Synchronisation Challenges for Wireless Networks

Jonathan Shannon [1], Hugh Melvin [2]
Discipline of Information Technology
NUI Galway
Galway, Ireland
Shannon.jonathan@gmail.com, hugh.melvin@nuigalway.ie

Abstract — *Many real world applications require a sense of global time in order to carry out time sensitive processes but providing these applications with an accurate measure of global time is no easy task. A computer system's clock is far from perfect and tends to drift from real time giving rise to significant clock errors. If computer clocks are to conform to some time standard they must be disciplined using some synchronisation technique. This is a challenging task when the computer systems are located across a dynamic and variable latency network but becomes even more challenging when the network contains wireless links.*

1 INTRODUCTION

The advent of the computer system has redefined the granularity of a useful clock measurement. Current CPU's can execute tens of billions of instructions per second meaning that the duration of a significant event or the interval between significant events in the life of a computer system may be in the order of microseconds or less. In order to provide a time sensitive application with this level of accuracy, it is necessary to discipline a clock such that it conforms to a time standard.

2 COMPUTER CLOCK

A typical computer clock consists of a quartz crystal that oscillates at a frequency which is dependent on its shape or cut. Each oscillation of this crystal results in the increment of a dedicated clock register which when full triggers a tick interrupt. This interrupt in turn results in the execution of a clock handler which increments the system time by one tick. The system time may be a simple counter in main memory which has been allocated by the underlying operating system. An application that requests the time does so via a system call which simply reads the system time from memory and possibly converts it to some accepted time format.
The frequency at which the quartz crystal oscillates dictates the maximum resolution of the hardware clock and this together with the size of the clock register dictates the size of a clock tick. For instance, a quartz crystal with a frequency of 4MHz provides a maximum resolution of 0.25μs. If the size of the clock register is 8 bits then the size of a clock tick is (256 * 0.25) 64μs. Thus, a tick interrupt is generated every 64μs and the counter representing the system time is multiples of this value. The resolution of the system time is determined by the size of a clock tick and so in this case is 64μs. If an application wishes to read the system time, it must do so via system calls which introduce a degree of latency. Thus, two consecutive readings of the system time may differ by multiple clock ticks. The minimum difference between two readings of the system time represents the precision of the clock.

The frequency at which a quartz crystal oscillates is dependent on its shape and its physical operating environment. If a crystal is incorrectly cut it may oscillate at a frequency below or beyond its stated value. Consequently, the crystal will have a permanent frequency offset that will cause it to constantly drift from real time. In addition, a change in environmental conditions such as temperature, pressure or voltage will cause the rate of oscillation to change and, thus, temporary frequency offsets will be observed.

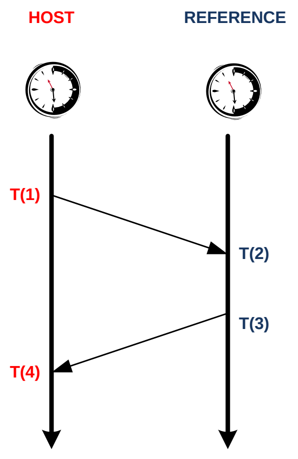
There are a number of methods of eliminating or reducing frequency offsets such that a clock's stability and accuracy are increased. One solution is to select the crystal's cut such that it has the lowest temperature coefficient for its operating environment (termed XOs). Another solution is to eliminate the effects of the clock's operating environment by placing it in a controlled environment such as an oven (termed OCXOs). The most effective approach is to replace the crystal clock with an atomic clock. While these approaches are all effective to some degree they can be unfeasible in many scenarios and so the use of a time synchronisation protocol is favoured.

2.1 CLOCK SYNCHRONISATION AND SOURCES OF ERROR

In general a clock synchronisation protocol corrects the clock of a host by referring to an external

1 Member of the Performance Engineering Laboratory (PEL) http://pel.it.nuigalway.ie/
2 Member of Energy Research Centre www.nuigalway.ie/energy and PEL

time reference which is connected via some communication link. A typical approach used in the wired domain is pair-wise synchronisation whereby a host node wishing to be synchronised sends a time request to a reference node and records the transmission time *t1*. The reference node replies with the reception time of the request message and transmission time of the reply message, *t2* and *t3* respectively. The host node records the reception time of the reply message *t4* and uses the four timestamps to calculate the propagation time and, thus, determine its clock offset from the reference.



$$\Delta = \frac{T(2) + T(3) - T(1) - T(4)}{2}$$

This technique operates on the assumption that the round-trip message delays between a sender and receiver are symmetric. If this is not the case, which in reality is generally true, then the result of the calculation of a clock's offset from its reference will differ from the true offset leading to a clock error. To understand the different sources of synchronisation errors one must understand the different components of a synchronisation message's latency. These components are the *send time, access time, propagation time* and *receive time*.

The *send time* represents the time taken for a host to construct a synchronisation message and transfer it to the network interface. The time taken to construct the message will be influenced by the underlying operating system. For instance, the process that constructs the message will be subject to some scheduling algorithm which may block it numerous times during its operation. Furthermore, additional delays may be incurred due to system call overheads.

The *access time* represents the delay incurred by the network interface while waiting to gain access to the communication medium. This is dictated by the Medium Access Control (MAC) protocol in use. For instance, the Ethernet and 802.11 protocols use contention based approaches meaning a node may not transmit until the channel is clear, thus, high traffic loads will most likely lead to large delays.

The *propagation time* represents the time taken for the message to travel to the receiver once it has left the sender. The propagation time will be minute if the sender and receiver are connected directly by the same physical medium. If, however, they are connected via multiple network nodes, this time will be much greater since the message will be subject to multiple queue and access delays.

The *receive time* represents the time taken for the receiver's network interface to receive the message from the communication medium, decode it and notify the host application that it has arrived. In addition it also includes the time taken to transfer the message to the host application and other latencies due to operating system overheads such as context switches and system calls.

It is important to note that, with pair-wise synchronisation, the magnitude of a message's delay is not the factor which results in a clock error but rather the difference between the delays of a request message and a response message. It is mainly this difference (asymmetry) that results in an incorrect calculation of the propagation delay between the sender and receiver and, thus, an incorrect calculation of the clock offset.

2.2 SYNCHRONISATION PROTOCOLS

Throughout the last number of decades a few notable time protocols have emerged, each one suited to a different environment. The complexity of the operating environment dictates the complexity of a time protocol and the accuracy achievable by that protocol and as such each one is tailored for a specific setting. The two most distinguished protocols are the Network Time Protocol (NTP) [1] and the Precision Time Protocol (PTP) or IEEE 1588 [2].

The NTP protocol is designed for synchronising the clocks of computer systems connected via variable latency and dynamic packet switched networks. NTP uses pair-wise synchronisation to correct a host's clock and in its most common configuration a host takes on the role of both a client and server, requesting and providing time from and to other NTP hosts. The protocol timestamps packets at the application layer and, as such, its packets are subject to all of the non-deterministic delays associated with the transmission and processing of time messages. To remedy this, NTP not only queries multiple time references, but also uses an array of sophisticated algorithms to mitigate the effects of these non-deterministic delays. The result is a protocol that can achieve sub-millisecond accuracies over wide area networks (WANs).

PTP or IEEE 1588 was designed to meet the accuracy requirements of applications that NTP and other time protocols could not provide. Akin to NTP, PTP uses pair-wise synchronisation to synchronise its hosts, however, in contrast to NTP, it is designed for local managed networks. Rather than use sophisticated algorithms to alleviate the effects of non-deterministic delays, PTP takes advantage of the fact that the network is managed and introduces the notion of PTP aware nodes. PTP aware nodes remove much of the un-deterministic delays associated with propagation time by eliminating the residence time of PTP packets. In addition, an ideal PTP network will include specialised PTP hardware at each node, allowing packets to be time stamped at the physical layer. In an ideal setup, PTP can achieve sub-microsecond accuracies.

3 WIRELESS ISSUES

Both NTP and PTP are particularly suited to wired networks but tend to degrade if forced to operate over wireless networks. This is mainly due to the large un-deterministic access times introduced by wireless contention delays. In an ideal PTP network containing specialised PTP hardware this will not be the case, though, in reality the cost of PTP hardware may limit a certain body to employing the protocol in software only. In this case PTP like NTP will be subject to un-deterministic access delays.

A wireless network replaces dedicated wired links with a shared wireless medium. Given that this medium is shared it must be allocated to individual nodes in a fair and orderly fashion. Depending on the technique used to allocate the wireless medium and the traffic load at a particular time, transmission of packets between two particular nodes may be subject to large contention delays.

The allocation of a wireless medium is typically performed by the MAC layer of the wireless protocol in operation. The most accepted and widely implemented set of standards for wireless communication over LANs is defined by IEEE 802.11. The basic 802.11 specification defines both the physical and data link layers for communication across a wireless network. Access to the wireless medium as defined by the 802.11 MAC layer is controlled by the distributed coordination function (DCF). The DCF controls access by using a CSMA/CA (carrier sense multiple access with collision avoidance) mechanism. It operates by first checking that the medium is clear using carrier sensing functions and if so begins transmission. If the medium is in use the station backs off for a random time interval to avoid a collision.

In the case of an 802.11 network, all nodes including the access point conform to the same access rules. Since the access point (AP) carries out extra work one should expect that an NTP or PTP station operating over an 802.11 network with competing nodes would experience significant asymmetric round-trip delays and, thus, significant clock errors. To illustrate this we simulated the transmission of NTP packets across an 802.11 network under varying traffic conditions.

3.1 NS2 SIMULATION

Simulation of NTP over 802.11 was carried out using the Network Simulator NS2 [3]. Each simulation replicated an NTP client polling an NTP server at minute intervals for an hour over a basic 802.11 network. All simulations included competing web-clients. The web-clients were configured to download average sized web pages (312KB) at minute intervals and commence downloads at random times. The upload times of NTP packets from the NTP client to the AP and vice versa were recorded and the difference between them calculated to determine the asymmetric delays.

Maximum asymmetric delays for 1, 5 and 10 competing web-client(s) were found to be roughly 140, 440 and 580 ms respectively. This concludes that pair-wise synchronisation across an 802.11 network would lead to significant clock errors unless there was some mechanism in place to eliminate or mitigate the effects. If PTP were forced to operate over an 802.11 network without specialised hardware it would be dramatically degraded and the clock would be constantly fluctuating. NTP, although equipped with special algorithms to diminish these effects would still be heavily degraded in terms of the accuracy it could provide although not to the extent of PTP.

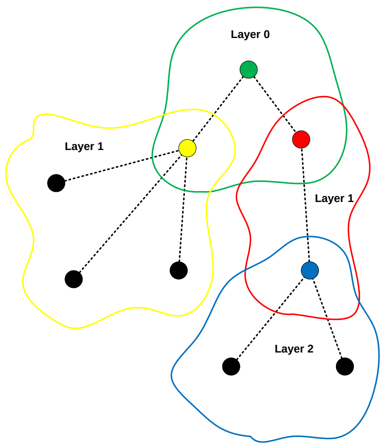
4 WIRELESS SENSOR NETWORKS

One important area where synchronisation over wireless networks is of critical importance is Wireless

Sensor Networks (WSNs). WSNs consist of a cluster of wirelessly interconnected autonomous sensing devices distributed over a specific area of interest, allowing for the acquisition and distribution of physical data within that area. The sensing devices are typically restricted in terms of their processing and memory capabilities and are designed with low energy consumption in mind. These restrictions contribute to their low cost, small size and extensive battery lifetime.

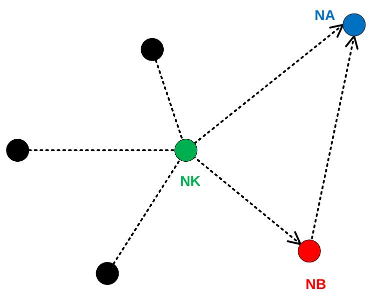
WSNs can be applied to a range of applications, however, regardless of the application, data collected from sensors within a WSN is generally of little use if the point in time it was acquired at is unknown. Time is crucial when it comes to the analysis of sensor data and in order to make accurate judgements or predictions about a system as a whole the issue of time synchronisation must be dealt with.

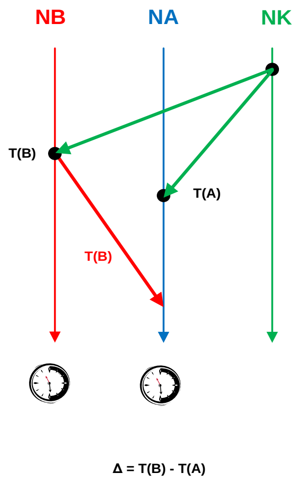
Time synchronisation in WSNs is typically performed using one of two techniques. They are pair-wise synchronisation and reference broadcast synchronisation. Pair-wise synchronisation is performed as explained in **Section 2.1**. An example of a WSN time protocol which employs this technique is TPSN [3]. To synchronise the sensors, TPSN first organises the network into a tree hierarchy with the root representing the origin source of time. The first layer of sensors then synchronises with the root using pair-wise synchronisation and adjusts their clock appropriately. The second layer then synchronises with the first layer and the process continues until the leaf nodes are corrected. To eliminate the effects of un-deterministic delays, particularly those associated with medium access, TPSN employs MAC layer time stamping.



Reference broadcast synchronisation is a synchronisation technique which is designed to eliminate the un-deterministic delays associated with the sender of a time message, namely, the send delay and the access delay. A protocol which uses this technique is RBS [5]. With RBS, a network of sensors is organized into clusters each of which contains a special beacon node. The beacon node periodically transmits a beacon message which is received by all nodes within the cluster. Two nodes which wish to synchronise with each other exchange the times that they received the beacon node. Each node can then determine their clock offset from the other node and build a relative timescale. Subsequently, time stamped data sent from one node to another can be converted to that node's timescale. To extend this to operate in a multi-hop scenario where the network contains multiple clusters, one or more nodes are placed in the vicinity of two or more clusters. These nodes act as gateways translating time from one cluster to another.

The two main sources of error with RBS are the un-deterministic delays associated with the propagation times and the receive times of messages. Errors associated with propagation delays are due to the differences in distance between nodes and their beacon node. However, propagation delays are overshadowed by the receive delays which typically accounts for the greatest clock error.





5 CONCLUSIONS

This paper has outlined how a typical computer clock operates and why a time sensitive application cannot rely on the accuracy of the time measurements of a free running computer clock. Time synchronisation protocols provide a feasible and effective means of synchronising a computer clock, however, the most distinguished of these protocols are aimed at wired infrastructure networks and, as such, their performance tends to degrade when forced to operate over a wireless network. Results of NS2 simulations replicating an NTP host operating over an 802.11 network support this statement.

WSNs highlight a domain where time synchronisation over a wireless medium is of crucial importance. Various WSN synchronisation protocols exist, for instance, TPSN and RBS, and they provide effective synchronisation solutions. Employing techniques used by these protocols in traditional protocols, such as NTP, could provide a means to remedy their shortfalls and, thus, provide all-round synchronisation protocols.

References

[1] Network Time Protocol (NTP).
http://www.faqs.org/rfcs/rfc1305.html

[2] IEEE-1588 - Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.

[3] The Network Simulator - ns-2.
http://www.isi.edu/nsnam/ns/

[4] Ganeriwal, S., Kumar, R., and Srivastava, M. B. *Timing-Sync Protocol for Sensor Networks.* The First AC Conference on Embedded Networked Sensor System (SenSys), p. 138–149, November 2003.

[5] Jeremy Elson, Lewis Girod, Deborah Estrin. *Fine-Grained Network Time Synchronization using Reference Broadcasts.* The Fifth Symposium on Operating Systems Design and Implementation (OSDI), p. 147–163, December 2002.