Internet-based Real-Time Computing ?

Ulrich Schmid
Technische Universität Wien
Department of Automation
Treitlstraße 1, A-1040 Vienna
Email: s@auto.tuwien.ac.at

Apart from the ever-growing need for more bandwidth, Internet-based applications are increasingly concerned about other quality-of-service issues as well. Among those are high availability and service-time guarantees, which are mandatory for next-generation applications like e-cash transaction processing. This fact creates unanticipated connections to the field of fault-tolerant real-time computing (FT-RTC), with a number of interesting consequences.

In what follows, we will argue that

(a) traditional paradigms of FT-RTC research are not applicable here, but

(b) the "dynamic school" of FT-RTC research could provide the required support and will hence gain in importance.

The major argument backing up (a) is the incompatibility of the underlying system models. "Hard" real-time computing is only possible in a synchronous model of computation. In fact, without bounded transmission delays/execution times and bounded-drift clocks, there is no way of guaranteeing even a simple real-time constraint. Consequently, traditional real-time systems are solely restricted to tightly coupled shared-memory multiprocessors or LAN-based distributed systems. Nevertheless, since shared resources like Busses/LANs or mutual exclusive data cannot be avoided completely, all processes must be carefully scheduled to allow them to complete within their deadlines.

The "static school" of real-time systems, which currently dominates FT-RTC, solves this problem by means of *time-driven systems*: All system activities are initiated according to an *a priori* fixed — usually cyclic— time schedule here. This paradigm is backed-up by the well-developed rate-monotonic scheduling theory and usually implemented atop of accurately synchronized clocks, which can easily be built in synchronous systems. It is important to note, however, that time-driven systems recognize external events at predetermined instants only, i.e., the event-flow from the outside world is effectively cut at the external inputs.

The time-driven approach is particularly attractive due to the fact that fault-tolerance mechanisms are easily incorporated. This is not easy in general, since FT-RTC needs to combine two issues that usually conflict with each other. For example, time redundancy techniques like retransmitting lost messages or roll-back recovery dramatically increase the overall transmission/execution time in case of a failure. Nevertheless, given the time-driven systems' total synchrony assumption in conjunction with complete control over scheduling instants, it is relatively easy to add both redundant processors/networks and multiple message transmissions for tolerating simple classes of failures.

By contrast, Internet-based FT-RTC imposes an asynchronous computational model, where tranmission delays and execution times are potentially unbounded. This is due to the fact that neither network congestion nor excessive workload can be ruled out in an open distributed system. In addition, simple failure models are no longer sufficient either, since byzantine (malicious) failures can easily occur in such systems. Unbounded execution times and transmission delays, however, violate a basic requirement of rate-monotonic scheduling theory, and deterministic clock synchronization is also impossible to achieve in asynchronous systems. Consequently, the "resource-adequate" design paradigm of time-driven systems cannot be applied here at all.

However, there is also a "dynamic school" in FT-RTC, which did not receive much attention up to now. Its major concern is to avoid a more restrictive system model whenever a weaker one would suffice. For example, instead of artificially cutting the external event-flow to make rate-monotonic scheduling applicable, online scheduling in conjunction with dynamic guarantees could be used. Processes are initiated dynamically upon occurence of external events, but they are only admitted to scheduling if the overall system load allows all deadlines to be met. If the load offered to the real-time system is less than the maximum sustained one,

all processes will be admitted. In case of overload, however, some processes' guarantee will be denied, which can be used to initiate graceful degradation measures.

Fortunately, many other services in distributed real-time systems can also be extended to be *fail-aware*, in the sense that they provide a dynamic failure indicator signaling abnormal operating conditions. For example, in case of clock synchronization, it is of course impossible to guarantee bounded accuracy during phases of excessive transmission delays. However, it is always possible to provide a reliable bound on the instantaneous accuracy locally at each node. Note anyway that just providing a failure indicator is of course not sufficient to ensure proper graceful degradation; a well-defined set of safety properties must always be maintained for this purpose as well. For example, there are communication protocols that provide certain liveness properties only when clocks are synchronized, whereas the most crucial safety properties are maintained under all circumstances.

There is strong evidence that research in dynamic real-time systems will eventually dominate the field of FT-RTC. Apart from the above mentioned Internet-based real-time applications, another major driving force will be the recent shift in policy of US organizations like DoD and NASA, which now favor COTS-based system designs instead of special-purpose ones. Consequently, even mission-critical systems will soon be built atop of commercially available system components, of course guarded by a suitable system software. Needless to say, given the complexity of the issues involved, it will take some time until mature solutions will be available.