2012 IEEE
SENSORS APPLICATIONS SYMPOSIUM
February 7-9 2012 - University of Brescia, Italy

The Recursive Time Synchronization Protocol for Wireless Sensor Networks

M. Akhlaq, and Tarek R. Sheltami
College of Computer Science & Engineering,
King Fahd University of Petroleum & Minerals, Saudi Arabia.
akhlaq@kfupm.edu.sa, tarek@kfupm.edu.sa

Outline

- Introduction
- Related Work
- System Model
- The Recursive Time Synchronization Protocol (RTSP)
- Sources of Efficiency
- Analysis of the Sources of Errors
- Performance Evaluation
- Conclusions and Future Work
- References

Introduction

- Applications of Wireless Sensor Networks need **accurate time synchronization** (usually < 1µs) for:
  - time-stamping of messages
  - data fusion and aggregation
  - time-based localization
  - TDMA-based medium access
  - sleep scheduling
  - cooperative communication
  - coordinated actuation, etc.

- Synchronization can be achieved through:
  - ordering of events
  - local time synchronization
  - **global time synchronization**.
    - synchronizes all network nodes with one global clock.
    - e.g., **RBS, TPSN, FTSP**, etc.

By: M. Akhlaq

Related Work

1. Reference Broadcast Synchronization (**RBS**)
   - a **reference node broadcasts a beacon**, receivers note the reception time, and then exchange it with each other to calculate the relative offset.
   - average accuracy of **29.1µs** in single hop network.

2. Timing-sync Protocol for Sensor Networks (**TPSN**)
   - makes a **spanning tree** of the network and then performs **pair-wise synchronization** along the edges at each level.
   - **MAC-layer time-stamping** at the sender side only.
   - nodes find the relative offset and prop. delay but **no drift**.
   - average accuracy is **16.9µs** for single hop and less than **20µs** for multi-hop network.

By: M. Akhlaq

Cont. . .

- Flooding Time Synchronization Protocol (**FTSP**)
  - elects a reference node to frequently **floods timing info**.
  - **MAC-layer time-stamping** at both sender and receiver sides, at the end of each byte after **SYNC byte** (normalized, averaged, error corrected, and final timestamp is used).
  - nodes collect at least **8 data points**, estimate the offset and skew using LSLR, be synchronized, and start flooding.
  - single hop accuracy of **1.48µs**; **0.5µs** per hop in multi-hop.
- Other protocols.
  - Time-stamping based on Start of Frame Delimiter (**SFD**) byte to enhance the accuracy and energy consumption [5].
  - **SFD**-based time-stamping and **Kalman-filter** [6] … to achieve single hop accuracy of **0.4µs.**

By: M. Akhlaq

Cont. . .

- **Issues**: Accurate but consume **high energy** and **poorly synchronize the distant nodes**.
- **RTSP** uses a novel combination of techniques:
  - SFD-based time-stamping
  - compensation of the propagation delay and adjustment of the timestamps at each hop
  - drift & skew estimation using LSLR on 2data points (2LR)
  - infrequent broadcasts by the reference node
  - adaptive re-synchronization interval
  - request aggregation
  - energy-awareness
  - ❖ average accuracy: **0.3µs per hop** in a multi-hop network; energy usage: **1/5th** of the FTSP .

By: M. Akhlaq

System Model

- **simple linear model** for clock [7]:

$$C_i(t) = \alpha_i + \beta_i t$$

  - where $C_i(t)$ is the time read by logical clock of the ith node, α_i is the clock offset at the reference time $t=0$, and β_i is the clock skew or frequency.

- re-synchronize within $\mathbf{T \leq \delta/2\rho}$, where δ is the relative offset and ρ is the skew rate.
- without frequent re-synchronizations, **estimate the relative offset and drift** to keep their clocks synchronized.

By: M. Akhlaq

Cont. . .

- Assumptions
  - sensor nodes have **unique IDs** from 0 to n–1.
  - messages can be **time-stamped at MAC-layer**.
  - wireless channel is **unreliable but error-corrected**.
  - **broadcasting** is supported.
  - **skew and connectivity remain constant** during the short interval between sync request and reply.
  - **Prop. delay** in one direction is equal to the other.
  - a **simple linear relationship** exists between the clocks of two nodes in a short duration.

By: M. Akhlaq

Cont. . .

- Time-stamping
  - **SFD-based** time-stamping at MAC-layer: **simpler, faster and more accurate** , but can't compensate prop. delay.

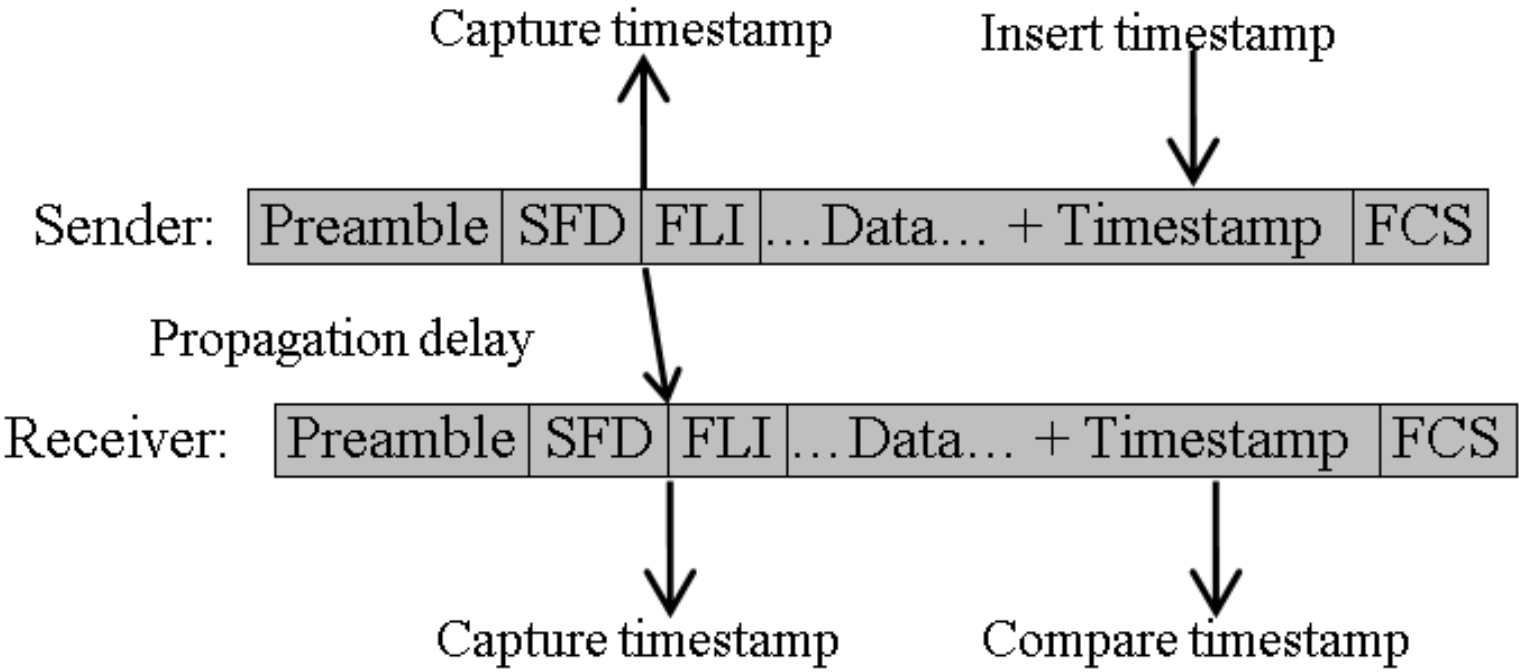


Figure 1. IEEE 802.15.4 MAC frame and the time-stamping

Cont. . .

- Structure of the RTSP messages

Field	Description
msgType	Type of the message: – ERN: enquiry/election of the reference node. – REQ: request for time synchronization. – REP: reply for time synchronization.
msgID	ID of the message.
originID	ID of the node that initiated the message.
imSrcID	ID of the immediate source that forwarded the message.
imDestID	ID of the immediate destination for message forwarding.
refNodeID	ID of the reference node that a node knows. (-1 for REF enquiry)
T1	Local time when the message was sent or forwarded.
T2	Local time when the message was received. (Null for a request)
T3	Local time when reply was sent or forwarded. (Null for a request)
Tr	Time at ref. node when reply sent or forwarded. (Null for request)

RTSP (msgType, msgID, originID, imSrcID, imDestID, refNodeID, T1, T2, T3, Tr);

By: M. Akhlaq

Cont. . .

- **Recursion** and Multi-hop Synchronization

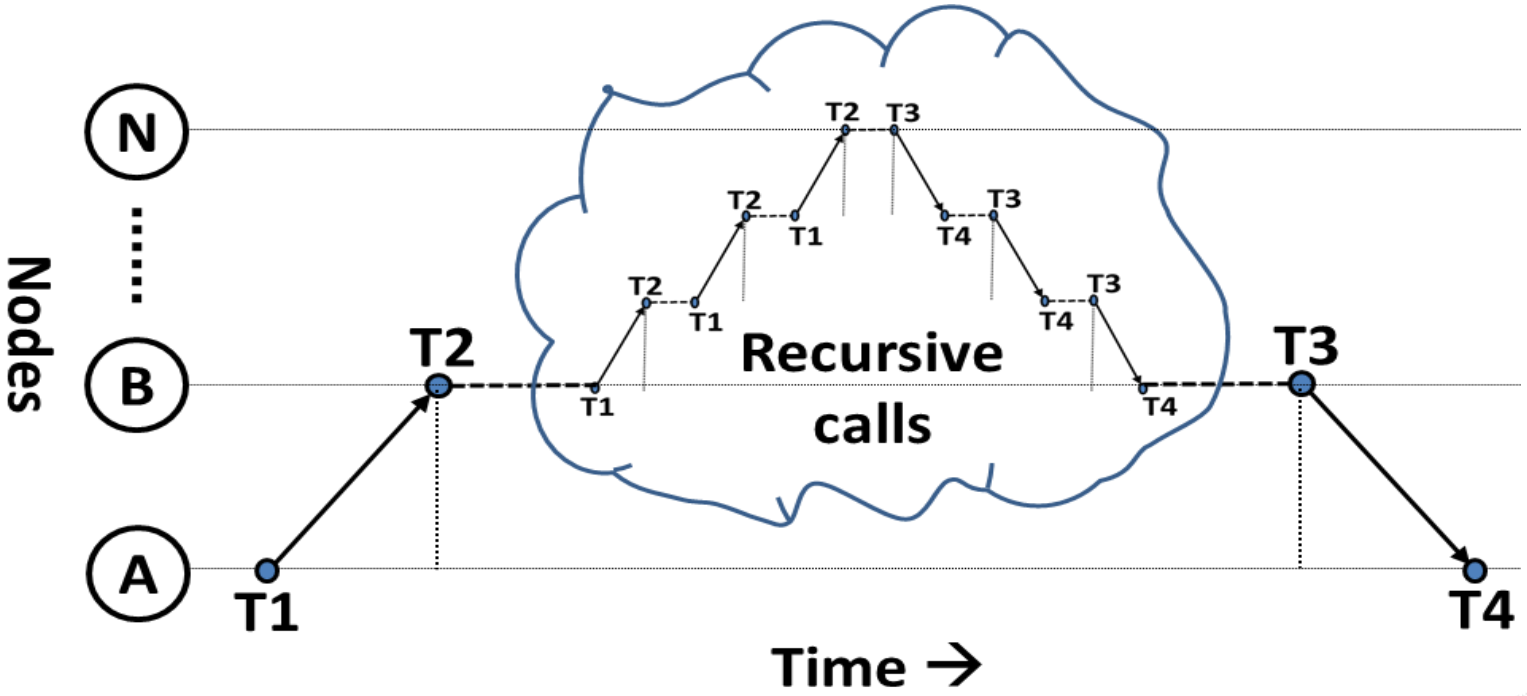


Figure 2. The request-and-reply mechanism in RTSP

The Recursive Time Synchronization Protocol (RTSP)

- After **WSN boots up**, nodes **wait** for some time, and then **broadcast an RTSP-ERN** message to enquire their neighbours about the ID of reference node. The nodes **wait** for the duration T or until a reply is received, and then run repeatedly the **adaptiveResync algorithm** (algorithm 3) which calls the **RTSP algorithm** (algorithm 2) to elect a single reference node, and compensate the offset and drift.

Algorithm 1: main

```
1.    Start up the node to join the network;
2.    Declare Variables: myRef=Null, myClient[]=Null,
3.    Wait (myID);      // smaller ID means shorter wait.
4.    RTSP("ERN", msgID, myID, myID, *, -1, T1, 0, 0, Tr);
5.    Wait for duration T or until a reply is received;
6.    Do
7.        Run adaptiveResync algorithm;    // uses RTSP algorithm
8.        // other processing.
9.    While (true);
```

By: M. Akhlaq

Cont. . .

Algorithm 2: RTSP

```
1.      // Runs on receiving a new msg that is authenticated as well.
2.      If msg-->msgType == "ERN" then
3.          T2 = timestamp after SFD byte has been received;
4.          If msg-->refNodeID < 0 AND myRef not empty then
5.              // it's an enquiry; reply if reference node ID known
6.              T3 = timestamp after SFD byte has been sent;
7.              RTSP("ERN", msg-->msgID, myID, myID,
8.                      msg-->imSrcID, myRef, msg-->T1, T2, T3, Tr);
9.          Else if (msg-->refNodeID >=0 AND myRef is empty) OR
10.                 myRef > msg-->refNodeID then
11.             // reference node unknown/expired; accept new one
12.             myRef = msg-->refNodeID;
13.             RTSP("ERN", msg-->msgID, msg-->originID,
14.                     myID, *, msg-->refNodeID, 0, 0, 0, Tr);
15.         Else if myID < msg-->refNodeID AND myEnergy > low then
16.             // smaller ID & fair energy; contest for reference node
17.             myRef  = myID;
18.             RTSP("ERN", msgID, myID, myID, *, myID, 0, 0, 0, Tr);
```

(Continued…)

Cont. . .

```
19.        Else if myRef == msg-->refNodeID AND
20.                  myID != msg-->originID AND msg-->msgID is new then
21.           // new msg from reference node; & hence rebroadcast
22.           RTSP("ERN", msg-->msgID, msg-->originID,
23.                     myID, *, msg--> refNodeID, 0, 0, 0, Tr);
24.        End if
25.     Else if msg-->msgType == "REQ" then
26.        T2 = timestamp after SFD byte has been received;
27.        If myID == refNodeID OR I'm synchronized then
28.           // reply as reference node or a synchronized node
29.           T3 = timestamp after SFD byte has been sent;
30.           Tr = T3;
31.           RTSP("REP", msg-->msgID, msg-->originID, myID,
32.                     msg-->imSrcID, myID, msg-->T1, T2, T3, Tr);
33.        Else  // save T1, T2, and source and make recursive call
34.           T1old = msg-->T1,   T2old = T2;
35.           myClient = msg-->imSrcID;
36.           T1 = timestamp after SFD byte has been sent;
37.           RTSP("REQ", msg-->msgID, msg-->originID, myID,
38.                     myRef, myRef, T1, 0, 0, Tr);
39.        End if
```

(Continued…)

By: M. Akhlaq

Cont. . .

```
40.     Else if msg-->msgType == "REP" then
41.         // calculate d, adjust & record timestamp, & forward msg
42.         T4 = timestamp after SFD byte has been received;
43.         d = ((msg-->T2 – msg-->T1) + (T4 – msg-->T3)) / 2 ;
44.         Tr = msg-->Tr + d;
45.         Record timestamps global=Tr & local=T4;
46.         If myID != msg-->originID AND myClient is not empty then
47.             // recover old values of T1 & T2; and forward the msg
48.             T1 = T1old,   T2 = T2old;
49.             T3 = timestamp after SFD byte has been sent
50.             Tr = Tr + (T3 – T4) ; // add the elapsed time since T4
51.             RTSP("REP", msg-->msgID, msg-->originID, myID,
52.                         myClient, msg-->refNodeID, msg-->T1, T2, T3, Tr);
53.         End if
54.     End if
```


By: M. Akhlaq

Cont. . .

Algorithm 3: adaptiveResync

1. Check if any new RTSP msg received (or overheard);
2. **If new msg is received** that is authenticated as well then
3. Run RTSP algorithm (algorithm 2);
4. End if
5. If no new ERN msg is received for period T AND
6. myID != myRef AND myEnergy > low then
7. **// no new msg for period T, so contest for new reference node**
8. myRef = myID;
9. RTSP("ERN", msgID, myID, myID, *, myID, 0, 0, 0, Tr);
10. End if
11. If myID == myRef AND didn't broadcast for period T AND
12. myEnergy > low then
13. **// an active reference node broadcasts every after period T**
14. RTSP("ERN", msgID, myID, myID, *, myID, 0, 0, 0, Tr);
15. End if

(Continued…)

By: M. Akhlaq

Cont. . .

16. If ((a new REP msg overheard OR ERN msg received) AND
17. myRef == msg-->refNodeID AND
18. local time is much different than msg-->Tr) OR
19. skew is much lesser or greater than 1 then
20. **// node is out of synchronization; it's time to send request**
21. RTSP("REQ", msgID, myID, myID, myRef, myRef, T1,0,0, Tr);
22. End if
23. **If two data points**, including one new, are available then
24. Calculate α and β using the last two data points;
25. Update the local clock to become synchronized;
26. End if

By: M. Akhlaq

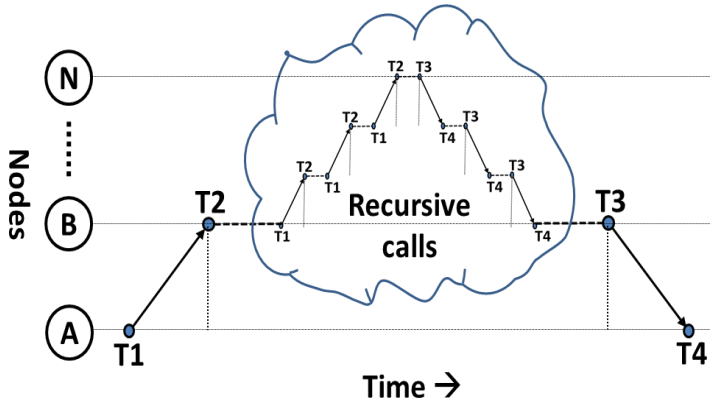
Sources of Efficiency

1. Adaptive Re-Synchronization Interval
   - send RTSP-REQ only when current **value of skew** is away form 1 or the **local time** is much deviated from global time
2. Aggregation of the Synchronization Requests
   - intermediate node temporarily saves **msgID, myClient, T1 and T**2 for recent requests and forwards only **one request**.
   - it forwards **reply to each node** in myClient accordingly.
3. Security
   - **Redundancy, authentication, discarding the corrupt info.**
4. Energy-awareness and Efficiency
   - high-energy reference node, **broadcast frequency 1/10th** of the FTSP, and due to above features.

By: M. Akhlaq

Analysis of the Sources of Errors

- The **request-and-reply** mechanism of RTSP is similar to TPSN but recursive



- **Let A sends request at T1 that is received by node B at T2 according to their local clocks.**

$$T_2 = T_1 + P_{A->B} + D_{t1}^{A->B}$$

$$RD_{t1->t4}^{A->B} = D_{t1}^{A->B} - D_{t4}^{A->B}$$ (relative drift from t1 to t4)

$$T_2 = T_1 + P_{A->B} + D_{t4}^{A->B} + RD_{t1->t4}^{A->B}$$ (by solving above eqs)

By: M. Akhlaq

Cont. . .

- **After the completion of recursive calls, node B at time T3 sends the reply to node A which receives it at T4 according to their local clocks.**

$$T_4 = T_3 + P_{B->A} + D_{t3}^{B->A} \quad \text{or} \quad T_4 = T_3 + P_{B->A} - D_{t4}^{A->B}$$

$$T_2 - T_4 = T_1 + P_{A->B} + D_{t4}^{A->B} + RD_{t1->t4}^{A->B} - T_3 - P_{B->A} + D_{t4}^{A->B} \quad \text{(sub)}$$

$$(T_2 - T_1) - (T_4 - T_3) = P^{UC} + RD_{t1->t4}^{A->B} + 2D_{t4}^{A->B} \quad \text{(re-arrange)}$$

$$2\Delta = P^{UC} + RD_{t1->t4}^{A->B} + 2D_{t4}^{A->B}$$

- **To correct the clock at T4 at node A:**

$$Error = \Delta - D_{t4}^{A->B} = \frac{P^{UC}}{2} + \frac{RD_{t1->t4}^{A->B}}{2}$$

By: M. Akhlaq

Cont. . .

- So, the **sources of errors** are:

1. **Variation in Propagation Delays (P^{UC}):**
   - Usually negligible in WSNs due to short range.
2. **Relative Drift between Local Clocks ($RD_{t1->t4}^{A->B}$):**
   - Calculate α & β using 2LR as:

$$\beta = (x1 - x2)/(y1 - y2)$$

$$\alpha = \bar{y} - \beta\bar{x}$$

where xi and yi are global and local timestamp

Performance Evaluation

- **Simulation parameters** for RTSP algorithm:

Parameter	Description
No. of nodes	20, 50, 100, 200, 500
Topology	Random
Deployment area	50m x 50m to 500m x 500m
Mobility	High (pause time 0)
Mobility model	Random Walk
Processing delay (at each node)	1ms – 100ms
No of simulations (results averaged)	1000
Channel	Wireless
MAC	IEEE 802.15.4/ ZigBee
Antenna	Omni
Transmission range	30 m
Routing protocol	AODV [13]

Cont. . .

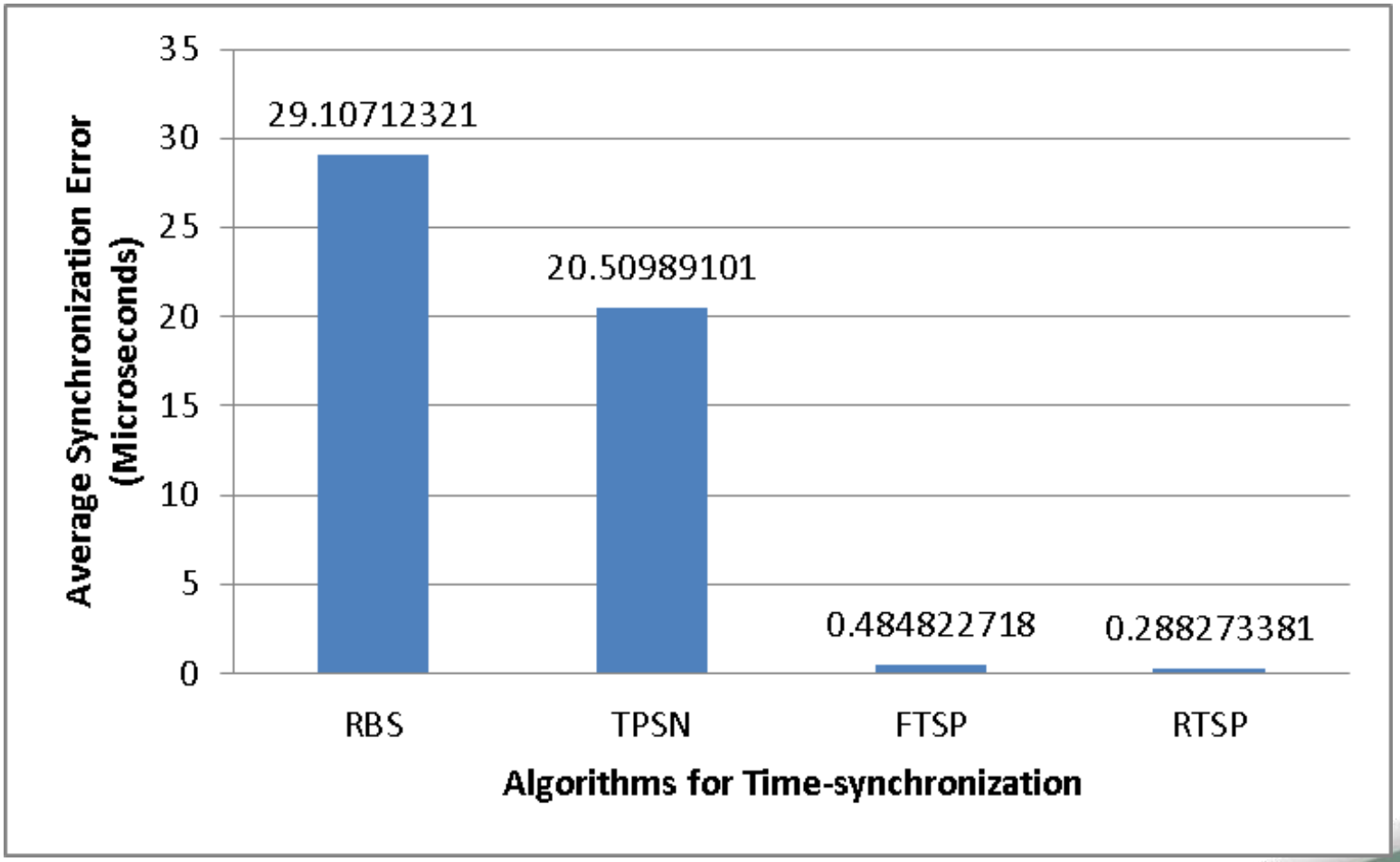


Figure 3. **Average absolute error** of time synchronization

Cont. . .

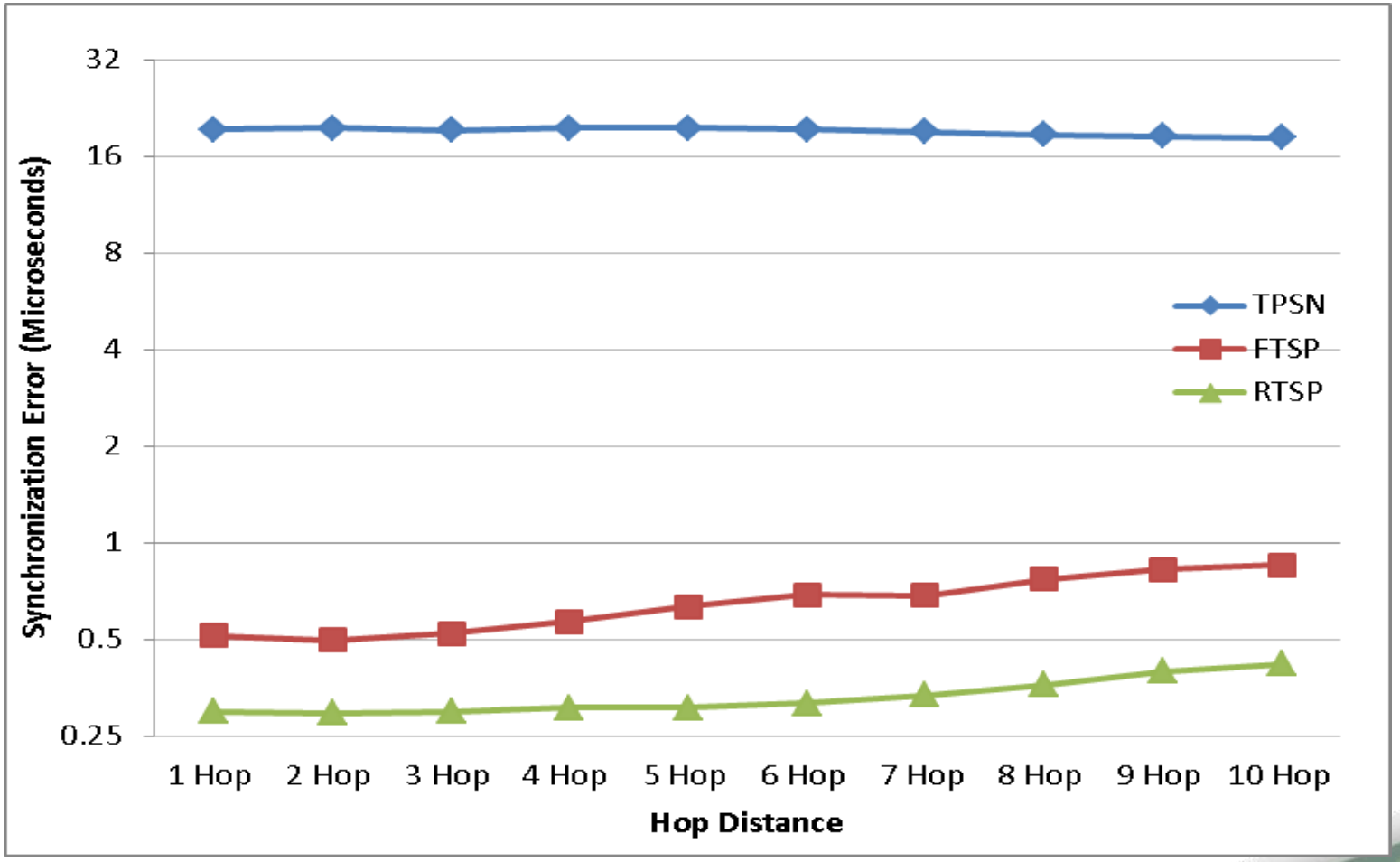


Figure 4. Average absolute error of time synchronization (**per-hop**)

Cont. . .

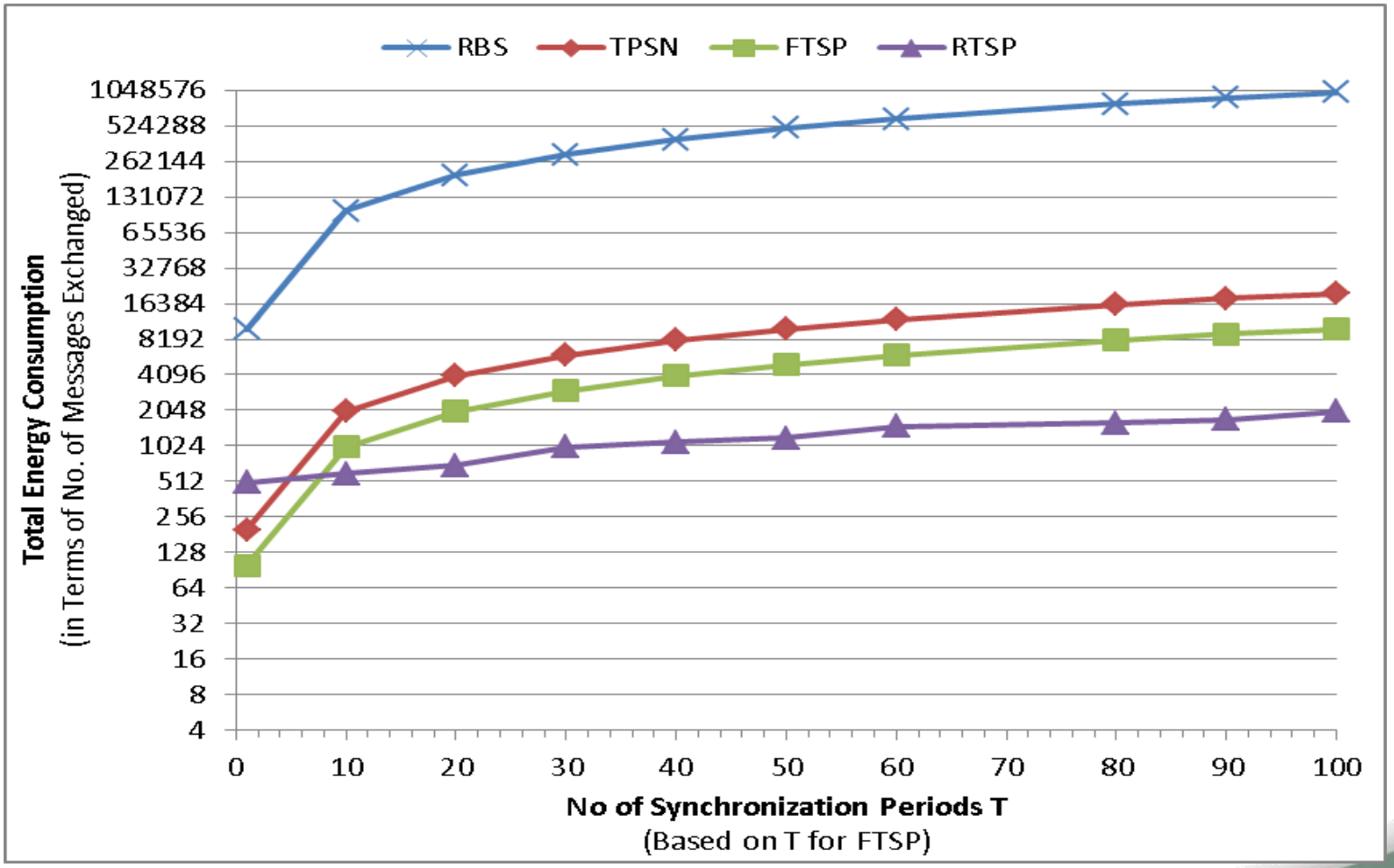


Figure 5. **Total energy consumption** in the long run (for 100 nodes)

Conclusions and Future Work

- **RTSP algorithm** for global time synchronization
- **main sources of errors**: the variations in propagation delays and relative drift. ..(are duly compensated).
- **average accuracy**: of **0.3µs** per hop; achieved by:
  - SFD-based MAC-layer time-stamping, compensation of the prop. delay and adjustment of timestamps at each hop.
- **energy usage** : **1/5th of the FTSP**; achieved through
  - infrequent reference broadcasts, and reducing the number of time synchronization requests through the adaptive re-synchronization interval and request-aggregation.
- **future work**:
  - performance evaluation on **real nodes** using TinyOS.
  - testing on **clustered WSNs**; better performance?.

References

1. M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04), New York, USA, 2004, pp. 39–49.
2. D. L. Mills, "Internet time synchronization: The network time protocol," IEEE Transactions on Communications, vol. 39, no. 10, pp. 1482–1493, 1991.
3. J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," ACM SIGOPS Operating Systems Review, vol. 36, no. SI, pp. 147–163, 2002.
4. S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in Proc. of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03), 2003, pp. 138–149.
5. D. Cox, E. Jovanov, and A. Milenkovic, "Time synchronization for ZigBee networks," in System Theory, 2005. SSST'05. Proceedings of the Thirty-Seventh Southeastern Symposium on, pp. 135–138.
6. M. Aoun, A. Schoofs, and P. van der Stok, "Efficient time synchronization for wireless sensor networks in an industrial setting," in Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys '08), New York, NY, USA, 2008, pp. 419–420.
7. A. Hu and S. D. Servetto, "Asymptotically optimal time synchronization in dense sensor networks," in Proc. of the 2nd ACM international conference on Wireless sensor networks and applications, 2003, pp. 1–10.
8. H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems," Computers, IEEE Transactions on, vol. 100, no. 8, pp. 933–940, 1987.
9. Y. C. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," Signal Processing Magazine, IEEE, vol. 28, no. 1, pp. 124–138, 2011.
10. M. Roche, "Time Synchronization in Wireless Networks," CSE574S: Advanced Topics in Networking: Wireless and Mobile Networking (Spring 2006), Washington University in St. Louis., 23-Apr-2006. [Online]. Available: http://www.cs.wustl.edu/~jain/cse574-06/ftp/time_sync/index.html. [Accessed: 23-Dec-2011].
11. Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," IEEE Comm. Surveys & Tutorials, vol. 8, no. 2, pp. 2–23, 2006.
12. V. C. Giruka, M. Singhal, J. Royalty, and S. Varanasi, "Security in wireless sensor networks," Wireless comm. and mobile computing, vol. 8, no. 1, pp. 1–24, 2008.
13. Das, S. and Perkins, C. and Royer, E., "Ad hoc on-demand distance vector (AODV) routing," , Mobile Ad-hoc Network (MANET) Working Group, IETF, 2002.

By: M. Akhlaq