The Timely Computing Base and its Future Trends

Pedro Martins
pmartins@di.fc.ul.pt
FC/UL[*]

Paulo Veríssimo
pjv@di.fc.ul.pt
FC/UL

Abstract

Future mobile applications with real-time requirements will demand solutions that traditional real-time approaches might not be able to fulfill. The emergence of new partial synchrony models, such as the Timely Computing Base (TCB) model, may provide the answers we are looking for. More than a framework to build applications, the Timely Computing Base is an adequate model to reason about timeliness issues when designing applications for mobile environments. In this paper we overview the current state of the art with respect to the TCB and we discuss some future trends of the Timely Computing Base towards its implementation in mobile computing.

1 Introduction and Motivation

The rapidly expanding wireless communication technologies will, in the near future, make computer systems accessible virtually anywhere and anytime. Moreover, we have been witnessing a striking user adherence to mobile devices like, for instance, laptop computers or small handheld devices (*PDA*, *Pocket PC*, *Microsoft Smart Phone*, etc.). Most of these mobile devices are capable to communicate through a wireless network to fixed parts of the overall network infrastructure and, possibly, to other mobile devices [9]. The resulting computing environment, mainly characterized by mobile devices travelling through several wireless network infrastructures, is often referred as *mobile* or *nomadic* computing.

With the progressive maturing of mobile computing several new classes of applications will be brought into the field, and some of them will require real-time capabilities from the computational infrastructures. Application domains such as mobile robotics, traffic management, automatic driven cars, or free-flight planes will be widely deployed in the future.

Nevertheless, mobility has raised several research issues namely in some areas considered pretty mature in ”traditional” computing systems. Some examples of the new challenges faced by the research community include new user interface designs, routing algorithms for multi-hop ad-hoc networks, new transport protocols for wireless links (e.g. *Indirect TCP*), energy aware hardware and software, or network routing protocols that support mobility (e.g. *Mobile IP*).

Addressing non-functional requirements, such as timeliness or dependability is also very important in several application domains. However, while there exist several solutions for wired real-time communication (e.g. *FIP - Factory Industrial Protocol* [1], *CAN - Controller Area Network* [6], etc.), when it comes to wireless communication the scenario is pretty much different. Moreover, the distinct nature of wireless and wired physical mediums disallows the applicability of some wired network protocols in wireless mediums

[*]Faculdade de Ciências da Universidade de Lisboa. Bloco C5, Campo Grande, 1749-016 Lisboa, Portugal. Navigators Home Page: http://www.navigators.di.fc.ul.pt. This work was partially supported by the EC, through project IST-2000-26031 (CORTEX).

(e.g. bit arbitration in CAN). In section 3 we discuss a few real-time services provided by some wireless LANs.

In figure 1a it is possible to observe an application scenario consisting of a fleet of automatic-piloted airplanes. Each airplane is equipped with a location information device (e.g. GPS) in order to be able to know its current position. One of the airplanes assumes the role of the fleet leader and all the other must follow it. Although being autonomous, airplanes have access to a short-range wireless network (e.g. 802.11 standard) that is used to support the communication among them.

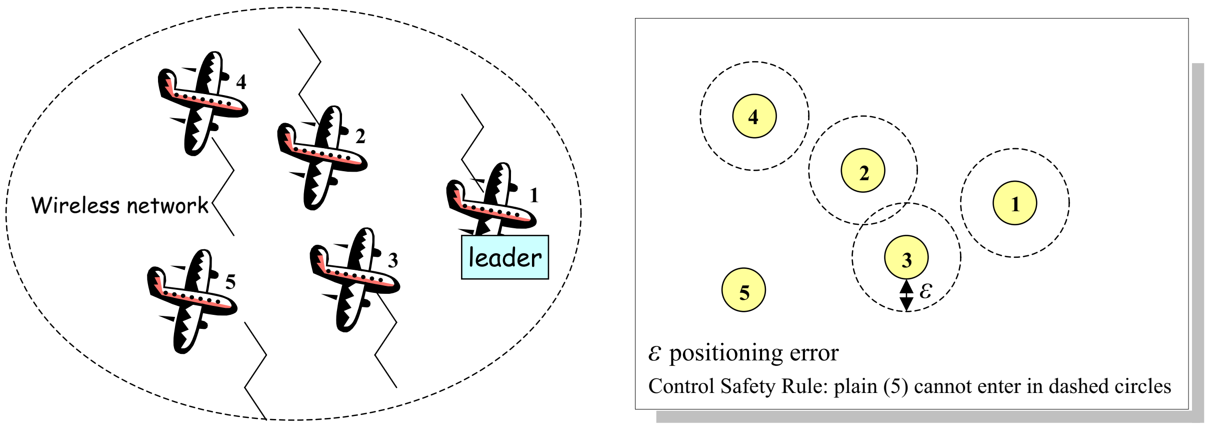
4
2
1
Wireless network
leader
5
3
4
2
1
3
5
ε
ε positioning error
Control Safety Rule: plain (5) cannot enter in dashed circles

Figure 1. a) fleet of plains application scenario; b) view of the environment from the perspective of plane number 5 (errors introduced by the imprecisions such as those resulting from GPS readings are not considered).

In this application scenario, several important requirements are raised. **Consistency** is needed to ensure that every plane has the same view (in time and space) of the environment, so that they can know each other's positions and so that crashes are avoided. **Coordination** is needed to make specific formations during the flight, or particular flight sequences. **Cooperation** is clearly necessary to achieve the previous goals, and also to ensure safety properties. For instance, it is necessary to ensure that every plane follows its leader, and maintains a certain distance to planes flying side by side.

Furthermore, given that: a) airplanes periodically disseminate information containing its position, and b) each airplane must achieve a consistent view in time and space of other plane positions, there exist some timeliness, or real-time, requirements (network communications and local processing) that must be verified. As a matter of fact, in order to ensure that airplanes only make decisions that preserve the safety of the overall system, it is necessary that, in every instant, each airplane knows the position of all other airplanes with a known and bounded positioning error (see figure 1b). To achieve that, a maximum airplane speed must be defined and each plane must perform the following computations:

- disseminate its position to the other airplanes in a timely manner, that is, periodically and with a bounded communication latency. Based on the positioning information received from other airplanes, each plane should represent internally its perception of the surrounding environment;

- timely process the positioning information when it is received from other airplanes. Actuation in response to specific conditions (e.g. change the plain direction to avoid crashing with another one) must be also timely undertaken.

Temporal correctness of the execution of the above computations implies that the computational infrastructure where the application is running enjoys real-time properties. In "traditional" hard real-time systems, timeliness is enforced by mechanisms such as real-time scheduling, real-time communication, etc. These mechanisms are built under strict assumptions about system loads, faults and topology, taking into account some infrastructure variables like CPU processing power or network latency and throughput.

However, in mobile environments, where devices may travel through several networks, real-time properties of the infrastructures where applications execute may be quite difficult to foresee. Moreover, mobile environments are inherently dynamic (e.g. several devices entering and leaving a network) and therefore timing properties of the services provided by a network (that is, its provided *Quality-of-Service (QoS)*) may oscillate over the time.

In essence, we have to face the problem of conciliating real-time requirements with the unpredictable nature of the environment. This raises the following two questions: 1) what does it happen if timing assumptions made about the infrastructure are not met during the execution of the application? 2) which time bounds should be assumed? If timing assumptions are not met during the execution of applications *timing failures* occur. Timing failures can affect applications in several ways [11]. For instance, considering the previous example of the flying fleet, the occurrence of a timing failure may lead to a positioning error bigger than the value (ϵ) tolerable by the control logic of the plains, and therefore put in risk the safety of the airplanes. The second question is motivated by the uncertain timeliness properties of target infrastructures. If we don't know in which infrastructures applications will execute, how can we make assumptions about timing variables like for instance the network transmission delay.

The Timely Computing Base model (TCB) [11, 12] proposes an architectural construct that constitutes a generic approach to the problem of programming applications with real-time requirements in the presence of uncertain timeliness. The TCB model assumes the existence of a component, capable of executing timely functions, which helps applications with varying degrees of synchrony to behave reliably despite the occurrence of timing failures. A TCB component provides services for timely detecting timing failures (in a complete and accurate way), for duration measurement, and for the timely execution of small functions.

The Timely Computing Base does not make the applications timelier than they are. However, it allows them to know if their timing assumptions are being fulfilled by the infrastructure, providing means to timely execute safety procedures if timing failures occur.

In our scenario, by using the timing failure detection service of the TCB, it is assured that if timing failures occur the TCB will timely undertake safety procedures. For example, a safety procedure would be to make an airplane going up to some other position considered safe.

Moreover, the TCB provides extensions for dependable QoS adaption [4], giving feedback to applications of the changing conditions of the infrastructure. Based on this information, applications must adapt their functionality updating their timing requirements to the ones offered by the infrastructure.

Once again, in the plane scenario, by using the QoS adaptation extensions of the TCB, plane speed would be adjusted in order to secure the allowed positioning error (ϵ) on each plain view.

One of the main advantages of reasoning about applications using the Timely Computing Base model, is to allow the encapsulation of real-time requirements of the infrastructures inside the TCB components. The synchrony requirements emerging in several application domains are thus confined to the feasibility of a TCB implementation.

In the next section, the Timely Computing Base model and an implementation of a TCB in RTLinux are described. The following section presents some future trends of the TCB towards its applicability in mobile environments. We finish the paper with some concluding remarks.

2 The Timely Computing Base Model

A system with a TCB is divided into two well-defined parts: a *payload* and a *control* part. The generic or *payload* part prefigures what is normally 'the system' in homogeneous architectures. It exists over a global network or *payload* channel and is where applications run and communicate. The *control* part is made of local TCB modules, interconnected by some form of medium, the *control* channel. Processes p execute on several sites, making use of the TCB whenever appropriate. Figure 2 illustrates the architecture of a system with a TCB.

Concerning the payload part, the important property is that the system *can have any degree of synchronism*, that is, if bounds exist for processing or communication delays, their magnitude may be uncertain or not known. Local clocks may not exist or may not have a bounded rate of drift towards real time. The system is assumed to follow an omissive failure model, that is, components *only do timing failures*— and of course, omission and crash, since they are subsets of timing failures— no value failures occur.

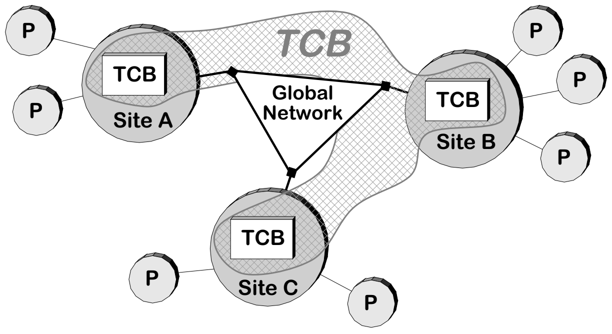

Figure 2. The TCB Architecture.

In the control part, there is one local TCB at every site. TCB modules are assumed to be fail-silent, that is, they only fail by crashing. Moreover, it is assumed that the failure of a local TCB module implies the failure of that site. In terms of synchrony, the TCB subsystem preserves, by construction, upper bounds on processing delays (property **Ps 1**), on the drift rate of local TCB clocks (**Ps 2**) and on the delivery delay of messages exchanged between local TCBs (**Ps 3**).

Given the above set of properties, a TCB can be turned into an oracle for applications (even asynchronous) to solve their time related problems. To accomplish this, a set of minimal services has to be defined, as well as the payload-to-TCB interface. In order to keep the TCB simple, the services defined are only those essential to satisfy a wide range of applications with timeliness requirements: ability to measure distributed durations with bounded accuracy; complete and accurate detection of timing failures; ability to execute well-defined functions in bounded time. A complete description of these services can be found in [11].

2.1 Real-Time Linux TCB

To demonstrate the feasibility of a TCB implementation, we have developed a prototype for the RTLinux system. A detailed description of this implementation can be found in [3].

RTLinux is a version of Linux that allows the execution of hard real-time threads and interrupt handlers in the kernel of the operating system. In a x86 generic system, RTLinux schedules periodic real-time tasks with up to a $35\mu s$ latency, and real-time interrupt handlers in $15\mu s$ [2, 14, 15].

A TCB module executes in Linux as a kernel module, using the services provided by RT-Linux executive to handle real-time tasks and hardware interrupts. From a Linux architectural point of view, applications are entities that run in user level space and the TCB is a component that runs in system kernel space. Figure 3 illustrates a composition of a system node with a TCB.

In our RT-Linux implementation, the TCB synchronous communication channel is based on a physically different network from the one supporting the *payload* channel. For the *control* channel we have used a switched Fast-Ethernet network, exclusively dedicated to connect local TCBs. The rest of the system is interconnected through a normal 10Mbit Ethernet network.

By exploiting some specific characteristics of switched Fast-Ethernet networks, it was possible to eliminate the unpredictability introduced by the Binary Exponential Back-off collision resolution algorithm of Ethernet networks. The solution simply consists in avoiding packet collisions. This can be achieved by connecting each port of the switch to a single station to ensure that at most one adapter may transmit on

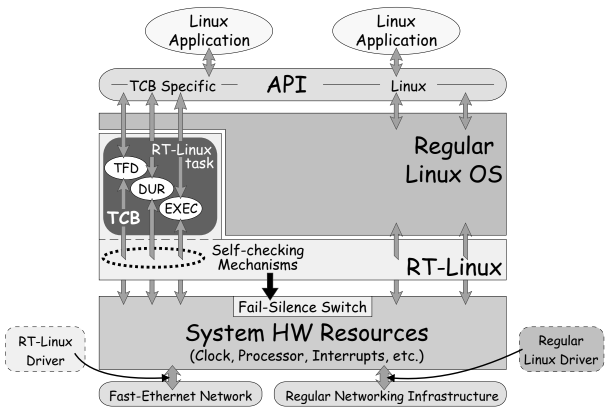

Figure 3. Block Diagram of a RT-Linux system with a TCB

each direction of the full-duplex link. Therefore, provided that the switch buffering capacity is not exceeded (which would introduce additional, possibly unpredictable delays), it is possible to guarantee a maximum transmission delay for each packet. The results of a set of experiences that we have done in order to evaluate the real-time capabilities of switched Ethernet networks can be found in [3].

3 TCB Future Trends

In order to develop a Timely Computing Base tailored for mobile environments, it must be first understood what kind of real-time capabilities are provided by existing mobile communication infrastructures and operating systems. An implementation of a TCB should preserve the properties **Ps 1 and Ps 3** [1]. Therefore we now review the Windows CE 3.0 operating system and 802.11 wireless LANs with respect to the fulfilment of these properties.

Windows CE 3.0 is an operating system designed for the embedded marketplace. Windows CE 3.0 is an evolution of Windows CE 2.12 with improved hard real-time capabilities [13, 7]. Real-time support enhancements that Microsoft introduced with Windows CE 3.0 to improve real-time performance can be divided into two major categories: interrupt handling and thread scheduling. The kernel itself was significantly rewritten to improve the handling of preemption and to reduce the time when the kernel is non-preemptible. Support was also added for nestable interrupts, that is, interruptible Interrupt Service Routines (ISRs). In previous versions of Windows CE, an ISR could not be interrupted, which meant that the possible delay for any given ISR to start was equal to the running time of the slowest ISR. The second major improvement is in the area of thread scheduling. Incidentally, thread scheduling improvements help interrupt handling, since an Interrupt Service Task (IST) must be scheduled like any other thread. The introduced improvements for thread scheduling were the following ones: new thread priorities, the addition of support for setting thread quantum, improved handling of priority inversion, improved timer granularity, the addition of semaphores, and better support for critical sections. Windows CE 3.0 provides a timer resolution of $1ms$, which is enough to allow the implementation of a TCB.

In terms of wireless communication, the 802.11 became the standard for wireless ethernet networking technology. An 802.11 wireless network adapter can operate in two modes, *infrastructure* and *ad-hoc*. In an infrastructure network there is an *access point* that is used in all communications between stations inside its coverage area. Each transmission takes two hops. First, the source station transfers the frame to the access point, and then the access point transfers it to the destination station. In ad-hoc mode, stations talk directly to each other and do not need an access point at all.

In 802.11, access to the medium is ruled by the *Distributed Coordination Function (DCF)*. The DCF is the

[1]It is assumed that all devices are equipped with precise clock crystals or UTC receivers (e.g. GPS) and therefore **Ps 2** is implicity ensured.

basis of the standard *carrier sense multiple access with collision avoidance (CSMA/CA)* access mechanism. Similarly to the wired Ethernet, the medium must be sensed before any data transmission can start. When a node receives a packet to be transmitted, it first listens to ensure that no other node is transmitting. If the channel is clear, it then transmits the packet. Otherwise, it chooses a random backoff factor, which determines the amount of time the node must wait until it is allowed to transmit its packet. During periods in which the channel is clear, the transmitting node decrements its backoff counter (when the channel is busy it does not decrement its backoff counter). When the backoff counter reaches zero, the node transmits the packet. Since the probability that two nodes will choose the same backoff factor is small, collisions between packets are minimized [10]. In addition, DCF may use the *Ready-to-send/Clear-to-send (RTS/CTS)* algorithm to further reduce the probability of collisions.

To support applications that require real-time - or guaranteed response time - services, the 802.11 standard includes a coordination function which allows a deterministic access to the medium. The *Point Coordination Function (PCF)* provides contention-free frame transfers [8].

When PCF is used, time on the medium is divided in two distinct periods: *contention-free period (CFP)* and *contention period (CP)*. During the contention period, access to the medium is ruled by the DCF described above. The PCF controls transmissions in the medium during contention-free periods.

The PCF acts as master central arbitrator for bus access during CFP, polling stations for data transmission according to a polling list. During the contention-free period, stations may transmit only if the PCF solicits the transmission with a pooling frame. Thus, contention on the access to the bus and transmission collisions are avoided.

Although the PCF appears to be an attractive solution to achieve real-time communications in 802.11 networks, point coordinators must reside in access points and, therefore, contention-free services are restricted to infrastructure network configurations. Providing contention-free services in ad-hoc networks is not covered by the 802.11 standard. To address the need for real-time behavior in ad-hoc networks another solutions must be envisaged. For instance, the *Time Bounded Medium Access Control (TBMAC)* [5] is a medium access control protocol that provides, with high probability, time-bounded access to wireless medium in ad-hoc multi-hop networks. The TBMAC protocol is based on a time-division multiple access with dynamic but predictable slot allocation.

It is thus clear that the construction of a TCB should be feasible in wireless environments. As future work, we intend to develop real-time control channels for the TCB using the Point Coordination Function of 802.11 and a TBMAC based network.

4 Concluding Remarks

In this paper we discussed some real-time requirements of the newly emerging mobile applications. The Timely Computing Base model seems to be an appropriate construct to address mobile applications with timeliness requirements. While it is possible to construct a TCB over a fixed infrastructure, as we have shown for the case of RTLinux and Fast Ethernet, more challenging is the implementation of a TCB in a wireless environment. The discussion presented in this paper allow us to be optimistic in this respect.

References

[1] J. D. Azevedo and N. Cravoisy. WorldFIP protocol, 1998. http://www.dia.uniroma3.it/autom/Reti_e_Sistemi_Automazione/PDF/WorldFip_Protocol.pdf.

[2] M. Barabanov and V. Yodaiken. Real-time linux. *Linux Journal*, Mar. 1997. http://rtlinux.cs.nmt.edu/rtlinux/papers/lj.ps.

[3] A. Casimiro, P. Martins, and P. Verssimo. How to Build a Timely Computing Base using Real-Time Linux. In *Proc. of the 2000 IEEE Workshop on Factory Communication Systems*, pages 127–134, Porto, Portugal, Sept. 2000.

[4] A. Casimiro and P. Verssimo. Using the Timely Computing Base for dependable QoS adaptation. In *Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems*, pages 208–217, New Orleans, USA, Oct. 2001.

[5] R. Cunningham and V. Cahill. Time Bounded Medium Access Control for Ad Hoc Networks. In *Proceedings of the Workshop on Principles of Mobile Computing, POMC 2002, Toulouse, France - to appear*, Oct. 2002.

[6] R. B. GmbH. CAN specification version 2.0, 1991.

[7] O. Grbner, H. Kfner, O. Stierschneider, and J. Schmitt. Assessing Microsoft Windows CE 3.0 Real-time Capabilities, 2001. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnce30/html/realtimecapabilities.asp.

[8] IEEE. IEEE std 802.11 - wireless lan medium access control (MAC) and physical layer (PHY) specifications, 1997.

[9] T. Imielinski and H. Korth. *Introduction to Mobile Computing*. Kluwer Academic Publishers, 1996.

[10] D. L. Lough, T. K. Blankenship, and K. J. Krizman. A Short Tutorial on Wireless LANs and IEEE 802.11. http://www.computer.org/students/looking/summer97/ieee802.htm.

[11] P. Veríssimo and A. Casimiro. The timely computing base model and architecture. *Transaction on Computers - Special Issue on Asynchronous Real-Time Systems*, 51(8), Aug. 2002.

[12] P. Veríssimo, A. Casimiro, and C. Fetzer. The timely computing base: Timely actions in the presence of uncertain timeliness. In *Proceedings of DSN 2000, the IEEE/IFIP Int'l Conf. on Dependable Systems and Networks*, pages 533–542, New York City, USA, June 2000. IEEE Computer Society Press.

[13] P. Yao. Windows CE 3.0: Enhanced Real-Time Features Provide Sophisticated Thread Handling, 2000. http://msdn.microsoft.com/msdnmag/issues/1100/RealCE/RealCE.asp.

[14] V. Yodaiken and M. Barabanov. A real-time linux. In *Proceedings of the USENIX conference*, 1997. http://rtlinux.cs.nmt.edu/rtlinux/papers/usenix.ps.gz.

[15] V. Yodaiken and M. Barabanov. Rtlinux version two. Technical report, VJY Associates LLC, 1999.