Economic Models for Resource Sharing in Wireless Networks

Maria Kazandjieva
Mount Holyoke College
makazand@mtholyoke.edu

Margaret Martonosi
Princeton University
mrm@princeton.edu

Abstract

Hardware and software resources in a mobile sensor network are limited and expensive, therefore we need an efficient way of managing their usage. One way to achieve this is by sharing resources among nodes. An economic model where nodes pay for resources they use and receive currency for resources they rent is one way of providing incentive for sharing. In networks where more than one node is ready to provide a resource, we need an effective way to pick the "best" seller. In this paper we present two decision schemes which aim to increase the total efficiency of the system in terms of successfully completed node request. We argue that a system where a node's quality of service and price of service are both factored into the final decision achieves best results. We test our hypothesis by running simulated experiments in software.

1 Introduction

Wireless sensor networks are becoming increasingly widespread in various contexts. For example, sensor nodes have been deployed on wildlife for tracking purposes [4], and could be used on vehicles for exchange of traffic information [8]. The nature of sensor networks introduces several challenges, including limited battery life, frequent network disconnections, and limited resources. Similar issues arise in peer-to-peer (P2P) networks where users join the network, request resources, use them and then leave. In this case, we are not pressed by the energy-efficiency problem, but we still have to consider the instability of the network in terms of dis- and reappearing peers. The ultimate goal in both sensor networks and P2P network is the efficient use of resources and transmission of data.

The problem we address is that of managing resources in a sensor network. Similar work has been done on P2P systems in [7]. In their work Turner and Ross point out that nodes often do not have an incentive to participate in resource sharing. Therefore they propose an economic market as a possible motivation; in this market nodes can offer and request resources in exchange for some monetary value. We take this idea and adapt it to the sensor network context. In our model there is a universal payment currency and each node receives the same initial budget. Sensor nodes request resources without specifying the "seller" they would like to buy it from. In a moderately dense network, on many occasions several potential sellers are available to serve the same request. Consequently, the main focus of this work is the design and implementation of algorithms for optimal decision between sellers.

Overall, the contributions of this work are the following:

- We built a software simulator, Atrium, which mimics the interactions between nodes in a wireless network. Atrium allows for the evaluation of node and system behavior.

- We proposed and implemented two algorithms for decision -making when a client node requests services from a seller node.

- We achieved an average improvement of 10% over the baseline case in terms of percentage of completed requests in the system.

The remainder of this paper is organized as follows. Section 2 discusses some related work. Section 3 presents the software implementation of the Atrium simulator. Section 4 presents the baseline and newly proposed decision algorithms. Section 5 presents our experimental results. Finally, Section 6 presents our conclusions and future work.

Event type	Description	Event Form	Example
move	Specifies the time and coordinates to which a specific node has moved	<nodeID> <time> move <x> <y>	6 41 move 3 3
failure	Specifies that a certain resource on a node is experiencing failure; the event determines how long the failure will last	<nodeID> <time> failure <resource> <length of failure>	10 49 failure radio 9
charge	Specify the time at which a node's battery is fully recharged	<nodeID> <time> charge	4 390 charge
request	Specifies what resource and for long a node would like to request	<nodeID> <time> request <resource> <length of request>	14 462 request radio 163
rate	Specifies the time at which *all* nodes' ratings are updated	0 <time> rate	0 50000 rate

Figure 1: Types of events for the Atrium simulator

2 Related Work

Several projects have looked at using economic models for managing of resources in both sensor and P2P networks [8] [7] [2] [6]. They all agree on the basics of such models - the existence of currency and budget allocation.[2] proposes a model in which nodes pay not for the usage of a specific resource but rather a combination of resources, needed for the completion of a single task.

The auction model is discussed in [1] and [2]. These works present the idea that in the case of several potential buyers of a resource and one seller, bidding is a good way to determine the winning client. They however do not discuss what happens if more than one seller are available. In [8] it is assumed that there is a way of dealing with interferences and conflicts. We can look at the multiple available resources as a type of conflict and use one of the decision schemes described in our work to solve it.

A good discussion of why economic models will be a useful approach to the problem at hand is given in [3]. The main point being that in a sensor network the decentralized allocation of resources is more efficient and desirable than a centralized approach.

The issue of determining the pricing of resources is not addressed directly in many of the related works. In [8] and [9] a relevance function is proposed. Based on this function a node can determine the price of a resource. These papers do not address the issue of dynamic price changes depending on the energy status of a node. On the other hand, [5] touches on the subject by proposing that price of a service needs to be directly related to its energy cost. We use a similar idea in our work and implement a pricing scheme in which prices reflect the energy budget of the seller.

3 Implementation

3.1 Atrium

Atrium is a software simulator implemented in Java. It creates a distributed network of nodes, where the size of the network and the number of nodes are configurable. Experiments ran for the purposes of this project were run in an environment of 20 nodes, spread out randomly in a 15 by 15 grid.

Each node in the simulated system owns one type of resource, say a radio transmitter. In addition each node begins the experiment with a budget allocation of 100, full battery, and a rating of 10.0. As the experiment progresses and nodes interact with each other, the characteristics of nodes change.

Atrium takes in a file that specifies the characteristics of each node. These include price of node resources, energy level, radio range, initial budget, and initial node rating. In addition Atrium reads in a trace file with a list of events. The events that are currently supported are movement of node, request for resource, battery charge, node rating, and node failure. The

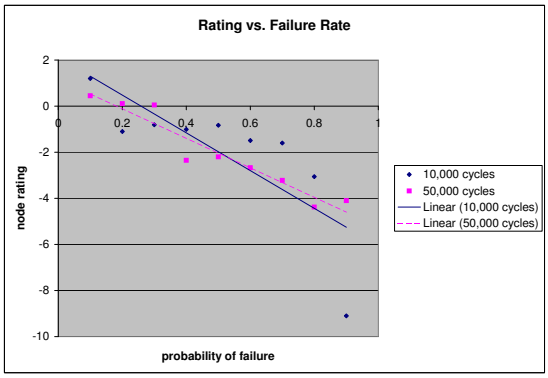

Figure 2: Correlation between failure rates and node ratings

table in Figure 1 summarizes the each of these events.

The simulator processes the list of events, updating node attributes and keeping statistics about the network's performance.

Some of the more important features of the Atrium are described below.

3.2 Dynamic Pricing

In order to take into account the energy-constrained environment of wireless sensor networks, we devise a scheme in which resource prices are adjusted based on energy. In particular, for every percent drop in battery, the price of node's resources goes up by half a percent. In this way when the nodes in the network start running low energy, they become more valuable and pricing reflects that. In Section 5.3 we look at some experimental data of how prices change under different decision schemes.

3.3 Node Rating Scheme

In order to reward nodes that are reliable and actively service requests, and to punish nodes that often fail, we have developed an "e-bay" style rating system. Nodes are judged on the number of accepted and the number of successfully completed requests. This is reflected in the formula below:

$$value = AcceptedReq/ReceivedReq - FailedReq/AcceptedReq$$

If the number of request the seller has received and/or accepted is 0, then the node is punished for not participating actively in the system and a constant 0.5 points are deducted from its rating. In all other cases, the above formula takes into account the willingness of a node to accept a request as well as the success rate of completing requests that have already been accepted.

The updated rating of a node is a weighted average of the current and the newly computed values. In our experiments we used 0.75 weight for the current node rating and 0.25 for the new one, but these parameters can be easily changed.

In order to evaluate our rating scheme we ran experiments with nodes whose failure rates varied from 10% to 90%. For each of those points we observed the end node rating. We ran the experiment twice - once for 10 000 time cycles and the second time for 50 000 time cycles. As expected the results were more stable for the second run, because nodes had more chances for interaction with each other. Ideally, there would be a linear correlation between failure rate and rating. Due to the random movement of nodes throughout the grid, however, we get certain indeterminism. In other words, not all nodes have equal chances of selling their resources because of their physical location. Therefore we cannot expect a perfect correlation between rating and quality, but we can still use the rating to predict the likelihood of a node not completing its task.

Figure 2 shows the results we obtained form the two runs. We can see that in both cases, the expected trend is preserved and nodes with higher failure rates received lower ratings. There is one major outlier in the 10000 time cycles run, and this is attributed to the fact that ratings did not have enough time to stabilize. The outlier is missing in the 50000-cycle run.

4 Decision Algorithms

Since the main question we want to answer with this project is, *What would be a good way to decide between several potential seller nodes offering the same resource?* we propose two different decision schemes. Using Atrium we were able to test the two algorithms and compare them to a baseline one. Below we describe all three of them.

4.1 Baseline Algorithm

In the baseline algorithm, a client node that is requesting resources always chooses the seller node with the lowest ID. This ensures that the decision scheme is completely oblivious to any characteristics that might differentiate potential sellers. An alternative scheme that might be used as a baseline is random choice of seller.

4.2 Price-Only Algorithm

Exploring the economic idea that cheap goods will allow for greater purchasing power, we devised an algorithm in which nodes always pick the resource seller with the lowest price. Intuitively this is a reasonable choice because spending less money on each transaction in the system will allow nodes to complete a greater number of transactions.

We predict that such an algorithm will perform very well in systems where cheap nodes have high availability. Unfortunately, the likelihood of operating in such a network is low, since quality usually comes at a greater price. An example of a network in which the price - only algorithm will not perform optimally is one where low-priced nodes have high rates of failure. It is possible to have a situation in which a node has price five to ten times lower than others, but also exhibit a rate of failure five to ten time higher. In this case, the price-only scheme will often pick nodes that are cheap but unreliable and will thus decrease the number of successfully completed resource requests.

Judging from real life economic models, we would expect networks of nodes where quality costs more and low prices are associated with suboptimal quality of service. For this reason we propose a scheme that takes into account not only the price of a resource but also the quality rating of the seller offering the resource.

4.3 Adaptive Algorithm

In the adaptive algorithm we consider both the resource price and the overall rating of the seller node. This way, we are able to make a better judgment of how worthy a node is. A simple way to convey this idea would be to take the ratio of price over rating and get a measure of currency per rating point. Even though this approach guarantees us to always make economically sound choices, it also has drawbacks. First, the rating of devices can only partially give us an idea of how reliable they are. Ratings are based on a nodes history of transactions with its neighbors and cannot always predict the future accurately. In addition, a ratio approach does not allow nodes to choose the importance of price and rating. For example, a node with a very high budget might want to purchase resources from the neighbor with highest rating without regard for price. Later, that same node, having had its budged drop drastically, might decide that it would go for a seller node of lower quality of affordable price.

In order to deal with the limitations of a simple price-rating ratio scheme, we propose an adaptive algorithm. The variables that go in the algorithm are the following: client node's initial and current budget allowance, and seller node's price and rating. We call the overall score of a seller s, where $s = \alpha * rating - \beta * price$. From this equation you can see that a very high score will be achieved by a node with high rating and low price, while a seller with a high price but low rating will receive a low overall score.

Below we outline the steps through which the adaptive algorithm determines the values of α and β.

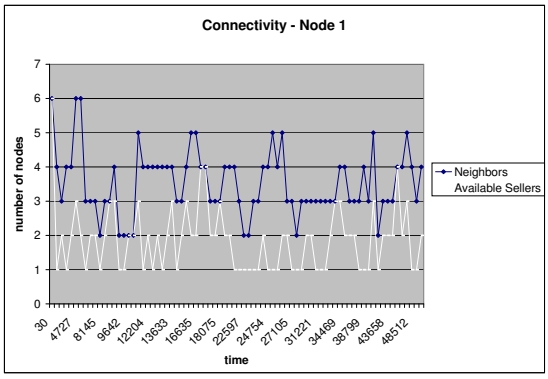

Figure 3: Connectivity information for a node in the network.

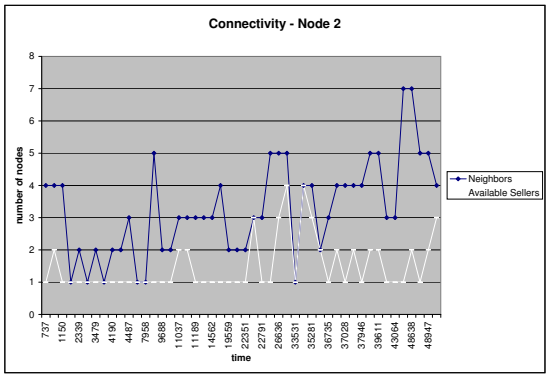

Figure 4: Connectivity information for a node in the network.

- If clients current budget is equal to initial budget, then seller price and seller rating receive equal importance:

  $\alpha = 0.5$

  $\beta = 0.5$

- Else:

  $\alpha = 0.5 + ((client.CurrentBudget() - client.InitBudget())/10.0) * 0.05$

  $\beta = 1 - \alpha$

The intuitive thought behind this scheme is that a rich node is willing to pay more for quality. As the wealth of a client node increases, the importance of price decreases. On the other hand, if a node is running low on currency, then it is willing to sacrifice quality of transaction and pay an affordable price.

5 Results and Observations

This section presents our experiment results and discusses some observations made during the experiments.

5.1 Network Connectivity

Before presenting a comparison between the baseline, price-only, and adaptive decision schemes, we discuss the effect of connectivity on those. After several experiments we discovered that in a sparse network all algorithms perform nearly the same. For example, some early experiment were ran in a simulated network of size 25 by 25. We randomly placed 10 nodes, each with range of 5, and tested the three decision schemes. In the experiments the percent of successfully completed transactions was 21.77%, 20.88%, and 21.4% for the baseline, price-only, and adaptive schemes respectfully. The lack of difference between different experiments is due to the low density and hence connectivity in the network. On most occasions when a node wants to request a resource from a neighbor, there is only one available seller. Therefore, the decision schemes do not come into use at all.

For the remainder of experiments and observations we worked with a much denser network. It consists of 20 nodes in a 15 by 15 grid. As before, each node had a radio range of 5. In order to get a visual idea of what kind of density this setup gives us, we present the connectivity graphs of 2 nodes - Figure 4 and Figure

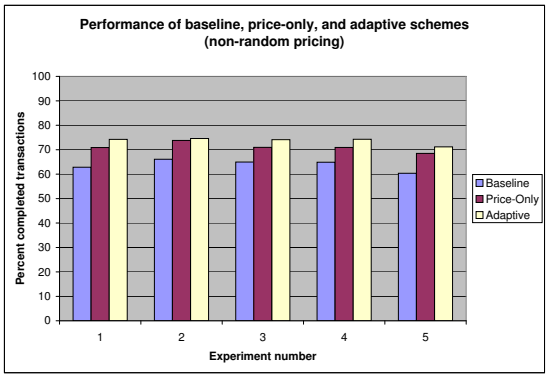

Figure 5: Performance of algorithms. Non-random pricing

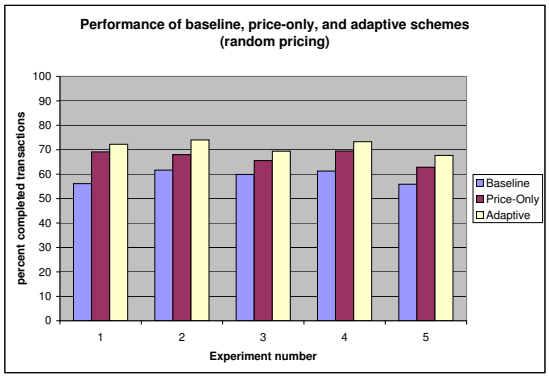

Figure 6: Performance of algorithms. Random pricing

5. Each data point on the upper curve represents the number of neighbors the node had in range at a request time. The lower line indicates the number of available sellers at request time. The difference between the two lines comes from the fact that often nodes are in range, but have unavailable resources.

5.2 Success rate of Algorithms

The main metric used in the evaluation of price-only and adaptive algorithm is the percentage of completed transactions between two nodes, for the entire system. Each tracefiles includes about 1000 request events. On average, the adaptive decision scheme increases the percent of completed request by 10% over the baseline case.

We ran two sets of experiments. In the first set prices of resources were determined randomly and did not in any way reflect the reliability of the node. In the second set of experiments, we chose prices that reflect node failure rates. We expected to see overall better performance of the adaptive algorithm, and higher percentage of completed requests in the second set of experiments.

Figures 6 and 7 show the results from both experiment sets. We can see that the price-only and adaptive decision schemes constantly perform better than the baseline algorithm. In addition, the adaptive runs outperform the price-only. Therefore we can conclude that in a moderately dense network, where prices reflect quality, using a smart algorithm to choose between potential resource sellers increases the overall performance of the system.

5.3 Price changes

It is interesting to observe what effect the decision schemes have on the dynamically changing resource pricing. As described in Section 3.2 a seller adjusts its price based on its energy budget. Figure 3 presents a timeline of these changes for one node. We see that when Atrium is ran with the adaptive scheme the peak prices are lower than those in the baseline case. This means that under the adaptive algorithm prices are more stable and remain more affordable to potential client nodes. Similar observations were made for other nodes.

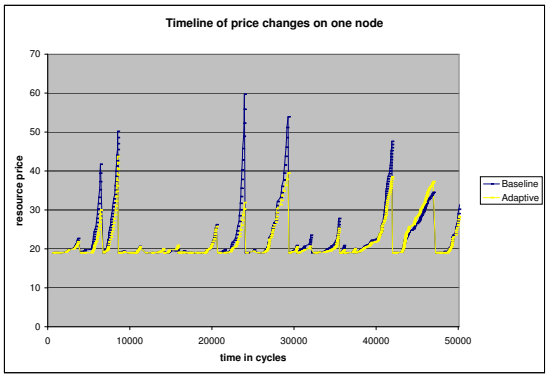

Figure 7: Dynamic change of price for one node under the baseline and adaptive schems

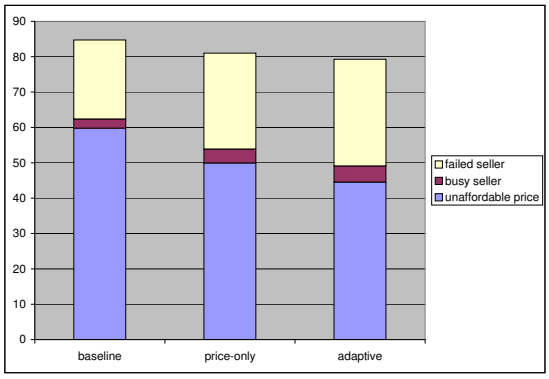

Figure 8: Distribution of reasons for seller rejection for all three schemes

5.4 Seller Rejection

In addition to discussing the success rates of different algorithms we also explore the various reasons seller nodes are rejected during our experiments. In some situations a request is never completed successfully because the client node has rejected all sellers in range. A deeper understanding of why sellers are unable to accept a request will give an idea of what aspects of individual nodes or the system as a whole need to change in order to achieve improved performance.

We identify three reasons for a rejection of seller:

- unaffordable price - the client node did not have enough currency to pay for the seller's resource
- busy seller - the resource is already being used by a different client
- failed seller - the node's resource is not functioning properly

In Figure 8 we see the percentage of rejections of each type for all three decision algorithms. We observe that overall the number of rejections is lowest for the adaptive scheme. This result is with agreement with the fact that the adaptive scheme has the most successful request completions. In addition, the number of times sellers are polled for their resources is minimized in the adaptive algorithm. In one experiment, the adaptive algorithm run considered about 9400 sellers, while the price-only polled 10000 and the baseline close to 11000. The conclusion is that with the adaptive algorithm we not only increase out success rate but also minimize the number of wasteful interactions between nodes.

Looking at the breakdown of seller rejections based on type, we can see that in all three cases the largest percentage was due to an unaffordable price on the seller's side. They account for about 45%, 50%, and 60% of all rejections for the adaptive, price-only, and baseline algorithms respectfully. Busy sellers account for a small percentage of rejections, with this percentage being a bit higher in the adaptive runs. The intuitive reason for this is that since more requests are getting completed, more sellers are busy when clients contact them. Failures among nodes' resources are the other big reason for uncompleted requests.

These results prompt us to consider ways of improving the system. It would be interesting to see what experiments would show if all all nodes

had increased budgets. Ideally there would be a way of determining how much money each node is allowed to spend on transactions based on its importance and energy profile.

Even though the graph does not indicate this, in addition to having a seller rejected because its resource has failed, a seller might interrupt a transaction that has already started. This would happen if a seller has accepted a request, lend its resource for a certain number of time cycles, but experienced a failure in the meantime. Experiments show that the adaptive decision scheme minimizes the number of such interruptions.

6 Conclusion and Future Work

This paper has presented the implementation of Atrium, a software simulator used to investigate resource sharing algorithms. We have also introduced two schemes for choosing between sellers offering the same resource. We have showed that a careful consideration of both node's quality and price performs better than a baseline scheme. In the work leading to our final results we also developed an "e-bay" style node rating scheme, as well as a system for dynamic adjustment of resource prices.

Future work might include testing of the proposed algorithms on a real sensor network. In addition, we would like to investigate the energy implications of using different schemes. In particular, it would be interesting to see whether a simple scheme, such as the price-only one, might be more efficient in terms of overhead compared to the more sophisticated adaptive scheme.

Taking this project one step further, it will be useful to see how the proposed algorithms need to be adapted to a system with multiple resources and requests for combination of resources.

Acknowledgements

This work was supported by the CRA-W Distribute Mentor Project.

References

[1] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing, 2002.

[2] B. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. Parkes, J. Shneidman, A. Snoeren, and A. Vahdat. Mirage: A microeconomic resource allocation system for sensornet testbeds, 2005.

[3] D. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic models for allocating resources in computer systems, 1996.

[4] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *ASPLOS, San Jose, CA*, October 2002.

[5] C. Saraydar, N. Mandayam, and D. Goodman. Efficient power control via pricing in wireless data networks, 2002.

[6] V. Siris, B. Briscoe, and D. Songhurst. Economic model for resource control in wireless networks, 2002.

[7] David A. Turner and Keith W. Ross. A lightweight currency paradigm for the P2P resource market, 2003.

[8] O. Wolfson, B. Xu, and A. P. Sistla. An economic model for resource exchange in mobile peer to peer networks. In *SSDBM*, 2004.

[9] B. Xu, A. Ouksel, and O. Wolfson. Opportunistic resource exchange in inter-vehicle ad hoc netowrks. In *MDM 2004*, January.