The timed asynchronous model and its application in time-triggered protocols

Ruiqi Hu

Abstract

Distributed real-time systems use both synchronous communication and asynchronous communication. The asynchronous communication has much more complicated semantics than the synchronous one. It is desirable to provide quasi-synchronous semantics for asynchronous communication networks. Time-triggered protocols in combination with the time asynchronous model can be used to form a hybrid system model with this property. This project is based on this research topic and focuses on the clock synchronization problem. An adaptive clock synchronization protocol functioning as a part of the hybrid system is fully discussed in this report.

1 Introduction

Traditionally, embedded real-time systems are implemented in dedicated hardware with simple software components in order to meet their stringent requirement on speed and size. However, with the rapid evolvement in hardware industry, hardware can be made much cheaper and faster than before. This enables the construction of real-time systems with more complicated software components.

Distributed real-time systems, which are based on network communication, are one important class of real-time systems because of their capability

and adaptability. These systems used to have dedicated communication networks which operate synchronously. The semantics of these communication networks are simple. However, asynchronous communication networks are beginning to find their role in more and more application areas. These communication networks are asynchronous in the sense that they can not provide certain communication in which messages are guaranteed to be delivered in time. The semantics of the asynchronous networks are more complicated. The problem of how to provide quasi-synchronous semantics for these asynchronous networks so that the system developers can still benefit from the simple semantics of synchronous networks becomes the research topic upon which this project is based.

Time-triggered protocols (TTPs), proposed by H. Kopetz et al.[7] [8] [9], are designed for the implementation of time-triggered hard real-time systems. They are suitable for high-speed, single-failure operational safety-critical applications such as automobile manufacture industry. TTP assumes the underlying communication network to be highly reliable.

The timed asynchronous (TA) model is introduced by F. Cristian et al. in [3] [4] to describe existing distributed systems built from network workstations connected by unreliable communication networks. It allows many needed services such as clock synchronization, membership, consensus etc. to be implemented.

A general idea of hybrid communication systems which accommodate both synchronous and asynchronous communication networks tries to achieve both the wide system coverage and the cost-effective properties at the same time. Inside the hybrid system, TA model can be used in cooperation with TTPs. Such a hybrid system consists of clusters (of processors) with

synchronous intra-cluster communication network and asynchronous inter-cluster communication network. A gateway node behaves as the interface between the cluster and the rest of the system. Gateways, on behalf of their corresponding clusters, communicate via the inter-cluster communication network and the TA model is used to model their behavior.

In a real-time system, there are fundamental services such as clock synchronization service, atomic broadcast service, and membership service etc. [2] [5] The clock synchronization protocol is much more important since many other protocols relies on the synchronized clocks among all the processors. This project focuses on the clock synchronization protocol in the hybrid system and proposed an adaptive clock synchronization protocol. In the following sections, the clock synchronization protocol, the adaptive protocol, and its implementation are discussed respectively.

2 Clock Synchronization

The clock synchronization in the hybrid system consists of two parts. Inside each cluster, where TTP is applied, the processors synchronize to the gateway. Dedicated hardware-supported clock synchronization is used since this kind of network usually has a fault-tolerant system BUS on which high precision, guaranteed clock synchronization can be achieved [6]. However, in the asynchronous communication networks that connect the gateways, which is modeled by TA, it is not possible to achieve guaranteed clock synchronization since the messages in this network can suffer performance failures, i.e. it is possible for the messages to be delivered late. Probabilistic clock synchronization [1] approach is used in this network.

The idea of probabilistic clock synchronization is illustrated in Figure 1.

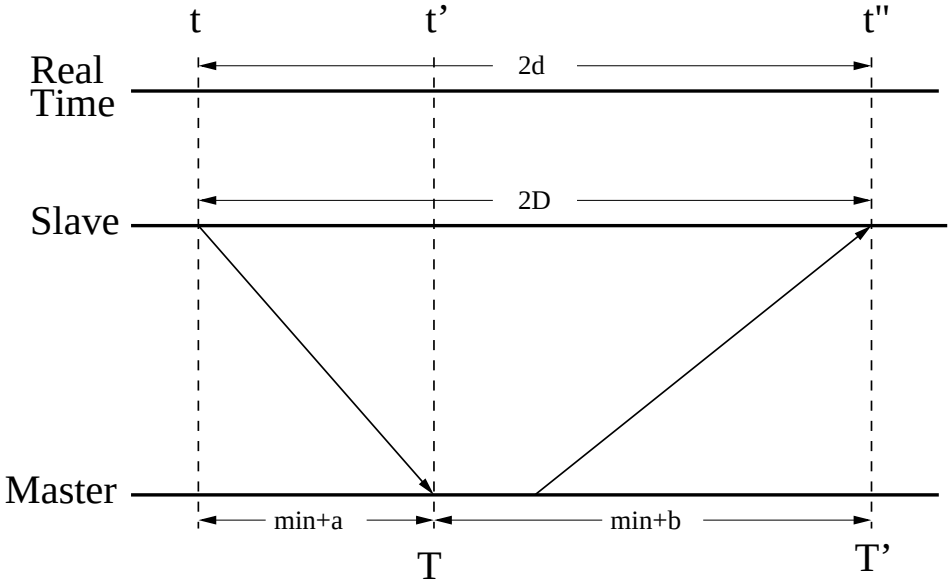

Figure 1: Probabilistic clock synchronization

Whenever a slave processor wants to synchronize with a master processor (at real time t), it sends an inquery message to the master processor. The master processor responds to this inquiry with an acknowledge message which contains its local time T' at which the inquiry arrives. The acknowledge message arrives at the slave processor at real-time t". The slave processor measures the round-trip delay $2D$ between real time t and t" in its local clock. Based on the assumption that the minimum message transmission delay is min and all the clock has a maximum drift rate of ρ, the local clock value of the master processor at real time t" lies in the time interval $[T + min(1-\rho), T + 2D(1+2\rho) - min(1+\rho)]$. Thus, the best estimation the slave processor can make about the master processor's clock value at real time t" is $T' = T + D(1+2\rho) - min\rho$, which is the midpoint of that interval. By doing this, the maximum deviation between the slave and the master clock

achieves its minimum, $e = D-(1+\rho)min$. Since the network is asynchronous, it is possible that the round-trip delay be an arbitrary large value. A timeout value $2U$ is thus introduced so that the slave processor is able to conclude that either the inquiry message or the acknowledge message suffers a performance failure when it does not receive the acknowledge message from the master processor within $2U$ time units by its local clock after sending the inquiry message.

The probabilistic clock synchronization algorithm has to decide when to start the next synchronization after achieving current synchronization. The real time delay to the next rapport, dnr, should be chosen to keep the deviation between the slave clock and the master clock within the desired precision. Given the maximum master slave deviation ms, dnr could be as as large as $\rho^{-1}(ms - e)$. Consider the possibility that a synchronization attempt might fail, k attempts are tried before the synchronization algorithm fails. If consecutive attempts has an interval of W time units($W > 2U$), the maximum real time delay dna between a rapport and the next attempt has to be smaller than $dna = \rho^{-1}(ms - e) - (1 + \rho)kW$.

3 Adaptive Clock Synchronization

Since the communication network is not dedicated in inter-cluster synchronization, it is possible that other communication protocols run simultaneously as the clock synchronization protocol. The network load can change over time, and this potentially affect the communication qualities. Figure 2 shows the round-trip delay in the light load scenario and the heavy load scenario. It is unfortunate to find out that the performance of the original probabilistic clock synchronization algorithm varies with the network load

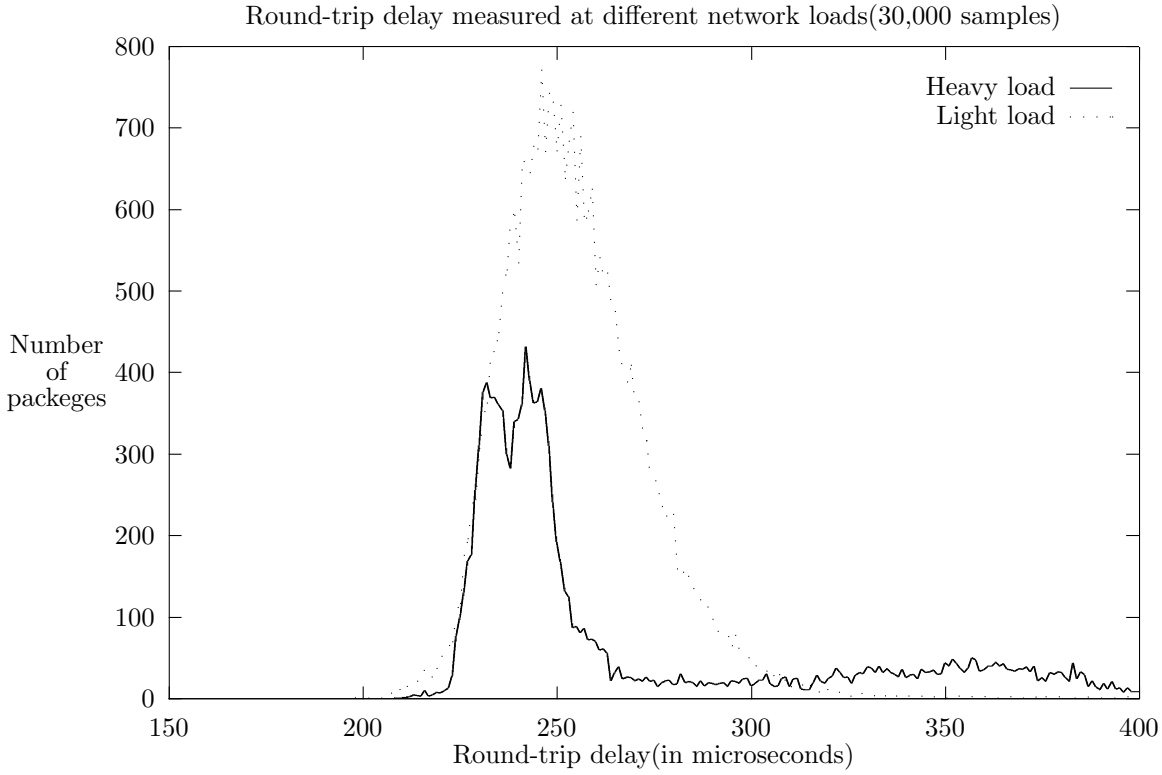

Figure 2: Round-trip delays

significantly. Table 1 shows some simulation result in a light load scenario and a heavy load scenario. As the table shows, the algorithm performs poorly in the heavy load scenario with a failure percentage as high as 20%. The reason is that the timeout value $2U$ is too small in this scenario. However, since it is not possible to predict the network load, there is no way to choose a fixed U that allows the algorithm to perform well in all possible scenarios.

One way to solve this problem is to make U adaptive, i.e. U is modified dynamically to reflect the current network load. The slave processor monitors the history of the round-trip delays. It increases U whenever the

	Light Load	Heavy Load
Probabilistic Protocol		
failure #	107/45068	3125/16194
failure percentage	0.24%	19.30%
Adaptive Protocol		
failure #	23/45227	578/22696
failure percentage	0.05%	2.55%

Table 1: Simulation results

network load is increasing and decreases U when the network load is decreasing. Currently, the adaptive protocol works like this: when it observes more consecutive timeouts, it concludes that the network load is increasing; and when it observes that the round-trip delays are consistently smaller than a given value, which is smaller than $2U$, it concludes that the network load is decreasing. As shown in Table 1, this simple adaptation approach improves the performance significantly, especially in the heavy load scenario(a failure percentage of only 2.55% is achieved).

4 Implementation

The hardware environment to implement the original probabilistic clock synchronization protocol and the adaptive synchronization protocol is Windows CE powered PCs(CEPCs). Windows CE is an operating system targeted on embedded systems. It has special design concerns to support real-time system developing. However, it is still far from a true real-time OS. The advantage of choosing this OS is that developers can benefit from its fully functional Win32 APIs. Its configurable nature also provides a new paradigm to develop embedded systems.

The protocols are developed under Windows CE Studio 3.0(Beta) and Windows CE Platform Builder 2.12. The datagram communication network is implemented using Winsock. The simulations are performed on two CEPCs which are connected via 10MBPS ethernet together with four Linux machines. The network load is simulated by having the Linux machines sending messages to each other at different rates. Some of the simulation results are presented in Figure 2 and Table 1.

5 Conclusions

As the simulation results shown, the adaptive clock synchronization protocol can achieve reliable performance despite network load viariation. This protocol can be further improved by providing fail-awareness [4] semantics with an indicator showing whether the clock is synchronized or not. If synchronization is lost, an application or another service that uses this clock synchronization service can observe this from the indicator and perform corresponding operations, such as degrading the service they provide. It is also possible to extend this protocol in such a way that not only U could be adaptive, but also other system parameters. An even more reliable protocol could be expected on this track. The hybrid system can provide quasi-synchronous semantics by using the fail-awareness approach. Further research work can be performed on how to formally specify the other fail-awareness protocols based on the clock synchronization protocol considered in this project.

References

[1] F. Cristian, "Probabilistic clock synchronization," *Distributed Computing*, vol. 3, no. 3, pp. 146-158, Mar. 1989.

[2] F. Cristian. "Synchronous and asynchronous group communication," *Communications of the ACM*, vol. 39, no. 4, pp. 88-97, Apr. 1996.

[3] F. Cristian and C. Fetzer, "The timed asynchronous distributed system model," *IEEE Trans. on Parallel and Distributed Systems*, vol. 10, no. 6, June, 1999.

[4] C. Fetzer and F. Cristian, "Fail-awareness in timed asynchronous systems," In *Proc. of ACM Symp. on Principles of Distributed Computing*, Philadelphia, May 1996, pp. 314-321a.

[5] C. Fetzer and F. Cristian, "A highly available local leader service," *IEEE Trans. on Software Engineering*, vol. 25, no. 5, Sep. 1999.

[6] H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems," *IEEE Trans. Computers*, vol. 36, no. 8, pp. 933-940, Aug. 1987.

[7] H. Kopetz and G. Grunsteidl, "TTP - A protocol for fault-tolerant real-time systems," *IEEE Computer*, pp. 14-23, Jan. 1994.

[8] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Kluwer Academic Publishers, Norwell (MA), ISBN 0-7923-9894-7, 1997.

[9] G. Kross, "The time-triggered communication protocol TTP/C," *Real-Time Magazine*, No. 4, pp. 100-101, 1998.