Spring 2008

Columbia University

ELEN E6951
PROJECT REPORT

WIRELESS TIME-SYNCHRONIZATION

A comparative study of NTP, RBS, and MPR

Shrivathsa Bhargav

Table of Contents

Abstract 3
1. Introduction 3
1.1. What is time sync? 3
1.2. Why do we need time sync? 3
2. Time sync in wired networks 3
2.1. Network Time Protocol 4
3. Time sync in wireless networks 4
3.1. Wireless pitfalls 4
3.2. Reference-Broadcast Synchronization 4
3.3. Multi-Packet Reception 5
4. Implementation 5
5. Results 5
5.1. Confidence testing 6
5.2. NTP without MPR 6
5.3. NTP with MPR 6
5.4. RBS without MPR 7
5.5. RBS with MPR 7
6. Conclusion 7
7. Future work 8
Acknowledgment 8
About the author 8
References 8
Appendix

Wireless Time Synchronization:
Reference-Broadcast Synchronization and Multi-Packet Reception

[ELEN 6951 – Wireless Networks Project Report, May 2008]

Shrivathsa Bhargav {sb2784@columbia.edu}

Abstract

Traditional methods of time synchronization in wired networks are no longer suitable for wireless networks, given the energy constraints, infrastructure deficiencies, and processing limitations. This paper explores one traditional method, Network Time Protocol, and looks at a simulation of it in a Multi-Packet Reception-enabled wireless scenario. A simulation of a more recent technique, Reference Broadcast Synchronization in a Multi-Packet Reception system is also analyzed.

1. Introduction

1.1. What is time sync?

Time synchronization (or time sync for short) can be defined as the process of converging the clocks of all the devices in any given network so that they agree by frequency, epoch, or both. Frequency sync is sometimes necessary because the oscillators used for clock generation in devices tend to have frequency "drift". Over time, this drift can cause gradual change in the timekeeping abilities of the device. Synchronization to a particular epoch refers to counting the number of ticks or seconds after a particular event. A popular epoch-based method is Coordinated Universal Time (UTC) (1), which is International Atomic Time with leap seconds added when necessary. In general, unless mentioned otherwise, synchronization will be considered to be both frequency **and** time synchronization in this paper.

1.2. Why do we need time sync?

A survey (2) of some 10,000 computers on the internet found that less than 40% of the clocks of these systems were synchronized (with less than a two minute offset to UTC). This survey is actually several years old. The current state of the internet is more chaotic, and this inaccuracy is only expected to increase.

Unexpected applications that necessitate moderately accurate timekeeping include document searches based on creation/modified dates, as well as online collaboration and communication, especially email. Online collaboration such as project development and wikis usually discard earlier edits in favor of later ones. Email timestamps are placed as the outgoing message is being sent, and an incorrectly-set system clock will result in an incorrect time stamp and therefore an incorrectly-ordered conversation upon receipt.

High-accuracy applications include military and scientific operations (such as physics and astronomy), as well as distributed computing.

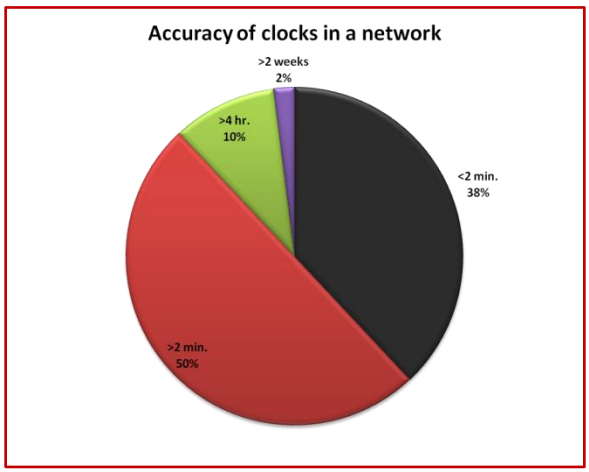


Figure 1 - Survey of time accuracy in 10,000 computers on the internet

Traditional methods of time sync including the Network Time Protocol and its drawbacks are discussed in section two. The challenges and techniques of syncing in wireless networks are discussed in section three. Section four provides implementation details of different techniques. The obtained results are presented in section four, while section five concludes this paper.

2. Time sync in wired networks

The simplest method of time sync is to manually set the system clock of every machine in the network. While this may work well for networks with a dozen or so nodes, it is obviously highly impractical for a corporate networks or the Internet.

One of the most prevalent syncing protocols is the Network Time protocol.

2.1. Network Time Protocol

The Network Time Protocol (NTP) (3) has been in use since 1985, and is a versatile and fairly robust protocol for time sync on the internet. In this protocol, one node (known as the reference or root node) knows the current time (usually UTC) via interaction with an external receiver or time-server. Note that UTC is an absolute, global time scale from the viewpoint of this network. The nodes in the network then organize themselves into levels, known as strata, determined by the number of hops to the reference node. Time data is then propagated through the network hierarchically as "streams". Each syncing event consists of at least four transmissions. (4)

NTP works reasonably well for networks with high-speed links. It is scalable and self-reconfigurable, which is a boon for large networks. It provides adequate accuracy while being robust and resilient to node/link failure and sabotage.

One downside of this hierarchical dissemination of time is the accumulated delay at each node. This delay has four main components. (5)

- Send Time – The time spent to construct the message at the sender node.
- Access Time – The delay while waiting for access to the channel. Nondeterministic.
- Propagation Time – The speed of the media, as well as any routing overhead. Nondeterministic.
- Receive Time – The processing time required to receive the message and notify the receiver.

Together, these times constitute the critical path of the transmission.

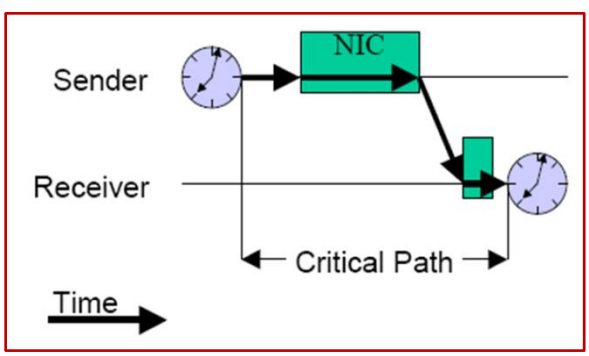


Figure 2 - Critical path delay of an NTP transmission

NTP works by trying to record the time delay between the source and itself, and accounting for this delay by adjusting the received time data. However, NTP is unable to account for the nondeterministic delays introduced by the access and propagation times. In large networks, this delay is considerable, and can lead to inaccuracies in the synchronization of nodes far from the root.

3. Time sync in wireless networks

3.1. Wireless pitfalls

Having been specifically designed with wired computer networks in mind, NTP does not perform well in wireless networks, especially sensor networks. This is because wireless sensor networks differ significantly from wired networks.

- Finite energy – Wireless sensor nodes have limited battery, and therefore some of the classic assumptions of NTP (especially that all nodes are tethered to a power outlet) do not hold. In particular,
    - The processor in these nodes is not always on.
    - Listening to the network is not "free", as far as energy-usage is concerned. In most wireless networks, listening for incoming packets is just as expensive as sending one.
    - Continuous transmissions, such as those required in an NTP stream are not feasible.
- No standard infrastructure – Sensor networks are often deployed in areas where there is no (or poor) infrastructure. This means
    - The network has a dynamic topology, with nodes coming in and out of the network often, and the self-reconfiguration option of NTP (being time and energy-costly), cannot be constantly deployed.
    - Higher-stratum nodes will have very poor sync, given the excessive number of hops a dynamic topology may demand.
    - A class-based hierarchy will not work, since no one node may be in a position to act as the reference node for a guaranteed length of time.

3.2. Reference-Broadcast Synchronization

Devised early this decade, Reference-Broadcast Synchronization (RBS) attempts to combat the flaws of NTP, while adapting to a wireless paradigm (6). This scheme is peer-to-peer instead of hierarchical, and it has

high accuracy across multiple hops. This works because RBS does away with a *global* timescale like UTC, and instead uses a *relative* timescale. The question is no longer "what time is it?" rather; it is now "if this is the time here, what time is it *over there*?"

In this system, nodes only keep track of the relative differences in the timekeeping abilities of their neighbors. When a node wishes to know the difference in time between two events that happen at different ends of the network, it looks at the local timestamps at the nodes closest to the events, and then adjusts the timestamps by the known relative difference between itself and the other node.

RBS, unlike NTP, is capable of working directly on the physical broadcast layer of the node. This does away with the nondeterminism at the sender's end, thereby making the synced time even more accurate.

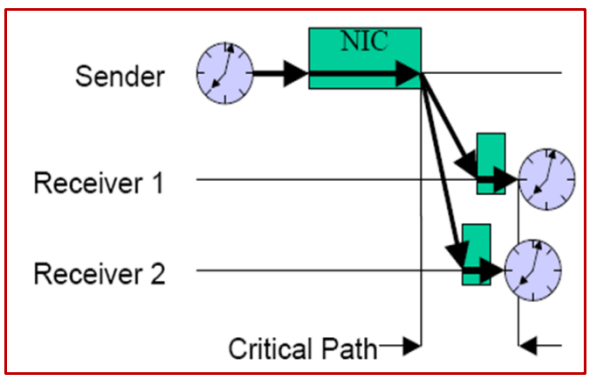


Figure 3 - Critical path delay of an RBS transmission

Additionally, RBS requires two nodes to exchange only one transmission in order to be synced. This does away with the streams needed to sync in an NTP-synced network. RBS consistently outperforms NTP in both lightly- and heavily-loaded networks. (7)

3.3. Multi-Packet Reception

Traditionally, wireless sensor nodes have only been able to receive one packet at a time; any more receptions lead to all packets being discarded upon arrival, since they collide. Advances in wireless transmission technologies have enabled multi-packet reception (MPR). (8)

In MPR-enabled networks, the number of simultaneously-received packets (referred to here as MPR-depth) is dependent on the network and transceiver architecture, but is greater than one. This is often achieved by transmitting and receiving on multiple frequencies simultaneously, known as Multiple-Input Multiple-Output (MIMO) (9), and represents the state of the art of wireless technology.

4. Implementation

The crux of this paper revolves around the development of a cycle-level simulation tool, in C, for time sync in wireless networks with MPR capabilities. The tool probabilistically simulates transmissions from randomly-placed nodes inside a predetermined grid. These nodes have controlled, but random, ranges. The tool functions using an event-driven queue with variable size. Different implementations were then attempted including NTP, RBS, and RBS with MPR. The primary goal of the simulation was to compare the performance of NTP and RBS under different conditions. The main result that was extracted was the number of cycles until the network is completely synced, averaged across thousands of runs with varying conditions.

Several simplifying assumptions had to be made during the design and development of the tool:

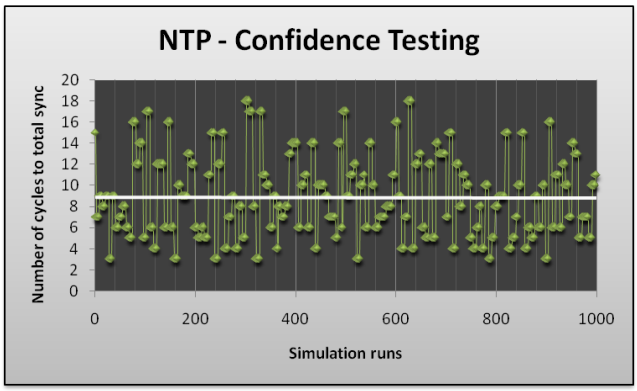
- All the nodes have fixed slots during which they may decide to transmit (similar to slotted Aloha)
- Nodes may not transmit and receive at the same time
- Nodes transmit probabilistically (all nodes have equal transmit probabilities); there is no dedicated back-off interval
- For NTP, a single node is picked as the reference node, and time-sync propagates hierarchically through the network
- For RBS, every node attempts to spread its relative time to its neighbors, and time-sync radiates through the network
- A collision occurs when a node receives more messages than the MPR-depth of the network allows
- A packet-receive takes only one cycle; when a node successfully hears a transmission, it is synced the next cycle

5. Results

As mentioned previously, the main result extracted from these simulations is the average number of cycles until total sync in all nodes of the network, aggregated across thousands of runs. The main elements of the results to be presented are: confidence testing, NTP, RBS without MPR, and RBS with MPR. Variables include the MPR-depth (where applicable) and the number of nodes.

5.1. Confidence testing

To gain confidence in the ability of the simulator, a confidence test was run. In this test, the topology of the network (comprised of the number of nodes, their positions, their transmit probabilities, and their ranges) were kept constant, and their deviance from the mean was studied.

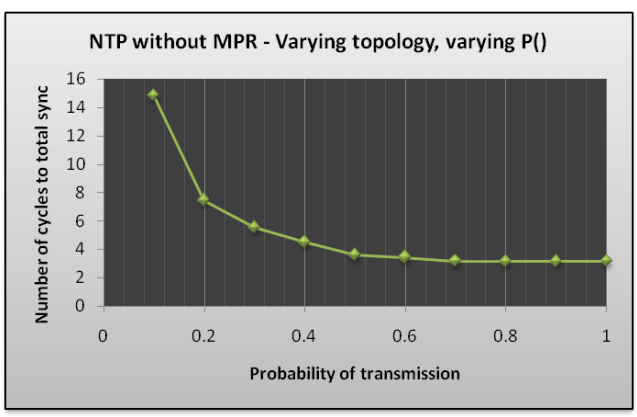


The average number of cycles to total sync across 1000 simulations was 8.75 cycles. While the per-run number of cycles seems to vary wildly, the linear trend-line (shown in white) is completely flat, and perfectly at the average of 8.75 cycles. Therefore, the simulator is consistent across a large number of runs.

The results for different technologies under different, varying conditions are detailed below. Unless mentioned otherwise, each data point is averaged across 300 simulations.

5.2. NTP without MPR

The topology is randomized each run, while the probability of transmission is increased from 0.1 to 1.0. MPR is disabled in this simulation.

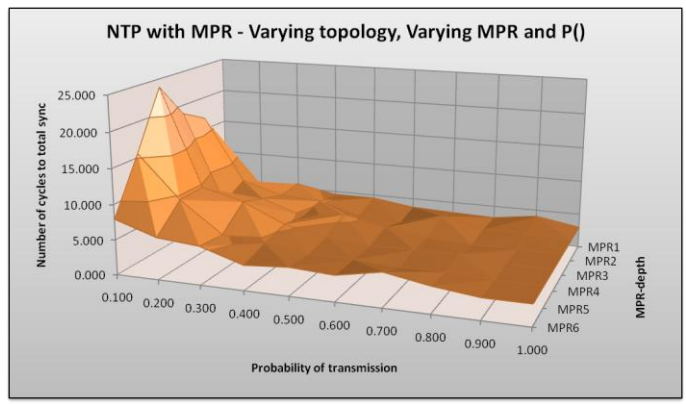


The result is an exponential decrease in the number of cycles which is expected, since nodes transmit more often.

The reason for a non-linear decrease in the number of cycles (and not a linear one) is because of the increased number of collisions that occur when all nodes have a higher transmit probability, since MPR is disabled. The simulation is able to run up to P(1.0) because the simulation ends when all nodes are synchronized. Were this not the case, the network would go into an infinite cycle of collisions at P(1.0), since every node is synchronized, and therefore every node transmits.

5.3. NTP with MPR

The topology is randomized each run, while the probability of transmission is increased from 0.1 to 1.0, and the MPR-depth is increased from 1 (MPR disabled) to 6 (full MPR, where there will be no collision even if all the nodes in the system are transmitting simultaneously).



The data is presented as a surface graph and a color-coded table.

Table 1 - NTP with MPR, recorded values

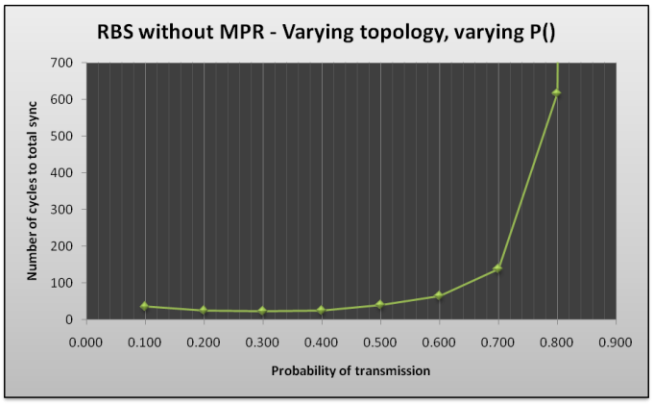
Probability \ MPR-depth	6	5	4	3	2	1
0.100	7.940	15.590	23.820	18.730	16.770	9.080
0.200	5.980	9.480	9.580	5.600	7.310	4.900
0.300	5.450	5.090	8.020	6.730	5.530	5.670
0.400	3.540	4.560	5.320	4.640	5.180	4.530
0.500	4.000	3.620	4.030	5.510	3.380	4.530
0.600	3.650	3.620	3.420	4.240	3.030	3.650
0.700	4.830	3.240	3.650	3.380	4.700	3.400
0.800	3.730	3.380	3.690	3.000	3.280	3.220
0.900	3.000	3.000	3.000	3.000	3.310	4.080
1.000	3.000	3.000	3.000	3.000	3.000	3.000

Having a low probability of transmission results in worse performance, since the nodes transmit far less often. However, higher MPR-depths usually result in better

performance. Higher probabilities have more of a positive effect on the number of cycles to sync than do higher MPR-depths. This shows that NTP is not able to take advantage of MPR to the level expected.

5.4. RBS without MPR

The topology is randomized each run, while the probability of transmission is increased from 0.1 to 1.0. MPR is disabled in this simulation.



The event queue limit of 9999 ticks was reached at P(0.9) (and so, it is safe to assume that the network will never get out of this halted condition at P(0.9) and above, and therefore it will take an infinite number of cycles to sync).

The general trend of the graph is to be expected, since MPR is disabled, and there is a large number of collisions at higher probabilities (larger, indeed, than NTP under similar conditions, because NTP nodes only transmit when they are synced to the reference node, whereas RBS nodes try to transmit all the time).

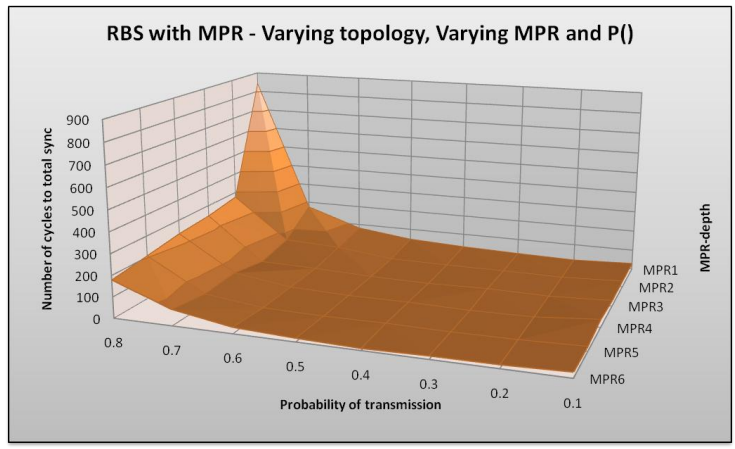
5.5. RBS with MPR

The topology is randomized each run, while the probability of transmission is increased from 0.1 to 0.8 (the event queue was being maxed out at lower MPR-depths and higher probabilities, so the results were only gathered up to P(0.8)), and the MPR-depth is increased from 1 (MPR disabled) to 6 (full MPR, where there will be no collision even if all the nodes in the system are transmitting simultaneously).

The data collected is recorded as a color-coded table, and illustrated as a surface graph.

Table 2 - RBS with MPR, recorded values

Probability \ MPR-depth	6	5	4	3	2	1
0.8	179.24	205.04	231.85	246.6	275.21	840.91
0.7	76.26	51.32	62.86	32.14	65.42	173.28
0.6	29.6	18.48	23.11	26.63	53.63	80.14
0.5	16.98	13.21	14.22	16.26	20.39	44.61
0.4	9.39	15.97	14.89	12.8	17.84	31.11
0.3	15.3	12.48	9.53	18.94	17.97	23.25
0.2	16.91	14.57	13.7	14.73	15.91	17.77
0.1	25.87	22.32	37.68	26.33	28.6	29.57



From the table of recorded values (illustrated in the graph), it becomes clear that there is actually a "sweet-spot" combination of probability of transmission and MPR-depth. In most cases, a higher MPR-depth for a fixed probability results in fewer collisions, and therefore quicker overall sync.

Interestingly, with higher probabilities, even higher MPR-depths do not result in as fast a sync as lower probabilities.

This is because nodes cannot transmit and receive at the same time (even on different channels).

One of the best possible sync cycle times (9.53 cycles) occurs even without high probabilities and high MPR-depths (P(0.3) and 4, respectively).

6. Conclusion

A cycle-level network simulator was built specifically to analyze different techniques of time-synchronization. The techniques studied were Network Time Protocol (NTP) and Reference-Broadcast System (RBS), both with and without Multi-Packet Reception (MPR).

It was found that RBS is able to make better use of MPR than NTP is. Additionally, increasing the probability of transmission for RBS did not always result in quicker total sync times in the network.

7. Future work

The level of detail in the simulator can be increased. Some of the changes that can be made include

- the ability to have moving nodes
- increasing the realism of the simulation by adding non-deterministic delay to received packets
- converting the cycle-level simulator into a discrete-time-based one

Acknowledgment

The author would like to thank Prof. Gil Zussman for providing help and guidance on this project.

About the author

Shrivathsa Bhargav graduated from SUNY Stony Brook with a Bachelors' Degree in Electrical Engineering (*Cum Laude*). He is currently pursuing a Masters' Degree in Electrical Engineering at the Fu Foundation at Columbia University, New York. His interests include digital circuit design and embedded systems.

References

1. **Wikipedia.** Coordinated Universal Time. *Wikipedia: The Free Encyclopedia.* [Online] [Cited: May 22, 2008.] http://en.wikipedia.org/wiki/UTC.

2. **Mills, David L.** On the accuracy and stability of clocks synchronized by network time protocol in the Internet system. *ACM Comput. Commun. Rev.* January 1990. Vol. 20, 1, pp. 65-75.

3. **Wikipedia.** Network Time Protocol. *Wikipedia: The Free Encyclopedia.* [Online] [Cited: May 22, 2008.] http://en.wikipedia.org/wiki/Network_Time_Protocol.

4. **Mills, David L.** Internet Time Synchronization: The Network Time Protocol. *Global States and Time in Distributed Systems.* : IEEE Computer Society Press, 1994.

5. **Kopetz, H. and Schwabl, W.** Global Time in Distributed Real-Time Systems. *Technical Report 15/89.* : Technische Universitat Wien, 1989.

6. **Elson, J., Girod, L. and Estrin, D.** Fine-Grained Network Time Synchronization using Reference Broadcasts. *OSDI.* 2002.

7. **Elson, J**. Time Synchronization for Wireless Sensor Networks. *Presentation.* s.l. : UCLA, May 2, 2003.

8. **Tong, L., Zhao, Q. and Mergen, G.** Multipacket reception in random access wireless networks: from signal processing to optimal medium access control. *Communications Magazine.* : IEEE, November 2001. Vol. 39, 11, pp. 108-112.

9. **Neelesh B. Mehta, et al.** *RF-based antenna selection in MIMO systems. 7327983* United States of America, February 5, 2008. Telecommunications.