This presentation was held at the 14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS 2006), a satellite workshop of IEEE 20th International Parallel & Distributed Processing Symposium (IPDPS 2006), April 25–29, 2006. Rhodes Island, Greece. It was made by Santiago Urueña Pascual, PhD Student working at the real-time research group (Sistemas de Tiempo Real y Arquitectura de Servicios Telemáticos, STRAST) of the UPM, Spain (Spanish: Universidad Politécnica de Madrid, Technical University of Madrid). Our research group is mainly working on hard real-time and embedded systems, especially critical ones. This was a 20 min. presentation, held at 15:30h, at the general session called “Real-Time Communication II”, session chaired by Giuseppe Lipari.

This presentation will talk about the Arbitrated Real-Time Protocol (AR-TP), a network protocol for hard real-time distributed systems whose main goal is predictability in the time domain. I will describe how it works, of course, but what I really want to transmit to you is why this is the main goal of the protocol, and the philosophy behind it. This presentation assumes knowledge on hard real-time systems, Real-Time Operating Systems, Schedulability Analysis, and Ethernet.
Schedulability Analysis of AR-TP, a Ravenscar Compliant Communication Protocol for High-Integrity Distributed Systems

I didn’t realized how long the title was until I have to write it down into a slide…

As can be read in the complete title of the paper, one basic technology to fully understand AR-TP is Ravenscar. You may have heard about Ravenscar before, it is a technology associated with High-Integrity Systems (HIS).

So, I’ll first clarify some terminology about high-integrity systems, next I’ll define Ravenscar, and finally I will describe our protocol.
High-Integrity Systems

- Safety critical
- Security critical

HISs can be described as those systems that cannot fail under any circumstance. In a HIS some software has a direct implication in the behavior of the system, so a software failure can result in the loss of human life or an environmental harm.

Here’s another definition extracted from the book *High-Integrity System Specification and Design*, by Jonathan P. Bowen, Michael G. Hinchey:

“High-Integrity Systems, or systems whose code is relied upon to be of the highest quality and error-free, are often both security- and safety-critical in that their failure could result in great financial losses for a company, mass destruction of property and the environment, and loss of human life.”

Some security critical systems are also HIS, but we are not interested in them because they usually don’t have real-time requirements. Examples of HISs are a nuclear power plant, the avionics system of an airplane or, of course, the International Space Station.

Anyway, because the software of a HIS cannot fail, a new step is needed in the development process: certification.

*Image by NASA*
Certification

- Validation & Verification techniques

*Every software code line* of a HIS must be certified using appropriate validation and certification techniques, like *formal code reviews or testing*, and also an *extremely high volume of documentation.*

And because safety critical systems usually have *hard real-time requirements*, so…

*Image by NASA*
“A Safety-Critical system has to be based
  - on technology that supports *predictability* in the time domain, and
  - the development process has to provide tools for *verifying* the correctness of the timing.”

— Alan Burns & Peter Puschner

…as professor Alan Burns states, a safety critical hard real-time system must be based not only on predictable technology, but the timing of the whole system must also be verified with the help of tools.

The certification of a system is not easy nor cheap, it is very expensive indeed. The software of this type of systems used to be very small to ease the certification process, but when more and more functionality is demanded to this type of systems new development techniques are needed.

This led us to Ravenscar…

The quote was extracted from the first paragraph of:

The complete quote is:

“A development process for safety-critical real-time computer systems has to emphasize the importance of time. On the one hand, such a development process has to be based on hardware and software technology that supports predictability in the time domain. On the other hand, the development process has to provide tools for assessing and verifying the correctness of the timing of both the hardware and the software components of the real-time systems being developed.”
What is Ravenscar?

- Computational Model
- Simplified run-time system
- Schedulability analysis

Ravenscar is a computational model designed for high integrity, hard real-time embedded systems.

It is a profile that specifies the set of operations that the real-time operating system (RTOS) has to provide, and also the set of forbidden operations, this is very important, that would made the system unpredictable.

So on the one hand, Ravenscar compliant real-time kernels have to provide less functionality than other RTOSs, and therefore they will be smaller and easier to certify. In fact, traditional RTOSs cannot be certified because they are too big and beyond the certification techniques, and HI software run without an operating system, only on a bare machine.

And on the other hand, applications developed under the Ravenscar restrictions can have a temporal analysis.

Actually Ravenscar is a village in England, where experts in high-integrity systems and hard real-time systems met (IRTAW’98) to define the profile.
We can say that Ravenscar follows the “less is more” philosophy. The kernel does not provide a lot of operations because actually applications do not need them, and moreover the simpler the kernel the easier to certify it.

Examples of Ravenscar restrictions are that the number of threads and mutexes must be static, or that conditional variables cannot have timed operations (no aborts). It also specifies the adequate scheduling policy: Fixed Priority Preemptive Scheduling (FPPS) for threads, and the Immediate Priority Ceiling Inheritance Protocol (IPCIP) for shared resources. This set of operations is expressive enough for hard real-time embedded systems.

The resulting run-time systems also have some nice features like blocking time minimization (it is worth noting that the maximum blocking time must be documented by the implementation, as well as other measures), an other side effects like high-performance and low memory footprint very useful for embedded systems with low hardware resources and applications with tight deadlines. Also, thanks to the IPCIP there cannot be deadlocks.

*Image by Eric Pierce, under the terms of the GFDL the Creative Commons Attribution Share-Alike.*
Applications

- Ada programming language
- Time & event triggered
- Off-line temporal analysis

On the other side, applications can still be implemented in Ada, the programming language usually employed in safety-critical systems due to its features, but now they are easier to implement because Ravenscar allows not only time triggered jobs, but also event triggered jobs. This is important because in the past HISs were only composed of sequential code because traditional real-time kernels were too big and therefore they could not be certified. But now, Ravenscar allows programming with threads, and it has really improved the development process of safety critical systems.

It is an ISO standard, and it is well accepted in the industry, specially in aerospace and transportation domains. For example, it is used by the European Space Agency (ESA).

However, currently it is only for monoprocessors, so our research group is working to extend Ravenscar for distributed systems. And one of the steps needed to extend Ravenscar to distributed systems is to be able to analyze the schedulability of the network. In some networks and buses this is nearly straightforward, but for those unpredictable networking technologies we have designed AR-TP…
AR-TP

- Research protocol
- Real-time communication over non-deterministic networks

AR-TP is a research network protocol based on RT-EP, Real-Time Ethernet Protocol, made by the University of Cantabria (Spanish: Universidad de Cantabria, UC), Spain. AR-TP allows hard real-time communication over networking technology without a deterministic media access control like half-duplex Ethernet or wireless networks.

It is like TDMA, the Time Division Multiple Access technology, in the sense that it avoids collisions on the shared medium because at every moment only one node of the distributed system has the right to transmit.
AR-TP is a software layer between the device driver and the middleware, that is:

• On the one hand it works with **unmodified networking hardware**. Our implementation of the protocol runs with off-the-self Ethernet interface cards and hubs.

• On the other hand, applications usually don’t use the protocol directly but through a middleware, and therefore **AR-TP doesn’t impose any communication paradigm to applications**.

Of course, in a HIS all its software is tailored to the system, including the hardware, the RTOS and the device drivers. It must be noted that the whole prototype implementation was done with standard hardware (a PC-104), but the whole software stack was designed for HISs:

- The Open Ravenscar Kernel (**ORK**) is our implementation of the run-time system [http://www.dit.upm.es/ork/]. It is a Ravenscar compliant RTOS for hard real-time HIS used by the European Space Agency.

- The Ada Run-Time System is a subset of the GNAT compiler standard libraries that follows the Ravenscar restrictions.

- The device driver for standard Ethernet cards is also Ravenscar compliant.

We currently don’t use any middleware, but we are working to integrate it with PolyORB.

So, how AR-TP works?

*Image made by Daniel Berjón*
Protocol Description

- Token passing
- Two phases:
  - Arbitration phase
  - Transmission phase

AR-TP is based on token passing, so only the station with that special message has the right to transmit, and therefore there are no collisions.

It has two phases: arbitration phase and transmission phase.
Arbitration Phase

In the arbitration phase the token is circulated among all nodes, and each node **writes onto the token** the priority of its highest-priority messages, so at the end of the arbitration phase **the priority of the highest priority messages of the whole system** is recorded on the token…

*Image by Stephane Tsacas, under the term of the GFDL.*
Transmission Phase

And at the transmission phase only those highest priority messages are sent. And at the end of this phase another arbitration phase starts.

The arbitration phase can be seen as an overhead because no actual data is sent in this phase, however it is intended to be short because the token is very small. And we also use the token for additional useful services, like congestion management.

Image modified by Santiago Urueña from an image by Stephane Tsacas, under the term of the GFDL.
Temporal Model

And finally, the temporal model.

*I will not describe in detail* the equations because they are explained in the paper.

They are *a tool that allows to*…

*Image by Nik Frey.*
Schedulability Analysis

\[ R_i = Q_i + Tr \]
\[ B = Ar + \max(Tr, W) \]

…to compute the maximum response time of each periodic and sporadic message.

The idea behind the first equation is that the worst case response time of a message depends on its priority, that is, it depends on ‘i’.

The second equation shows that the blocking time is bounded, and it is always constant.

The blocking time can be adjusted by configuring some parameters of the protocol, like the number of messages sent in the transmission phase of the maximum size allowed for a data message.
AR-TP Features

- Hard real-time
- Event triggered
- Fully distributed
- …but still some overhead

So AR-TP allows hard real-time communication, and maybe it is only for hard real-time systems because probably there are better protocols for soft real-time.

It is event triggered, rather than time triggered as TDMA, and therefore it follows the philosophy of Ravenscar. But of course TDMA can also be used with Ravenscar, and in fact we’re investigating in which cases it is better than AR-TP.

And other feature is that it is fully distributed, there is no single point of failure, this is important for dependability.

But we know in some cases the total predictability of AR-TP has a cost.
Conclusions

- Certification is hard
- Importance of schedulability analysis
- Work in progress

**In summary**, certification of a system is hard, specially when it has to address hard real-time requirements.

New development methods are needed, and these must take into account the schedulability analysis.

And finally, AR-TP is a work-in-progress protocol that *should be seen as a step in the process* of extending the Ravenscar profile to distributed systems.

*Image by Hajor, under the terms of the Creative Commons Attribution ShareAlike License v. 1.0 and/or GFDL.*
So, thank you very much for your attention.
The handout of this presentation will be available at our web (http://www.dit.upm.es/rts/papers/wpdrts2006/handout.pdf), as well as other papers about AR-TP and Ravenscar.

Further reading:

The following slides shows the performance of the prototype, and some additional features. They are extracted from the first article above.
Effective bit-rate for 3 stations sending messages of 1500 octets. The main bottleneck is the delay between message. When this delay is small, the bit-rate can be over the 75%, higher than pure Ethernet.
Max. blocking time

Blocking time for a system of 3 stations
Other services

- Congestion management
- Heart-beat
- Fault-tolerance

See: