increased collisions.

Fig. 4(d) shows the average energy consumption at each

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7294
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7294

node. The 802.11 alone consumes much more energy even when the traffic load is light, since its idle listening energy is much higher. When the traffic load exceeds the data rate limit of IEEE 802.15.4, its energy consumption does not increase much since the IEEE 802.15.4 network is already saturated. However, the energy consumption of the switch agent remains close to that of the 802.15.4 alone, since their IEEE 802.11 interfaces were selectively waken up.

Fig. 4(e) shows the number of bits received on every unit of energy consumption. When traffic load is light, the 802.15.4 alone and the switch agent have better energy efficiency than the 802.11 alone. The switch agent surpasses the 802.15.4 alone when the data rate reaches 25Kbits/sec and remains the highest efficiency among the three afterwards. That is because the IEEE 802.11 interfaces on the switch agent are selectively waken up.

D. Bursty Traffic with Varying Burst Time

Fig. 5 shows the simulation results for bursty traffic with varying burst time periods. In this simulation, we have kept the average idle time to be fixed (10 seconds), and investigated the performance under various average bursty time. The rate during burst time period is 10Kbits/sec.

Fig. 5(a) shows an increase of goodput with longer period of burst time for all the three scenarios. The switch agent has a goodput very close to that of the 802.11 alone, which is also much better than that of the 802.15.4 alone.

Fig. 5(b) shows that the switch agent has a delivery ratio close to that of the 802.11 alone, which is also close to 1. The delivery ratio of the 802.15.4 alone decreases with the increase of the burst time period, because the longer burst time could cause more collisions.

Fig. 5(c) shows the average end-to-end delay. The 802.15.4 alone has the largest delay and the 802.11 alone has the smallest delay. The switch agent falls in between. When the burst time period increases, the delay of the switch agent is getting closer and closer to that of the 802.11 alone, since more traffic is being sent through the higher-bandwidth radio.

Fig. 5(d) shows the average energy consumption at each node. The switch agent consumes much less energy than the 802.11 alone, since only the nodes involved in data communication will stay awake. It is a little more than what the 802.15.4 alone consumes.

Fig. 5(e) shows the energy efficiency defined by the number of bits received on every unit of energy consumption. Since the switch agent has a goodput very close to the 802.11 alone but with a lot less energy consumption, it yields a much better energy efficiency compared to the 802.11 alone without compromising the throughput and end-to-end delay. The energy efficiency of the switch agent will surpass that of the 802.15.4 alone when the traffic load gets heavier.

V. Conclusions and Future Work

This paper proposes a switch agent at the routing layer, sitting on top of dual routing agents. The switch agent monitors the traffic flow and switches on the interface with higher-bandwith and longer transmission range whenever the traffic rate becomes high. To save energy for using the high-bandwidth interface, the switch agent caches the routes established previously so that a unicast wake-up message can be sent out to selectively wake up the high-bandwidth interface at the downstream nodes. The switch agent also keeps a registry for flows which already have the high-bandwidth interfaces awake so that no further wake-up message transmissions are incurred for subsequent requests.

The simulations shows that the switch agent yields throughput, delay and packet delivery ratio comparable to the higher-bandwidth interface alone, without incurring much energy wastage.

In the future, we plan to extend the current work by allowing each sensor node to make its own switching decision based on its traffic condition. More flexible wakeup-sleep patterns along a path can be achieved by doing this. We also plan to allow the sensor nodes to switch based on the end-to-end delay for delay-bound applications. Different routing protocols, like OLSR [16], will be tried out in comparison with AODV.

References

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA'02*, Atlanta, Sep. 2002, pp. 88–97.

[2] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *INFOCOM'02*, New York, Jun. 2002, pp. 1567–1576.

[3] T. V. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *SenSys'03*, Los Angeles, Nov. 2003, pp. 171–180.

[4] T. Zheng, S. Radhakrishnan, and V. Sarangan, "PMAC: An adaptive energy-efficient MAC protocol for wireless sensor networks," in *IPDPS'05*, Denver, Apr. 2005, p. 237a.

[5] T. Pering, Y. Agarwal, R. Gupta, and R. Want, "CoolSpots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces," in *MobiSys'06*, Uppsala, Sweden, Jun. 2006, pp. 220–232.

[6] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting heterogeneity in sensor networks," in *INFOCOM'05*, Miami, Mar. 2005, pp. 878–890.

[7] T. Stathopoulos, M. Lukac, D. McIntire, J. Heidemann, D. Estrin, and W. J. Kaiser, "End-to-end routing for dual-radio sensor networks," in *INFOCOM'07*, Anchorage, Alaska, May 2007, pp. 2252–2260.

[8] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, Feb. 1999, pp. 90–100.

[9] Network simulator ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[10] *Standard for Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Std. 802.15.4, 2003.

[11] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.

[12] S. Y. Shin and W. H. Kwon, "Packet error rate analysis of IEEE 802.15.4 under IEEE 802.11b interference," in *Proc. Wired/Wireless Internet Communications*, 2005, pp. 279–288.

[13] K. J. Myoung, S. Y. Shin, H. S. Park, and W. H. Kwon, "IEEE 802.11b performance analysis in the presence of IEEE 802.15.4 interference," *IEICE Trans. Commun.*, vol. E90-B, no. 1, pp. 176–179, 2007.

[14] T. Hubler. Worry-free wireless networks. [Online]. Available: http://sbt.siemens.com/bau/products/Wireless/HPACEprint.pdf

[15] Datasheet for Chipcon CC2420 2.4GHz IEEE 802.15.4/ZigBee RF transceiver. [Online]. Available: http://www.chipcon.com/files/CC2420_Data_Sheet_1_2.pdf

[16] P. Jacquet, P. Mhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," *IEEE INMIC*, pp. 62–68, 2001.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7294
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7294