Evolving Game Strategies to Minimize Power Consumption in Agent Based Multi-Hop Wireless Networks

Muthukumar Udaiyanathan
Dept. of Industrial &Manufacturing Engineering
Pennsylvania State University
State College, PA 16802

Aditya Kaul
Dept. of Industrial &Manufacturing Engineering
Pennsylvania State University
State College, PA 16802

Abstract

In this paper, each node in a multi-hop wireless network is modeled as an agent. The nodes utilize their resources for both processing information, sending their own packets and for forwarding packets of other nodes. Though each node is competitive and aims to conserve its resources, cooperation is essential for the network to function. An Iterated Prisoner's Dilemma game has been used to help the nodes make decisions regarding utilization of resources. The strategies used by the nodes to make these decisions are evolved to improve transmission efficiencies, while conserving the node resources. The initial results obtained show that evolving these game strategies could bring in coordination within the network.

1 INTRODUCTION

The multi-hop wireless network consists of nodes that communicate with each other by tranferring packets to intermediary nodes until the packets reach the final destination. The nodes represent the processing elements such as computers or hand-held devices. The arcs represent the ability to communicate between these nodes. These geographically distributed nodes can be stationary or in motion. The main difference between a multi-hop network and other communication networks is that there is no centralized controller or access point that accepts data packets from the sender nodes and forwards them to the destination nodes. Each node has limited resources in terms of bandwidth and power. The nodes may be involved in sending and/or receiving packets at the same time.

Thus in order to communicate with a destination node the sender is dependant on other intermediary nodes to receive and forward its packets. At higher levels, the problems deal with routing and quality of service. The primary concern at this level for any node, is to conserve power while performing its tasks. Each node expends power in performing its internal functions, sending out or receiving its packets and fowarding the packets that were generated by other nodes. Thus each node has to decide how much power it is willing to expend in order to forward packets generated from other nodes. In other words, how many packets of data is a node willing to receive and forward even though these packets are not relevant to itself. Though a particular node can use power exclusively to process and send its own packets, the network as a whole would fail unless the node cooperates. In the absence of a centralized controller it is difficult to coordinate the communication process between nodes.

Each node has two functions to perform - One is to process information and transmit its own packets, the other is to forward packets of other nodes. The first functions ensures the proper functioning of the node locally and the latter is necessary for the network as a whole to function effectively. The resources available at each node are finite and the nodes therefore have to balance the alloctation of resources for these two competing functions.

One of the approaches to resolve these conflicting interests is to let each node follow a set of strategies or actions. In this paper the nodes are represented as multi-agents and are equipped with a set of initial random strategies. The agents make the decision of accepting packets from other nodes based on these strategies. These strategies can be evolved to improve them at each generation. The strategies of individual agents or nodes are evolved in such a way that both power conservation at a local node is minimized and the transmission efficiency across the network is improved. The experimental results obtained show that evolving these strategies lead to an emergence of unplanned coordination among the nodes. This minimizes the power consumed across the nodes while maintaining efficiency of communication.

It must be noted that the results presented in this paper are based on the analysis done on a static network. The size of packets exchanged between the different nodes is assumed to be equal. The power expended in sending or receiving a packet is assumed to be proportional to the distance between the nodes. This is an ongoing study during the course of which we will eliminate these assumptions and continue the experiments so as to develop a more generic schema.

2 AGENT BASED NETWORK MODEL

The model used considers each node to be an autonomous software agent. Each agent has a set of strategies that are randomly initialized. The agents use these strategies for both, choosing partner nodes to whom they send their packets and to decide the partner nodes whose packets they will accept and forward. The strategies followed by the agents are different and so the network consists of a set of heterogeneous agents. This reflects the state that the nodes would be in real world networks.

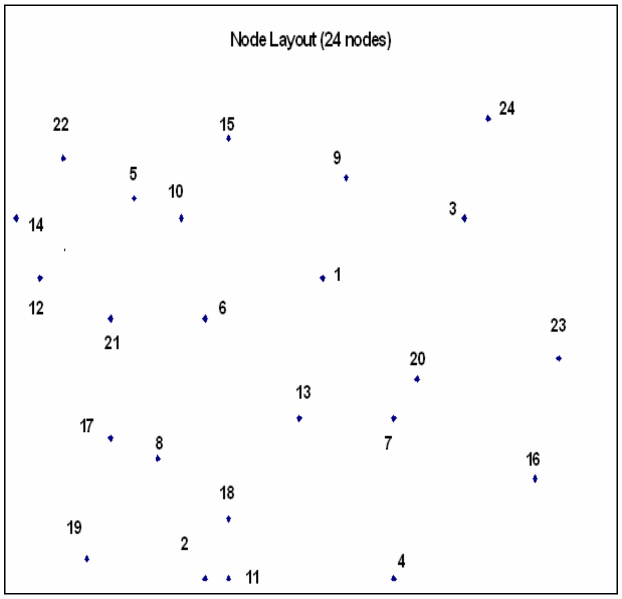

Figure 1: Layout of a 24-node network

Whenever a node needs to send out a packet of data to a destination node outside its range, it makes a request to other nodes to forward its packets. The requests are made only to those nodes, such that the power expended by the sending node is less than a prefixed threshold value, which is the *tolerable* range for a given node. When the request sent out from a node is rejected it then resends a request to another *tolerable* node. The node continues to make these requests until all *tolerable* nodes have rejected its requests or all its requests have been accepted. Similarly a node, that receives requests for forwarding packets, rejects a request unless the power it will need to expend is within the *tolerable* range. Upon receiving the requests the node makes a list of all *tolerable* requests in decreasing order of preference. The current most preferred request is replaced by a new request, which is better than the current best.

Once the potential partners have been chosen, the receiver nodes need to decide the sender nodes from which they will accept and forward packets. This decision is made through a two-player prisoner's dilemma (PD) game. Nodes that have accepted the requests play a PD game with each of the nodes, which fall within its *tolerable* threshold, starting with the most preferred node. During the game the nodes either decide to co-operate or defect. These decisions are made based on their expected payoffs and the set of strategies that they currently follow. The nodes record the previous moves of the other nodes to make subsequent decisions. Once the games are completed and the nodes reach an agreement, the packets are exchanged.

The Model explained above has been implemented with the *SimBioSys* platform developed by (McFadzean, 1995) The pseudo code for the above-described model is shown below in figure 2. The detailed implementation of the model using the *SimBioSys* platform is described in detail in section 2.1.

```
int main () {
Init();                                   // Construct initial node generation
                                          // with random game strategies.
For (G = 0, . . . , Max_Gen – 1) {        // Enter the generation cycle loop.

// Generation Cycle:
InitGen();                                // Configure nodes with user-supplied
                                          // parameter values (initial expected
                                          // payoff levels, resource quotas, . . . ).
For (I = 0, . . . , Max_Game – 1) {               // Enter the game cycle loop.

// Game Cycle:
MatchNodes();                             // Determine game partners,
                                          // given expected payoffs

PlayGame();                                       // Implement games and
                                          // record trade payoffs.
UpdateExp();                              // Update expected payoffs
}                                         // using newly recorded payoffs.

// Environmental Step:
AssessFitness();                          // Assess and output node information.
Dump();                                   // Output fitness statistics for the
                                          // current node generation.

// Evolution Step: Evolve a new node generation.
EvolveGen();
}
Return 0;
}
```

Figure 2: Psuedo Code for Model Implementation

2.1 MODEL IMPLEMENTATION

The simulation starts by initializing a network. The nodes are first defined and are given a random location within the X-Y plane and the corresponding x and y coordinates are stored. At a given instant of time a node is either a sender or a receiver or both. The initial matching of nodes is done based on the initial expected payoff of each node. At the end of this initial matching the receiver nodes have a list of sender nodes whose requests they find tolerable. They maintain this list in the decreasing order of preference.

After the matching is complete, each receiver node starts playing a prisoner's dilemma game with the sender nodes. The decisions of whether to cooperate or to defect are made based on the set of strategies that the node follows and the previous move made by the opponent. The strategy for a given node is represented in the form of a finite state machine explained in 2.1.1. For each combination of decisions made by the nodes they get an associated payoff. The payoff is computed as a linear sum of the effects of loss in power, successful transmission and a trust factor (see section 2.1.2). At the end of a cycle of games the payoffs of all the nodes are updated.

The nodes are listed in the order of decreasing payoffs. An elite fraction of this list is chosen which are carried on to the next generation. These nodes act as parents and through mutation and crossover they produce children. These child nodes replace the lower one-third nodes in the list. Thus only the strategies that were successful at the end of a cycle of games are allowed to carry on to the subsequent generation of nodes. The strategies are evolved until a pre-specified number of generations.

2.1.1 Strategy – Finite State Machine

The strategy followed by a particular node is represented as a finite state machine (FSM). The state of the node captures the current conditions the node faces, such as the amount of power remaining and the current processor load. The number of states that a node may be in is infinite in reality. This paper assumes that this is a finite quantity. The input given to the node is the previous action of the opponent. Based on this and the current state the node takes a particular action of cooperating or defecting, which may cause a change in state. Thus throughout the game the node keeps hopping between these finite states. Figure 3 shown below depicts a three state FSM. Depending on the previous move made by the opponent the agent takes an action to defect or to cooperate. Once this action is taken the agent experiences a change in state. Therefore the agent keeps transitioning between the three states that this FSM captures.

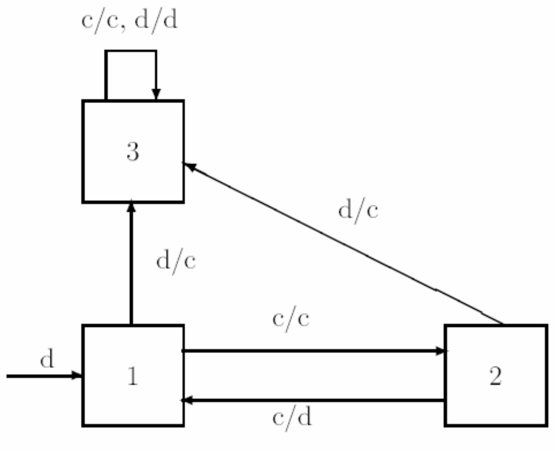

Figure 3 : Representation of a Three State FSM

The initial input or previous action of the opponent given to the agent is a *defect* and the agent now starts at state 1. Each arrow shown within the figure is a transition to the next state. The decisions of the opponent and the action taken are represented along the arrow [opponents action/ agents action]. In this case if the opponent had cooperated then the agent decides to cooperate and it moves to state two. On the other hand if the opponent had defected it would still cooperate and move to state three. It can be seen that the agent keeps taking actions and transitioning between the states. The set of actions and state transitions represent the strategies of an agent.

The finite state machine associated with each node is randomly initialized. The number of states is fixed at the beginning of the simulation. The FSM is represented as a series of binary digits. The FSM implementation uses a lookup table, which is indexed by the input and the current state, each index corresponds to a certain output and the subsequent state. Thus the node uses the lookup table to determine its next move and the corresponding state transition.

2.1.2 Payoff Function

The payoff that a particular node gets is determined by the outcome of its actions during the game. The payoff is computed as a linear sum of the following three factors.

<u>Successful Transmission</u> – Every time a sender node is able to successfully transmit a packet it gets a positive payoff of one unit.

<u>Power expended</u> – Each node gets a negative payoff proportional to the power that it would expend if it takes a particular action. Earlier in the paper it was mentioned that the power expended is assumed to be proportional to the distance between the nodes, therefore the power expended is computed as a function of this distance.

<u>Trust Factor</u> – Each time an opponent decides to cooperate with a node, it assumes that the opponent might do the same in future and so receives a positive payoff of one unit.

The total payoff received is the linear sum of these three elements. The individual payoffs that the nodes receive for different combination of actions are shown in Table 1.

2.1.3 Evolving the Strategies

When a cycle of games is completed the binary string representation of the FSM is evolved using mutation and crossover operators. Based on the payoffs received by the nodes during the cycle of games the nodes are arranged in decreasing order of payoff, thus keeping the best or most successful strategies on the top of the list.

The best two-third strategies are then selected. In other words the elite percentage for selection is kept constant at 2/3, throughout the simulation. These are then paired to mate. Two points x and y are chosen randomly in each pair. The bits x through y of one of the parents in each pair replace the bits x through y of the other parent, thus producing one offspring per pair. The offspring generated then undergoes mutation. Each bit in the offspring is flipped with a pre-specified mutation probability. Following these operations the best two third strategies along with the newly generated one-third offspring strategies enter the next generation. Since the number of strategies equals the number of nodes/agents, the individual agent strategies are evolved from generation to generation. The elitism, recombination, mutation operations on the current generation strategies have the following effects: Elitism preserves successful actions, recombination rearranges the blocks of successful actions with little disruption and mutation changes individual actions. Thus the behavioral effects of these genetic operations are that successful strategies are mimicked, unsuccessful strategies are discarded and new strategies are injected for experimentation.

2.2 VARIABLES AND NOTATION

N_i	: Node i in the network
$D_{i,j}$	: Distance between node I and node j
P_i	: Power available at node I
K	: Power expended in transmitting one data packet over unit distance
T_i	:Transmission payoff =1 if a packet is transmitted by node I = 0 otherwise
$F_{i,j}$	: Trust factor = 1 if node j cooperates with node i = -1 otherwise
TP_i	: Total Payoff for node i following a transaction between nodes i & j

$$\mathbf{TP_i = T_i + F_{i,j} + P_i - K{*}D_{i,j}}$$

Max_Gen : Total number of generations for which the nodes are evolved

Max_Game: Maximum number of game cycles in a generation

μ : Mutation rate

FSM_States : Number of internal states in the FSM

FSM_Mem: Number of memory bits in the FSM allocated to the previous move

State_Size : The number of bits used to represent a state

FSM_Size : Total number of bits representing an FSM $2^{(\text{State\_Size}+\text{FSM\_Mem})}$

Agent_Count: Total number of nodes in the network

Elite_Frac : Fraction of the subpopulation that is considered elite

Sen_Quota :The quota of resources (bandwidth, channels, Power) available at a sender node

Rec_Quota :The quota of resources (bandwidth, channels, Power) available at a receiver node

$\text{Init\_Exp\_payoff}_{i,j}$: Initial expected payoff of an agent i when playing with agent j

	$(\text{Cooperate})_j$	$(\text{Defect})_j$
$(\text{Cooperate})_i$	[(Ti + 1) + Fij +(Pi – K*Dij)], [(Tj +(Fji + 1) + (Pj – K*Dij)]	[Ti + Fij – 1 + Pi], [Tj + (Fji + 1) + Pj]
$(\text{Defect})_i$	[Ti + (Fij – 1) + Pj], [Tj + (Fji – 1) + Pj]	[Ti + (Fij – 1) + Pi], [Tj + (Fji – 1) + Pj]

Table 1: Payoff Matrix of Prisoners Dilemma Game

3 EXPERIMENTAL RESULTS AND ANALYSIS

The above model was implemented on the SimBioSys platform. The experimental simulations were done with Sen_Quota = 5 and Rec_Quota = 2, Elite_frac = 2/3, μ = 0.005, $\text{Init\_Exp\_payoff}_{i,j} = 1$.

The performance of the network is measured as the average fitness of all the nodes in a particular generation where fitness is the total payoffs received by the nodes. The payoff function captures the number of successful transmissions and is negatively correlated to the power expended. Therefore an increased fitness shows that more transmissions are made while lesser power is consumed. Moreover this increase in overall fitness over subsequent generations validates the utility of this model. The network reaches a steady state behavior after a considerable number of generations and the overall fitness shows minimal variation. One of the key factors that affect the time taken to reach a steady state behavior for the network is the FSM_Size. Figures 4 & 5 show the variation in fitness of the nodes for FSM_Size = 8 and 64.

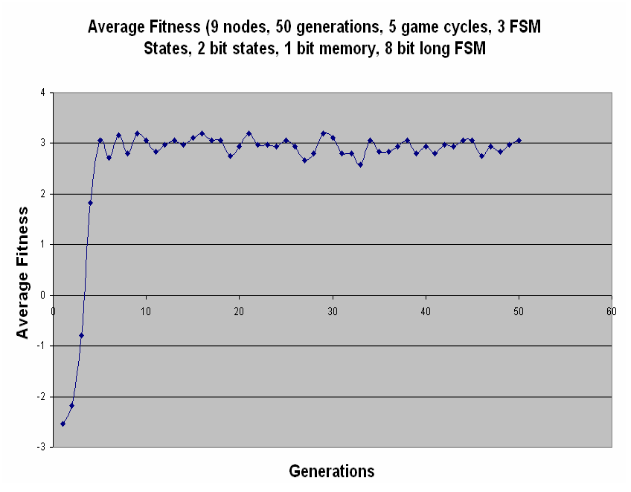

Figure 4

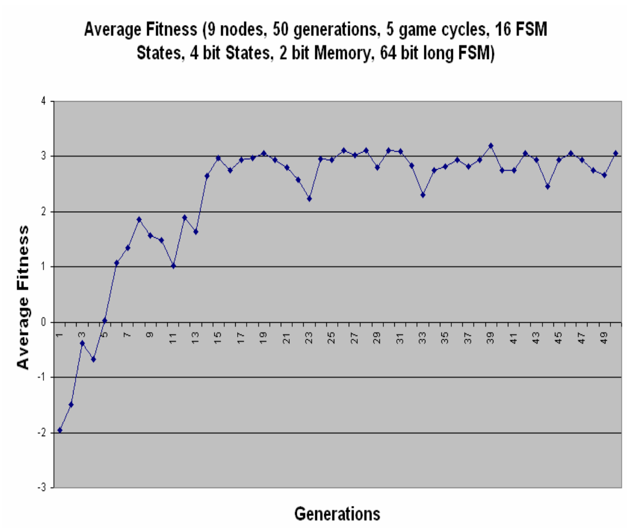

Figure 5

It can be seen that, larger the FSM_Size the more number of generations that the system spends before achieving a steady state behavior. Once a steady state is reached the amount of variance in fitness is greater for a larger value of FSM_Size.

When the number of nodes within the network is increased the algorithm still manages to reach steady state and improve the overall fitness. The actual numerical values of fitness are seen to be more in the case of fewer nodes because the total power consumed is less compared to when more nodes successfully send packets. This is show in Figure 6.

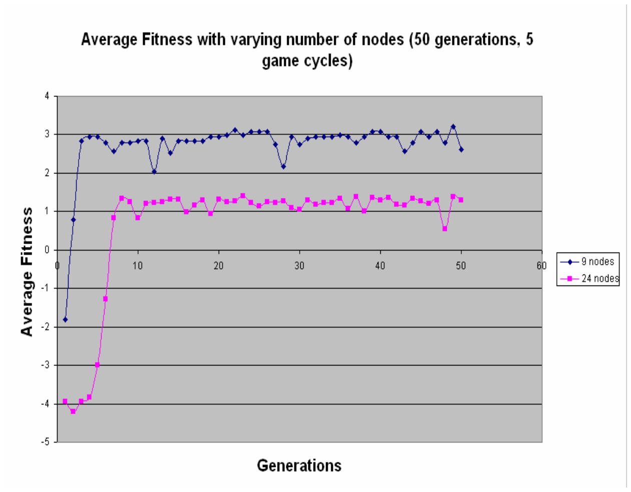

Figure 6

The effect of mutation rate (Figure 7) on the evolution of the behavior of the nodes was studied and the results clearly show that very low values of mutation (< 0.005) show greater efficiency. We recommend this value for mutation rate when conducting these experiments.

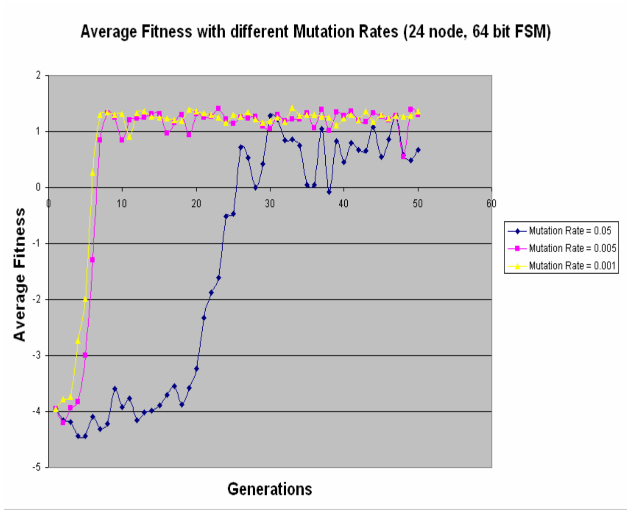

Figure 7

The effect of ignoring the power consumed at each node was studied and the results are shown in the Figure 8. It is evident that when we ignore power consumption most nodes within the network do not achieve good fitness values and so the average fitness for the nodes is lesser than when power consumption is considered. This shows the importance of considering the power consumed for our wireless network model. It results in improvement of the global performance. At the same time since selfish motives are included within the payoff function power is conserved at individual nodes.

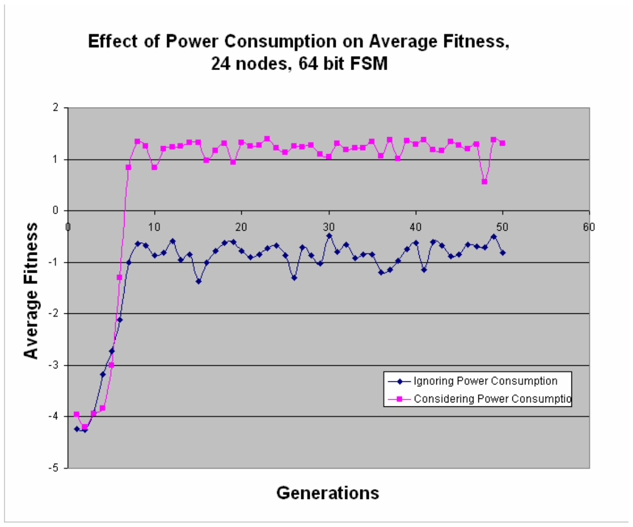

Figure 8

4 CONCLUSIONS

Treating the nodes within the multi-hop network as autonomous agents enables them to function in the absence of a centralized controller. Modeling the interaction as a prisoner's dilemma game resolves the conflicting interests of allocating resources for information processing and forwarding the packets of other nodes. The results of the study show that though decisions are made by nodes in an attempt to conserve their individual resources, the network as a whole also functions efficiently. This could be due to an emergent unplanned coordination. Further analysis will be performed to investigate the same. The results obtained show that there is improvement in transmission efficiency and reduction in the amount of power expended across the network. These conclusions have been validated by appropriately defining the payoff and fitness functions. The results clearly indicate that the average fitness of the nodes increases and the system achieves a steady state behavior through implementation of this network game model.

Acknowledgments

We are grateful to Dr. Patrick Reed for his guidance and constructive criticism during the course of this study. We look forward to his continued support as we carry on this study to the next level.

References

Axelrod.R (1987), The evolution of strategies in the Iterated Prisoners Dilemma, *Genetic Algorithms and Simulated Annealing*, Morgan Kaufman, Los Altos, 32-41

Fogel.D (1993), Evolving behaviors in the Iterated Prisoners Dilemma, *Evolutionary Computation*, vol 1, no. 1, 77-97

Ishibuchi, H.; Nakari, T.; Nakashima, T.(1999), Evolution of neighborly relations in a spatial IPD game with cooperative players and hostile players, *Evolutionary Computation,* 1999. CEC 99. Proceedings of the 1999 Congress Vol 2, 929- 936

Ishibuchi, H.; Sakamoto, R.; Nakashima, T.;(2001), Evolution of unplanned coordination in a market selection game*, Evolutionary Computation*, IEEE Transactions, Volume 5, Issue: 5, Oct. 2001, Pages: 524 - 534

McFadzean, David; Tesfatsion, Leigh, (1999), A C++ Platform for the Evolution of Trade Networks, *Computational Economics*, Volume 14, Issue 1-2 October 1999, 109 – 134

Monks, J.P.; Ebert, J.-P.; Wolisz, A.; Hwu, W.W. (2001) A study of the energy saving and capacity improvement potential of power control in multi-hop wireless networks *Local Computer Networks*, 2001. Proceedings. 26th Annual IEEE Conference, 550 - 559

Song Guo; Yang, O, (2003), Minimum-energy broadcast routing in wireless multi-hop networks**,** *Performance, Computing, and Communications Conference*, 2003. Conference Proceedings of the 2003 IEEE International, 273 – 280

Sun Xuebin; Zhou Zheng, (2003), Energy conservation and routing efficiency tradeoff in multi-hop wireless ad hoc networks**,** *Communication Technology Proceedings*, 2003, vol.2, 1278 - 1281