Building a dependable smart home system

Swadhin Pradhan, Qiren Chen, and Xiaofan Lu
{swadhin, qiren, xiaofan}@cs.utexas.edu
Computer Science Department
The University of Texas at Austin, USA

I. Motivation

Internet of Things (IoT) is gaining popularity with an array of consumer wearable devices and smart home solutions. In such a smart home scenario, we have a new computing environment comprising a set of sensor nodes and hosts traveling in and out the rooms and communicating in an opportunistic manner via wireless links. The lack of wired network infrastructure imposes a set of new challenges on how to build a dependable smart home system which can withstand many failures in timely manner. For example, disconnection with certain user doesn't mean that the user is actually leaving, as wireless connection is unreliable and disconnection is frequent.

II. System Description

We define our smart home system in the following manner:

(1) A set of static sensor nodes at home whose positions are known to every node.
(2) A set of mobile nodes. These can roam around home but when outside connected via internet.
(3) A central node at home with connectivity to internet.
(4) When inside home, the nodes communicate via bluetooth and subset of mobile/central node can communicate through internet.
(5) All mobile nodes have energy budget.
(6) Human-in-the-loop model can be modeled via different instructions given through a mobile node (e.g. smartphone).

In the following sections, we will discuss a set of challenges and corresponding solution in this kind of smart home system.

III. Clock Synchronization

It is necessary for performing cascading operations within nodes. Clock synchronization among sensor nodes can be performed using Christian's algorithm where we can assume central node as master (assuming central node has a hardware clock with good precision and the failure probability of center node is very low). But, if we assume mobile nodes (in or out home), it is very difficult to estimate D as they are moving. Systematically, we can enumerate the following cases for clock synchronization in mobile nodes in smart home scenario :

1) Mobile node is available at home and it can sense different sensor nodes (except center node) in their vicinity.
2) Mobile node is available at home but it can not sense different sensor nodes in their vicinity. However, the mobile node can connect to center node.
3) Mobile node is roaming outside and can connect to the home center node through internet.

In all the cases, we assume that all sensor nodes including the center node knows the locations of each other and calculate the delay among themselves using heartbeat messages. The clock of these sensor nodes are synchronized using Christian's algorithm within δ duration. Moreover, each mobile node has a parameter γ which is the period in which it searches for the nearest neighbor set. For the first case, mobile node will run Christian's algorithm at every γ duration (assuming $gamma$ is set to be very large compared to the worst case delay between any two nodes in the smart home system) with its neighbor sensor nodes and calculate the ranges of possible times at the master. Then, it will consider the lowest and highest possible value in those range set and take the middle most value for least error consideration. For the second case, it will try to run Christian's algorithm with the sensor node every γ period using the assumption that there is no drastic changes in the mobile node location at home. For the last case, if the mobile node is outside then it is very hard to estimate D to run the Christian's algorithm like second case if the last hop to the mobile node is wireless. Otherwise in case of wired connection, we

can run Christian's algorithm for time synchronization. If the last hop is wireless, we will use multiple APs or multiple cell towers similar to the first case to estimate the last hop propagation delay and thus the time estimate.

IV. Atomic Broadcast

Atomic broadcast is necessary if group membership information is needed. This is because it is important for the smart home system to know about any component failure. It may become critical like fire alarm sensor failure. It is very difficult to estimate node-to-node delay for mobile nodes. With the number of processors in an open system changing, we have to re-consider the problem of broadcasting messages. Typically there are several processors in such a system, and we have no fixed infrastructure in the configuration of the network. In Christian's algorithm, there are $f+1$ wired channels among processors. Here, in wireless network, we have infinite channels. Another difference between these two networks is in an open system a processor can move in or move out freely. The membership is always changing. Not only processors can crash or move out, but also new processors can join dynamically. In a wireless network, the range of a message is bounded. Messages can be directly transmitted by a processor to other processors that are no more than a particular distance D. To be energy efficient, one processor needs to transmit as few messages as possible.

A. System Model

First we want to describe our system. As the definition of smart home system, there are a set of static sensor nodes and slow-moving mobile nodes. As well there is a central node in this system. But in such system, not only central node will broadcast messages but also any node may do broadcasting. So in smart home system, the central node can be viewed as a normal static node as others.

In the Smart Home Broadcast (SHB) the system is a graph, where a vertex of the graph represents a node and there is directed edge from node i to node j if node j is in the range of node i. The communication graph is connected all the time and of maximum diameter D. We refer the transmission range D as one hop. In SHB, we have infinite wireless channels instead of $f+1$ channels in wired network. Thus, we don't care about channel failure in our protocol. In wired network, a node sends message on different channels without considering about the destination nodes. However, in SHB, though we have infinite channels, we need the information of neighbor nodes. When nodes move out or move in, nodes already in SHB need to know their existence. In our protocol, nodes broadcast beacon messages to advertise their own existence and also discover the existence of neighboring nodes within the one hop.

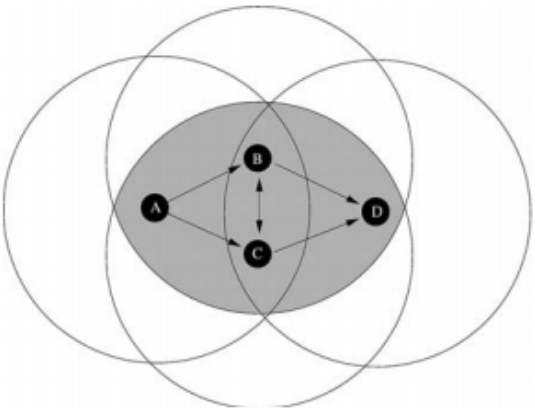


Fig. 1: Example of redundant rebroadcasts and channel contention

B. Simple Flooding

The simplest form of broadcast in SHB is simple flooding. In simple flooding, a node transmits a packet, which is received by all neighboring nodes that within one hop. Upon receiving a broadcast packet, each node determines if it has transmitted the packet before. If not, then the packet is retransmitted. Simple flooding terminates when all nodes have received and transmitted the packet being broadcast at least once. As all nodes participate in the broadcast, blinding flooding suffers from the Broadcast Storm Problem. In the worst case there are $N-1+(N-1)*(N-2)$ messages. Simple flooding may result in several problems:

1) Redundant rebroadcasts: a node sends the same message to its neighbor more than once.
2) Channel contention: when neighbor nodes receive messages, they may contend channels to broadcast simultaneously.
3) Out of Memory: every node in broadcast may get N redundant copies for one message. This may cause nodes to run out of memory.

The above figure shows us an example of redundant rebroadcasts and channel contention.

- Since node A within transmission distance of B and C, both B and C will get messages from A. For D, it will receive two copies from B and C. This redundancy can cause out-of-memory problem.
- Nodes B and C may contend for the broadcast medium in the shadow area. If there are more nodes in the shadow area, there will be an increase in contention for the broadcast medium.

C. Neighbor based Smart Home Broadcast

In SHB, nodes periodically broadcast beacon messages to advertise their own existence. Nodes update neighborhood list by receiving beacon messages. One beacon message may contain its own identity (ID), its address and the neighboring nodes that the node may be aware of. Thus, the information of neighbor topology within two hops can be obtained. Information of neighbor topology can be used to determine link state topology. GPS may not work well at home. Neighbor information is helpful in such situation. Once a node has its one- and two-hop neighbor set, it can select a minimum number of one-hop neighbors which covers all its two-hop neighbors.

The goal of our protocol is to minimize the number of rebroadcast while forwarding a broadcast message. This protocol checks the states of its neighbors and decides which set of its neighbors to re-transmit. This set is kept small as much as possible by efficiently selecting the neighbors which covers the same network region as the complete set of neighbors does.

We denote N(x) to the set of one-hop neighbors of node x and S(x) to the set of two-hop neighbors.

Here are the steps to decide the set of neighbors to broadcast for one node.

1) Start with an empty set to broadcast
2) Find all the one-hop nodes which connect with only one or no two-hop node. Add these one-hop nodes into broadcast set.
3) From the rest of one-hop nodes, select a node which connects with most two-hop nodes not in the set. Add this node into broadcast set.
4) Repeat Step 3 until all the two-hop nodes are covered.

Here is an example for this protocol:

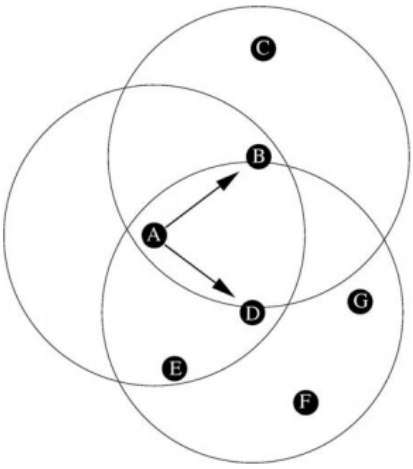


Fig. 2: Example run of the protocol

In this case, node A wants to broadcast some message. In this graph, B, D, E are one-hop nodes of A and C, G, F, are two-hop nodes. A broadcast is initiated by A. First add B into the broadcast set since only C is reachable from B. D connects two two-hop nodes F, G, which is more than E. Thus next add B into the set. All two-hop nodes are covered and protocol ends. E is not added into the set since its neighbor F is added into the set. For the second-round broadcast, the broadcast set of B will contain C and the set of D will contain E, F, G. By now, every node is covered in this case. For not rebroadcast, we can add a path message in each broadcast message. A path message tells the path of this message in avoid of duplicate rebroadcast.

This protocol works much better than simply flooding since we have seen we don't need to broadcast to every neighbor. In this protocol, every node has information of two-hop nodes and it ensures every one- or two-hop node is covered. Nodes of three hops will be covered by one-hop nodes and so on so forth. Since we avoid rebroadcast, there are O(N) messages in one broadcast, where N is the number of nodes.

In this part, we analyze the difference of a smart home system and a wired system. Based on the difference, we build a protocol to decrease the total number of messages forwarding. We have shown that this protocol reduce the flooding test, utilizing neighborhood information.

V. Group Membership

If atomic broadcast works, we can run two-tier group membership protocol. Static sensor nodes can run attendance list protocol whereas mobile nodes can run neighbor surveillance protocol among them. This group membership is important because every sensor might take some decision or run some protocol depending upon the availability of some sensors.

A. Why would we need group membership?

With the advent of both mobile and communication technology, group membership is useful in a lot of applications:

- Consensus Preference
  The group ranking problem consists of constructing coherent aggregated results from preference data provided by decision makers. The fundamental of the above problem is an accurate group membership. In our smart-home setting, different family member may have different preference in certain setting in home. Consider the following toy example:

	music	Temp
Alice	on	76^oF
Bob	-	76^oF
Carol	off	70^oF

  If both Alice and Bod is at home, the smart-home keeps the music on and set the room temperature at 76^oF. However, if Carol return home, we need to turn-off music and turn-down temperature a little bit.

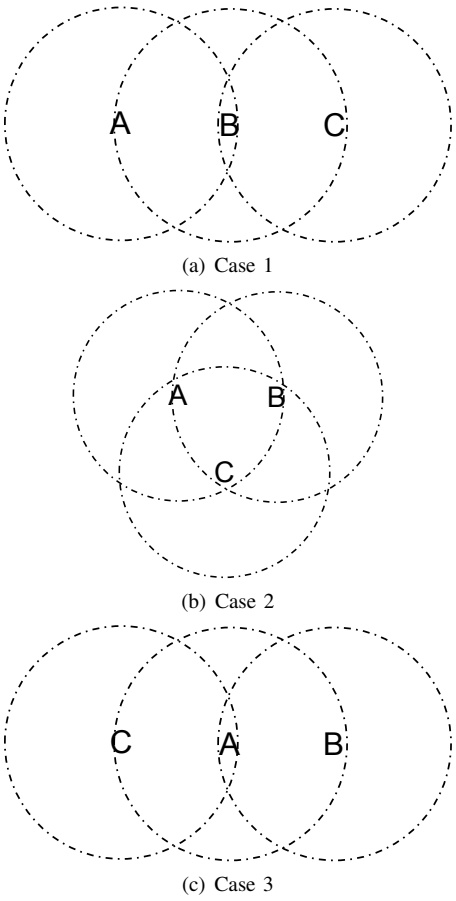
- File sharing
  File sharing is a common and useful application nowadays. Airdrop by Apple is a good example. It is an ad-hoc service in Apple Inc.'s OS X and iOS operating systems that enables users to transfer files to another supported Mac computer and iOS mobile device without using email or a mass storage device. Again, the basic of file sharing is to discover nearby host and reach mutual agreement on the group membership.

B. Why the protocol from class doesn't work?

One of the fundamental difference between the system in class and our system is that we don't have a wired infrastructure.

The unreliable wireless network with frequent disconnection make it impossible to set a bound on the transmission time. In other word, the communication system now suffer omission failure instead of performance failure. This makes the time-out mechanism of the protocol from class less useful in the new mobile system. However, this kind of link-failure can be marked with low layer retransmission. The only different is that we might have a much longer timeout time.

The real issue is that the topology of the system might change with the movement of the mobile host while the group membership may not always change. Consider the following example:



(a) Case 1

(b) Case 2

(c) Case 3

Fig. 3: Different configuration of mobile host with same group membership

In the above three different cases, the group membership is exactly the same, $\{A, B, C\}$. However, for

each case, each host's view of its own neighbor is different.

	Case1	Case2	Case3
A	$\{B\}$	$\{B, C\}$	$\{B, C\}$
B	$\{A, C\}$	$\{A, C\}$	$\{A\}$
C	$\{B\}$	$\{A, B\}$	$\{A\}$

TABLE I: View of neighbor host of each host

C. Problem definition

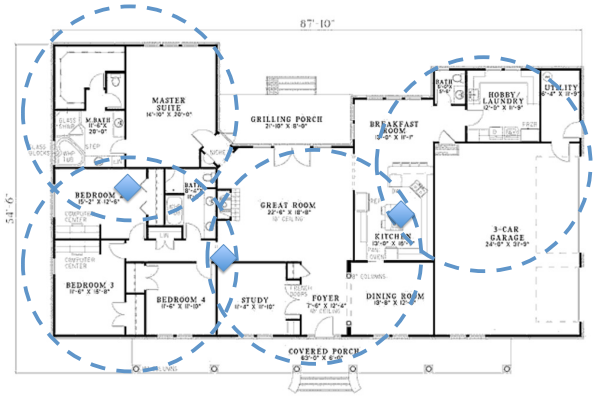
Our goal is to provide upper layer application with a consistent global data structure in a setting in which mobile hosts come and go as they please and engage in transient collaborative activities [ref].

D. Protocol

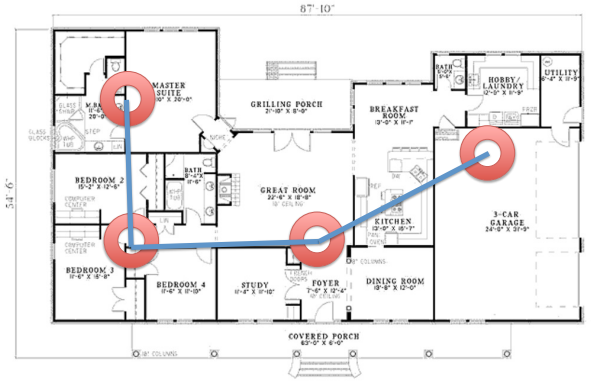
1) Subgroup formation: Due to the limit on communication range, instead of forming a big group all at once, we form some local subgroups first. It is very easy for each node to discover its neighbors with wireless beacon. In our subgroup formation protocol, the node with smallest ID (can be MAC address) stands out and acts as group leader. For other nodes with larger ID number, there might be two cases. The first case is an easy one. If a node only receive one subgroup formation request, it simply joins that subgroup. In the second case, where a node receives multiple subgroup formation requests, things are more complicated. The presence of multiple subgroup requests indicates that the given node is in the intersection of multiple groups and the group leaders can't communicate with each other. Otherwise, they should have been in the same group already. In this case, the node will join the group with smaller leader ID. It will also remember the two closest groups and act as a bridge between them later on.

Thus, we can summarize the subgroup formation protocol as follow:
Each node checks the ID of all its neighbors

- If the node's ID is smaller than all its neighbors ⇒ Send out subgroup formation request with its ID.
- Otherwise
  - If receive only one subgroup formation request ⇒ Join that subgroup
  - If receive multiple subgroup formation requests ⇒ Remember the two requests with highest RSSI (received signal strength indicator) ⇒ Join the group with lower ID among the above two request.



(a) subgroup view



(b) logic group view

Fig. 4: Smart home example

2) Logic Group Formation: When the subgroup formation is done, the bridge node (the node in the intersection of two subgroups) can initiate logic group formation to connect every subgroup into a big logic group. In the logic group, we can view every subgroup as vertice and bridge node as an edge connecting two subgroups. Within a single house, the logic graph won't be large. In the following example, we present both the subgroup and logic group view of one smart home application. The bridge node sends CONNECT request

to both subgroups in its neighbor list. The subgroup leader need to respond to this request. As there might be more than one bridge nodes between two subgroups, both subgroup leaders will send ACK to the bridge node with lowest id and NACK to other bridge node. The bridge node may view timeout as an implicit NACK.

3) Group Membership Query: As a wireless distributed system is more dynamic than a wired one, it is really hard to keep the group membership information updated within each node. Actually, this kind of real-time group membership is not required as the not all the nodes need this information all the times. We adopted stateless group membership in our protocol design. That is, instead of storing group members in every node, we only store subgroup membership in the subgroup leader. In the case that some node need the global membership (membership of the nodes of the entire home), it would send MEMBERSHIP query to it's subgroup leader and the subgroup leader would gather that information from all the subgroup leaders through bridge nodes and return it to the original node. In this way, we simplify the group membership problem to subgroup membership problem, which is much easier to solve.

4) Node Joining: The case of joining is easy to handle. We don't want the new node to change the overall structure of the logic group. Thus, the new node just select the subgroup leader with the largest RSSI and join that group. In the rare case where no subgroup leader is visible. It will form a single node subgroup itself and and some other node may become neighbor node.

5) Node leaving: The case of node leaving the group is more complicated. We need to consider three different cases:

- Normal node
  This is the most straight forward case, as we only need to delete it from subgroup leader's list.
- Bridge node
  In the case that a bridge node fails, both the subgroup leaders will realize it as the bridge node will no longer show up in its subgroup member list. Thus, the leader sends BRIDGE request to all of its group members. Any node receive both BRIDGE requests may respond with CONNECT and the rest is similar to Logic group formation. If no such bridge node is found, the two subgroup is not directly connected in logic group graph.
- Subgroup leader
  In the case of the failure of subgroup leader, we'll run the group formation protocol again and rebuild the group structure.

VI. Summary

In this paper, we discussed how to build a dependable smart home system. Our discussion focused on there aspect: Clock Synchronization (by Swadhin Pradhan), Atomic Broadcast (by Qiren Chen) and Group Membership (by Xiaofan Lu). Due to time limit, we were unable to provide formal proof on our protocol. We would like to leave the proof as a future work.

Acknowledgments

We would like to thanks Professor Aloysius K. Mok for his guidance on our project. We learnt a lot about Dependable Computing System through the lecture.