Volume 2, Issue 9, September 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Survey on Neighbor Discovery in Asynchronous Wireless Sensor Network

K.Sreekanth
Asst Professor
Department: CSE Dept
Malla Reddy College Of Engineering & Technology
E-mail:sreeinfo23@yahoo.com

S.Sarfaraj
M.Tech student,
Malla Reddy College Of Engineering &Technology
CSE Dept.
sarfaraj06@gmail.com

Abstract:The Neighbor Discovery is a process of identifying the nearest node. So the identification can done through Neighbor Discovery Protocol (NDP) is a protocol in the Internet Protocol Suite used with IPv6. It is responsible for address autoconfiguration of nodes, discovery of other nodes on the link, and maintaining reachability information about the paths to other active neighbor nodes. Node connectivity is subject to changes because of mobility in wireless communication, transmission power changes, or loss of synchronization between neighboring nodes. Hence, even after a sensor is aware of its immediate neighbors, it must continuously maintain its view, a process we call continuous neighbor discovery Each sensor employs a simple protocol in a coordinate effort to reduce power consumption without increasing the time required to detect hidden sensors.

Keywords: Sensor network, ndprotocol, ndpmodel, hidden link analysis

I. Introduction

Sensor networks are large sets of small, inexpensive devices with hardware for sensing and a radio for communication with the other sensors. Sensor networks are being enabled by the convergence of several technologies at once. The advent of cheap, low-power microprocessors, sensor technology, and low-power RF design has made it possible to conceive of large networks which can together do what might be impossible (or too costly) to do with fewer, more expensive nodes.

Wireless sensor networks are “ad hoc” networks, which means that the topology of the network is not planned, but must be decided by the network nodes themselves. Many fundamental questions about wireless ad hoc networks remain unanswered. Among the questions considered in this dissertation are: To what extent is “layered networking” satisfactory for sensor networks? Is scheduling or contention a better way to control medium access in sensor networks? With what power should nodes in the network transmit? Will nodes determine their neighbors, and if so, how?

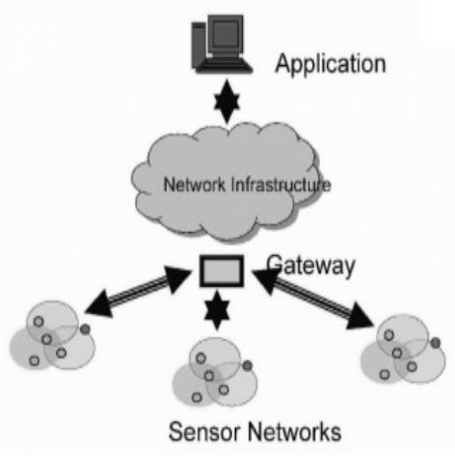
Ad hoc sensor networks are likely to be designed to fit one application verywell. This is because sensors will in many cases face a short lifetime as a result of the limitations of battery technology, and energy can be conserved by removing extraneous functions. The result of applying this principle is that sensor architectures

will be optimized for two things: lasting a long time, and doing well the one thing for which they were designed.

If a given wireless sensor network is to have only one or two objectives, there is little reason to maintain the generality of the OSI model (“layered networking” [1]) within the network stack of the sensor. We should optimize lifetime or performance by making connections across layers of the network stack. In a general purpose computer, which must handle a large set of applications and conform to many networking standards, layering is a reasonable response to the great complexity of software. In a sensor which has a single application and no requirement to conform to a standard (at this time), layering serves as an obstacle to performance, lifetime, or both. For example, some routing algorithms decide which link to forward a packet onto by choosing the path with smallest end-to-end expected delay. A better fit for an energy-constrained node might be to choose the path consuming least total energy. However, energy is a physical layer parameter, which would not be available in the routing calculation if we maintained a strict boundary between the routing, link and physical layers. It is quite unclear how the optimal communication system should work, absent layering. We largely keep the existing layered model, but allow information to be shared between the layers where there are clear advantages to doing so.

Contention-based medium access control (MAC), such as that which is in 802.11 wireless LANs, is often preferred to scheduled access to the wireless medium. However, contention-based MACs have well-known disadvantages, including wide variability in delay of transmissions, poor performance in heavily loaded networks, and wasted energy when multiple users attempt to transmit simultaneously. For energy-related metrics typical of sensor networks, scheduling would clearly be superior for mediating access.

II. Architecture:



III. Existing System

Initial neighbor discovery is usually performed when the sensor has no clue about the structure of its immediate surroundings. In such a case, the sensor cannot communicate with the gateway and is therefore very limited in performing its tasks.

Disadvantages:

1. In networks with continuously heavy traffic.
2. Long-term process.
3. Greater expense of energy than required in our scheme.

IV. Proposed System

We distinguish between neighbor discovery during sensor network initialization and continuous neighbor discovery. We focus on the latter and view it as a joint task of all the nodes in every connected segment. Each sensor employs a simple protocol in a coordinate effort to reduce power consumption without increasing the time required to detect hidden sensors.

Advantages:

1. Detect their immediate neighbors.
2. Message does not collide with another.
3. Every node discovers its hidden neighbors independently.

Technics:

1. Client – Server
2. Detecting all hidden links Inside a segment
3. Detecting all hidden links Outside a segment
4. Neighbor Discovery Model

Client – Server:

Client – Server computing is distributed access. Server accepts requests for data from client and returns the result to the client. By separating data from the computation processing, the compute server's processing capabilities can be optimized. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system.

Hidden link participate Inside a segment:

This scheme is invoked when a new node is discovered by one of the segment nodes. The discovering node issues a special SYNC message to all segment members, asking them to wake up and periodically broadcast a bunch of HELLO messages. This SYNC message is distributed over the already known wireless

links of the segment. Thus, it is guaranteed to be received by every segment node. By having all the nodes wake up .almost at the same time. for a short period, we can ensure that every wireless link between the segment's members will be detected.

Hidden link participate Outside a segment:

A random wake-up approach is used to minimize the possibility of repeating collisions between the HELLO messages of nodes in the same segment. Theoretically, another scheme may be used, where segment nodes coordinate their wake-up periods to prevent collisions and speed up the discovery of hidden nodes. Since the time period during which every node wakes up is very short, and the HELLO transmission time is even shorter, the probability that two neighboring nodes will be active at the same time.

Neighbor Discovery Model:

Neighbor Discovery is studied for general ad-hoc wireless networks. A node decides randomly when to initiate the transmission of a HELLO message. If its message does not collide with another HELLO, the node is considered to be discovered. The goal is to determine the HELLO transmission frequency, and the duration of the neighbor discovery process.

Neighbor discovery is the determination of all stations with which a station may communicate directly. It is an important and non-trivial task in wireless networks, particularly for sensors. In the case where sensors are immobile, it may make sense to pay a one time price to learn of the existence of all one's neighbors in order to
optimize medium access and to enable routing.
There are several ways to discover neighbors. In one class of methods, there is a central controller. All stations report to the controller, which determines the positions of the stations, computes their neighbors, and informs each station. Neighbor discovery is expected to cost a lot of energy, particularly when the number of nodes is large. Distributed algorithms have no central controller.
In the distributed algorithm of Baker , all nodes participate in a two-round round-robin schedule. In each round each station is assigned a single slot to announce its identity and the identities of neighbors discovered so far. Stations listen in the other slots, and can determine all their neighbors, and all their neighbors' neighbors, within two rounds, under the assumption that nodes receive messages from neighboring nodes without errors.
Although the algorithm of [41] is distributed, it requires global synchronization
of the network. In particular, it assumes:
(G1) schedules may be formed among sets of ≥ 3 nodes, and
(G2) the nodes know ahead of time the time when the algorithm is supposed to begin.

The additional advantage that it does not require global synchronization. When the algorithmconcludes, each node has a (possibly incomplete) list of its neighbors.
The ability to work without global synchronization is useful for two reasons. First, neighbor discovery may be the very first algorithm run in a network. Subsequent network behavior may presume that some prior mechanism enforces synchronization, but for the first algorithm there is no basis for that presumption. Second,
the burden of maintaining global synchronization through algorithmic means, or the expense of maintaining it through hardware such as GPS, would increase with the size of the network. A large wireless network implies a large expense to maintain global synchronization.

The usefulness of an asynchronous algorithm for wireless networks goes beyond the ability to compile a list of neighbors. The algorithm is a protocol for passing arbitrary messages between neighboring nodes. Some messages that could be passed are the exchange of messages in support of a transmission scheduling
algorithm; the exchange of information about one's neighbors, eventually allowing several-hop-away neighbor information to be obtained, which may be used to support routing algorithms; the exchange of physical layer parameters such as received signal strength; and so on. It is even possible for this algorithm to carry messages
which will allow for the subsequent synchronization of the network, thereby allowing algorithms which require synchronization to follow it.

Node x is a neighbor of node y if x can exceed y's signal-to-noise ratio requirement.
We assume that:
1. each node can broadcast its transmission, or receive up to one broadcast at any time, but cannot simultaneously transmit and receive. Broadcasts from neighbors are received free of errors, provided only one neighbor is transmitting.
2. each node has and knows its own unique identifier.

3. all nodes have an estimate # *N* of the number of neighbors each is expected to find. The requirement that each station have a unique identifier can be achieved in various ways. Each station might have a Network Interface Card ID, or a CPU ID, which it can access and whose uniqueness is assured by device manufacturers. Or,
if the nodes are truly indistinguishable from a hardware point of view, a sufficiently large pool of numbers may be chosen from, such that it is highly unlikely two nodes near each other will collide in their choices.
The requirement for an estimate of the number of neighbors will often be satisfied in sensor networks. A known number of sensors may be deployed within a known area, and an estimate follows from experience of the radio range of the devices within the expected propagation environment. Any neighbor discovery scheme requires there be a message *m* identifying the sender. We assume that *m* can be successfully transmitted in time *Tm* to a receiver. The receiver successfully receives *m* if no other station within distance *d* of the receiver is transmitting simultaneously. That is, transmissions colliding at a receiver are destroyed. To counter the lack of global synchronization we include a preamble in the message *m*. It is therefore important that the receiver of *m* receive *m* from the beginning; no cut-ins are allowed.

Node x discovers y if x and y are neighbors and x receives m fromy at least once. Note that *x* may discover *y* without *y* discovering *x*.

Problem: Given a set of *K* immobile wireless nodes whose locations are unknown a priori, we seek a distributed algorithm, that does not require global synchronization of the network, and that maximizes the performance metric described below. Our performance metric is the following.
Let *A* be a neighbor discovery algorithm. Given a finite set of nodes, each node has a finite set of neighbors. There is a finite set of neighbor relations for the network, including both (*x, y*) and (*y, x*) if *x* and *y* are neighbors.

Algorithm description

We will describe a neighbor discovery algorithm *A*. We assume each station has its own time slotting, with equal slot lengths, but at a random offsets to others. In each slot each node chooses among the states *{*T,
R*}* corresponding to transmit and receive, with probabilities *pT* and *pR* satisfying *pT* + *pR* = 1. The nodes act independently of each other and choose their states independently in each slot.
We determine the slot lengths as follows. During a slot when a node has decided to transmit, it transmits *W* copies of message *m*, where *W* is a positive integer fixed throughout the network. In the case of slotted operation *W* > 1 is unnecessary, but for unslotted operation the optimal value of *W* remains to be determined.
Depending on the radio technology in use, there will be a maximum transmission rate and some time required to transit between states, which we assume is negligible. We assume that a period *Tm* is required to transmit *m* once, and that *WTm* is required to transmit *W* copies of the message. Thus a single slot has duration *T = WTm*.
During a receive slot, a node turns on its receiver and decodes its input. The node processes the input to determine whether an error-free message was received. If one was, then the identity of the transmitter is determined from the message contents, and if the transmitter was heretofore unknown to the receiver, he is added to a local list of neighbors. Any additional information in the message is stored or updated at the receiver.
To overcome the lack of an agreed start time for *A*, we propose an additional algorithm *B*, which is compatible with *A*. *B* is described in Appendix 5.9. The effect of *B* is that nodes will begin to run *A* not all at once but at various times, such that neighbors' runs of *A* overlap.

Asynchronous analysis of *A*

The primary goal of the asynchronous analysis of *A* is to determine the optimal values of *W* and *pT* . The larger *W* is, the longer the slots are. Large *W* gives a better chance for the message *m* to be successfully received during a slot, but for a given period *t* over which *A* is to be run, large *W* reduces the number *S* of slots
We have assumed that if two neighbors simultaneously transmit their messages, *X* hears garbage during the period of their overlap; *X* must hear exactly one of his neighbors transmitting a complete message. If multi-user detection were possible, then more transmissions would be successfully received and the performance of this
algorithm would be improved.
The algorithm produces diminishing returns as it runs. At first, every reception of the message *m* from a neighbor is new; as the algorithm runs and more messages are received, more of them duplicate messages already received; finally, a node may continue to run *A* well after it has (unknowingly) discovered all its neighbors. If *A* runs long enough, a majority of the energy used by *A* is wasted, since only the first time one transmits successfully to a neighbor is one getting useful information across. Later successful transmissions to that neighbor are redundant. In this was assumed that the running time *t* is fixed. If instead we were to choose a stopping point, we could use Equation (5.5) as a basis for determining performance as a function of time.
The nodes might adapt local parameter settings based on observations. During *A*, they listen during every slot they do not transmit. If a nodes hears few transmissions, from the fact that *pT* is common across the network, it could conclude that it has few neighbors. If this were the case, a higher *pT* might be called for. Performance might be improved by simple adaptive behavior such as this.

V. Conclusion

Neighbor discovery is an important task in sensor networks. This paper presented the asynchronous, distributed neighbor discovery algorithm. The algorithm *A* impose a state machine structure on the participating nodes and promise simple operation. Furthermore, the probabilistic nature of *A* makes it robust in the event that the actual number of neighbors is different from what was expected. The algorithm's performance was analyzed in slotted and unslotted time. A major difference between the algorithms is that *A* is probabilistic, so not all neighbor relations are necessarily discovered. However, simulations indicate that *A* can discover nearly all neighbors with proper parameter settings.

References

[1] S. Vasudevan, J. Kurose, and D. Towsley, .On neighbor discovery in wireless networks with directional antennas,. in *INFOCOM*, vol. 4, 2005, pp. 2502.2512.
[2] R. Madan and S. Lall, .An energy-optimal algorithm for neighbor discovery in wireless sensor networks,. *Mob. Netw. Appl.*, vol. 11, no. 3, pp. 317.326, 2006.
[3] M. J. McGlynn and S. A. Borbash, .Birthday protocols for low energy deployment and _exible neighbor discovery in ad hoc wireless networks,. in *MobiHoc: Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing*. New York, NY, USA: ACM Press, 2001, pp. 137.145.
[4] D. Baker and A. Ephremides, .The architectural organization of a mobile radio network via a distributed algorithm,. in *IEEE Transactions on Communications*, vol. 29, Nov. 1981, pp. 1694.1701.
[5] A. Keshavarzian and E. Uysal-Biyikoglu, .Energy-ef_cient link assessment in wireless sensor networks,. in *INFOCOM*, 2004.
[6] E. B. Hamida, G. Chelius, and E. Fleury, .Revisiting neighbor discovery with interferences consideration,. in *PE-WASUN*, 2006, pp. 74.81.
[7] S. A. Borbash, .Design considerations in wireless sensor networks,. Ph.D. dissertation, ISR, August 2004.
[8] G. Alonso, E. Kranakis, R. Wattenhofer, and P. Widmayer, .Probabilistic protocols for node discovery in ad-hoc, single broadcast channel networks,. in *IPDPS*, 2003, p. 218.
[9] C. Perkins, E. Belding-Royer, and S. Das, .Ad hoc on-demand distance vector (AODV) routing,. RFC 3561, July 2003.
[10] J. Haartsen, *Bluetooth Baseband Speci_cation v. 1.0*.
[11] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. O. LaMaire, .Distributed topology construction of bluetooth personal area networks,. in *INFOCOM*,2001, pp. 1577.1586.
[12] *IEEE 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Speci_cations for Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE 802.15 WPAN Task Group 4 (TG4), 2006.
[13] P. Dutta and D. Culler, .Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications,. in *SenSys: Proceedings ofthe 6th ACM conference on Embedded network sensor systems*. New York, NY: ACM Press, 2008, pp. 71.84.
[14] J. Hill and D. Culler, .A wireless embedded sensor architecture for system-level optimization,. Technical report, U.C. Berkeley, 2001.

