Modeling and Simulation Comparison of Two Time Synchronization Protocols

Tsung-Han Lin[a], Keng-hao Chang[e], Jr-ben Tian[d], Hao-hua Chu[bd], Polly Huang[abc]

[a]Department of Electrical Engineering
[b]Graduate Institute of Networking and Multimedia
[c]Graduate Institute of Communication Engineering
[d]Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan

[e]EECS, University of California at Berkeley, Berkeley, US
b90901046@ntu.edu.tw, kenghao@gmail.com, r93045@csie.ntu.edu.tw, hchu@csie.ntu.edu.tw, phuang@cc.ee.ntu.edu.tw

ABSTRACT

To infer correctly application semantics, sensor network applications often need accurate times on observations that are reported from distributed sensor nodes. Since the nodes' local clocks can go out-of-sync due to clock drifts, a networked time synchronization protocol is needed to synchronize their clocks to a reference clock. This paper provides performance modeling and comparison between two time synchronization protocols: TPSN clock synchronization (clock-sync) and TSS event synchronization (event-sync). Their main difference is that the TPSN clock-sync synchronizes all nodes' local clocks to a global reference clock, whereas TSS event-sync synchronizes events' generation times from different local nodes to their sink nodes' clocks. Although these two time synchronization protocols have their respective limitations in application scenarios, they are comparable in that they also share a large domain with none of these limitations. This paper evaluates these two protocols by considering different ad-hoc network sizes, node mobility levels, and traffic volumes. In order to fully understand the tradeoffs between these two time synchronization protocols, we have derived analytical models on their performances and conducted simulations to measure the impact of these variables. Both the simulation results and analytical models show that (1) event-sync provides much better accuracy than clock-sync, (2) under very high node mobility level, clock-sync may achieve better accuracy than event-sync, and (3) under increasing traffic volume clock-sync scales better.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks – *Distribute Systems*

General Terms

Performance

PM2HW2N'08, October 31, 2008, Vancouver, BC, Canada.

1. INTRODUCTION

Many recent works have devised very creative and successful applications using wireless sensor networks (WSNs) to address a wide array of real-world problems [1]-[3]. In such networks, distributed sensor nodes are used to observe various aspects of the physical world. In order to correctly infer application-level semantics from the sensor data, accurate timing information needs to be attached to observations reported from these distributed sensor nodes [4][5].

In the wide area network (WAN), time synchronization protocols, such as the widely used Network Time Protocol (NTP) [6] have been extensively studied. However, in the wireless sensor network (WSN), these WAN time protocols, e.g., NTP, cannot work well due to different network characteristics and requirements, such as limited energy, limited bandwidth, constrained hardware (e,g., may not be possible to strap a GPS receiver on a small sensor node), and unstable network [4].

In WSN, the most commonly used approach is to synchronize sensor nodes' local clocks to a global reference clock. In this paper, we refer to this class of synchronization mechanisms as clock synchronization (clock-sync). However, not all applications require their nodes' local clocks to be synchronized to a global clock. For example, if we only need to know the relative time of the observations, it is sufficient to synchronize the timestamps among these observations at a sink (or gateway) node to correctly infer application semantics. This class of time synchronization methods is called event synchronization (event-sync). The clock and event synchronization mechanisms are different from the event order synchronization mechanisms such as Lamport's logical clock scheme [7] where the finer-grained timing information is no longer available.

These two classes of time synchronization have different assumptions on and limitations for applications. For examples, the clock-sync does not work in a sparse wireless sensor network in which the sensor nodes are not always fully connected [4]. On the other hand, the event-sync does not provide a global reference clock to applications. Despite their differences, these two classes of time synchronization protocols share a large domain of sensor network applications, in which the networks are not sparse and

the knowledge of relative event time is sufficient. In some sensor network applications [3], there are a limited number of more powerful sink nodes that collect observations from sensor nodes. Then, they execute application logics on these observations to process temporal information. In these applications, it is sufficient to synchronize the timestamps contained within observations according to the sink node's clock, rather than to synchronize the sensor nodes' clocks to the sink node's clock or a global clock. Under this common application domain, these two classes of time synchronization mechanisms are both applicable. Therefore, it becomes meaningful to understand and compare their performance tradeoffs. This helps application developers choose the appropriate class of time synchronization under different network and traffic scenarios.

One synchronization protocol from each class was selected: TPSN [8] represents the clock-sync class and TSS [9] represents the event-sync class. We present a thorough analysis and comparison between the two protocols. From the analytical models and simulation results, we are able to quantify the impact of different network and traffic factors on the synchronization error.

2. MECHANISMS and ANALITICAL MODELS

2.1 TPSN (Clock-sync) Protocol Mechanism

TPSN [8] has two phases in its process: "level discovery" and "synchronization". A hierarchical structure with a root node is first created in level discovery phase. Then in synchronization phase, nodes synchronize their clocks to the root node's clock using the hierarchical structure constructed earlier.

1) Level Discovery Phase: This phase of TPSN happens at the beginning, after the network has been setup. To start, the root node assigns itself a level 0 and broadcasts a *level_discovery packet*. This packet holds the node identity and the level number of the root node. When its neighbors receive this packet, they assign themselves a greater level number than received in the packet, say level 1. Then they continue to broadcast *level_discovery packets* with their own node identity and level number. This process lasts until every node in the network is assigned a level number. At the end of this phase, a hierarchical structure with a root node is created for use in the next phase.

2) Synchronization Phase: The root node starts this phase by broadcasting a *time_sync packet*. Upon its reception, the nodes on level 1 wait for a random time then send a *synchronization_pulse packet* to the root node. The randomized waiting prevents collisions caused by contention for *media* access. The root node replies accordingly with *acknowledgement packets*. Therefore, all nodes belonging to level 1 can correct their clocks according to the clock of the root node. In addition, the nodes on level 2 will overhear the message from at least one neighbor on level 1. Consequently, the nodes on the next level will each send a *synchronization_pulse packet* to their level-1 neighbors for synchronization. This is applied recursively until every node in the network has its clock synchronized to the reference clock of the root node.

In this phase, pair-wise synchronization is achieved across the edges of the hierarchical structure built in the previous phase. As depicted in Fig. 1, there are two nodes called N_i and N_{i-1}. t_1 and t_4 are the times measured according to node N_i's local clock; t_2 and t_3 are the times measured according to node N_{i-1}'s clock. At time t_1, node N_i sends a *synchronization_pulse packet* to node N_{i-1}. The *synchronization_pulse packet* holds the value of t_1. Node N_{i-1} receives this packet at t_2, where t_2 is equal to $(t_1 + \Delta + d)$. Δ represents the clock drift between the two nodes, and d represents the sending delay (including the time to send, propagate, and receive the packet). At time t_3, node N_{i-1} sends back an *acknowledgement* packet to node N_i. This acknowledgement packet holds the values of t_1, t_2, and t_3, and node N_i receives the packet at t_4. TPSN assumes the delays of *synchronization_pulse* packet and *acknowledgement* packet are the same, so t_4 is equal to $(t_3 - \Delta + d)$. Assuming that the clock drift and the propagation delay do not change in this small period of time, node N_i can calculate the clock drift and propagation delay using the following formula:

$$\Delta = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}; \quad d = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \qquad (1)$$

Node N_i can therefore synchronize its local clock to N_{i-1}'s since it has information about the clock drift between them.

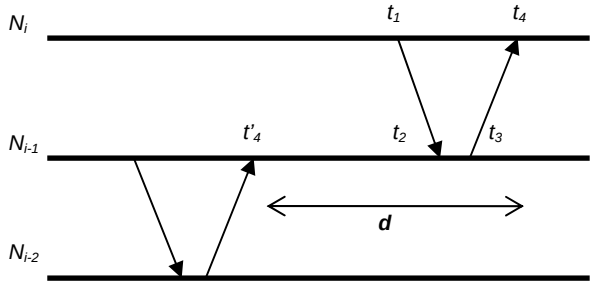

Figure 1. Pair-wise Synchronization of TPSN. t_2 and t_3 are measured in node N_{i-1}'s clock, and t_1 and t_4 are measured in node N_i's clock.

2.2 TPSN Analytical Model

As shown in Fig. 2, TPSN synchronization error is composed of three error components: *pair-wise synchronization error (E_{sync})* caused by the delay estimation when a parent node is exchanging the global clock value with its one-hop child nodes, *external clock skew error (E_{ext})* caused by clock skews of intermediate nodes on the transmission path while they are forwarding the synchronization message from the root node to the target node, and *internal clock drift error (E_{int})* caused by clock skew of the target node as its local clock drifts away from the most recent synchronization time point:

$$E^{tpsn} = E_{sync} + E_{ext} + E_{int} \qquad (2)$$

The pair-wise synchronization error *(E_{sync})* comes from TPSN's assumption that sending delay of *synchronization_pulse packet* and that of *acknowledgement packet* are the same. However, in real deployment, the forward and reverse link delays can be asymmetric. This leads to incorrect calculation of the clock drift value, i.e., Δ in Equation (1). Denote the average time difference between the forward and reverse link delays as u. At the end of each pair-wise synchronization (i.e., t_4 in Fig. 1), the asymmetric link delay will cause the clocks between the parent and child nodes to be off by $u/2$ (on average). Consider a target node at level l, since there are l number of such pair-wise synchronizations occurred between pairs of intermediate nodes on

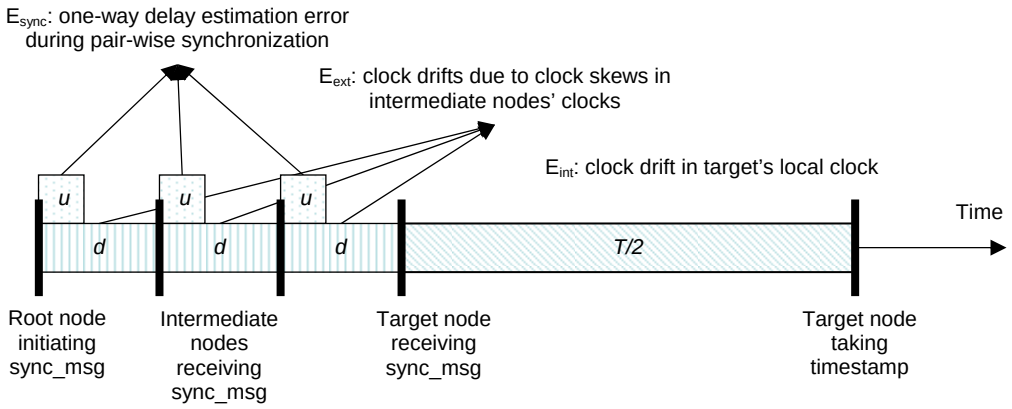

Figure 2. TPSN synchronization error decomposed into three error components

the path between the root node and the target node, the total synchronization error is the sum of all pair-wise synchronization errors on that path. It can be written as follows.

$$E_{sync} = \frac{l \cdot u}{2} \qquad (3)$$

The external clock skew error *(E_{ext})* is caused by clock skews of *intermediate nodes* as the sync message (containing the global clock) is pushed from the root node down the hierarchy to target nodes. Since each pair-wise synchronization takes some amount of processing and transmission time, this hop latency needs to be accounted for by each intermediate node using its local clock, added to the global clock, and then passed it down the hierarchy. This clock skew error is *external* in the sense that the error is not contributed by the target node, but rather clock skews from these external intermediate nodes. Denote the average clock skew (i.e., clock drift rate) on any non-root nodes as *r*. Denote the average hop latency time for a pair-wise synchronization as *d*. At the end of each pair-wise synchronization, clock skew will cause the clock on an intermediate node to drift apart on average by (*d* r*) from the global clock. Consider a node on level *l*, since there are *l* number of pair-wise synchronizations occurred on the path between the root node and the target node, the external clock skew error is the sum of clock skews on that path.

$$E_{ext} = l \cdot d \cdot r \qquad (4)$$

The internal clock drift error *(E_{int})* is caused by the drift of the clock on the *target node* between the current time and the most recent synchronization time point. This clock drift error is called *internal* in the sense that the error is contributed solely by the target node's local clock. Denote this synchronization time interval as *T*. If the data generation time occurs uniformly over this time interval, the average amount of clock drift away from the global clock (since the last synchronization time point) can be derived as follows.

$$E_{\text{int}} = \frac{r \cdot T}{2} \qquad (5)$$

The aggregate synchronization error for TPSN (E^{tpsn}) is the sum of these three error components. Combining Equations (3), (4), and (5) gives the following equation for E^{tpsn}.

$$E^{tpsn} = l\,(\frac{u}{2} + r \cdot d) + \frac{r \cdot T}{2} \qquad (6)$$

2.3 TSS (Event-sync) Mechanism

Rather than synchronizing every node's clock to a global clock, TSS [9] estimates and accumulates the hop-by-hop latency. When a data packet arrives at the sink, the packet generation time relative to the sink's clock can be traced back from the accumulated end-to-end latency. TSS then determines the relative data generation time to the sink's clock. Wireless links among the nodes are assumed to employ a CSMA/CA-like MAC-layer mechanism where an acknowledgement is sent for each data to assure the reception of the data packet. The hop latency, *d*, can be estimated using the following formula.

$$d = (t_4 - t_1) - (t_3 - t_2) - Est_{D_{ACK1}} \qquad (7)$$

As depicted in Fig. 3, t_4 - t_1 can be obtained using the receiver's clock and $t_3 - t_2$ from the sender's clock. The value of $t_3 - t_2$ can be piggybacked on the Data2 packet to the receiver. $Est_{D_{ACK1}}$ is the estimation of the delay of ACK1. We can use the transmission delay to estimate this value, as in (8).

$$Est_{D_{ACK1}} = \frac{ACK1\_packet\_size}{bandwidth} \qquad (8)$$

With the above information, the hop latency *d* of Data2 at the

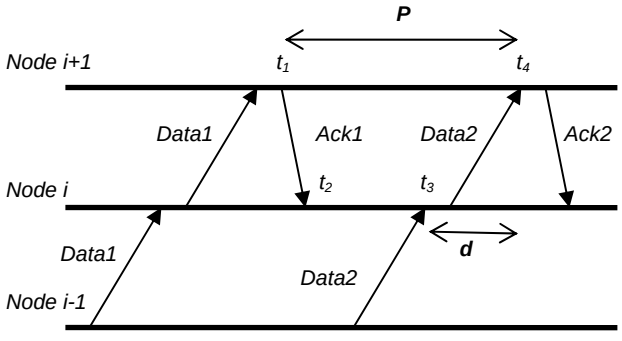

Figure 3. Event- Synchronization of TSS. ACK1 departs at *t1* and arrives at *t2*. Data2 arrives at the sender at *t3*, and arrives at the receiver at *t4*. *d* is the hop latency of Data2 at the sender**.**

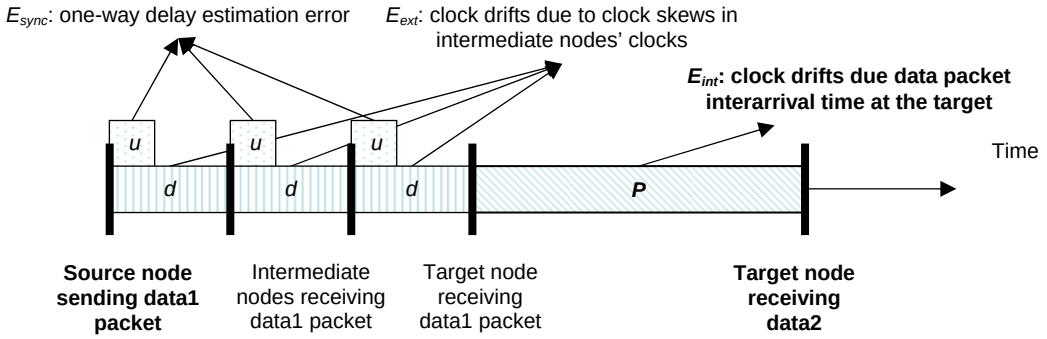

Figure 4. TSS synchronization error decomposed into three error components. Note that the differences between TSS's error components (shown here), and TPSN's error components (shown in Figure 2) are highlighted in bold letters.

receiver node can be calculated. Immediately, the latency can be accumulated and carried along with the data packet. To estimate the hop latency, each node needs to keep two extra states: the ACK departure time and the ACK arrival time of the latest data packet. When a data packet is ready to be sent but cannot find the state information to be piggybacked, e.g. the first flow of packets, an additional overhead will be sent in order to set up the state information.

2.4 TSS Analytical Model

The error model of TSS is similar to that of TPSN, composing of three components shown in Fig. 4: *pair-wise synchronization error (E_{sync})* caused by the delay estimation on its transmission path, *external clock skew error (E_{ext})* caused by clock skews of intermediate nodes while they are forwarding data packets from the source node to the sink node, and *internal clock drift error (E_{int})* caused by the clock skew of the target node as its local clock drifts between the arrival time of a packet and the arrival time of its subsequent packet carrying its end-to-end delay.

$$E^{tss} = E_{sync} + E_{external} + E_{internal} \qquad (9)$$

The pair-wise synchronization error comes from hop delay estimation in TSS. TSS assumes that the ACK packet transmission time (t_2-t_1 in Fig. 3) can be estimated according to Equation (8). However, in real deployment, other small delay factors such as the protocol processing time are not considered. As a result, TSS has the delay estimation error. Denote the average delay estimation error as u. Consider a data packet with a path-length of l between a source node and a sink node. Since there are l numbers of such forwarding hops, the total pair-wise synchronization error is the sum of all pair-wise synchronization errors on that path. It can be written as follows.

$$E_{sync} = l \cdot u \qquad (10)$$

The external clock skew error (E_{ext}) is caused by clock skews of intermediate nodes as the data packet is forwarded from a source node to a sink node, and its per-hop delays are accumulated in the subsequent data packet. Denote the average per-hop delay time as d. At the end of each hop transmission, clock skew will cause the clock of an intermediate node to drift apart on average by ($d*r$) from the global clock. Consider a data path of length l. The total external clock skew error is the sum of individual clock skew error over these l intermediate nodes. It can be written as follows.

$$E_{ext} = l \cdot d \cdot r \qquad (11)$$

The internal clock drift error *(E_{int})* is caused by the clock drift of the *sink node* between the arrival time of a data packet (e.g., *Data2* packet) and the arrival time of the previous data packet (e.g., *Data1* packet) carrying the accumulative hop-by-hop delay of the previous packet. This clock drift error is internal in the sense that the error is contributed solely by the sink node's local clock. Denote the average inter-arrival time of a packet stream as P. The amount of clock drift from the packet inter-arrival times can be derived as follows.

$$E_{\text{int}} = r \cdot P \qquad (12)$$

The aggregate synchronization error for TSS (E^{tss}) is the sum of these three error components. Combining Equations (10), (11), and (12) gives the following equation for E^{tss}.

$$E^{tss} = l\,(u + r \cdot d) + r \cdot P \qquad (13)$$

3. TYPESET TEXT

TPSN and TSS were implemented on the *ns-2* simulator. We describe the details for the simulation setup, evaluation metric, and evaluation variables. Based on the analytical models derived in Section 2, we validate the analytical models and analyze the impacts of changing these evaluation variables on the evaluation metrics

3.1 Simulation Design

In all simulations, the sensor nodes are placed on a predefined grid in a uniformly random fashion. The data sink is fixed in one corner of the grid, while other nodes are randomly chosen as data sources. Each node has a constant clock skew rate selected from $0.5*10^{-6}$ to $1.5*10^{-6}$ seconds, which is applicable to common crystal oscillators in markets [5]. Other setup aspects include directed diffusion [10], a well-known data-centric routing mechanism, and IEEE 802.11, a popular wireless link technology.

For all of the evaluation parameters, the base case is defined to have 40 nodes on an 80x80m^2 grid with 40 meters communication range. 10 of these sensor nodes are data sources. Each source sends a 100-byte data packet every 5 seconds. Unless specified otherwise, these are the default values for the parameters.

We compare TPSN and TSS based on one metric, *Synchronization Error*. This represents the difference between actual data generation time and estimated data generation time. To understand the scaling behavior of the two time synchronization mechanisms, scenarios are simulated with varying *network sizes*, node *mobility levels*, and *data rate*s.

- *Network Size*: The number of nodes is changed from 20 to 140 with incremental steps of 20 nodes. In order to fix the network density with increasing number of nodes, the grid size is configured accordingly.
- *Node Mobility*: Each node has a randomly generated target location and moves to that location with a random speed (maximum speed 10*m/s*). To change the levels of node mobility, the pause time between the target locations is adjusted from 0 to 400 seconds. A smaller pause time means higher mobility.
- *Data Rates*: We vary the packet sending rates from a fixed-size packet every 4 (2^2) seconds to every 0.015625 (2^{-6}) second.

3.2 Simulation Results

Fig. 5 shows the synchronization error decomposed into three individual error components for the TPSN and TSS under different network sizes, node mobility levels, and data rates. Each plot contains the following five lines:

- Line (1) shows the actual simulation results of E^{tpsn} and E^{tss};
- Lines (2-4) show the expected E_{sync}, E_{ext} and E_{int} obtained by applying the simulation's values to *Equations (3) ~ (5)* and *(10) ~ (12)* in the analytical models; and
- Line (5) shows the expected synchronization errors $E^{tpsn'}$ and $E^{tss'}$ as the sums of the above three error components. Note that they are expected values different from the actual values (E^{tpsn} and E^{tss}) measured in the simulation.

In order to compute the expected synchronization error from the analytical models, it requires plugging in the correct values for parameters (*u, l, P, T,* and *d*) in Equations (6) & (13). The correct values for these parameters: *u* (the average hop delay estimation error) and *r* (the average clock skew rate) can be obtained directly from the network topology and traffic volume settings in our simulation scenarios. The correct values for the parameters: *l* (the average node level in TPSN or the average path length in TSS), *d* (the average elapsed time for pair-wise synchronization), *T* (the synchronization period in TPSN), and *P* (the average packet inter-arrival time in TSS) can be observed during simulation.

3.2.1 Impact of network size on accuracy

Fig. 5(a) shows the synchronization error of TPSN under different network sizes. The actual synchronization error from the simulation is shown as line (1), and it exhibits a growing trend as the network size increases. This is expected given that a larger network size implies a higher level of the clock synchronization hierarchy, therefore, lengthens the path for a sync message to travel from the root node to a target node. The expected synchronization error derived from the analytical model is shown as line (5). Close similarity between lines (1) and (5) shows that simulation results are consistent with our TPSN analytical model. E_{int} is the dominant component in the overall error. It is shown as

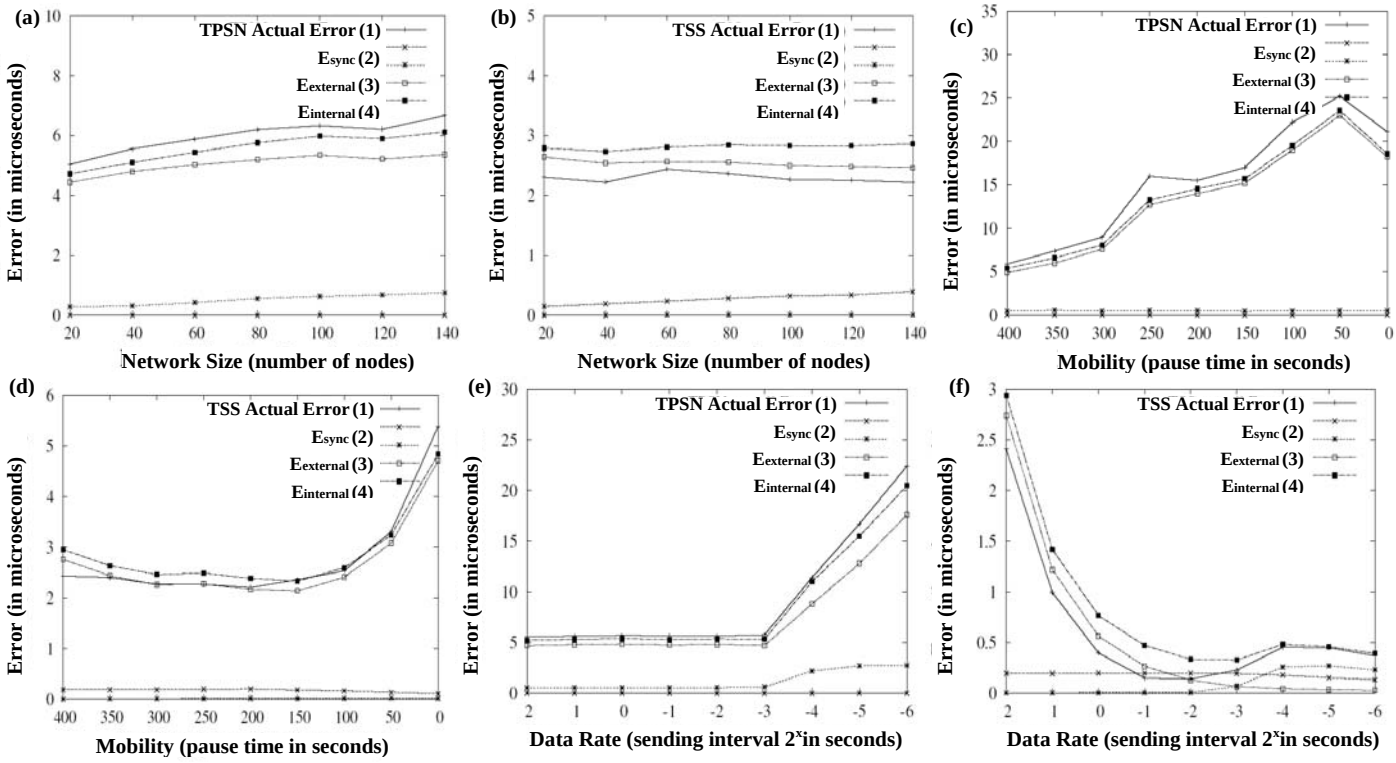

Figure 5. Graphs in (a), (c), and (e) represents synchronization errors for TPSN under the following dynamic factors: different network sizes, node mobility levels, and data rates. Graphs in (b), (d), and (f) represent synchronization errors for TSS under these dynamics factors. Note that the actual simulation errors (E^{tpsn} and E^{tss}), shown as line (1), can be smaller than the expected synchronization error ($E^{tpsn'}$ and $E^{tss'}$), shown as lines (5). The reason for this small difference is that the expected synchronization error is derived from analytical models that use observed values from the simulation. This small difference shows that our analytical models are consistent with the simulation results.

line (4) in Fig. 5(a). It exhibits a growing trend with increasing network size. This is somehow unexpected by our analytical models, given that the clock skew rate (*r)* and synchronization period *(T)* are fixed in the simulation setup. After analyzing the simulation results, we find that when the sync packets are lost, the actual synchronization period can be double, triple, or more.

Fig. 5(b) shows the synchronization errors of TSS under different network sizes. The actual synchronization error from the simulation, shown as line (1), matches well with the model, shown as line (5). In other words, simulation results are consistent with our TSS analytical model. Similarly, we separate the error of TSS mechanism into three components (E_{sync}, E_{ext} and E_{int}). E_{int} is not affected by the changing network size, because the network size has no effect on any factors in Equation (12). Although both E_{sync} and E_{ext} grow with increasing network size, their scales (< 0.1 microsecond) are one order of magnitude smaller than E_{int} (2~3 microseconds). As a result, the overall TSS error is dominated by the flat E_{int}.

3.2.2 Impact of node mobility on accuracy

Fig. 5(c) shows the synchronization error of TPSN under different node mobility levels. The actual synchronization error from simulation exhibits a growing trend as node mobility level increases. E_{int}, again the dominant component, exhibits a growing trend with increasing levels of node mobility. Higher node mobility implies higher probability of broken hierarchy, which in turn leads to higher probability that nodes may fail to be synchronized with upper level nodes, and raises the average synchronization time interval *T*. An interesting phenomenon in TPSN is that when the node mobility level moves to the high end of the extreme (pause times = 50~0 seconds), the synchronization error actually *falls*. Since high mobility enables the root node to encounter more nodes during movements; therefore, mobility helps to spread the root node's global clock to other nodes. This is an example where mobility can sometimes be a positive factor.

Fig. 5(d) shows the synchronization error of TSS under different node mobility levels. E_{int} exhibits a growing trend with increasing node mobility. This growth is mainly due to a larger data packet inter-arrival time (*P)*. The rise in packet inter-arrival time is caused by frequent switching of routing paths. Switching of routing paths is a result of intermediate nodes repeatedly moving in and out of routing paths, making use of previously established time-sync states on nodes. Since E_{int} is the dominant factor, the overall synchronization error exhibits the growing trend.

3.2.3 Impact of data rates on accuracy

Fig. 5(e) shows the synchronization error of TPSN under different data rates. The actual synchronization error from the simulation exhibits a two-step behavior. These two steps can be explained as follows. High traffic volume eventually leads to network congestion and packet loss. In the simulation scenario, this *congestion point* is the point connecting these two steps (i.e., the packet sending interval is approximately 2^{-3} second). If the lost packets contain sync messages, nodes will miss synchronization rounds. Missing synchronization rounds has the same effect as increasing the sync period *(T), and* leads to higher synchronization error. This is the reason that the dominant error component E_{int}, shown as line (4), exhibits a growing trend after the congestion point.

Fig. 5(f) shows the synchronization error of TSS under different data rates. Interestingly, the actual synchronization error exhibits a decreasing trend before the congestion point, and remains flat after the congestion point. We can explain this two-step behavior from the analytical models and by decomposing the overall synchronization error. E_{ext} remains flat before the congestion point, but grows after the congestion point. This is due to the fact that network congestion increases the queue length and the hop delay (*d)*. E_{int} exhibits a downward trend. A high data rate means a small packet sending interval, and this leads to a shorter packet inter-arrival time (*P)* at the sink node. When data rates are low with no congestion, E_{int} falls linearly because the amount of decrease in *P* is linear to the data sending interval. However, when data rates are high with congestion presence, *P* does not hold a linear relation with the data sending interval, because network congestion can slow down the actual data receiving rate. Therefore, E_{int} decreases at a slower rate. The overall synchronization error in TSS drops under low data rates because E_{int} dominates the overall error. However, under high data rates, E_{int} is small and E_{ext} becomes the dominant factor. This explains the slight-upward trend in the second step of E^{tss}.

4. CONCLUSION

Although TSS outperforms TPSN in accuracy under most of the cases, TSS has a fundamental limitation. It only synchronizes the events' generation times to the local clock of the events' sink node. Consider the case that there are more than one sink nodes in the wireless sensor network. Since the clocks of different sink nodes are not synchronized, the generation times of events that go to different sink nodes are not synchronized. If the application requires events to be labeled using a global reference clock, TSS is not applicable. In short, we achieve in (1) modeling the average error of two time synchronization mechanisms, TPSN and TSS, (2) validating the analytical models with simulations, and (3) identifying that the dominant error component in TPSN is the periodicity of synchronization and in TSS is the data rate.

5. REFERENCES

[1] A. Cerpa, J. Elson, M. Hamilton, J. Zhao, "Habitat Monitoring: Application Driver for Wireless Communications Technology," *ACM SIGCOMM'2000*, Costa Rica, April 2001.

[2] Z. Zhang, "Investigation of Wireless Sensor Networks for Precision Agriculture," Paper number 041154,*2004 ASAE Annual Meeting*. H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

[3] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, M. Yarvis, "Design and Deployment of Industrial Sensor Networks: Experiences from a Semiconductor Plant and the North Sea," *In Proceedings of the Third International Conference on Embedded Networked Sensor Systems (Sensys '05)*, November 2005.

[4] J. Elson, K. Römer, "Wireless Sensor Networks: A New Regime for Time Synchronization," *ACM Computer Communication Review (CCR)*, Vol. 33 No. 1, pp. 149-154, January 2003.

[5] J. Elson, L. Girod and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," *In the proceedings of the fifth symposium on Operating System*

Design and Implementation (OSDI 2002), December 2002.

[6] D. L. Mills, "Internet Time Synchronization: the Network Time Protocol," *IEEE Trans. Communications,* Vol 39, no 10, pp. 1482–1493, Oct. 1991.

[7] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM*, 21(7), 1978

[8] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-Sync Protocol for Sensor Networks," *In First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[9] K. Römer, "Time Synchronization in Ad Hoc Networks," *In ACM Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, October 2001.

[10] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, August 2000.