SOLAR-SKIN

TEAM: AMEY PENDHARKAR, MIRAJ SHAH, NEEL S SHAH

Motivation

Today, a lot of thought and effort goes into making buildings energy efficient, especially commercial offices and educational buildings. The indoor environment of many such buildings is regulated as per the requirements with the help of an HVAC (Heating, Ventilation and Air-Conditioning) system. The question that needs answering here is: "Is it possible to reduce the energy consumption by operating the HVAC system in an intelligent way"?

As a means to answer this question is a simple observation - Any building on an average day observes the following scenario: The outer Building-Surface facing the Sun (say the East Wall) starts heating up and eventually the heat penetrates inside the building through the walls and windows, heating up the indoor environment. When this happens, the HVAC system powers up to regulate the indoor environment to the desired setting. However, it takes some time say half an hour or more to get the indoor environment to the desired setting. What if we can avoid the excessive energy usage here and power up the HVAC system beforehand? We may notice here that if we can actually record the temperatures on the building surfaces and inside the buildings for a day on an hour by hour basis and repeat the same for several days, the resulting temperature profile can tell us the trend followed by temperature on any average day. Based on this knowledge, we can give our HVAC system a head start early in the day (usually temperatures increase as the Sun comes up) before the inner environment of building starts to heat up. Similarly, we can also shut down or suspend the HVAC system a little early in the evening (when temperatures usually start to drop)! This way we could save energy…

This motivated us to implement our knowledge in embedded systems and set up a Wireless Sensor Network consisting of multiple sensor nodes on the Building Surface that we wish to study. We currently use *"Mbed"* as our hardware platform for the sensor nodes (Each Mbed has a small board containing one Temperature and one Light sensor mounted on it) and our own *modified protocol* based on *Dynamic Source Routing*. The idea is as follows: Each sensor node collects data and transmits the data in form of packets to the Gateway node which is operated by the user. The collected data is used to prepare a statistics chart which can then be analyzed to get some good information about the Building's Temperature and Light profile throughout the day. Suggestions on the best way to operate the building's HVAC system can then be made based on this analysis.

Wireless Sensor Network Protocol Design based on DSR

Dynamic source routing relies on source routing instead of the conventional approach of routing table maintenance at each intermediate node, so the packet itself contains the entire information about the route that needs to be traversed to reach the destination. Since this protocol is an "on demand" protocol design, it restricts the control packets used to periodically update the routing tables at each node.

The two major phases of DSR are **"Route Discovery"** and **"Route Maintenance"**.
During the *Route Discovery* phase the network is flooded with the ***"route request"*** packets. As each node broadcasts the packet, it appends its own address in the existing route record. Once the packet reaches the Gateway node which is the destination, the Gateway node sends out a ***"route reply"*** packet which is unicast via the same route from where the corresponding route request packet arrived.
For example, source S initiates the route request to destination D; on receiving the route request packet, the destination initiates a route reply which is sent via the exact reverse route that exists in the route request packet (remember that the packet saves the address of each intermediate node through which it passes. So the destination node can use the same path to send out the reply packet).

Route Maintenance is the phase when source S has a route to destination D which doesn't work anymore (maybe some intermediate node broke down or the network topology changed somewhere in the middle and needs to be updated). *Route Error* and *Route Salvage* are used in the route maintenance phase and eventually a route request might be reinitiated

We assume in our case that most of the time after the network deployment, our nodes are going to be nearly stationary. We use this to our advantage and maintain a ***cache*** at each node so that once the routes are cached by the nodes; we do not need to do a route discovery again. The cache is also used to initiate a route reply if the path to the destination is already known from the intermediate node's cache.

When a link in the route is broken (maybe a node gets killed), a **"*route error*"** packet containing information of that link is sent back to the source of the transmission. Further, to update all other nodes (which may be transmitting packets via this dead node) about the dead node, a **"*route salvage*"** packet is broadcasted in the network. Every node having the non-functioning link in their cache will initiate a route discovery again and obtain a new optimal path to the destination.

Packet Structure

Description	*Size*
Delimiter (0x7E)	1 byte
Length (cmdID to cmdSpecific)	1 byte
TTL	1 byte
cmdID	1 byte
cmdSpecific	2-120 bytes
CRC	1 byte

The various ***cmdID packet types*** are as follows:

Description	*Size*
DATA *cmdID*	1 byte
Source	1 byte
Route record	0-MaxHop bytes
Destination	1 byte
sensorData	4 bytes

Description	*Size*
RREQ *cmdID*	1 byte
UniqueID	1 byte
Destination	1 byte
Source	1 byte
Route Record	0-MaxHop bytes

Description	*Size*
RREP *cmdID*	1 byte
Source (destination of reply)	1 byte
Route Record	0-MaxHop byte
Destination (source of reply)	1 byte

Description	*Size*
RSAL *cmdID*	1 byte
Link Broken Node1	1 byte
Link Broken Node2	1 byte

Description	*Size*
DACK *cmdID*	1 byte
Source (destination of ACK)	1 byte
Route Record	0-MaxHop bytes
Destination	1 byte

Route Discovery

When some node S originates a new packet destined for some other node D, it stores the source route (the sequence of hops that the packet should follow on its way to D) in the packet's route record. Normally, S will obtain a source route by searching its ***cache***. If no route is found, it will initiate a Route Discovery to dynamically find a new route to D. In this case, we call S the initiator and D the target of the Route Discovery.

For example, say node A is attempting to discover a route to node E. To initiate the Route Discovery, A transmits a ***"route request"*** message as a single local broadcast packet, which is received by (approximately) all nodes currently within wireless transmission range of A. Each route request message identifies the Initiator and Target of the route discovery and also contains a *unique request_id* determined by the initiator of the request. Each route request also contains a record, listing the address of each intermediate node through which it has been forwarded. This Record keeps updating (adding addresses of intermediate nodes) dynamically as the packet passes via more number of hops (nodes).

When any intermediate node receives a route request, it checks if it is the target of the Route Discovery. If so, it sends a ***"route reply"*** message to the initiator of the route discovery, giving a copy of the accumulated route record from the route request packet; when the initiator receives this route reply, it caches this route in its route cache for future use in sending subsequent packets to this destination. However, if the node receiving the route request has recently seen another route request message from the same initiator bearing the same request id or if it finds that its own address is already listed in the route record in the route request message, it will simply discard the request. Otherwise, this node appends its own address to the route record in the route request packet and propagates it by transmitting it as a local broadcast packet (with the same request id).

Route Maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that the packet has been received by the next hop along the source route; the packet is retransmitted (until the maximum attempts limit is reached) until the confirmation of receipt is received. If the confirmation is not received, that particular link is considered dead and Route Maintenance is performed.

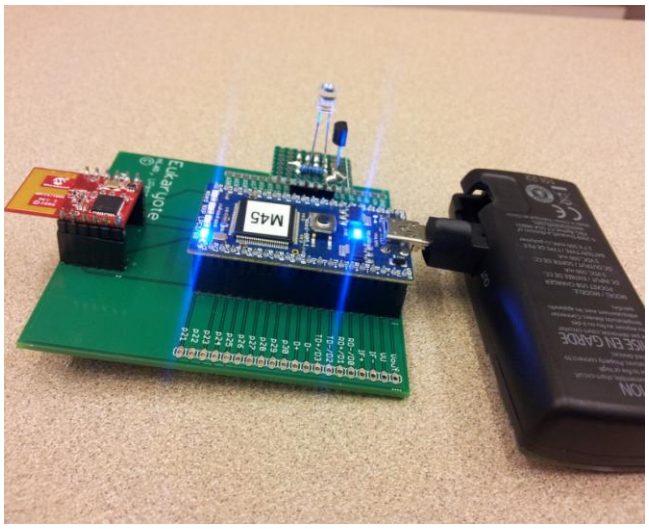
Route Caching

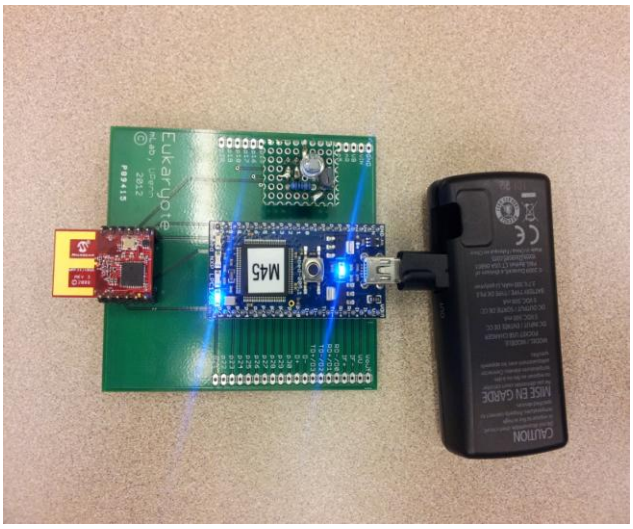
As the route reply propagates through the intermediate nodes, the intermediate nodes start caching the routes from the route records of the data packets and also from the route replies. Caching reduces the number route discoveries and hence reduces the number of packets transmissions and the time wasted to get the route from the source to destination. Also when route request are received and the particular node has the cache route to the destination of the route request then it will initiate the route reply by copying the whole route record form the route cache.

The Experiment Setup

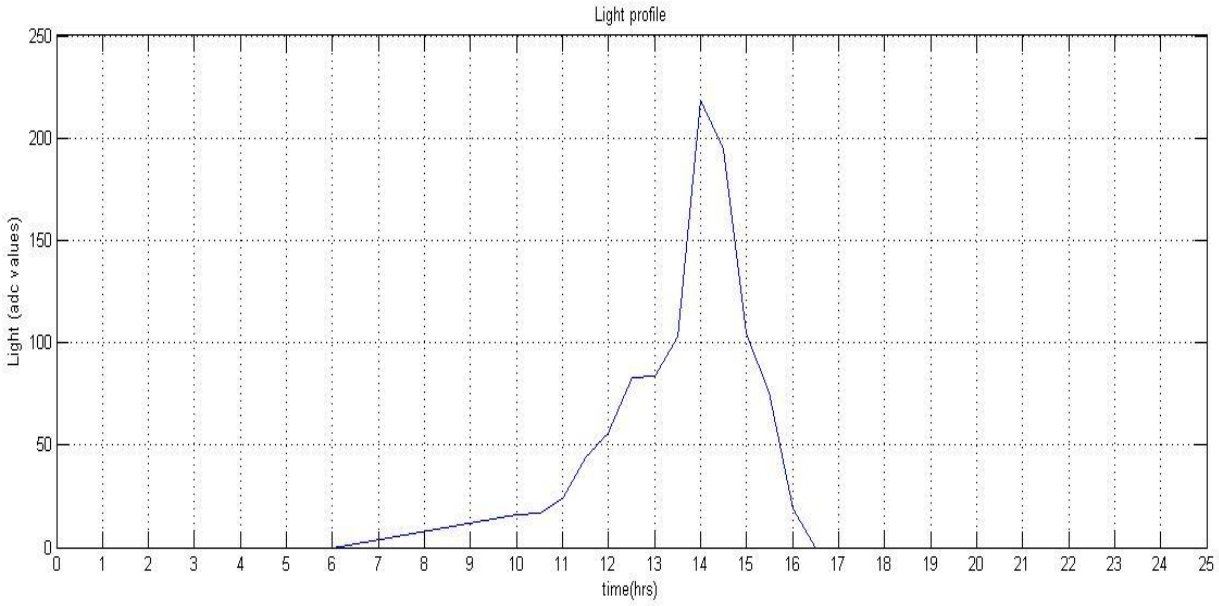
Now comes the fun part of deploying the nodes and collecting the data !

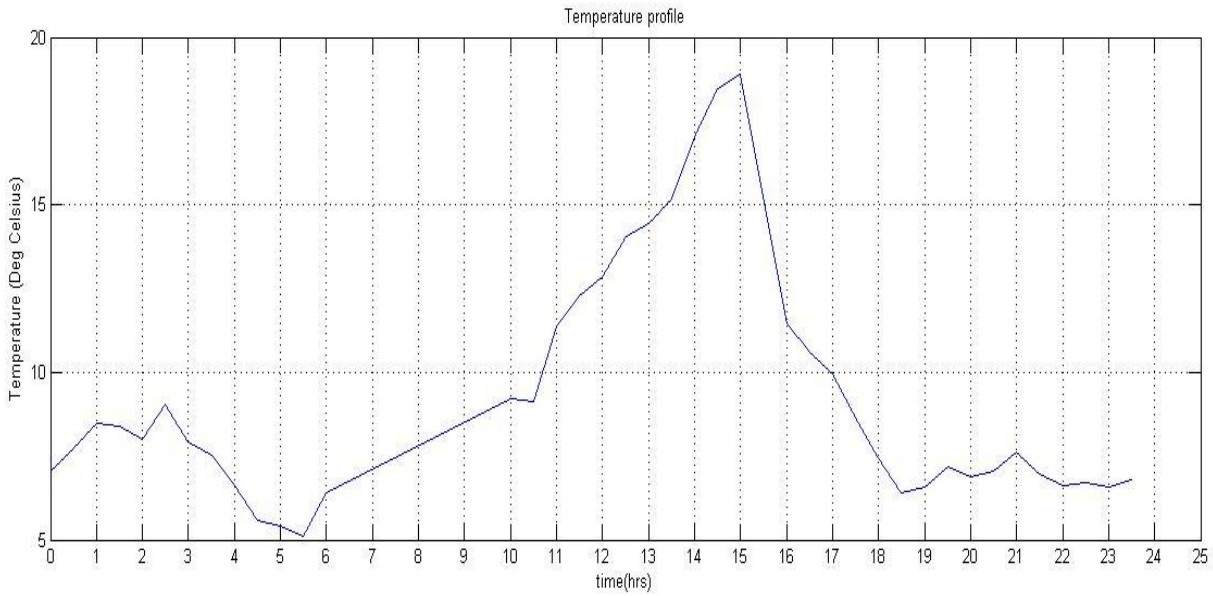
Our hardware setup consists of Temperature sensors (Digikey:MCP9700A-E/TO-ND), Photodiodes (Digikey:PDB-C139-ND) and Eukaryotes. Following are some pictures of our sensor nodes





We deployed 16 of these sensor nodes in Towne 303, Towne 305 and Towne 307 which gave us access to the West and South walls of the Building. Following are plots of sensor readings of a node deployed overnight outside a window in Towne 303.

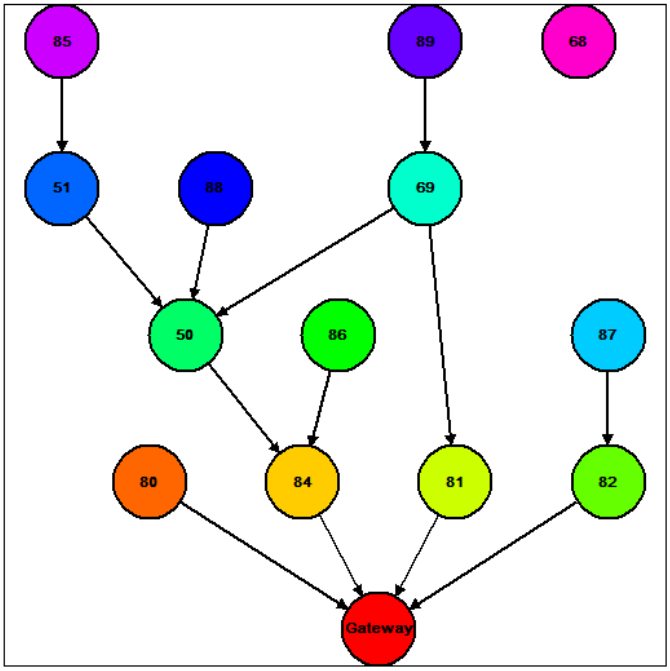


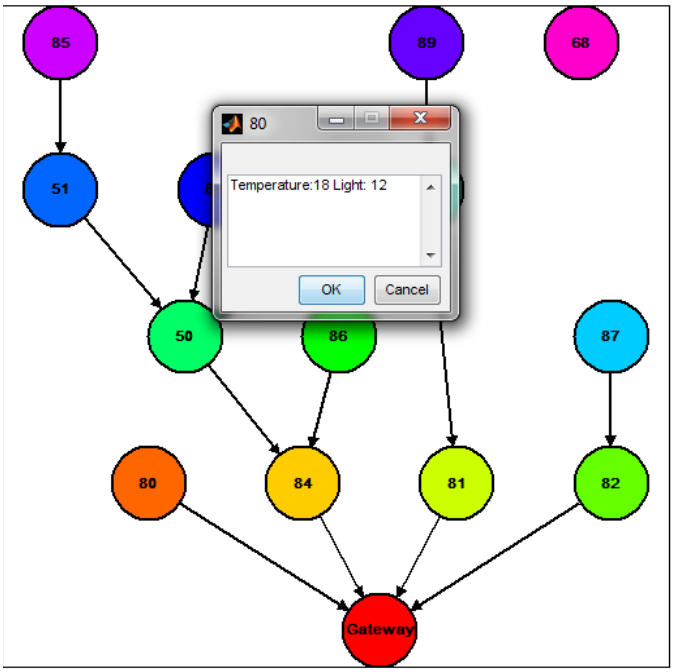


Note that these plots convey the general scenario that the Temperatures outside increase as the day progresses and start to drop late in the afternoon. These are the readings taken on a sunny day, which give us a good insight into the behavior of Temperature and Lighting conditions.

The GUI

We have a GUI showing the Connectivity Graph of our Network Topology. It dynamically updates the changes in Topology as the wireless sensor nodes move. Also, we can check the Temperature and Light readings of any node by clicking it on the GUI. Following is a snapshot of our GUI showing the Connectivity Graph of the Sensor Network.





Note that these numbers are the numbers mentioned on the mbeds. We use the same numbers to help us identify the sensor nodes easily. The arrows show the direction of travel of data packets. For example, the data packets from node #85 reach the Gateway node by hops via nodes #51 -> #50 -> #84

References

1. **Dynamic Source Routing in Ad Hoc Wireless Networks** *By David B. Johnson, David A. Maltz*
2. **Enhanced Route Maintenance for Dynamic Source Routing in Mobile Ad hoc Networks** *By Bin Xiao, Qingfeng Zhuge*
3. **DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks** *By David B. Johnson David A. Maltz Josh Broch*