Oh, the tangled webs we weave…
From digital systems to complex systems-of-embedded-systems

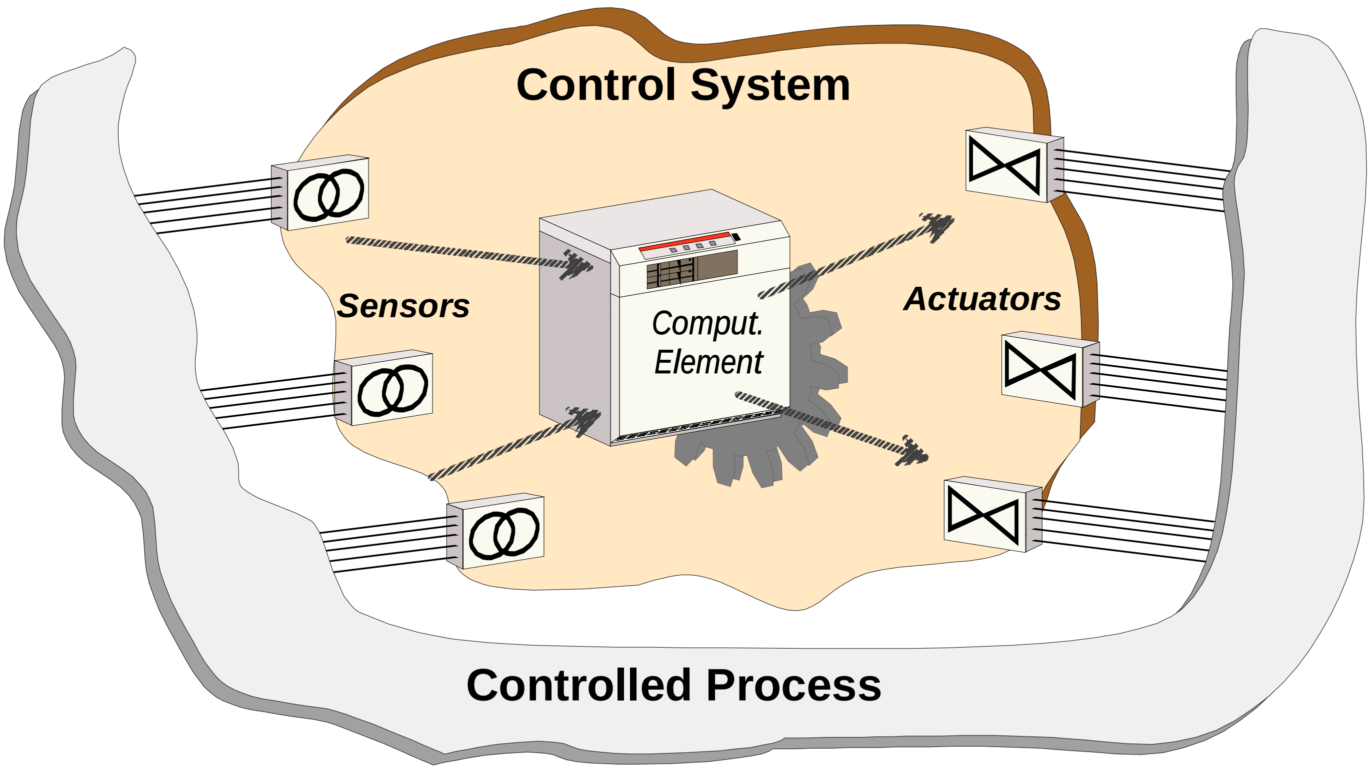
Paulo Esteves Veríssimo

Navigators Group,
LaSIGe, Laboratory for Large-Scale Informatic Systems

Univ. of Lisboa, Portugal
pjv@di.fc.ul.pt
http://www.di.fc.ul.pt/~pjv

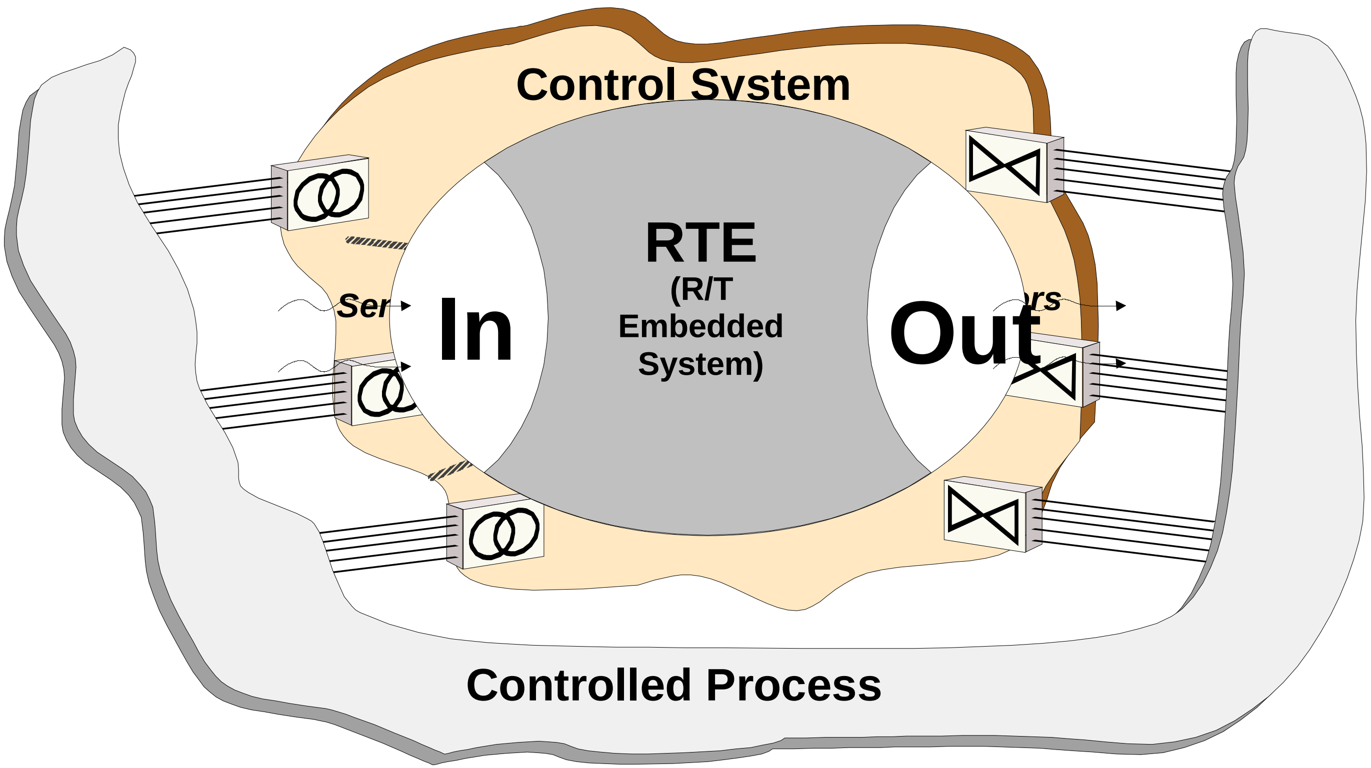
- the paradigms and technologies that we use to automatically observe and control our environment give quantum leaps every decade.
  - from relays, to transistor logic, to microcontroller embedded systems, to field buses and networked embedded systems.
- So far, we could pretend these are all *digital systems*,
  - the magic of the time-triggered abstraction and the synchronous programming languages.
- But how much further can we push the metaphor?

Flying over R/T computing evolution



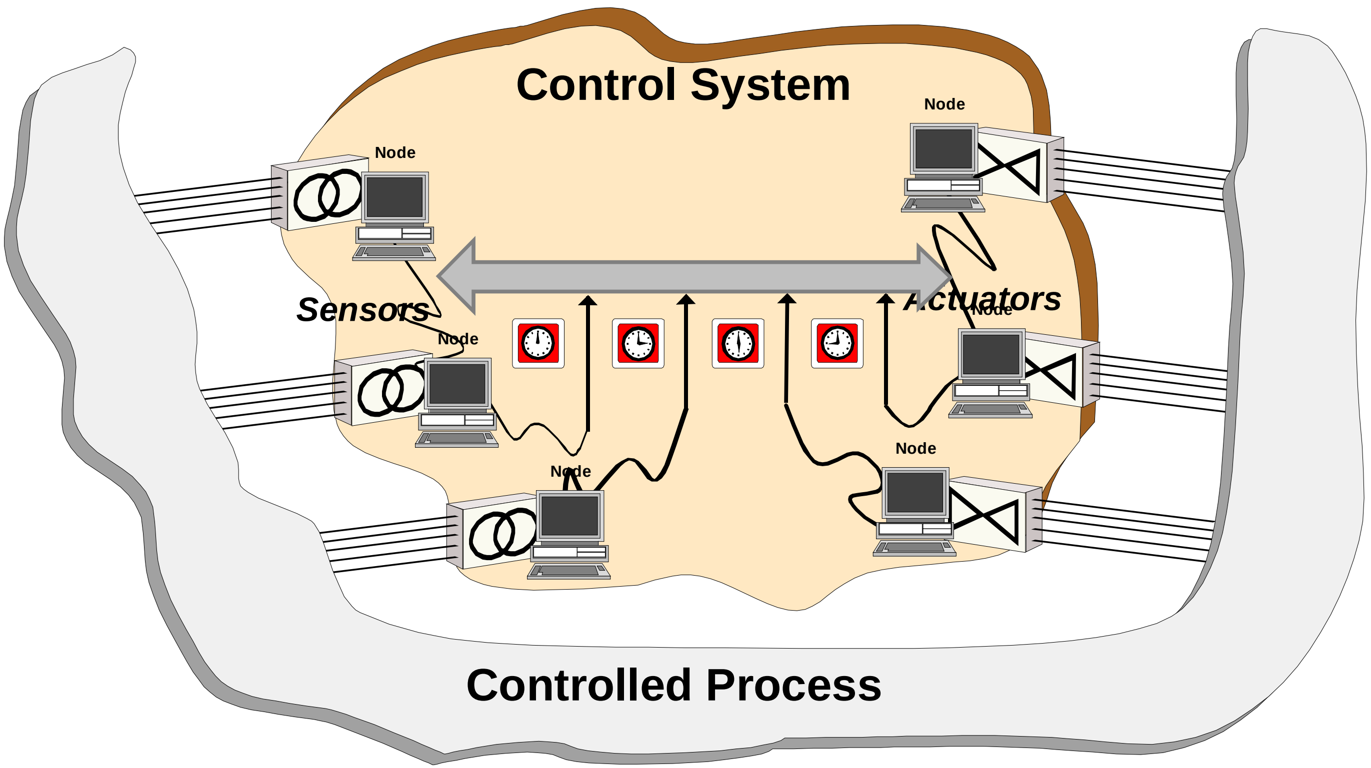
- Centralized control architecture

Flying over R/T computing evolution



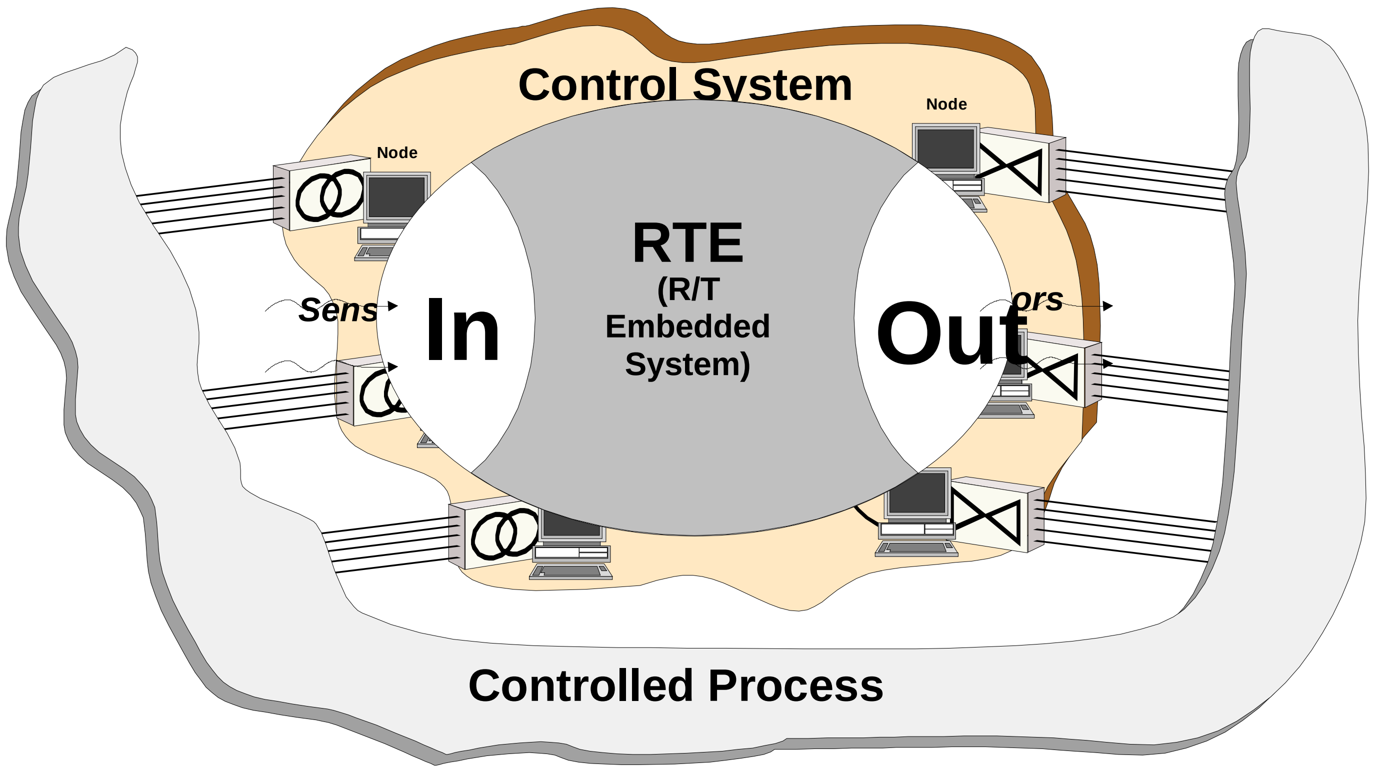
- Compatible with centralized control model

Flying over R/T computing evolution

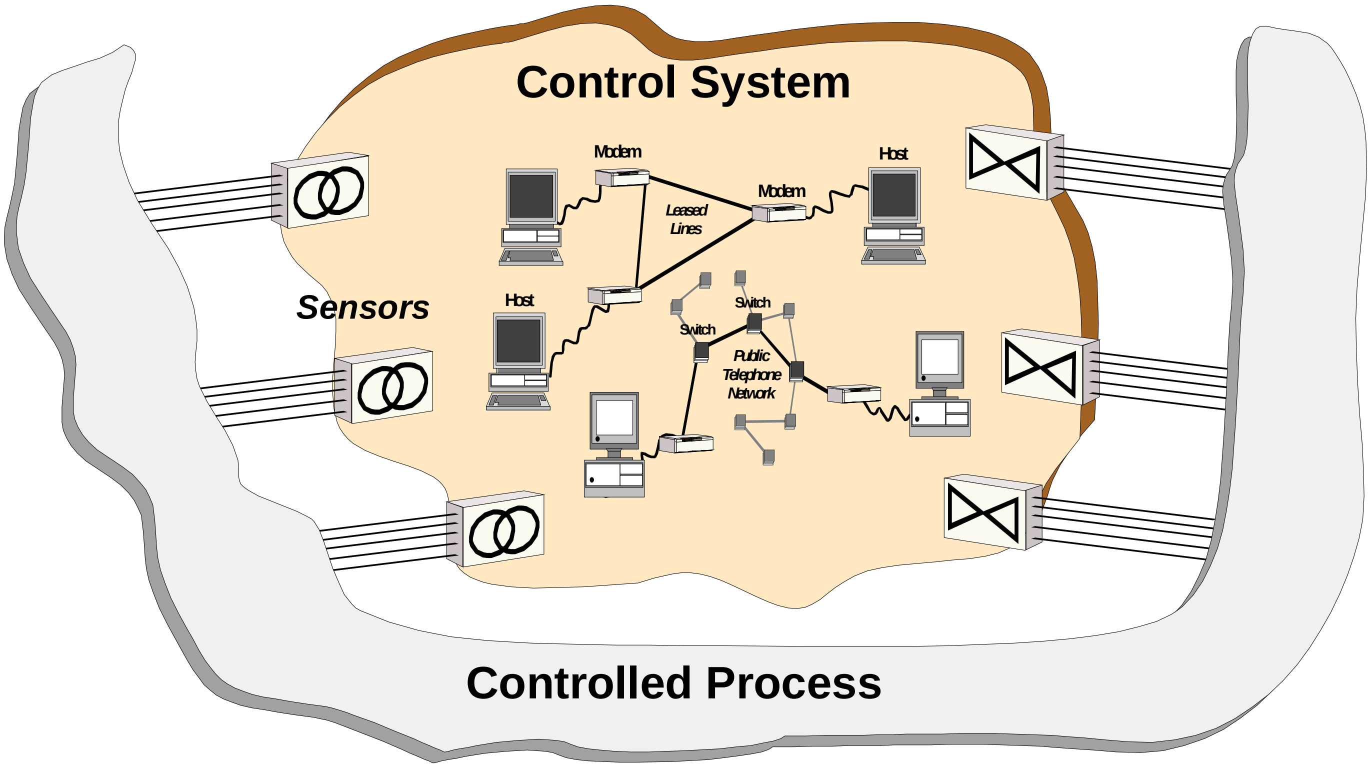


- Digital-Bus centralized control architecture

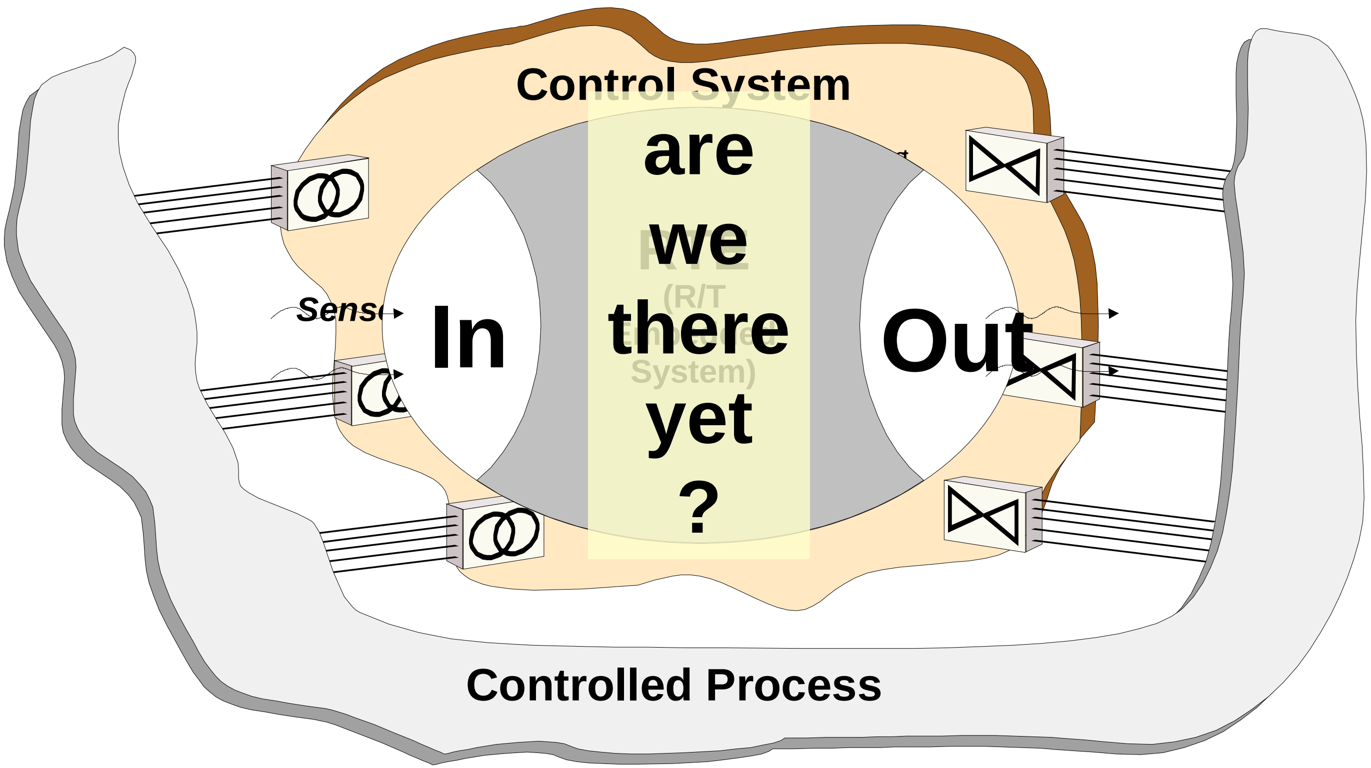
Flying over R/T computing evolution



- Compatible with centralized control model



- Truly distributed control architecture



- Non-compatible with centralized control model
- What is the adequate model?

- Advent of complex systems of embedded systems reveals real nature of interconnected embedded systems:
- DISTRIBUTED (real-time) SYSTEMS
- which must imperatively be studied under their theory, their assumptions, and their possibility and impossibility results
- However, some misconceptions stand in the way...

On Misconceptions

Misconceptions on R/T systems

- The community never got past the sterile TT vs ET debate, and this has been very damaging:
  - It created a ‘shoot-on-sight’ attitude against any distributed systems research that would not smell TT (e.g. CAN-based systems)
  - The word ‘events’ would trigger laser-guided weapons control devices to explode the offender
  - Radicalized ET hardliners would do pretty much the same, in opposite sense: ‘did you say time?’, Bang!
  - ... eliminating the last hopes of a healthy scientific discussion

Misconceptions on R/T systems

- Furthermore, tragic misunderstandings were caused that permeated other sub-communities:
  - Considering TT <=> ‘synchronous digital system’
  - Wrongly Read: a synchronous distributed system is a “big” integrated circuit
  - Wrongly conclude: So let’s continue using our tools for integrated HW systems (e.g. architecting methods, formal spec/verif lang/tools), for any (distributed) R/T system

Misconceptions on R/T systems

- Or:
  - Considering ET <=> ‘asynchronous digital system’
  - Wrongly Read: ET systems are asynchronous distributed systems, and thus fall to the FLP result of impossibility of consensus/atomic broadcast
  - Wrongly conclude: So TT is the *only* way to build F/T R/T systems

Misconceptions on R/T systems

- These struggles obscured the real reason why we should build time-sensitive systems:
  - the environment evolves at its own pace, which we usually observe through an artifact: *real* time
  - our system must sync with the environment
  - Wrongly Read: It's all about time
  - Wrongly conclude: deadlines are the one and only thing that matters

Misconceptions on R/T systems

- Some clarity lacking, distributed theory and algorithms communities abstracted one single thing from this:
  - Time is always an artifact
  - Wrongly Read: thus can be ignored, or afterthought of
  - Wrongly conclude: So let's continue *only* using asynchronous (time-free) models

On the ET vs. TT debate

Motivation

- Over the past years there has been a classical of the conference debates: ET vs TT
- Incidentally, sometimes people did try to analyse the problem objectively
- But for several reasons, the question has been re-amplified again later
- It is worthwhile to try and identify whether "being TT" versus "being ET" is a fundamental question

Is "being TT" or "being ET" a fundamental question?

- If it is not, then people's research on both sides has been obscured by that struggle of schools, and perhaps, despite the fact that good TT and ET systems have been built, *better and more generic systems might have been built*
- In fact, I think *ET vs TT* is not a fundamental question, because I *never could find enough evidence on fundamental issues separating them*, and over the past few years, that evidence kept shrinking

No point in talking about ET vs TT

- What exist are schools of system design (not models) none of them perfect, none of them complete for all applications.
- As early as in [AW93], a few points were identified as being problems common to both schools:
  - information flow control
  - responsiveness
  - predictability and assumption coverage
  - efficiency and versatility
  - extensibility

- **[AW93] – Distributed Systems, 2 Ed., Addison-Wesley, Ch.16-19, Kopetz & Veríssimo**

Examples

- "TT" performs *peripheral* event-to-state transformation (PES), "ET" performs *central* event-to-state transformation (CES)
    - "the door is open" vs. "the door opened“
        - *But couldn't we do CET-TT ? Or PES-ET?*
- "TT" follows a DSM or shared tuple space (STS) computing paradigm, "ET" follows state machine (SM)
    - *But couldn't we do DSM-ET ? Or SM-TT?*

Examples

- Sporadic (ET) systems have high jitter, since they do not have a notion of global time
  - **NO:** There is *clock-driven* and *timer-driven*, and that is independent from ET-TT. Ex. Δ-prots (Cristian) are ET-CD
- ET systems are subject to event showers
  - **NO:** 500 "fire alarm" event messages for the same fire are useless repetitions in an ill-designed ET system

Examples

- ET systems must be infinitely fast since they do not define minimum spacing between events
  - **NO:** because this is a fundamental issue--- so would TT systems need to be infinitely fast, if they were to capture any amount of info: infinite information => infinite BW and MIPS
- TT yields faster error detection, because all messages are expected at a Tm, so an omission is immediately detected at Tm^+
  - **NO:** if there exists a Td<Tm at which we already know of a failure, then ET error notification at Td will be faster: asap

- TT is inherently deterministic because all system progress is paced by the clock
  - **NO:** so are most CPUs, and some are not deterministic at all.
- TT ensures a predictable control system
  - **NO:** it allows a technically predictable solution for a controlled system which is technically modeled or artifacted as being predictable. If the environment (a.k.a. controlled system) is not predictable, we don't know how to reach a correct solution anymore

Beware of zealots

Some reality

- So would this mean that TT is inadequate?
  - Of course not, it has proved to be an excellent abstraction
  - It simply does not solve all the problems in the world
- We should avoid to have TT-hammers or ET-hammers, otherwise all problems look like TT- or ET-nails...

SAE Control challenge

- Solved by Kopetz et al. with TT approach (TTP)
- Solved by Burns et al with ET approach (CAN)

Distributed fault-tolerant control problem

- Solved by Kopetz et al. with TT approach (TTA)
- Solved by Rufino et al with ET/TT approach (CANely)

Can we live without time then ??

- It is costumary to consider the asynchronous time-free model as the baseline for designing resilient algorithms
- Furthermore, considering security, one also assumes arbitrary faults
- This has been the almost exclusive workhorse of algorithmists
- But this has a cost

Taking detours…

- OBJECTIVE:
  - solve most non-timed problems with highest possible coverage

- tone down determinism (e.g., randomisation)
- tone down liveness expectations (e.g., indulgence)
- use weaker semantics (e.g., thresholds, quorums)
- tone down allowed fault severity (e.g., hybrid faults)
- tone down asynchrony (e.g., parsync protocols, FDs)

- OBJECTIVE:
  - solve timed problems with highest possible coverage

- tone down asynchrony (e.g., sync/parsync protocols)

- Maybe the key lies with models and architectures that address time in a way adequate to complex distributed real-time systems scenarios

How sacred are deadlines?

Observation 1

- Any system is described by a set of safety and liveness properties
- Some of the former may be timeliness properties (time as a first class citizen)
- Deadline specs are certainly amongst the latter
- But many others are not, like e.g. control error variables, safety distances, etc.

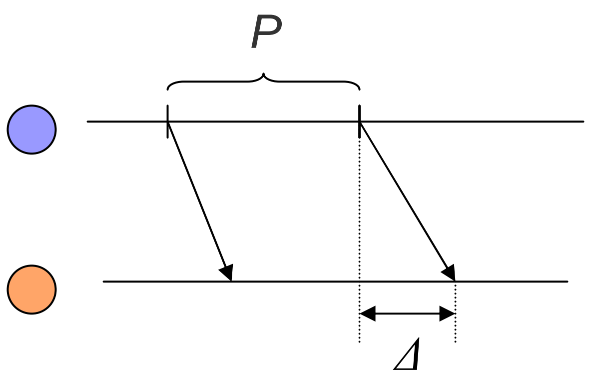
Observation 2

- When deadlines fail, we have a HRT system failure?
- This is the normal philosophy, but not necessarily true
- Think of:
  - what safety properties to preserve
  - missed deadlines as faults (component failures)
  - detected by timing failure detectors or masked
  - i.e., timing fault tolerance

Classical approaches to R/T progr.

Consider a car driving control example: avoiding collision between two cars

- Traditional hard real-time approach is deadline-driven:



- Given target speeds, devise R/T schedule so that corrections made suffic. often.
- Static schedule loaded onto R/T executives
- Periodically, with a deadline of *P* units, cars exchange information and trajectory is corrected
- Missed deadline is a failure in HR/T system

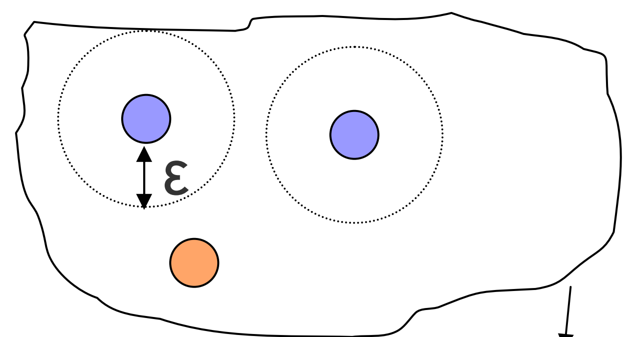
- Consequence:
- The deadline became the goal
- The safety distance became accessory

An alternative approach to R/T progr.

Consider a car driving control example: avoiding collision between two cars

- Our approach:

- SAFETY DISTANCE Property: A car cannot “enter” the dashed circles of other cars, i.e must remain at a distance ε
- Each car must know other cars’ positions with a bounded error
- Distance ε proportional to the error
- Error depends on physics (fixed) and on period and delay of comm’s (variable)
- Allowed speed proportional to ε



Orange car’s view of the environment

- Consequence:
- The safety distance is the goal
- The speed and deadlines are accessory
- They become timed actions, which can have timing errors,
- Errors can be handled by timing fault tolerance

On the new world

Vision: the Future

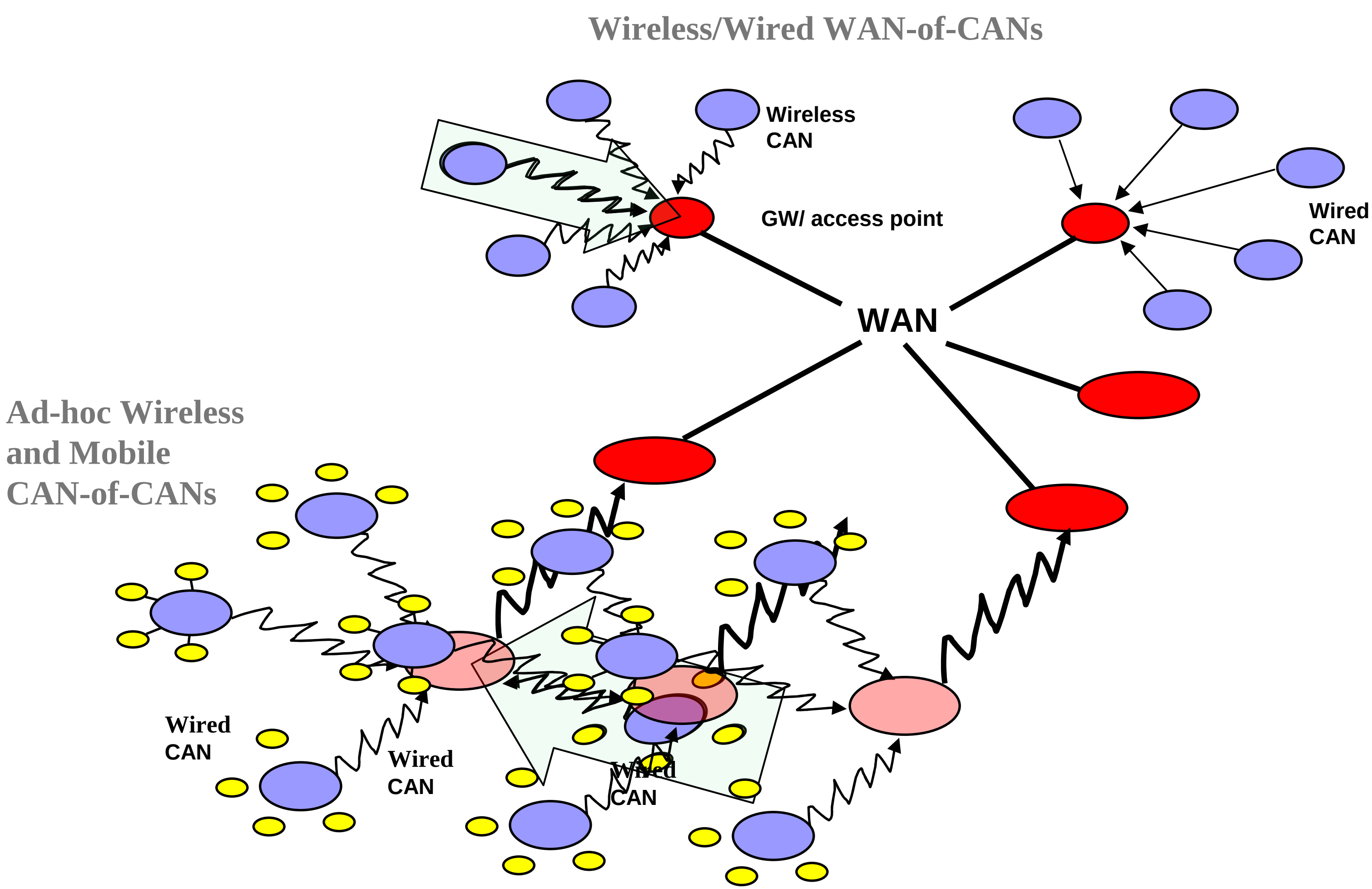
- The future lies with a new generation of systems:

  large-scale, complex and networked

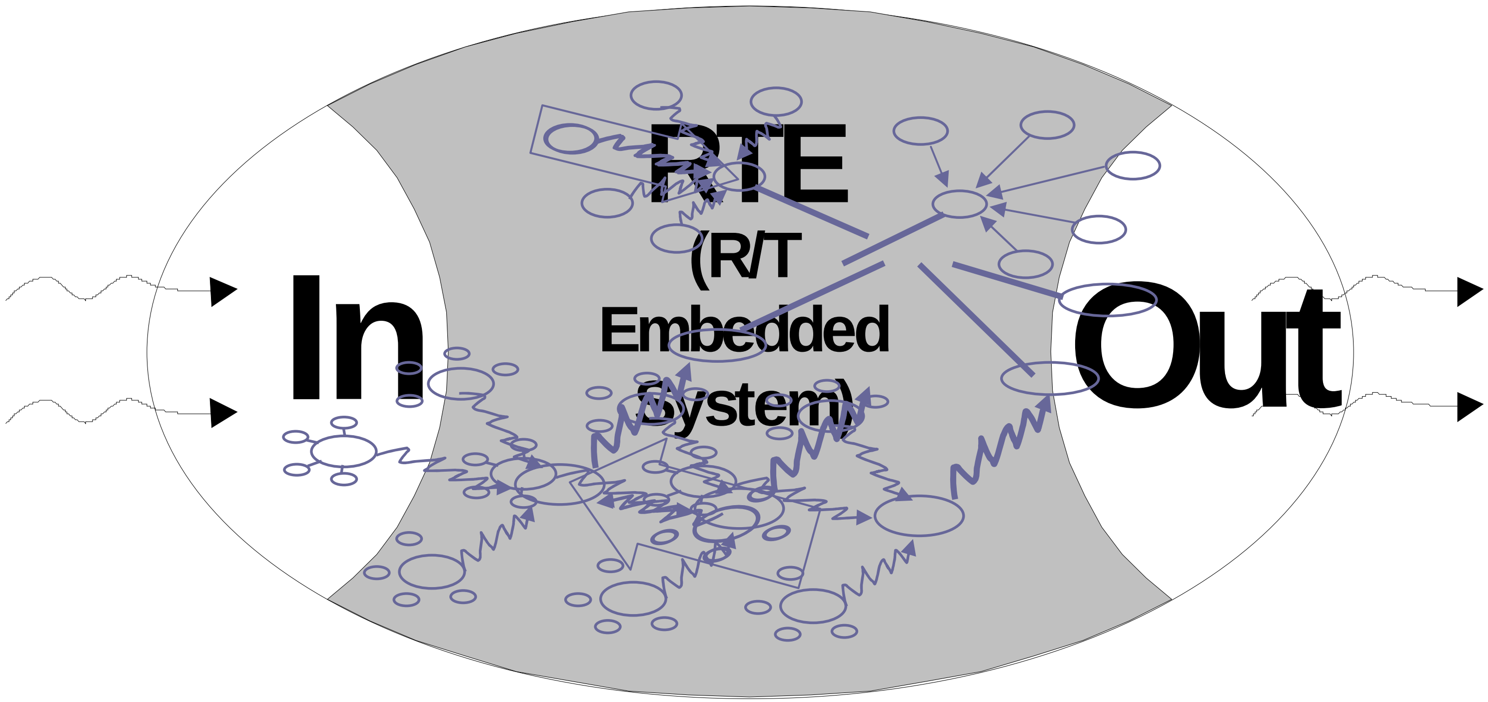
  systems-of-embedded-systems

- This is a grand challenge

Systems of Embedded Systems



Correct and trustworthy design of Systems of Embedded Systems: are we there yet?



The Challenge

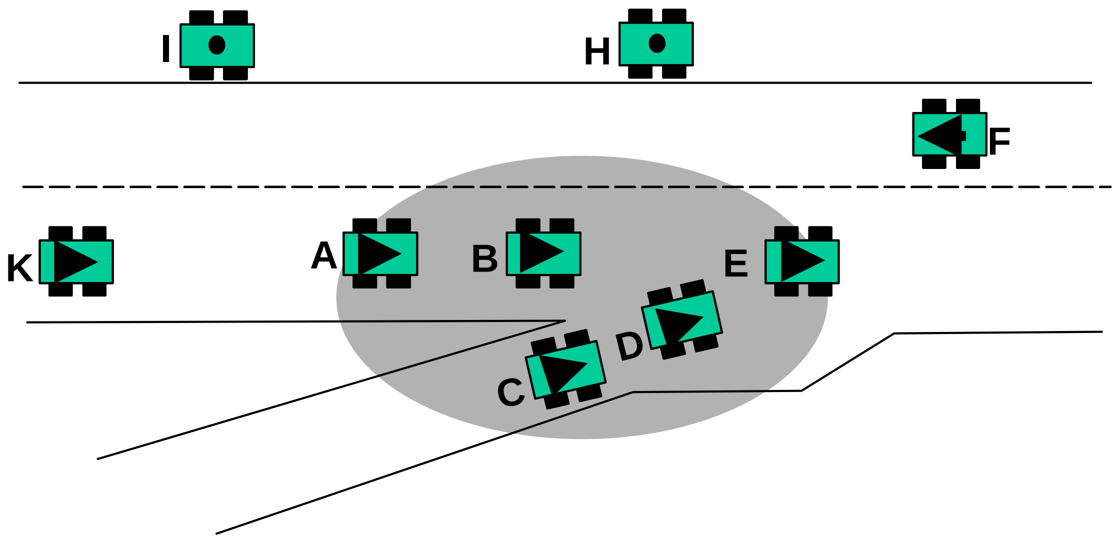
- What is the next challenge in the road ahead, for Embedded Systems research?
- To master complexity, modularity, autonomy, dynamics of configurations, heterogeneity of compositions
- But also
- pervasiveness of devices, ubiquity of computations, lack of perceived global state, unreliability of communication, uncertainty of timeliness (delays), insecurity
- In other words, think about:

Complex R/T systems of embedded components

Complex systems-of-embedded-systems

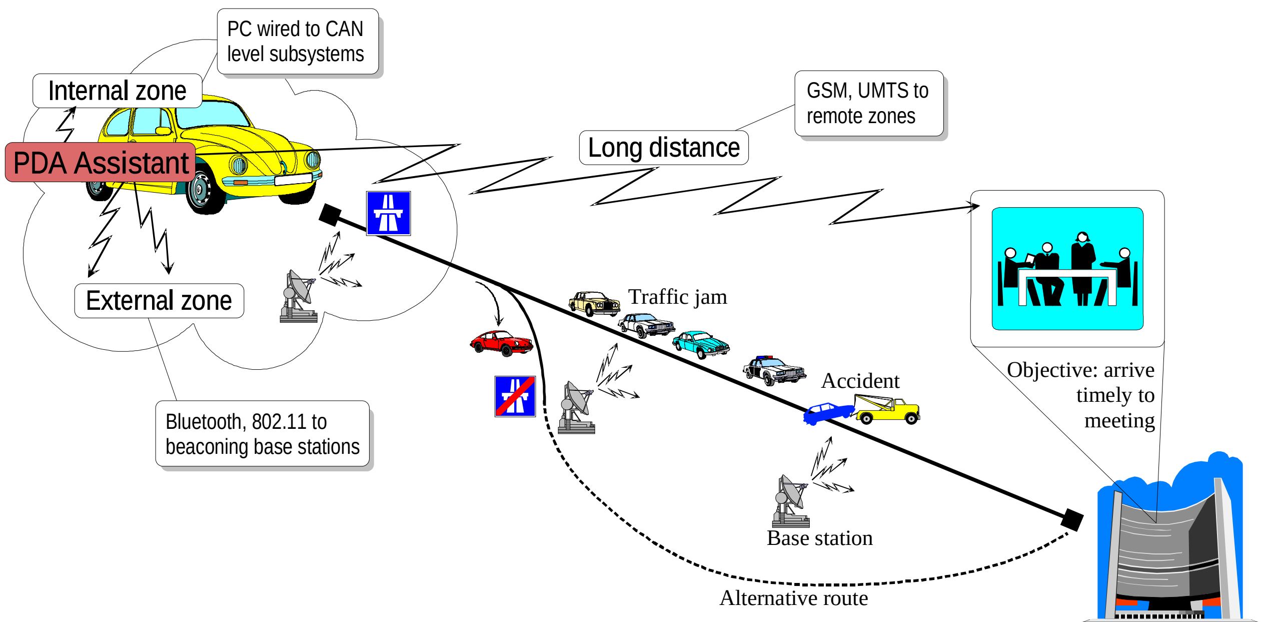
Application scenarios

- Cooperating Cars



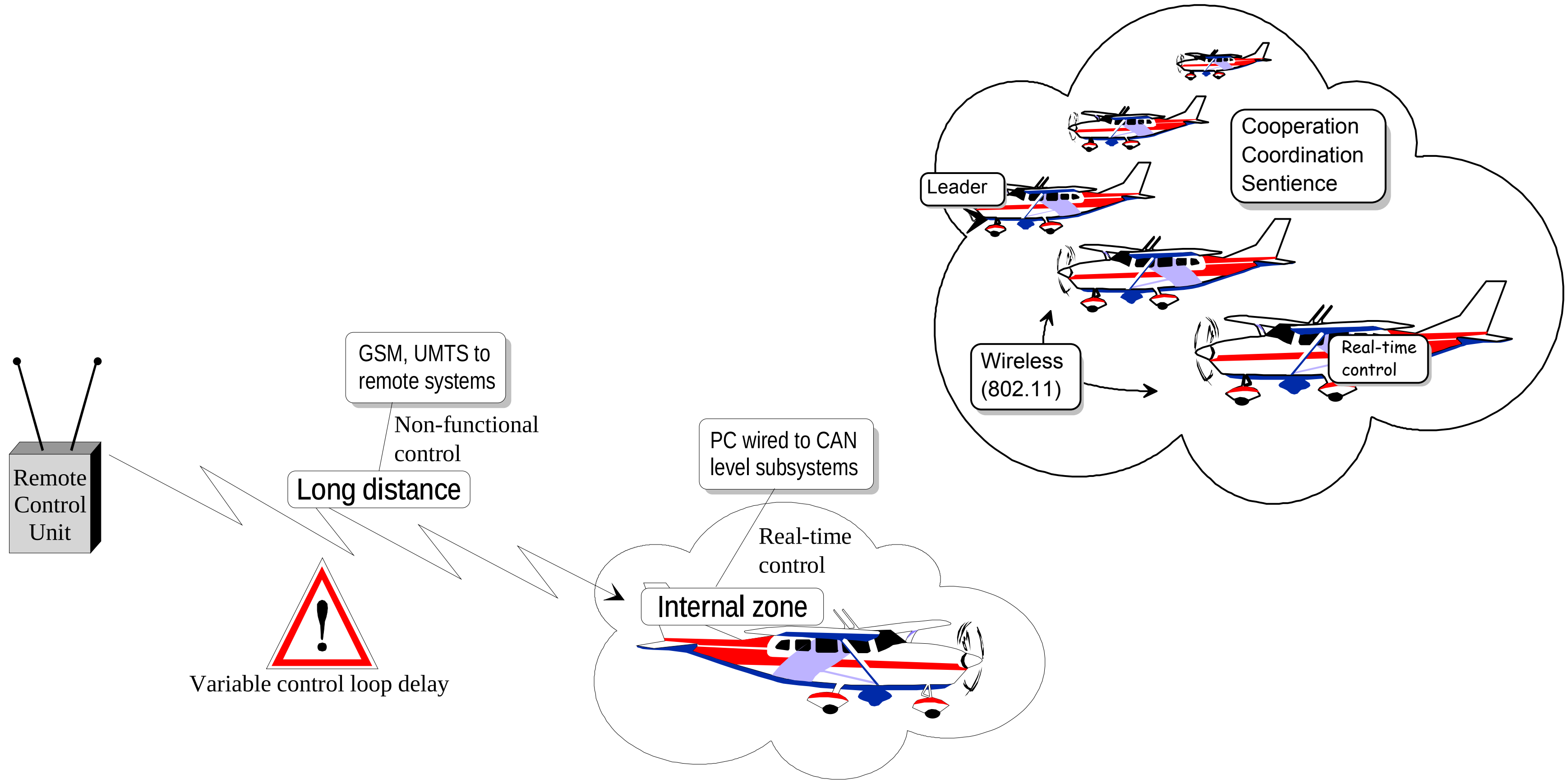
Application scenarios

- Assisted Terrestrial Transportation Systems
- Other wireless/mobile/ambient-intelligent appls



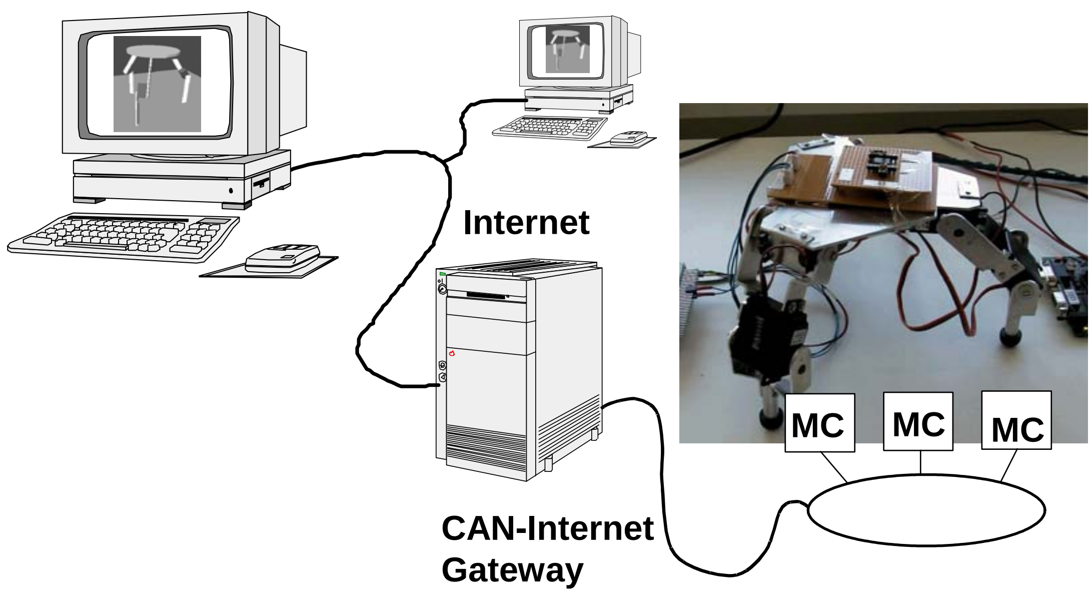
Application scenarios

- Autonomous or Remote control of real-time operations (e.g. free-flight, satellite constellations)



Application scenarios

- Remote control of a grabber robot
- Autonomous teams of robots or enhanced humans
- Other wireless mobile gadget based control or ubiq. comp. appls



The hackers are coming

Threats

- Back to the digital system metaphor
- Assuming we can design a complex networked RTE as a huge hardware digital system:
  - time-triggered, global state, synchronous execution, "happened at once“, quasi-stationary approximations
  - these are simplifying assumptions that render the problem more tractable (simplify it)
  - all these normally hold inside single or small systems, not so much in large systems of embedded systems
- We can't, and we better get ourselves convinced of it very quickly

Threats

- But if we can't, how and why does it work?
  - deterministic proofs of correctness are based on assumptions
  - system assumptions have a certain coverage
  - in mature technologies, coverage for accidental failures tends to be high, even if assumptions pushed to limit: good mastering of failure stochastics
- However... Hackers don’t like stochastics:
  - They will attack the system by its weakest link: the assumptions (time, clock, phase, etc.)

Cyber Security for embedded control systems: how much time do we have?

- It is common knowledge among Sec&Dep people that :
  - Assumptions are vulnerabilities that are attacked by hackers in ways much more severe than accidental faults would
  - The less coverage an assumption has, the more fragile to attack it is
- It is a matter of time until hackers understand how to attack control systems underlying critical infrastructures, cars or trains
- Maybe all it takes is a **www.scada_rootkit.com**

The road to embedded systems security (1)

- Securing individual components (e.g. chips, PLCs, industrial PCs) is important, but does not solve the problem:
  - Cannot assert the security of the overarching system
  - There are many legacy devices
  - Classical security techniques hamper R/T operation
- So:
  - We will not deploy really secure RTE components in a near future
  - Maybe we will never be able to deploy completely secure RTE components (e.g. vulnerability-free)

The road to embedded systems security (2)

- What we want is to deploy **secure-enough RTE systems**
- How? We must learn how to use:
  - Mostly insecure components (untrusted comp's)
  - Some secure components (trusted-trustworthy comp's)
  - Modular interconnection techniques/architectures
  - trustworthy and resilient glue algorithms
- So that the whole is better than the sum of the parts:

TRUSTWORTHY EMBEDDED SYSTEMS (-of-SYSTEMS) OUT OF NON- TRUSTWORTHY EMBEDDED COMPONENTS

Epilogue

- Advent of complex systems of embedded systems reveals real nature of interconnected embedded systems:
- DISTRIBUTED (control) SYSTEMS
- Subject to:
  - accidental and malicious faults
  - Uncertainty of the environment
  - Uncoverage of assumptions
- which must imperatively be studied under both theories, their assumptions, and their possibility and impossibility results

In Conclusion

Past

- Historically, Real-Time Computing has realized great breakthroughs
- In scientific terms, fundamental results have been published in scheduling, communication, architecture, etc.
- In industrial terms there have been major achievements:
  - R/T kernels and executives
  - fly-by-wire, drive-by-wire
  - Formal spec/verif on non-distributed timed systems
  - …

Present

- But if you ask me about the current slope/momentum...☹
- Current reality is about
  - **distributed, dependable, secure, real-time**
- Are these bodies of knowledge recognised within the real-time community is their own?
- Are:
  - clock synchronisation and time services, distributed R/T protocols, R/T agreement, R/T causal ordering, R/T replication management, temporal consistency, timed consensus, R/T databases, fault-tolerant fieldbuses
- considered core R/T subjects?

... the Future

- In the meantime, life goes on, into new, better things, like
  - Ambient Intelligence and Pervasive Computing, Complex Systems-of-Embedded-Systems, Global Critical Information Infrastructures, etc.
- Obviously, this increases the slope at which the need for DistDepSecRT computing raises, and ... brings new challenges, such as:
  - dependable adaptability
  - reconciling uncertainty with predictability
  - dynamics and evolvability
- If we do not increase the slope at which we create knowledge in DistDepSecRT computing...
- ... we are going to have a problem...
- ... Lots of problems!

Some Recent Publications (w/ urls)

- *On Wormholes and Dependable Adaptation*
- Travelling Through Wormholes: a new look at Distributed Systems Models. **P. Veríssimo**, SIGACTN: SIGACT News, ACM Special Interest Group on Automata and Computability Theory, 37(1), MArch 2006.
- Uncertainty and Predictability: Can they be reconciled? **Paulo Veríssimo**, Future Directions in Distributed Computing, pp. 108-113, Springer Verlag LNCS 2584, May, 2003
- *The Timely Computing Base: Timely Actions in the Presence of Uncertain Timeliness*. **Paulo Veríssimo, António Casimiro, C. Fetzer. I**n Proceedings of the 1st International Conference on Dependable Systems and Networks, New York, USA, June 2000.
- The Timely Computing Base Model and Architecture. Paulo Veríssimo, António Casimiro. IEEE Transactions on Computers - Special Issue on Asynchronous Real-Time Systems, vol. 51, n. 8, Aug 2002
- The Timely Computing Base. Paulo Veríssimo and António Casimiro. **Technical Report DI/FCUL TR 99-2, Department of Informatics, University of Lisboa,** May 1999. *(original paper, improved in TOCS02)*

- *Implementing Wormholes*
- *Measuring Distributed Durations with Stable Errors*. **António Casimiro, Pedro Martins, Paulo Veríssimo, Luís Rodrigues.** Proceedings of the 22nd IEEE Real-Time Systs Symposium, London, UK, December 2001
- *How to Build a Timely Computing Base using Real-Time Linux*. **António Casimiro, Pedro Martins, Paulo Veríssimo.** in Proceedings of the 2000 IEEE International Workshop on Factory Communication Systems, Porto, Portugal, September 2000.
- *Timing Failure Detection with a Timely Computing Base*. **António Casimiro, Paulo Veríssimo.** 3rd Europ. Research Seminar on Advances in Distr. Sys (ERSADS'99), Madeira Island, Portugal, April 23-28, 1999
- *The Design of a COTS Real-Time Distributed Security Kernel*, ***Miguel Correia, Paulo Veríssimo, Nuno Ferreira Neves***, *Fourth European Dep. Comp. Conf., Toulouse, France, October 2002 © Springer-Verlag.*

A New Programming Model for Dependable Adaptive Real-Time Applications

- MAIN FEATURE of May 2005 issue of IEEE Distributed Systems On-Line Journal:
  - http://dsonline.computer.org
  - http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso_level1&path=dsonline/0505&file=o5001.xml&xsl=article.xsl&
- *A New Programming Model for Dependable Adaptive Real-Time Applications*
  Pedro Martins, Paulo Sousa, António Casimiro, Paulo Veríssimo
  IEEE Distributed Systems Online, vol. 6, no. 5, 2005.

you may also get there from our web site,
www.navigators.di.fc.ul.pt under "Recent Documents“.

Some Recent Publications (w/ urls)

- *Using Wormholes*
- *Using the Timely Computing Base for Dependable QoS Adaptation*. **António Casimiro, Paulo Veríssimo.** Proceedings of the 20th IEEE Symp. on Reliable Distributed Systems, New Orleans, USA, October 2001
- *Generic Timing Fault Tolerance using a Timely Computing Base*. **António Casimiro, Paulo Veríssimo.** Procs of the Intern'l Conference on Dependable Systems and Networks, Washington D.C., USA, June 2002
- *Efficient Byzantine-Resilient Reliable Multicast on a Hybrid Failure Model*, ***Miguel Correia, Lau Cheuk Lung, Nuno Ferreira Neves, Paulo Veríssimo.*** *Proc's of the 21st Symp. on Reliable Distributed Systems (SRDS'2002), Suita, Japan, October 2002*
- *How to Tolerate Half Less One Byzantine Nodes in Practical Distributed Systems*
  Miguel Correia, Nuno Ferreira Neves, Paulo Veríssimo
  In Proceedings of the 23rd IEEE Symposium on Reliable Distributed Systems. Florianopolis, Brasil, pages 174-183, October 2004
- *Low Complexity Byzantine-Resilient Consensus*
  Miguel Correia, Nuno Ferreira Neves, Paulo Veríssimo, Lau Cheuk Lung
  Distributed Computing, 2005.
- *Solving Vector Consensus with a Wormhole*
  Nuno Ferreira Neves, Miguel Correia, Paulo Veríssimo
  Transactions on Parallel and Distributed Systems,2005

- Paulo Veríssimo:
  - http://www.di.fc.ul.pt/~pjv
- Navigators group:
  - http://www.navigators.di.fc.ul.pt/

Thank you !