International Journal of Computer Science and Engineering (IJCSE)
ISSN 2278-9960
Vol. 2, Issue 2, May 2013, 1-8
© IASET

DISTRIBUTED COOPERATIVE CLUSTER BASED COMMUNICATION PROTOCOL FOR ENERGY EFFICIENCY IN WIRELESS SENSOR NETWORKS

G. NALLASIVAN[1] & S. LALITHA[2]

[1]Associate Professor, Department of CSE, P. S. R. Engineering College, Sivakasi, Tamil Nadu, India

[2]Assistant professor, Department of MCA, P. S. R. Engineering College, Sivakasi, Tamil Nadu, India

ABSTRACT

Energy constraint in wireless sensor networks has received an increasing research interest in recent years. Radio irregularity, channel fading and interference results in larger energy consumption and latency for packet transmission over wireless channel. One recent technology that has the potential to dramatically increase the channel capacity and reduce transmission energy consumption in fading channels is cooperative communication. The increase in the channel capacity results in reduced error rate . In this paper, one cooperative communication technique is proposed by constructing energy efficient sending and receiving clusters for each hop. It consists of two phases namely routing phase, recruiting-and-transmitting phase. In the routing phase, the initial path between the source and the sink nodes is discovered. In the second phase, the nodes on the initial path become cluster heads, which recruit additional adjacent nodes with lowest energy cost from their neighborhood then the packet is transmitted from the sending cluster to the newly established receiving cluster. The simulation results show that the reduction in error rate and the energy savings translate into increased lifetime of cooperative networks.

KEYWORDS: Sensor Networks, Clustering, Cooperative Networks Energy-Efficient Protocols, Cooperative Transmission, Fading Channel

INTRODUCTION

In Wireless networks nodes have limited energy resources, techniques designed should be energy-efficient. Wireless ad hoc networks have matured as a viable means to provide ubiquitous untethered communication. The basic idea of the cooperative communications is that all users or nodes in a wireless network can help each other to send signals to the destination cooperatively. Each user's data information is sent out not only by the user, but also by other users. Thus, it is inherently more reliable for the destination to detect the transmitted information since from a statistical point of view, the chance that all the channel links to the destination go down is rare. Multiple copies of the transmitted signals due to the cooperation among users result in a new kind of diversity, i.e., cooperative diversity that can significantly improve the system performance and robustness. In this paper, we use a cooperative communication model with multiple nodes on both ends of a hop and with each data packet being transmitted only once per hop. A key advantage of cooperative transmission is the increase of the received power at the receiving nodes. This decreases the probability of bit error and of packet loss. Alternatively, the sender nodes can use smaller transmission power for the same probability of bit error, thus reducing the energy consumption.

Recently, many efforts have also been focused on design of cooperative diversity protocols in order to combat the effects of severe fading in wireless channels. In [1] Cooperation Along Non-cooperative path is formulated. First the "*non-cooperative path*" between the source and the sink is found, then the last m predecessor nodes along the non-cooperative path is used for cooperation to transmit to the next node on the path. The work in [2] uses the model with only one helper node at each hop in addition to the sender and the receiver.

The authors of [3]-[4] proposed MAC layer design for cooperative transmission. The MAC protocol in [3] addresses the problem of low rate transmission in Wireless LAN with the assistance of high rate station. The author of [4] proposed a MAC in which a set of relays determine their needed transmission power to participate in the cooperative communication, while only the "best" one is chosen to minimize the overall energy consumption. The relay selection is done in a distributed manner with minimum overhead. This incur less cooperation overhead .Besides, this MAC can potentially achieve the same diversity multiplexing.

In the MIMO systems, each node is equipped with multiple antennas. Information is transmitted from the sender node by multiple antennas and received by multiple antennas at the receiver node [5], [6]. In [7], a MAC protocol for MIMO systems is described, which is based on centralized cluster architecture. This protocol uses clustering mechanisms like LEACH [8]. Nodes in a cluster cooperate to forward the data to only the next cluster head on the path to the sink. However, the centralized architecture leads to higher energy usage for the cluster maintenance. In contrast, distributed mechanisms are more efficient in the cluster maintenance operation and lack the single-point-of failure vulnerability. Thus, they may be better suited for sensor or mobile networks.

Finally, the significant cost increase in MIMO to implement multiple antennas at each node would be most often considered impractical in many wireless networks and, in particular, in sensor networks.

Several cross-Layer approaches are also developed in [9]-[10]. In[9] a Cross-Layer Medium Access Control (CL-MAC) protocol using two adjacent layers (MAC and Network) to economize energy for WSN is proposed. The basic idea behind this protocol is to wake-up only nodes belonging to a routing path from the source to the base station (Sink) by exploiting routing information while other nodes leave maintained as long time as possible in a sleep mode. The approach in [10] spans the physical, medium access control and routing layers, and provides: (a) a significant improvement in the end-to-end performance in terms of throughput and delay, and (b) robustness to mobility and interference induced link failures. MAC layer finds the list of neighbors by the HELLO messages of the routing protocol. But the selection of the nodes to cooperate is done randomly, without regard to how useful these nodes could be in improving the cooperative communication.

In our model of cooperative communication [Fig. 1] the initial path between the source and the sink nodes is discovered and every node on the path from the source node to the destination node becomes a *cluster head*, with the task of recruiting other nodes in its neighborhood and coordinating their transmissions. Consequently, the classical route from a source node to a sink node is replaced with a multihop cooperative path, and the classical point-to-point communication is replaced with many-to-many cooperative communication. The rest of the paper is organized as follows. Section II presents our proposed protocol. The simulation results are presented in Section III. Finally, Section VI concludes our paper.

PROPOSED SYSTEM

The routing phase is implemented using energy aware AODV.During this phase information about the energy required for transmission to neighboring nodes is computed. This information is then used for cluster establishment in the "recruiting-and-transmitting" phase by selecting nodes with lowest energy cost. Medium access control is done in the "recruiting-and-transmitting" phase through exchanges of short control packets between the nodes on the "one-node-thick" path and their neighbor nodes.

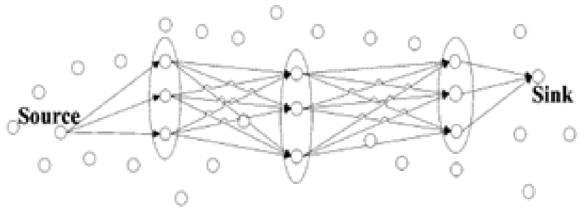

Figure 1: Our Cooperative Transmission Protocol

Operation of the Routing Phase

The initial path between the source and the sink nodes is discovered using modified AODV protocol with the links' transmissions energy used as the links' cost. Route discovery is based on a route request/route reply query cycle. Once discovered, a route is maintained as long as needed by the source. To guarantee loop freedom, AODV utilizes per node sequence numbers. A node increment the value of its sequence number whenever there is a change in its local connectivity information.

Route Discovery: Route discovery begins when a source node needs a route to some destination. It places the destination IP address and last known sequence number for that destination, as well as its own IP address and current sequence number, and its link cost into a Route Request (RREQ). It then broadcasts the RREQ and sets a timer to wait for a reply. When a node receives the RREQ, it first creates a *reverse route entry* for the source node in its route table. It then checks whether it has an unexpired route to the destination node. In order to respond to the RREQ, the node must either be the destination itself, or it must have an unexpired route to the destination whose corresponding sequence number is at least as great as that contained in the RREQ. If neither of these conditions are met, the node rebroadcasts the RREQ with updated link cost.

Route Reply: On the other hand, if it does meet either of these conditions, the node then creates a Route Reply (RREP) message. It places the current sequence number of the destination, as well as its distance in hops to the destination, total link cost into the RREP, and then unicasts this message back to the source. The node from which it received the RREQ is used as the next hop. When an intermediate node receives the RREP, it creates a *forward route entry* for the destination node in its route table, and then forwards the RREP to the source node. Once the source node receives the RREP, it can begin using the route to transmit data packets to the destination. If it later receives a RREP with a greater destination sequence number or equivalent sequence number and smaller link cost, it updates its route table entry and begins using the new route. If the source node does not receive a RREP by the time its discovery timer expires, it rebroadcasts the RREQ. It attempts discovery up to some maximum number of times. If no route is discovered after the maximum number of attempts, the session is aborted.

Route Maintenance: An active route is defined as a route which has recently been used to transmit data packets. Link breaks in nonactive links do not trigger any protocol action. However, when a link break in an active route occurs, the node *upstream* of the break determines whether any of its neighbors use that link to reach the destination. If so, it creates a Route Error (RERR) packet. The RERR contains the IP address of each destination which is now unreachable, due to the link break. The RERR also contains the sequence number of each such destination, incremented by one. The node then broadcasts the packet and invalidates those routes in its route table. When a neighboring node receives the RERR, it in turn invalidates each of the routes listed in the packet, *if* that route used the source of the RERR as a next hop. If one or more routes are deleted, the node then goes through the same process, whereby it checks whether any of its neighbors route through it to reach the destinations. If so, it creates and broadcasts its own RERR message. Once a source node receives the

RERR, it invalidates the listed routes as described. If it determines it still needs any of the invalidated routes, it re-initiates route discovery.

Operation of the Routing Phase

This phase is starting from the source node and progressing, hop by hop, as the packet moves along the path to the sink node. Once a data packet is received at a receiving cluster of the previous hop along the path, the receiving cluster now becomes the sending cluster, and the new receiving cluster will start forming. The next node on the “one-node-thick-path” becomes the cluster head of the receiving cluster.

The receiving cluster is formed by the cluster head recruiting neighbor nodes through exchange of short control packets. Then, the sending cluster head synchronizes its nodes, at which time the nodes transmit the data packet to the nodes of the receiving cluster.

The example in Fig. 2(a)–(f) demonstrates the operation of the “recruiting-and-transmitting” phase. In the current hop, node 2 is the sending cluster head and has a packet to be sent to node 5. Node 2 sends a request-to-recruit (RR) packet to node 5 [Fig. 2(a)], causing node 5 to start the formation of the receiving cluster, with node 5 as the cluster head. From the routing phase, node 5 knows that the next-hop node is node 8.Node 5 broadcasts to its neighbors a recruit (REC) packet [Fig. 2(b)]. The REC packet contains: the id of the previous node (2), the id of the next node (8), and the maximum time to respond, denoted as .

Each node that receives the REC packet, which we call *potential recruits* (nodes 4 and 6 in our example), computes the sum of the link costs of the following two links: a link from the sending cluster head to itself (the *receiving link*) and a link from itself to the next node, such as the receiving cluster head or the sink node (the *sending link*). In our example, node 4 computes the sums of the energy costs of the links (2,4) and (4,8), i.e., , while node 6 computes the sum of the energy costs of the links (2,6) and (6,8), i.e., .

A *potential recruit* replies to the REC packet with a grant (GR) packet that contains the computed sum [Fig. 2(c)] after a random backoff time drawn uniformly from (0,). The GR packets inform the cluster head that the nodes are available to cooperate in receiving on the current hop and in sending on the next hop.

After waiting time and collecting a number of grants1, the cluster head (node 5) selects cooperating nodes with the smallest reported cost to form the receiving cluster of nodes. (The value of is protocol-selectable.) If the cluster head node received less than grants, it forms a smaller receiving cluster with all the nodes that sent the grants.

Node 5 then sends a clear (CL) packet [Fig. 2(d)] that contains the ids of the selected cooperating nodes (4 and 6 in our example). The CL packet serves two purposes: 1) it informs the sending cluster head (node 2) that the cluster has been formed; and 2) it informs the *potential recruits* whether they have or have not been chosen to cooperate.

Upon receiving the CL packet from node 5, node 2 sends a confirm (CF) packet to the nodes in its sending cluster (nodes 1 and 3) to synchronize their transmission of the data packet [Fig. 2(e)].

The CF packet contains the waiting-time-to-send and the transmission power level . The transmission power level is the total transmission power (a protocol-selectable parameter) divided by the number of the nodes in the sending cluster. In the case of our example, the value of is divided by 3 (nodes 1–3 are cooperating in sending).

After the waiting-time-to-send expires, sending cluster nodes 1–3 send the data packet to the receiving cluster nodes 4–6 [Fig. 2(f)].

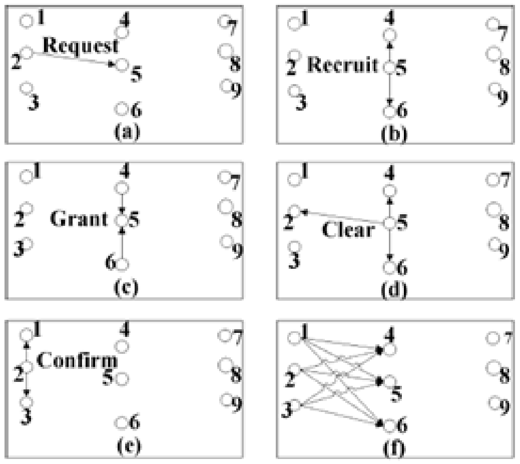

Figure 2: Example of the Recruiting Phase Operation (a) Request-to-Recruit (RR) Packet (b) Recruit (REC) Packet (c) Grant (GR) Packet (d) Clear (CL) Packet (e) Confirm (CF) Packet (f) Transmission of the Data Packet

Calculation of the Cost of Links

The cost of a link from node to node i to node j , is calculated by node i as: $C_{i,j} = (e_{i,j})^{\theta} / (R_i / R_{avg})$ where $e_{i,j}$ is the energy cost of the link, R_i is the residual battery energy of node , and R_{avg} is the average residual battery energy of the neighbors of node . Energy cost of a link is the transmission power required for reception at a particular bit error rate. Nodes determine the energy costs of links by listening (or overhearing) transmissions during the routing phase. The protocol-selectable parameter θ controls the weight of each factor in the total cost. With this definition of the cost, nodes with small residual battery capacity are less likely to be recruited in this phase.

Details of the Control Packets

The format of an RR packet includes: the id of the sender node (node 2 in our example), the id of the receiver node (node 5 in our example), the sink node id, and the NAV field that contains the estimated transmission time of the data packet. The NAV field serves to indicate when the channel will become available again for other transmissions. The REC packet contains the sender node id, the receiver node id, the id of the next node on the path (node 8 in our example), and the maximum time-to-respond. The GR packet sent from node contains the id of the originator of the REC packet and the sum of the link costs of the *receiving link* and the *sending link*. A node can be involved in a single recruiting process at any time; i.e., a node can have only one outstanding GR packet. A node chosen to cooperate cannot be involved in another recruiting process until the transmission of the current data packet is fully completed, i.e., received and sent to the next cluster by the cooperating node. A CL packet contains the id of the cooperating nodes and an updated value of NAV. Nodes that see their ids in the CL packet form the receiving cluster for this hop and the sending cluster for next hop. Other neighbor nodes that sent GR packets but do not see their ids in the CL packet will not participate in the cluster. To avoid interference, any node that receives an REC packet, whether cooperating or not, has to wait for the transmission of the data packet to be fully completed before it can get involved in another recruiting process. Similarly, to avoid interference, any node that overhears any of the control packets sent by any other node will not get involved in any recruiting or any transmission operation until the transmission of the data packet is fully completed. If a data packet was not received at the receiving cluster head node, or was received in error, the packet is deemed lost, and the whole “recruit-and-transmit” phase will restart again. A timer is associated with every exchange of control packets, so that if a critical control packet is lost, the “recruit-and-transmit” phase will restart again.

SIMULATION RESULTS

We use simulation to evaluate the performance of our protocol by comparing it to the CAN protocol. We use ns2 simulation package implementation.

We run two sets of experiments. In the first set, nodes are positioned on a grid[fig. 3] to compare our simulation results to our analytical results. In the second set of experiments, nodes are randomly placed for a more realistic scenario.Unless otherwise stated, we assume that the channel bandwidth is 1 Mb/s, the length of the data packets is 1 kB, is 3, the maximum

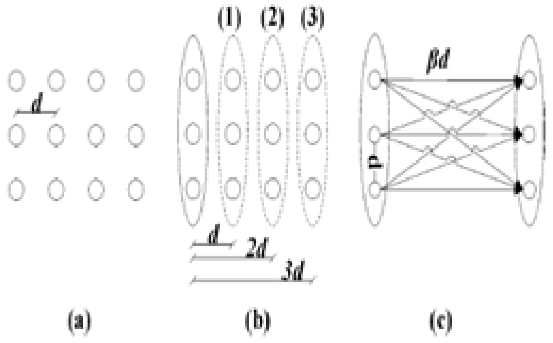

Figure 3: Grid Topology (a) Placement of Nodes (b) Formation of Clusters (c) Intra-Versus Inter-Cluster Distances

waiting time is 1.5 ms, and the maximum retry time is 50 ms. We use . Each of our simulation results represents an average of 10 random runs, and each simulation run represents a real time of 100 s. We establish one route of 5 hops between a source node in the first column and the middle row and a sink node in the same row. The sink's column varies depending on the parameter . In our protocol, the initial path is set as the middle row, and clusters are formed from nodes in the same column. In the CAN protocol and the one-path scheme, the cooperative path is set as the middle row.

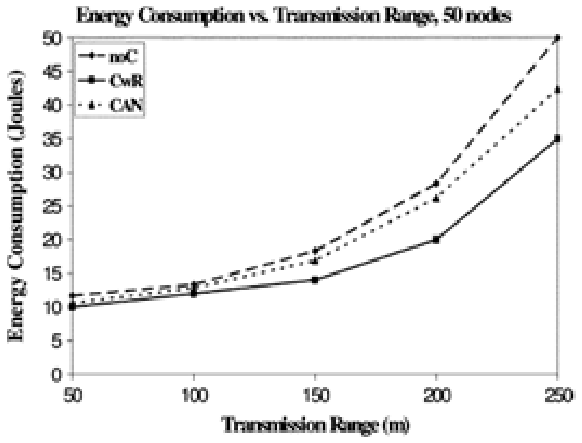

Figure 4: Effect of Transmission Power on Consumption

In Fig. 4,which shows the effect of the transmission range on the total energy consumption. Here, we sum the energy consumption for all packets transmitted (control and data packets). Our cooperative transmission protocol saves between 6% and 20% of the energy consumption compared to the CAN protocol. As the transmission range increases, the contention increases and the noise power increases. This increases the energy consumption. The elevated contention increases the retransmission of control and data packets, which, in turn, increases the total energy consumption.

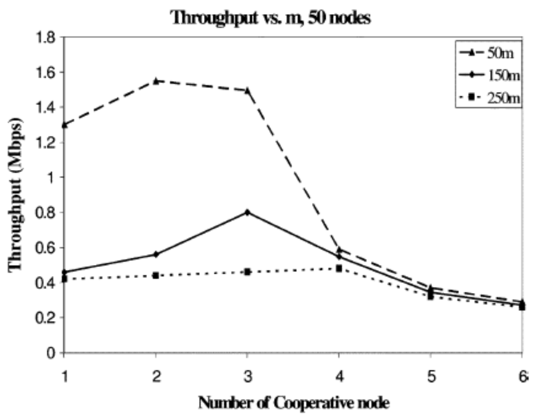

Figure 5: Effect of the Number of Cooperative Nodes

In Fig. 5, we study the effect of the number of cooperative nodes on the performance of our cooperative protocol. We fix the packet loss probability at 0.2.We plot the capacity versus the number of cooperative nodes for three different transmission ranges: 50, 150, and 200 m. Each point in the figure represents the maximum load that can be pushed through the network. There is a tradeoff between the delay of recruiting the cooperative neighbors and the robustness to packet loss. At small, the delay is small, but the effect of packet loss is more significant on the performance of our cooperative transmission protocol.

CONCLUSIONS

In this paper, we evaluated the performance of cooperative transmission, where nodes in a sending cluster are synchronized to communicate a packet to nodes in a receiving cluster. In our communication model, the power of the received signal at each node of the receiving cluster is a sum of the powers of the transmitted independent signals of the nodes in the sending cluster. The increased power of the received signal, vis-à-vis the traditional single-node-to-single-node communication, leads to overall saving in network energy and to end-to-end robustness to data loss. We proposed an energy-efficient cooperative protocol, and we analyzed the robustness of the protocol to data packet loss.

REFERENCES

1. A. Khandani, J. Abounadi, E. Modiano, and L. Zheng, “Cooperative routing in static wireless networks,” IEEE Trans. Commun., vol. 55, no. 11, pp. 2185–2192, Nov. 2007.
2. J. Nicholas Laneman, Member, IEEE, David N. C. Tse, Member, IEEE, and Gregory W. Wornell, “Cooperative Diversity in Wireless Networks: Efficient Protocols and Outage Behavior”, IEEE Transactions on Information Theory, Vol. 50, No. 12, December 2004.
3. Pei Liu, Zhifeng Tao, Sathya Narayanan, Thanasis Korakis, and Shivendra S. Panwar, “CoopMAC: A Cooperative MAC for Wireless LANs”, IEEE Journal On Selected areas In Communications, VOL. 25, NO. 2, February 2007.
4. Zhong Zhou, Shengli Zhou, Jun-Hong Cui, and Shuguang Cui, “Energy-Efficient Cooperative Communication Based on Power Control and Selective Relay in Wireless Sensor Networks”, IEEE Journal On Selected areas In Communications, VOL. 14, NO. 2, February 2008.

5. D. Hoang and R. Iltis, “An efficient MAC protocol for MIMO-OFDM ad hoc networks,” in Proc. IEEE Asilomar Conf. Signals, Syst. Comput., Pacific Grove, CA, Oct. 2006, pp. 814–818.

6. K. Sundaresan, R. Sivakumar, M. Ingram, and T. Chang, “A fair medium access control protocol for ad-hoc networks with MIMO links,” in Proc. IEEE INFOCOM, Hong Kong, Mar. 2004, vol. 4, pp. 2559–2570.

7. Y. Yuan, M. Chen, and T. Kwon, “A novel cluster-based cooperative MIMO scheme for multi-hop wireless sensor networks,” EURASIP J. Wireless Commun. Netw., vol. 2006, no. 2, pp. 38–38, Apr. 2006.

8. W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocols for wireless microsensor networks,” in Proc. IEEE Int. Conf. Syst. Sci., Jan. 2000.

9. Bouabdellah Kechar, Ahmed Louazani, Larbi Sekhri, Mohamed Faycal Khelfi, “ Energy Efficient Cross-Layer MAC Protocol for Wireless Sensor Networks”,IEEE Trans. Commun., vol. 25, Dec. 2008

10. G. Jakllari, S. V. Krishnamurthy, M. Faloutsos, P. V. Krishnamurthy, and O. Ercetin, “A cross-layer framework for exploiting virtual MISO links in mobile ad hoc networks,” IEEE Trans. Mobile Comput., vol. 6, no. 6, pp. 579–594, Jun. 2007.

11. C. Perkins and E. Royer, “Ad-hoc on-demand distance vector routing,” in Proc. IEEE WMCSA, New Orleans, LA, Feb. 1999, pp. 90–100.

12. I. Hen, “MIMO architecture for wireless communication,” Intel Technol. J., vol. 10, no. 2, pp. 157–165, May 2006.

13. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE 802.11 Working Group, 1997.

14. Q. Dong, S. Banerjee, M. Adler, and A. Misra, “Minimum energy reliable paths using unreliable wireless links,” in Proc. ACM MobiHoc, Urbana-Champaign, IL, May 2005, pp. 449–459.