ABSTRACT

MCCLUSKY, DOUGLAS RANDALL. Ad-hoc Wireless Routing for Wildlife Tracking with Environmental Power Constraint. (Under the direction of Kazufumi Ito.)

The purpose of this paper is to suggest an algorithm by which mica motes can organize themselves into a network to relay packets as quickly as possible under energy constraints from environmental harvesting. This problem is part of a larger project to develop a means to monitor red wolves using a mica mote network. The network has three parts: sensor motes attached to collars on the wolves, a base station or base stations that receive packets and display them in useable form for scientists and relay motes that forward packets from the sensor motes to a base station. The proposed algorithm adapts Hohlt et al's Flexible Power Scheduling to work under Kansal et al's Environmental Harvesting power constraint. Employing this strategy changes energy consumption from a performance objective to a constraint, allowing me to add my own throughput maximizing piece to the algorithm, based on dynamic programming and microeconomics. I also discuss the ongoing development of a simulation of this algorithm, designed to test its performance and to solve implementation problems.

Ad-hoc Wireless Routing for Wildlife Tracking with Environmental Power Constraint

by

Douglas McClusky

A thesis submitted to the Graduate Faculty of

North Carolina State University

In partial fulfillment of the

Requirements for the degree of

Master of Science

In

Operations Research

Raleigh, NC

2006

Approved by:

Dr Kazufumi Ito _____

Dr Robert Fornaro _____

Dr Robert Buche _____

BIOGRAPHY

Douglas McClusky was born. He grew up doing all sorts of cool stuff that you should ask him about if you're interested, as well as becoming an Eagle Scout. He went to the NC School of Science and Math before going to NC State University for his undergraduate degree in Computer Science and then his MS in Operations Research. Now, he hopes to go on to other interesting personal adventures, perhaps involving multi-agent systems and wireless sensor networks. Tune in for the next thrilling episode.

Table of Contents

List of Figures

I. Introduction

The purpose of this paper is to suggest an algorithm by which mica motes can organize themselves into a network to relay packets as quickly as possible under energy constraints from environmental harvesting. This problem is part of a larger project to develop a means to monitor red wolves using a mica mote network. Mica motes are tiny computers designed by Berkeley, part of a class of "smart dust" devices equipped with sensors and radio communication hardware. Particularly useful for wildlife tracking are mica motes' GPS and radio communication capabilities and their relatively inexpensive unit price. The network has three parts: sensor motes attached to collars on the wolves, a base station or base stations that receive packets and display them in usable form for scientists and relay motes that forward packets from the sensor motes to a base station.

Wildlife tracking is an integral part of understanding ecosystems and the impact we have on them. It also is necessary to maximize the efficiency of repopulation efforts, like the current initiative to bring back red wolves. Red wolves were once common in the Eastern United States, but were hunted nearly to extinction. Currently, efforts are being made to reintroduce them into their former habitats. A mica mote sensor network is one proposed alternative for gathering information about the red wolves that have been released into the wild at Alligator National Wildlife Reserve in Eastern North Carolina.

If a solution to this problem is found for the wolves, it will also benefit other wildlife tracking projects. Mica mote tracking networks promise to gather more data and cost less to deploy than current technology. Current wildlife tracking uses radio telemetry. This requires expensive, bulky collars and costly, labor-intensive searches on foot or in cars and airplanes. It also only gives the location of the wolves and only at the time of the check. Mica mote sensors, however, can take GPS readings much more often and supplement them with

information about temperature, light and other sensor readings. They also cost a small fraction of a radio collar. For this reason, wildlife tracking as an application of wireless sensor networks has been the subject of research for several years.

Current power-management algorithms for sensor networks are designed to maximize some measure of performance over time subject to the finite battery lives of the nodes. This allows a one-time deployment network to last longer or extends the time between costly battery replacements. Sensor network power management literature falls into 4 categories: topology control, data management, packet routing and sleep cycle management.

Topology control, packet routing and sleep cycle management all involve decisions about how and when a mote communicates with its radio. Topology control is manipulating the way a network constructs itself. This can involve organizing motes into smaller structures within the network, like clusters[1,2], or choosing between using direct or indirect connections, based on power requirements[3]. Packet routing algorithms decide where to send packets at each step along the way. Power-aware routing algorithms make routing decisions to maximize measurements of utility like the network's coverage time[3] or total lifetime information flow[4].

Sleep management, as its name suggests, is simply deciding when a mote should power down. Since components like the mote's radio take nearly as much power to listen as to transmit, knowing how to shut it down when it will not be needed significantly reduces power consumption. Data management uses properties of the information passing through the network to reduce the number of packets required. One example of this is Goel and Imielinski's adaptation of MPEG compression for sensor networks[5].

Within the current literature, all of these methodologies are applied to maximize some utility measure of a finite battery life. If the cost of physically

accessing nodes is large enough, even this level of utility may not justify deploying a system for just one battery lifetime, and battery replacement may be too expensive to implement. For the purpose of wildlife tracking, where nodes are often deployed in hard-to-reach places, this is likely to be the case.

An alternative strategy proposed by Kansal et al is to power the nodes with a constantly replenishing source of energy, like solar panels[6]. Under this strategy, a network node that does not use more power than is provided by its environment can continue functioning until it suffers a hardware failure. Employing this strategy changes energy consumption from an objective to a constraint, allowing us to focus on some other performance objective for the network, such as maximizing network throughput within this energy constraint. This paper will apply routing and sleep cycle management to this new paradigm of sustainable performance within energy harvesting constraints. With this in mind, I will adapt a routing algorithm called Flexible Power Scheduling, designed by Hohlt et al specifically for mica motes[7], to work within the Environmental Harvesting power constraint. I will then add my own throughput maximizing piece to the algorithm, based on dynamic programming and microeconomics.

II. Theoretical Foundations

Environmental Harvesting

The simplest way to power a mica mote with a solar panel is to connect the mote to a huge solar panel and a huge battery. However, this is hardly cost effective for a practical wildlife tracking implementation. Kansal et al's paper on Environmental Harvesting answers just how big a solar panel and battery have to be for a given level of performance. Their theory of sustainable performance at infinity is based on their characterizations of the energy provided by a power source and the energy used by a device.

$A(\rho, \sigma_1, \sigma_2)$ - source satisfies the equation, $\rho T - \sigma_1 \leq \int_T E(t) \geq \rho T + \sigma_2$.

$A(\rho, \sigma)$ - consumer satisfies the equation, $\int_T E(t) \leq \rho T + \sigma$.

for any period of time, T, where ρ is a measure of average power and the σ's are upper and lower bounds on energy input or output.

If power availability is independent of consumption, they prove that a $(\rho, \sigma_1, \sigma_2)$ - source can sustain a $(\rho*, \sigma)$ - consumer with battery capacity $\sigma + \sigma_1 + \sigma_2$ indefinitely.

Kansal et al built a solar panel attachment for the mica motes and wrote software to manage power according to this constraint. The 'Heliomote' sampled power supplied by the solar panel attachment on the mote's sensor board and changed its performance to remain within its long-run power constraint. They also tested a simple relaying application on their Heliomote design.

FPS

The basis of Hohlt et al's Flexible Power Scheduling (FPS) protocol is that each mote keeps track of its own power schedule of T_s length time slots grouped into cycles of length $m*T_s$. Motes coordinate their schedules by advertising open slots to receive data and by responding to others' advertisements. In addition, each mote has a demand equal to the number of packets it has scheduled to

receive per cycle and a supply equal to the number of forwarding slot reservations per cycle it has made with motes closer to the base station. These are both integers and their sum is less than m. The protocol begins when the base station starts advertising available slots. Each mote starts with a demand of one more than the number of packets of data it generates in one cycle. When a mote whose demand exceeds its supply hears an advertisement, it attempts to schedule that slot. When a mote satisfies its demand, it begins advertising slots and sleeps during empty slots to conserve energy. As motes are added to the network, their cycle length and their position within their schedules are synchronized with their parents'.

Hohlt et al implemented a small prototype tracking network to test FPS. They report that FPS's scheduling results in significant energy savings. They also found that FPS adapts quickly to changes in the distribution of demand over the network.

III. Environmentally Constrained Latency Bidding

Energy Constraint

Adapting the FPS protocol to solve the EH-constrained shortest path problem requires finding a way for each mote to keep the average power use of its schedule below its ρ and getting motes to negotiate slot reservations in a way that fairly distributes network resources and gives each mote the shortest available expected time to a base station achievable with this distribution.

The first problem is fairly simple. Assuming batteries are big enough, the sustainable performance constraint is satisfied when $\Sigma \varphi_{i,s} P_s \leq \rho_i$, where $\varphi_{i,s}$ is the proportion of time mote i spends in state s, and P_s is the power expended in state s. This is a reasonable assumption, since upper and lower bounds would be known prior to deployment for both power provided by a given solar cell and power consumed by a mote running a given program. Choosing battery size would therefore be part of the engineering decisions of building a network.

Since each mote already keeps track of its power schedule, it just needs to choose the number of slots each cycle it spends in each state to keep average power consumption below ρ.

Hohlt et al used six states in their description of FPS:

- (T)ransmit
- (R)eceive
- (A)dvertise
- (RP) Listen for reservations
- (TP) Send a reserve request
- (I)dle

To write the power constraint, I divide Idle into

- (L)isten for advertisements
- (S)leep

This makes the power constraint

$$\left(\varphi_T + \varphi_A + \varphi_{TP}\right) P_T + \left(\varphi_R + \varphi_{RP} + \varphi_L\right) P_L + \varphi_S P_S \leq \rho$$

or

$$\frac{\left(t_T + t_A + t_{TP}\right) P_T + \left(t_R + t_{RP} + t_L\right) P_L + t_S P_S}{m} \leq \rho$$

where each t_k is the time, in slots, spent in a given state.

Shortest Path Formulation

A Dynamic Programming formulation of the shortest path problem facing each mote is my jumping off point to solve the second problem. While dynamic programming is used in static routing problems routinely, it was not relevant to the multi-destination network lifetime problem that has been the focus of sensor network power management research because this problem is NP complete in the general case.

One property of a wildlife tracking network that I will use in my analysis is the unidirectionality of information flows. Data is taken by motes on the wolves

and is relayed by the network to one or more base stations for collection and analysis. This unidirectionality is what allows for a simple DP solution of the routing problem. Motes do not need to make routing decisions for arbitrary destinations. They need only find the shortest path to the nearest base station. This problem formulation differs from most sensor network literature because of the unidirectionality assumption and because relay motes are not collecting the data they are sending to the base station.

Assuming that $p_{i,j}$ is the probability of mote i successfully transmitting to its parent, mote j, the expected time from when i receives a packet to when it successfully forwards that packet is $(q_i*m/(p_{i,j}*r_i))*T_S$, where r_i is the supply of reservations at mote i and q_i is the queue at mote i. Since supply and demand must be balanced to prevent the queue from exploding and minimize underutilized supply, q_i is generally also mote i's demand. Since T_S is common to all motes, I will use this as my unit of time and drop it from future calculations. This means that in the simplest case, where each mote forwards to one receiver, the DP shortest path problem for a mote whose shortest path is some k hops from its current mote is

$$D_k = \min_{u_k} \left\{ \frac{q_k m_k}{p_{k,u_k} r_k} + D_{k-1} \right\}$$

$$D_0 = 0$$

Extending this to a mote i that has reservations with motes in set $n(u_i)$,

$$D_i = \min_{u_i} \left\{ \frac{q_i m_i}{r_i^2} \sum_{j \in n(u_i)} \frac{r_{i,j}}{p_{i,j}} + \frac{1}{r_i} \sum_{j \in n(u_i)} r_{i,j} D_j \right\}$$

Where $r_{i,j}$ is the supply of reservations i has made with j.

Demand-Driven Path Bidding

FPS can be adapted to approximate the solution to this problem. By keeping track of dropped transmissions, motes can estimate the p's between themselves and their neighbors. In fact, keeping track of link strengths is part of the standard communication platform for mica motes. Also, each mote's power schedule already contains all the other variables required to estimate its piece of the DP-formulated shortest path problem and to manage its power schedule as outlined above.

Each advertisement includes the mote's estimate of the time it will take packets from a respondent to reach the nearest base station. Motes with some demand attempt to satisfy that demand with the lowest-estimated slots available by responding with their estimated reduction to their estimate of the time required to get packets to a base station, D. Since supply and demand at each mote will be roughly equal at each mote, this means both supply and demand should be prioritized by $\Delta D_i/\Delta d_i = \Delta D_i/\Delta s_j = \Delta D_i/p_{i,j}$. This insures the most change in D for any given supply. Advertising motes award their slots to the responses with the highest $\Delta D/p_{i,j}$. When mote j awards a slot to mote i in this manner, it increases i's supply and j's demand by $p_{i,j}$. Since motes seek to optimize the utility of both their demand and supply, the motes in any given neighborhood will function like an economy. This property ensures optimal slot assignments for any pattern of demand over the network in equilibrium.

To maintain the accuracy of motes' shortest path estimates, each mote's supply should be greater than, but as close as possible to its demand. Excess demand would make paths through a mote longer than advertised. Excess supply would make the mote's ΔD estimates larger than its actual change would be and would not contribute to path throughput, since the mote cannot forward packets it does not have. To maintain balanced supply and demand while maximizing its D, a mote will follow this pruning procedure.

If $d > s$, drop $\min_{\{n_d\}}(\Delta D / \Delta d)$ until $d \leq s$. The drop message is sent on the next confirmation to chosen neighbor. Until the mote gets new supply or one of its demand reservations is dropped, only give reservations to motes whose $\Delta D / \Delta d$ Bid $> \min_{\{n_d\}}(\Delta D / \Delta d)$.

If s > d after some time waiting for new demand, drop reservation $\min_{\{ns \mid \Delta s < (s-d)\}}(\Delta D/\Delta s)$ until $\{ns \mid \Delta s < (s-d)\} = \emptyset$. Drop messages are sent on next scheduled send to chosen neighbors. Until the mote gets new demand or one of its supply reservations is canceled, Bid only if new $\Delta D / \Delta s > \min_{\{n_s\}}(\Delta D / \Delta s)$.

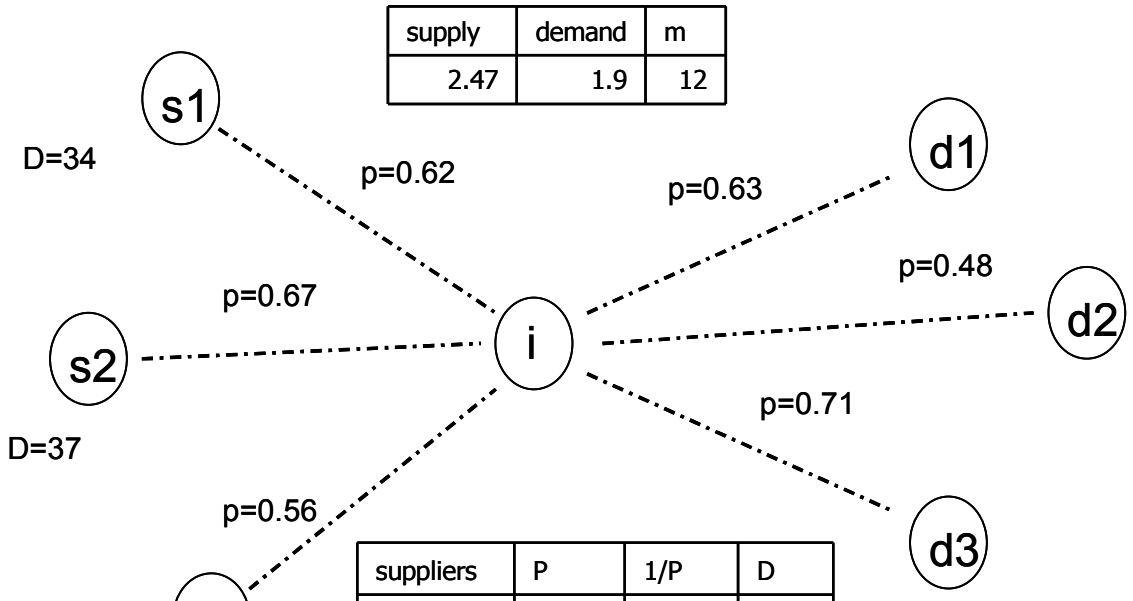
Finally, to make sure that each mote becomes aware of new, higher bids than those currently awarded slots, each should reserve at least one advertise slot per cycle. Also, current slot holders and their suppliers should update each other about changes in their D's, since this may change each's available options. This could be integrated into the scheduled send/confirm exchange. The adapted protocol starts when base stations begin advertising its time-left-0 slots. Other motes start by filling their schedules with as many listen slots as they can within their power constraints. Since ΔD is infinite for any mote's first link, some tie-breaking rule will decide the order in which its neighbors get their first slots. After that, slots are awarded to the highest bid.

IV. Numerical Examples

Each mote manages its power schedule with three types of interactions: advertise/bid, trim excess demand, and cut persistently excess supply. Here are numerical illustrations of each.

Advertise/Bid

An Advertise/Bid interaction begins when a mote sends an advertisement to its neighbors with its estimate of how much time packets routed through it would take to reach a base station. In this case, mote i has four reservation slots with link strengths and server D estimates as in the table below. By calculating one iteration of the DP formulation above, mote i's D approximation is 36.22 cycles.



supply	demand	m
2.47	1.9	12

suppliers	P	1/P	D
s1	0.62	1.61	34
s1	0.62	1.61	34
s2	0.67	1.49	37
s3	0.56	1.79	31
D	36.22		

Fig. 1 Calculating D

When neighboring motes hear this advertisement, they decide whether to make a bid for the slot.

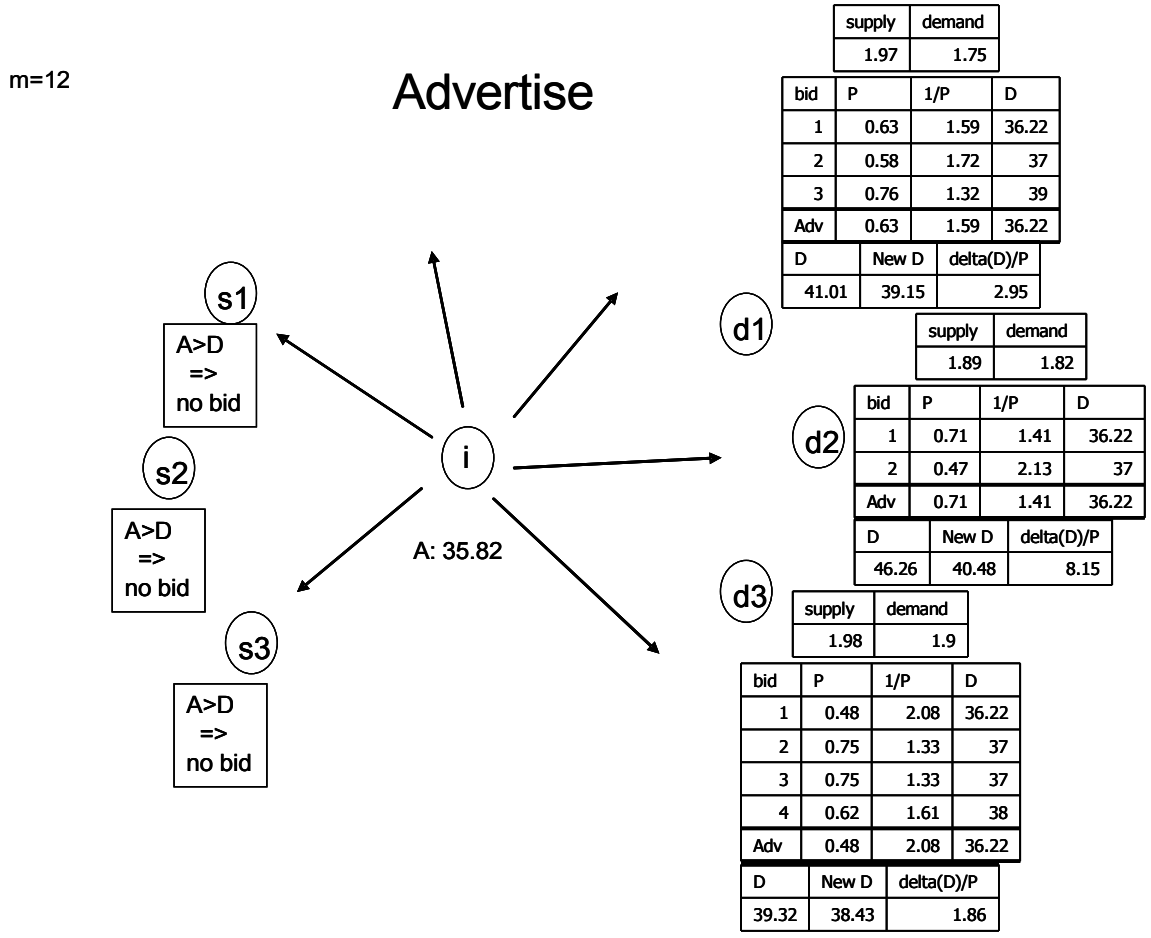


Fig. 2 Advertisement and Bid Determination

Award Slot/Trim Demand

In this case, d2's bid is the highest, so it wins the slot, but the additional 0.71 demand puts i's demand above its supply. Therefore, it checks the slots it has awarded and drops the lowest bid, setting its demand back below its supply. Until i's supply increases, it will ignore bids below the 3.39 of its last dropped slot.

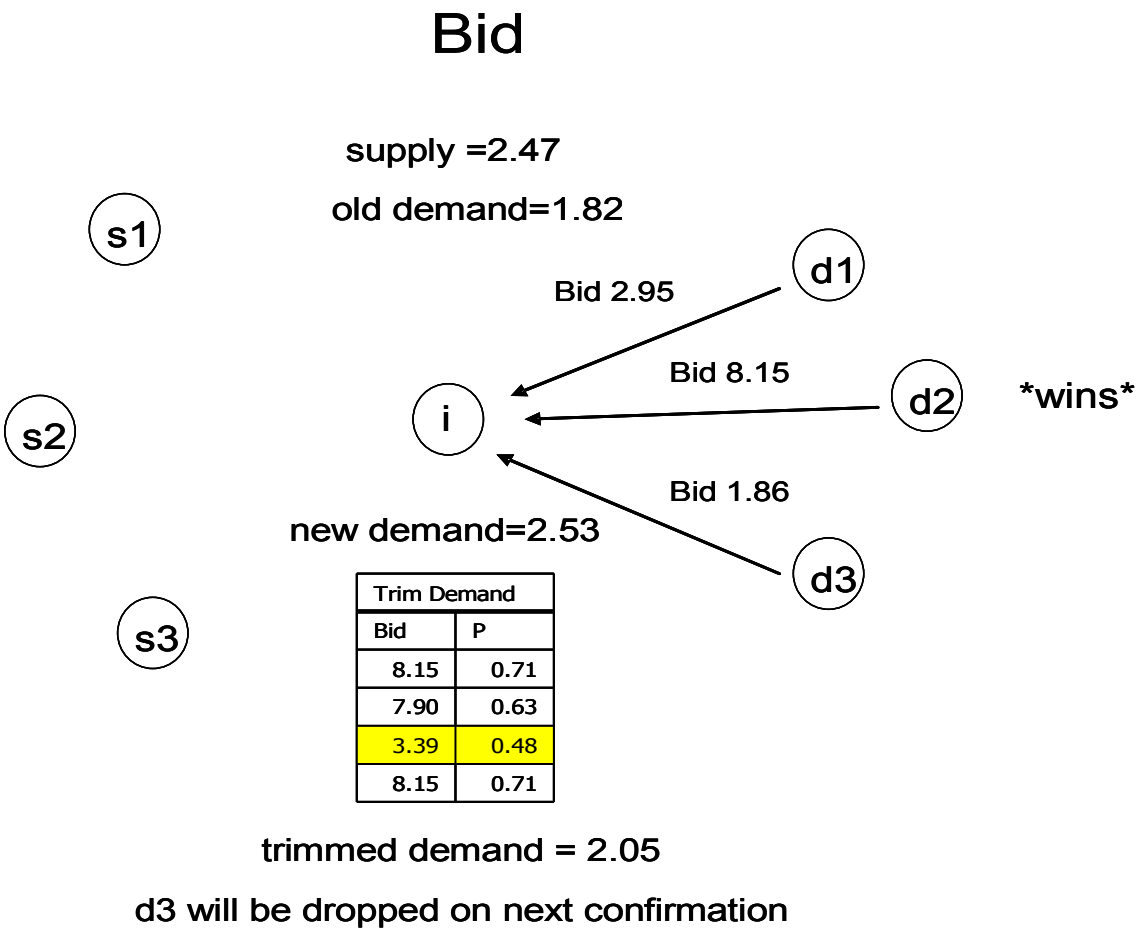


Fig. 3 Bid Cycle, Slot Awarding and Demand Trimming

Cut Supply

Some time later, network demand has shifted, and i's demand has dropped to 1.77. Therefore it checks the supply slots whose link strengths are less than the excess supply for the minimum bid. In this case, all slots' strengths are less than the excess supply, and slot 3 has the minimum bid.

supply	demand
2.47	1.77

suppliers	P	1/P	D(s)	new D(i)	delta(D)/P
s1	0.62	1.61	34	37.85	2.72
s1	0.62	1.61	34	37.85	2.72
s2	0.67	1.49	37	36.94	1.17
s3	0.56	1.79	31	38.71	4.56
D	36.16				

supply	demand	D
1.80	1.77	36.94

Fig. 4 Surplus Supply Cutting

After dropping slot 3, i's supply drops to 1.8, so it has no more slots whose link strengths are less than its 0.03 excess supply. It will update its clients about its new D estimate in its next confirmation messages, and will not respond to advertisements unless its bid is more than its minimum bid slot, now 1.17, until its demand increases.

V. Simulation

Motivation

To get a better understanding of how this idea will perform when implemented, I built a computer simulation using Rockwell Software's Arena simulation package. This simulation and the animation example that goes with it should make it easier to visualize the way my proposed network algorithm would work. Arena's statistic gathering and numerical optimization capabilities may help solve engineering questions like the best choice of m.

One question about this protocol's performance is simply how quickly it can relay data, and consequently what data sampling rate it could accommodate given some density of wolves in one area. In the simulation, each wolf generates packets at some constant rate and keeps them in a queue until they can be relayed to the nearest network mote. This assumes constant sampling and that the wolf mote will always attempt to send data to the nearest relay

mote. In reality, this generation and relaying is another problem which must be solved before a network can be deployed, but this abstraction is sufficient to illustrate the relay network's performance. When more is known about a solution to the generation problem, its rules can be added to this piece of the simulation.

Another performance question is how much of a mote's available power is dedicated to the upkeep of its schedule instead of being used to relay data. One issue with asking this question to compare with other protocols, however, is that the renewable power constraint makes it rational for motes to increase upkeep interactions up to their power constraints, regardless of the proportion of power for overhead that this causes. For instance, since a mote cannot know when advertisements will come, it is better for it to listen than to sleep if power permits. Similarly, more advertise and bid messages increase a mote's chance to find better clients and suppliers, respectively, in a given period. Therefore, higher overhead increases the rate at which a mote gets to equilibrium.

How quickly the network adapts to changes in the environment is also important. One of the properties of FPS is that it adapts quickly to changes in network demand. The new protocol also reacts to changes in link strength and power availability. This simulation can help answer whether the new protocol inherited FPS's adaptation to demand and how quickly it can adapt to changes in these other two environmental properties. The simulation includes the ability for the operator to cause each of these environmental variables to change in the middle of a run. This allows for tests of the protocol's adaptability to these changes individually or in combination.

Simulation Specification

There are n motes in the network. For the example animation, this is 20. Since queue allocation and set definitions in Arena are difficult to automate, the operator will have to manually change these to run with a different n. The

power schedule for each mote is kept in an nxm matrix, PS. P, an nxn matrix, represents the actual link strengths between motes. This can be written as some P_0 + (t>threshold)*ΔP to test the protocol's adaptation speed for link strength changes.

P^ is an nxn matrix of the motes' estimations of P. Since the underlying link strength and not the chance that another mote may not be listening is the desired quantity to measure, motes only sample P when they know that the other mote will be receiving. Therefore, P^, the estimated probability of successful transmission and confirmation, will be the same for both motes in a link. This means P^ does not need to be represented by n different estimations, just one matrix.

Simulation events come from three types of entities. The first entity flow is for the packets. The second is for solar panel sampling events. The third flow executes the action specified by each mote's power schedule at each tick.

Packet entities are divided for record keeping and animation by wolf. Each wolf generates packets at some constant rate. Packets are sent to the relay at the wolf's position if the relay is listening. Wolves have discrete positions, each of which has a link strength with its nearest relay. Each wolf's position is determined by a random walk. For the first animation, I simplify this further and give each of the five wolves a constant position. Once at a relay mote, the packets wait in its queue until that mote successfully sends them to another mote. When packet entities reach a base station, they are destroyed.

Solar events occur at some constant rate. Each triggers a draw from the solar power distribution and updates the mote's average power, rho. I started trying to characterize the solar power distribution from solar panel data gathered here at NC State for another research project, but I realized that this is not enough information to answer the implementation question of optimal panel size.

What I will need for that is a characterization of power provided by a panel with a given cost, which will require further research.

Schedule management events drive all the intricate routing behavior of the protocol in the simulation. Every tick, this event executes the action in each mote's schedule. Once each period, this event also clears and randomizes listen and failed advertise slots for the next period. This will insure that each mote has at least one advertisement per period and that motes do not reach a pathological case where two neighbor motes never share and advertise/bid interaction. In addition to following the scheduling algorithm outlined earlier in this paper, entities in this part of the simulation log power used for overhead, according to power use data found by Davis and Miller[8].

Arena's integrated statistics taking mechanisms will provide statistics for time packets stay in each queue and overall throughput and speed as well as other user-defined variables like overhead. Combined with the ability to change the simulation's three environmental variables, these statistics will allow users to evaluate the protocol's performance and adaptability. In addition, the simulation's animation will allow users to visually check how packets move through the network.

During the validation phase of this simulation, I discovered that the network protocol a mica mote sensor network implementation would use already performs link-strength testing. Consequently, the application would not control power dedicated to this task. This does not affect the validity of the model's structure, but it will require some minor changes and some additional research before the simulation can be used to make implementation decisions. Specifically, the model will need a characterization of the power used by link updating so it can deduct this from each mote's rho.

Once the solar and link update issues are resolved, the model must be validated. This process should include comparing the model's output to a small,

controlled prototype network. A validated motel will allow quick initial performance tests of this algorithm under different assumptions. With a validated model, Arena's optimization tools can also be applied to mathematically cumbersome implementation problems like the most cost-effective combination of solar power cells, choice of m and solar sampling rate. The model will randomly assign solar intensities and link strengths at the beginning of each replication. Arena's Output Analyzer uses numerical techniques to search for the optimal average value of some objective function over many replications. While numerical methods like this do not assure that the optimum setup will be found, they only improve the best known implementation setup, and are the only methods available when relationships between variables are very complex.

VI. Extensions

Because the formulation of this protocol does not specify only one base station, multiple base stations or even moving base stations could fit into this protocol. In fact, deploying multiple base stations would be one way to fix the problem of more wolves sending to the same base station than the network can accommodate.

Data collecting motes on the wolves can easily fit into the network structure by running some adaptation of FPS with supply and demand defined as in the new protocol. If suitable hardware is developed, the wolf mote programming could simply extend this protocol. One possible way to do this would be some sort of kinetic generator, like some wrist watches use. Sensor readings could then either be some constant whose power use is deducted from rho for the network protocol or be another state within the mote's schedule. The first would be better suited to wolves moving in and out of network coverage, whereas the second would better fit wolves that are always within range of some relay mote.

As long as sensor motes have finite power reserves, shifting power use from the sensor motes to the relay motes, like having relay motes be responsible for notifying sensors when they are within range, can increase their battery lives. Since this protocol has relay motes advertise regularly, some of this load shifting is already built into it, but further shifting might be possible.

Finally, the assumptions of unidirectionality and no data generation at network nodes can be relaxed if appropriate adaptations are made to the algorithm. Network nodes generating their own data would have to subtract the power this requires from their ρ's and add the data generation per cycle to their demands. Bidirectional communication could be accomplished by passing data inside the confirmation messages of suppliers. If each network node keeps a list of those nodes that contribute to its demand, data could be sent from an arbitrary network node to another in this way. Messages addressed to nodes not on a network node's list would just be forwarded towards the base station. However, this additional traffic towards base stations would need to be accounted for when determining demand and supply.

Works Cited

[1]Pan et al. "Topology Control for Wireless Sensor Networks." Proceedings of MobiCom '03. Sept. 2003. Online. ACM Digital Library. Nov. 2005.

[2]Shu et al. "Power Balanced Coverage-Time Optimization for Clustered Wireless Sensor Networks." Proceedings of MobiHoc '05. May 2005. Online. ACM Digital Library. Nov. 2005.

[3]Xing et al. "Minimum Power Configurations in Wireless Sensor Networks." Proceedings of MobiHoc '05. May 2005. Online. ACM Digital Library. Nov. 2005.

[4]Chang, Jae-Hwan and Leandros Tassiulas. "Maximum Lifetime Routing in Wireless Sensor Networks." IEEE/ACM Transactions on Networking. May 2003. Online. ACM Digital Library. Nov. 2005.

[5]Goel, Samir and Tomasz Imielinski. "Prediction-based Monitoring in Sensor Networks: Taking Lessons from MPEG." ACM SIGCOMM. Online. ACM Digital Library. Nov. 2005.

[6]Kansal et al. "Performance Aware Tasking for Environmentally Powered Sensor Networks." ACM SIGMETRICS Performance Evaluation Review , Proceedings of the joint international conference on Measurement and modeling of computer systems SIGMETRICS 2004/PERFORMANCE 2004. Jun 2004. Online. ACM Digital Library. Nov. 2005.

[7]Hohlt et al. "Flexible Power Scheduling for Sensor Networks" IPSN. Apr 2004. Online. ACM Digital Library. Nov. 2005.

[8]Davis and Miller. 2006.