FAQs

Sayed Ahmed and Dinesh Bhat

May 16, 2004

Course: 74.783 Mobile Networking

Project: Time Synchronization in Sensor Networks Using Flooding Protocol

Email IDs: sayed@cs.umanitoba.ca & bhatdinu@cs.umanitoba.ca

Instructor: Dr. Jelena Misic

1. What is the component that the beacon (reference) broadcaster runs?

Answer: BeaconBroadcasterC

2. What is a Base Station component?

Answer: BaseAppC

3. What are the components that run on the clients?

Answer: All clients as well as the root run FTSPAppC which includes both FTSPDebuggerC and FTSPC components. The implementations of these components are provided in FTSPDebuggerM.nc and FTSPM.nc file. FTSPDebuggerC is a component which works in parallel with FTSPC component in the FTSPApplication module. FTSPDebuggerC and FTSPC both are compiled automatically in each FTSPAppC application (i.e. when FTSPAppC is compiled)

4. What does FTSPDebuggerM do?

Answer: As said before FTSPAppC uses FTSPDebuggerC component. As seen in the FTSPDebuggerM.nc file, a task sendstatistics() procedure is using DiagMsg interface, that is responsible for sending the data to the base station from the client at the receive of BeaconBroadCaster message.

5. What are the components that run on the root?

Answer: Same as the components that run on the clients. See 3.

6. Who is the root?

Answer: The root is always the node with the smallest node-id participating in FTSP. Nodes are explicitly programmed with node -id. The node that is programmed with the lowest node-id becomes the root.

7. I plan to work in two steps.

1) place a base station and some clients(i.e. single hop).

2) A Multi hop environment.

What are the considerations?

Answer: If you want to work one hop, please make sure that you don't define MULTIHOP variable, which is enforcing multi-hop routing based on node ID (as described in the installation manual). For multihop, you need to either move motes far from each other, to get them out of radio range, or define MULTIHOP, which allows to simulate multihop (by throwing out radio messages based on node ID) - you need to assign the node IDs to the motes in a special way - please refer to the installation manual.

8. what is -DMULTIHOP in Makefile?

Answer: MULTIHOP -DMULTIHOP means that the local addresses of the motes can decide whether 2 motes can communicate together in FTSP

protocol. FTSP assumes the addres is given as XY where XY are (x,y) coordinates of a mote. The two motes can then communicate with each other only if the distance between 2 motes is = 1. This is good for multihop testing

9. Facts about clocks

Answer: First of all, there are 2 possibilities for TimeStamping - using the slow clock(32kHz) or the fast clock(7MHz). We use the fast clock. The needed module is found under tinyos/system directory. tinyos-1.x/tos/platform/mica2/SysTimeStmpingM.nc is used in our program.

In short, the purpose of TimeStamping code is to accurately timestamp the arrival and the transmission of a radio message. The radio chip does not fire an interrupt soon after the message comes - interrupt comes only after some delay. We need to correct this delay - therefore, we need the BIT_CORRECTION[] array. Obviously, this delay depends on the speed of the radio communication and the frequency of the clock as well as bit offset of the receiver from the sender. Therefore, these values need to be calibrated, if you want to use different radio settings or clock settings.

10. What is DiagMsg?

Answer: DiagMsg is a tinyos service that encodes the type of data along

with the data itself in a radio message. It is then possible to decode this data automatically e.g. a java program running on your laptop can show them in the correct form (eg. double is encoded into 4 bytes, transmitted by a mote and then decoded and shown as double at java side). Of course, you need to run proper software at java side that connects to a base station. (you can find more in tinyos-1.x/tos/DiagMsg/DiagMsg.txt). Our ShowTimeSyncStat application that connects to a base station and report the data in a window.

If you need to know how DiagMsg works, you can check report() task in FTSPDebuggerM.nc, which defines the data transmitted: the address of a mote, ID of timesync msg, local and global time. For the analysis, (1)TOS_LOCAL_ADDRESS, (2) global and (3)synced are the most useful data that get transmitted. synced indicates whether the mote is synchronized (if it's 0, you should not consider the data to be valid). Also, the msgID can be helpful if you have many poller rounds and many motes and you want to do statistics per round (msgID is the same for all motes in the same round).

11. Is it that when a client receives a message from root, it calculates the difference value and stores in the table?

Answer: Yes, (1)a message arrives, (2)the timestamp of arrival of message

is stored and task processMsg() is posted, (3)processMsg() checks if the mote needs to store synchronization points, checks if the incoming synch_point is new (i.e. has not been stored before) and then stores the synchronization point which is a pair: [local time, offset of this local time from the global time].

12. Do clients send global time estimate with time stamping when they get the polling messages?

Answer: Yes, this is just sending debug information to the PC. The poller msg comes to all the FTSP motes at the same time. All of these motes timestamp the poller message with global time and report the timestamps. The received time stamps should be the same.

13. What is the purpose of precision timer?

Answer: It is needed for the proper functioning of TimerC.nc component together with TimeStamping. It provides high precision of TimeStamping.

14. What is linear regression function is used in the program to calculate skew?

In our case, linear regression equation is y = m*x + q, where m = skew, and q = offset

y=time_offset and x=local_time in our case (so if we are given local_time

we can compute time_offset).

15. What are the functions of interfaceFTSP.localToGlobal and interfaceFTSP.globalToLocal procedure?

Answer: Each mote keeps track of the skew and offset of its local time vs the global time. For this, it has the skew and offset variables. We assume that the drift of the local clock against global clock is linear. Therefore, the equestion is, global_time = skew*local_time + offset holds. Skew can be assumed to be a value very close to 1 (as explained before that the skew is typically 1 unit into 1 million). However, the precision of float is limited and is better, if the skew is close to 0 rather than 1. Subtracting skew by one gives the equation for timeOffset (=global-local): timeOffset = skew'*local_time + offset then we count global = local + timeOffset and local = global - timeOffset in the two functions.

16. In the code for root, in case of message sending, a calculation is done as follows

if ((int32_t)(localTime - localAverage) >= 1073741824) {} what does it mean and why?

Answer: The root does not update its localAverage time, therefore after some time localTime - localAverage becomes large which is causing loosing of the precision of representation, therefore we need to recalculate

localAverage time to time even for the root.

17. outgoingMsg → sendingTime = globalTime - localTime; why??

Answer: This field is initially set to the offset between global time and local time. The TimeStamping component adds the current local time when the message is actually transmitted. Thus, the receiver will receive the global time of the sender when the message is actually sent.

18. Do I need skew? I guess all of my motes are operating in the same clock?

Answer: You need the skew. It is necessary because the crystals at the different motes have small differences in frequencies. Even though the differences are small, our clock precision is in microsecond range and therefore, we need to compensate for it. The specifications for MICA2 says that CPU frequency should be 7.368 MHz. In reality, they have different values. This means that this small error in frequencies would transform into quite a big errors over time (like 2 frequency ticks per 1 second). We want to compensate for this and therefore, we need skew.

19. Should the globaltime and local time in output be equal. I ran ShowTimeSyncStat and I got them to be equal for multihop,

but for single hop thing the values were different?

Answer: Most probably the mote IDs of your motes do not correspond to what they should. If you don't program mote IDs carefully, you get disconnected network and then each subnetwork will have its own root. If the global time is equal to the local time and time sync bit is 1, than the mote is the root. This means that the root has never heard sync message from a mote with smaller ID than itself (however if a mote is a root, then, global time is not necessarily be equal to its local time).

20. Sometimes skew is not 0 but synchronized bit is shown as 1?

Answer: This means that the mote is the root.

21. I think you never changed the system time (hardware time)?

Answer: This is correct, we never touched the hardware time, we just keep track of offsets.

22. What does calculateSkewOffset() procedure in FTSPM do?

Answer: Motes keep track of the offset and skew of their local clock with respect to the global time. They calculate the skew and the offset from synchronization points (global_time, local_time) obtained over last couple of rounds. Motes stores last 8 sync points in a table and every time it receives a new sync point, it removes the oldest point from the table and

inserts a new point in the table (function addNewEntry is doing this). However, after changing the values in the table, the skew and offset need to be recalculated (linear regression). calculateSkewOffset does this job.

23. In your implementation, is it that both root and the poller are broadcasting messages in every 30s?

Answer: Actually, all the motes broadcast time_sync messages with 30s period, not only the root. you can change this period in makefile by defining TIMESYNC_INTERVAL (-DTIMESYNC_INTERVAL=60 would change it to 1 min) or similarly for the poller, by defining TIMESYNC_BEACON_INTERVAL

24. When we read time from hardware. is it that we get answer in noofclocks? can we convert the time in hh:mm:ss format?? is it that for 32KHZ if we divide the time with 32768 then total no of sec is got?

Answer: Yes, time unit of the value which you get from read time is in clock ticks: - for 32kHz clock, one unit is 1/32768 sec = 30.5 microseconds - for 7 MHz clock, SysTimeC provides the clock and it actually provides 921.6 kHz timer for Mica2 (1 unit is 1 microsecond) or 500kHz time for Mica2Dot (1 unit is 2 microseconds)

so yes, you need to divide the time by 32768 for the slower clock, and by 10^6 for the faster clock to get the time in seconds.