Trends in Cyber Physical Systems: A Historical Perspective

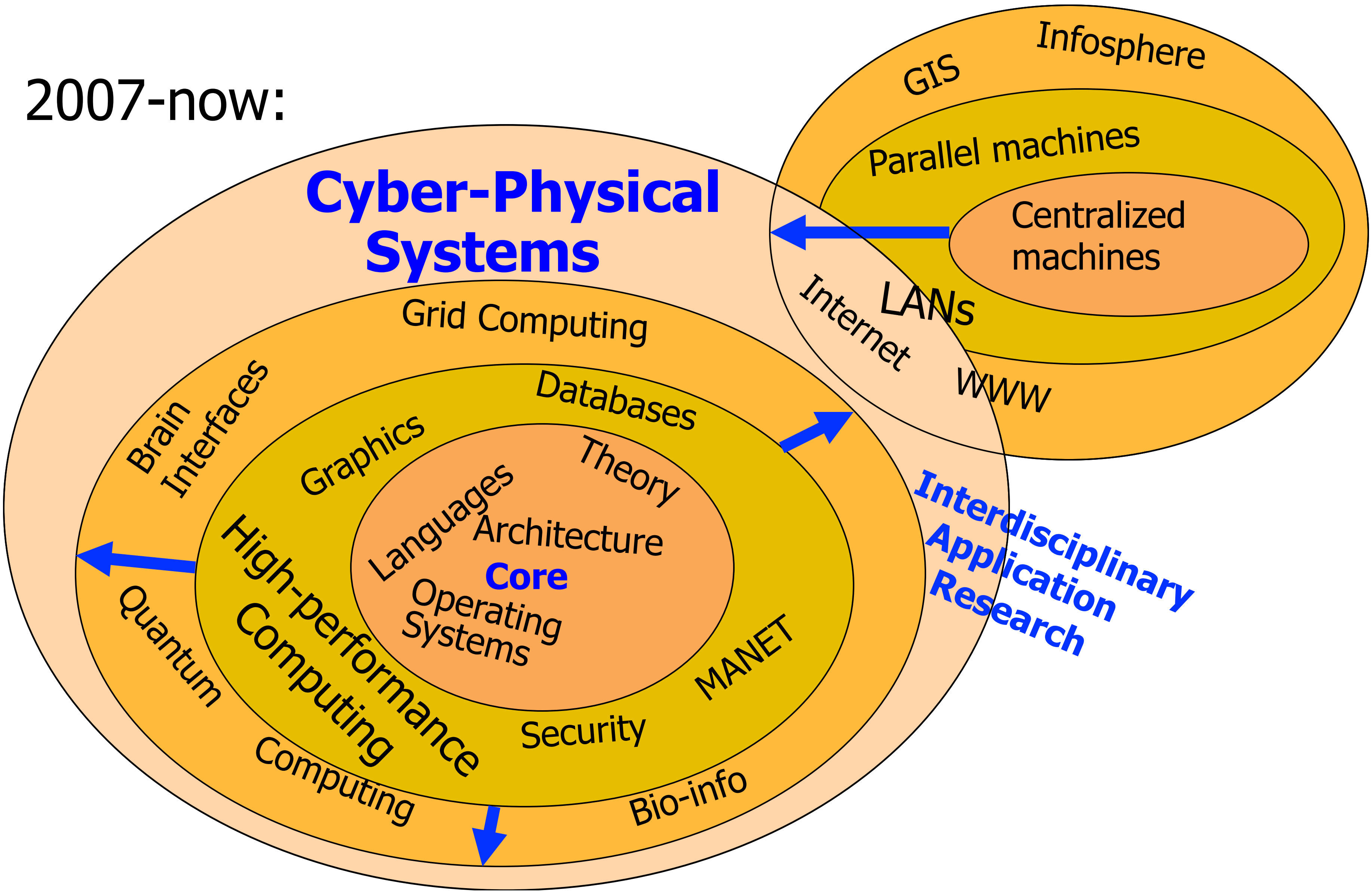
Systems that Interact with the Physical World

CSE 40437/60437-Spring 2015

Prof. Dong Wang

Last Lecture

2007-now:



This Lecture

- History and Origin of CPS
- Two Fundamental Challenges of Traditional CPS
- Real-world Examples

History: The Beginnings

- NSF Workshop on Cyber-Physical Systems, October 16-17, 2006, Austin, TX.
- National Meeting on Beyond SCADA: **Networked Embedded Control** for Cyber Physical Systems, November 8-9, 2006, Pittsburgh, PA.
- National Workshop on **High-Confidence Software** Platforms for Cyber-Physical Systems (HCSP-CPS), November 30 - December 1, 2006, Alexandria, VA.
- NSF Industry Round-Table on Cyber-Physical Systems, May 17, 2007, Arlington, VA.
- Joint Workshop On High-Confidence **Medical Devices**, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability, June 25-27, 2007, Boston, MA.
- National Workshop on **Composable Systems** Technologies for High-Confidence Cyber-Physical Systems, July 9-10, 2007, Arlington, VA.
- National Workshop on High-Confidence **Automotive** Cyber-Physical Systems, April 3-4, 2008, Troy, MI.
- CPSWeek, April 21-24, 2008, St. Louis, MO.
- CPS Summit, April 25, 2008, St. Louis, MO: NSF Announces new CPS Initiative
- The First International Workshop on Cyber-Physical Systems, International Conference on Distributed Computing Systems (ICDCS), June 20, 2008, Beijing, CHINA.
- Workshop on CPS Applications in Smart **Power** Systems, Raleigh, NC, 2011

History: The Beginnings

- NSF Workshop on Cyber-Physical Systems, October 16-17, 2006, Austin, TX.
- National Meeting on Beyond SCADA: **Networked Embedded Control** for Cyber Physical Systems, November 8-9, 2006, Pittsburgh, PA.
- National Workshop on **High-Confidence Software** Platforms for Cyber-Physical Systems (HCSP-CPS), November 30 - December 1, 2006, Alexandria, VA.
- NSF Industry Round-Table on Cyber-Physical Systems, May 17, 2007, Arlington, VA.
- Joint Workshop On High-Confidence **Medical Devices**, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability, June 25-27, 2007, Boston, MA.
- National Workshop on **Composable Systems** Technologies for High-Confidence Cyber-Physical Systems, July 9-10, 2007, Arlington, VA.
- National Workshop on High-Confidence **Automotive** Cyber-Physical Systems, April 3-4, 2008, Troy, MI.
- CPSWeek, April 21-24, 2008, St. Louis, MO.
- CPS Summit, April 25, 2008, St. Louis, MO: NSF Announces new CPS Initiative
- The First International Workshop on Cyber-Physical Systems, International Conference on Distributed Computing Systems (ICDCS), June 20, 2008, Beijing, CHINA.
- Workshop on CPS Applications in Smart **Power** Systems, Raleigh, NC, 2011

Technologies

History: The Beginnings

- NSF Workshop on Cyber-Physical Systems, October 16-17, 2006, Austin, TX.
- National Meeting on Beyond SCADA: **Networked Embedded Control** for Cyber Physical Systems, November 8-9, 2006, Pittsburgh, PA.
- National Workshop on **High-Confidence Software** Platforms for Cyber-Physical Systems (HCSP-CPS), November 30 - December 1, 2006, Alexandria, VA.
- NSF Industry Round-Table on Cyber-Physical Systems, May 17, 2007, Arlington, VA.
- Joint Workshop On High-Confidence **Medical Devices**, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability, June 25-27, 2007, Boston, MA.
- National Workshop on **Composable Systems** Technologies for High-Confidence Cyber-Physical Systems, July 9-10, 2007, Arlington, VA.
- National Workshop on High-Confidence **Automotive** Cyber-Physical Systems, April 3-4, 2008, Troy, MI.
- CPSWeek, April 21-24, 2008, St. Louis, MO.
- CPS Summit, April 25, 2008, St. Louis, MO: NSF Announces new CPS Initiative
- The First International Workshop on Cyber-Physical Systems, International Conference on Distributed Computing Systems (ICDCS), June 20, 2008, Beijing, CHINA.
- Workshop on CPS Applications in Smart **Power** Systems, Raleigh, NC, 2011

Applications

Original Focus: Mission-critical Systems

Building Timely, Predictable, Reliable Systems

Two Classical Challenges

- ***Establish Functional Correctness:*** How to build functionally correct systems from possibly flawed components?
- ***Establish Temporal Correctness:*** What are the analytic foundation for robust timing guarantees in highly dynamic, time-critical software systems?

Two Classical Challenges

- ***Establish Functional Correctness:*** How to build functionally correct systems from possibly flawed components?
- ***Establish Temporal Correctness:*** What are the analytic foundation for robust timing guarantees in highly dynamic, time-critical software systems?

Rate of Innovation and Development Time Issues

- Near the turn of the 20th century products had a 20-30 year life-span before new "versions" were developed
- At present, a product is obsolete in 2-3 years
  - No time to discover and "debug" all possible problems
  - New problems introduced in new versions
  - Component reuse generates additional problems

Software: Increasingly the Primary Cause of System Failure

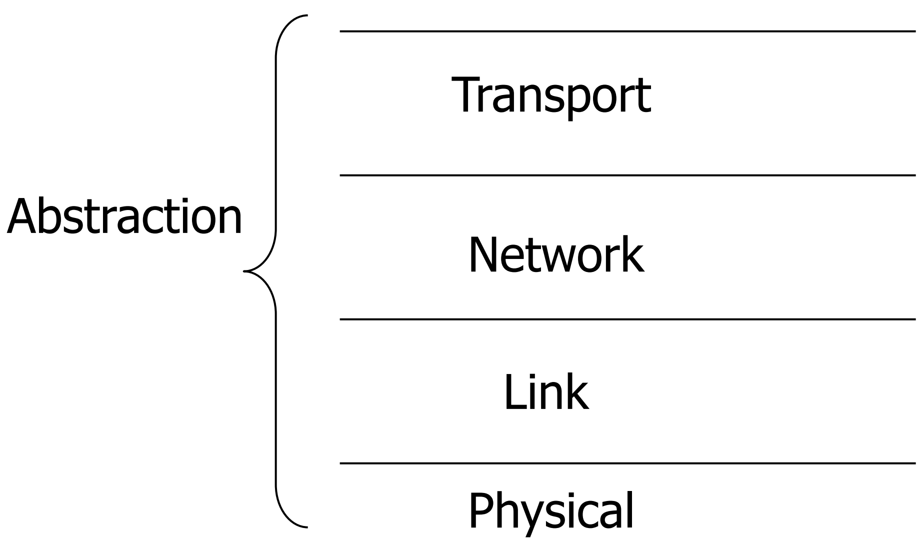
- Arbitrary component interactions unconstrained by physical laws of nature (algorithms can do anything)
- Fast error propagation (at computing device speed)
- Software that interacts with the physical world is buggy!

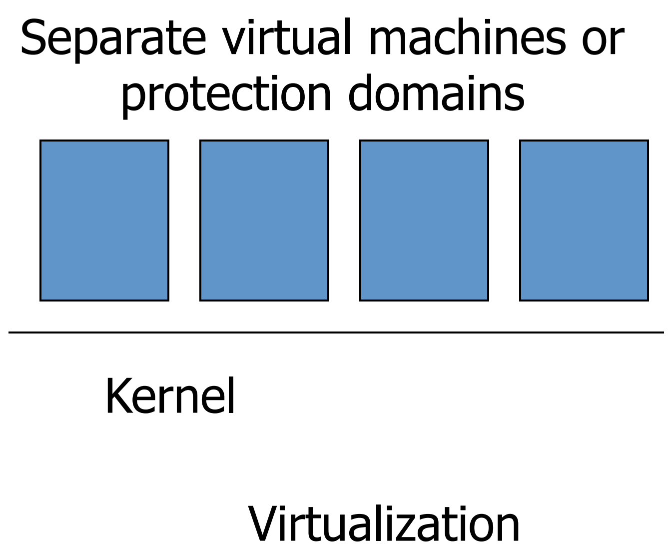
Typical Isolation Techniques

- Abstraction
- Separation of concerns

Typical Isolation Techniques

- Abstraction
- Separation of concerns





Abstraction → Solution?

- Complexity
  - → More levels of abstraction
    - → Narrower specialization
      - → More details are "abstracted away"
        - → Myopic view. Less knowledge of possible adverse interactions
          - → More potential for interaction or incompatibility errors

The Curse of Component Re-use

The Ariane 5 Explosion

- On June 4, 1996, the maiden flight of the European Ariane 5 launcher crashed about 40 seconds after takeoff (0.5 Billion Dollars)

The Curse of Component Re-use

The Ariane 5 Explosion

- On June 4, 1996, the maiden flight of the European Ariane 5 launcher crashed about 40 seconds after takeoff (0.5 Billion Dollars)
- Cause of problem?
  - An inertial reference software component.
    - Not needed during flight. Should be stopped before takeoff but is allowed to operate for up to 50 additional seconds to avoid expensive restarts should countdown be interrupted
    - Component was designed for Ariane 4. Ariane 5 was a faster system. Velocity variable overflowed.
    - Overflow causes an exception that is not caught and crashes the software

Example 1: Interactive Complexity in Distributed Protocols

- Interactive complexity means:
  - Simple individually insignificant failures interact to compound into system failures, or even…
  - Sets of correctly operating components interact to produce a system failure
    - Example:
      - Shortest hop routing
      - Adaptive rate control

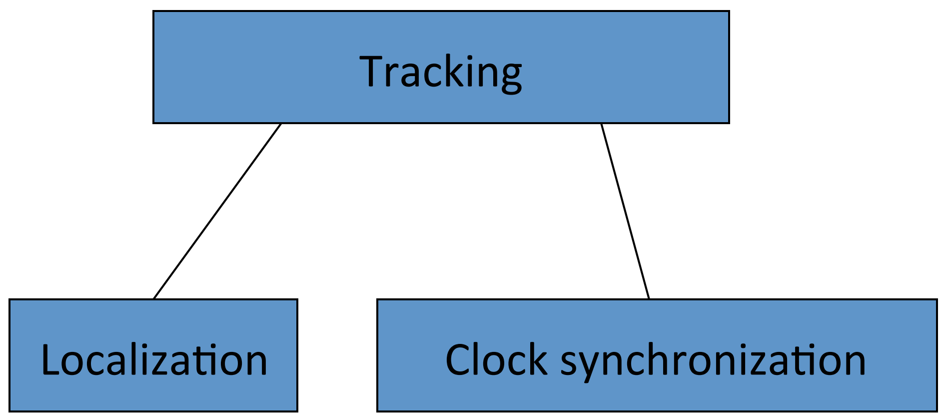
Example 1:

- Shortest hop routing
  - Find shorter path (fewer hops that are longer)
- Long wireless hops → poor channel quality
- Adaptive rate control
  - Reduce transmission rate to improve quality
- Reduced transmission rate

  → Has longer transmission range

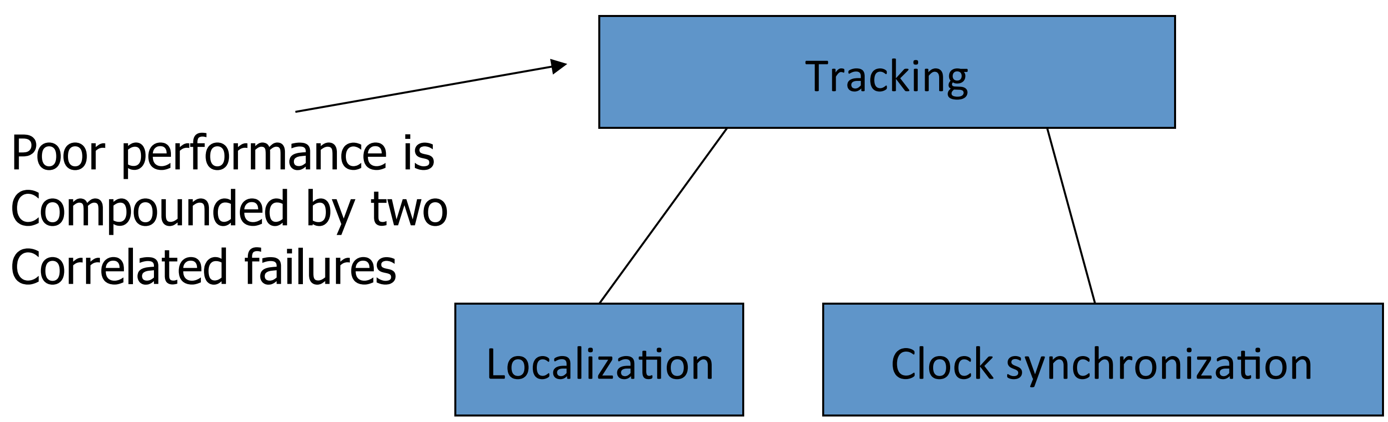
Example 2:
Correlated failure modes between "independent components"

- Localization (determining a node's location) fails in a correlated manner with failure to synchronize clocks. Why?
  - Note: None of the two components uses the other



Example 2:
Correlated failure modes between "independent components"

- Localization (determining a node's location) fails in a correlated manner with failure to synchronize clocks. Why?
  - Note: None of the two components uses the other
- Answer: communication problems. Both subsystems rely on distributed protocols



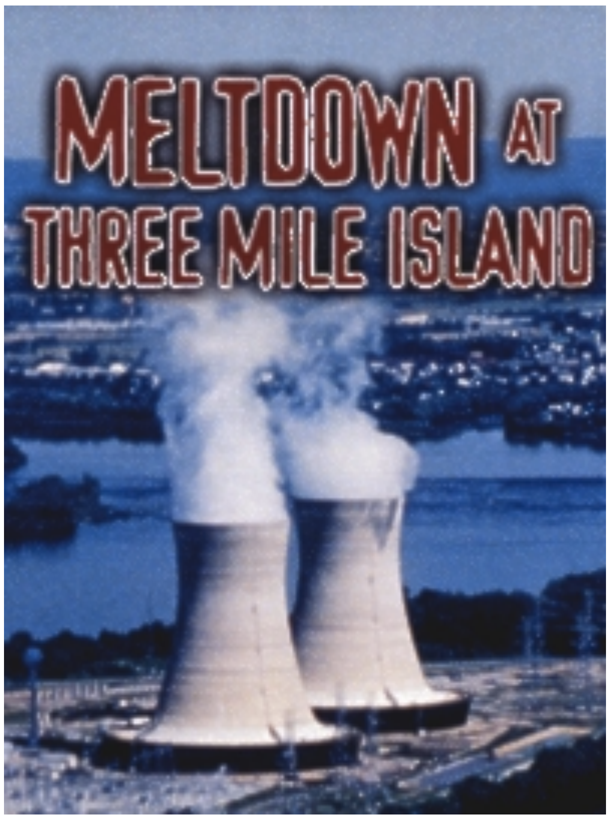
Example 3:
More on hidden interactions

- Magnetic tracking system operates perfectly in calm weather but fails under strong wind conditions. Why?
  - Wind should not change magnetic sensor reading

Example 3:
More on hidden interactions

- Magnetic tracking system operates perfectly in calm weather but fails under strong wind conditions. Why?
  - Wind should not change magnetic sensor reading
- Explanation
  - Wind caused node antenna to vibrate
  - Moving (metal) antenna caused a lot of noise on the magnetic sensor
  - Noise filter adapted noise threshold to remove background noise (and in this case the signal too)

Example 4: Three Mile Island Nuclear Reactor Failure



March 28, 1979

Example 4: Three Mile Island Nuclear Reactor Failure

Core temperature and pressure continue to build up

Coolant pressure relief valve opens to reduce pressure

Core overheating triggers emergency shutdown

Pressure drops. **Valve is stuck open**. Coolant boils off. Core temperature rises. Reaction resumes.

Valve failure indicator light turns on but is **occluded by repair tag** on another device

Core is flooded with water

Failure to open valves

Water at very high temperature oxidizes metal fuel rod coating (rusting)

Open emergency feed-water pumps from emergency tank to cool coolant

Heat exchange stops between primary and secondary cooling Systems. Primary overheats.

Hydrogen is released and eventually leads to explosion

Stop secondary coolant flow and turbine

False alarm of minor secondary system coolant leakage through seal

Ensuring Software Correctness

- The physical world has no “reset” button
  - When failures occur, they can be costly!
- Must reduce:
  - Interactive complexity
    - Unexpected interactions between seemingly correct components
  - Coupling
    - Fast propagation of effects of failure to other system components

Designing Complex Systems
(Example: Air-traffic control)

- Reduce interactive complexity
  - Air traffic is restricted to non-intersecting "corridors" that separate flight paths in the sky
- Reduce coupling
  - Separate aircraft by a substantial distance to reduce cascaded failure effects (think: multiple-car pile-ups in freeway accidents)

Interaction Examples

- Function calls
- Resource sharing
  - One module crashes → overwrites memory of another → second "unrelated" module crashes (analogy to physical proximity and correlated damage)
  - One module is overloaded → another starves
- Timing and synchronization constraints
  - Precedence constraints (one module must execute before another)
  - Exclusion constraints (cannot operate at the same time)
- Assumptions
  - I thought you submitted our paper?
  - No, I thought you did?

Question: How to Build Reliable Software?

- Common approaches:
  - Tracing, source level debugging
  - Simulation/emulation
  - Log and replay

Candidate Approach: Formal Methods

- Express safety properties (e.g., task A will never miss its deadline)
- Prove that safety properties hold
  - If proof fails, counter example is presented (a sequence of events that leads to failure)
- Problem:
  - Proofs require axioms. Axioms may make incorrect assumptions (e.g., circular sensing range)
  - Interactions must be explicitly modeled. Failure to model interactions (e.g., between wind and magnetic sensor) may overlook some failure modes.

Living with Buggy Systems

- If errors cannot be avoided (even using formal methods), we must design systems to tolerate them
  - Architectures for “living with bugs”
  - Fast diagnosis and recovery
  - Issues
    - Problem must be observable (or else cannot diagnose)
    - Observation must be in time so that recovery is possible (observing that you forgot your parachute *after* you jump will not help you)

Simplicity to Conquer Complexity

Prof. Lui Sha UIUC

- Elements of a good design
  - Simple safety core
  - Complex enhanced mission functionality
  - Formal proof of core correctness
  - Well formed dependency (core may use but will not depend on any other components)

Sha, Lui. "Using simplicity to control complexity." IEEE Software 18.4 (2001): 20-28.

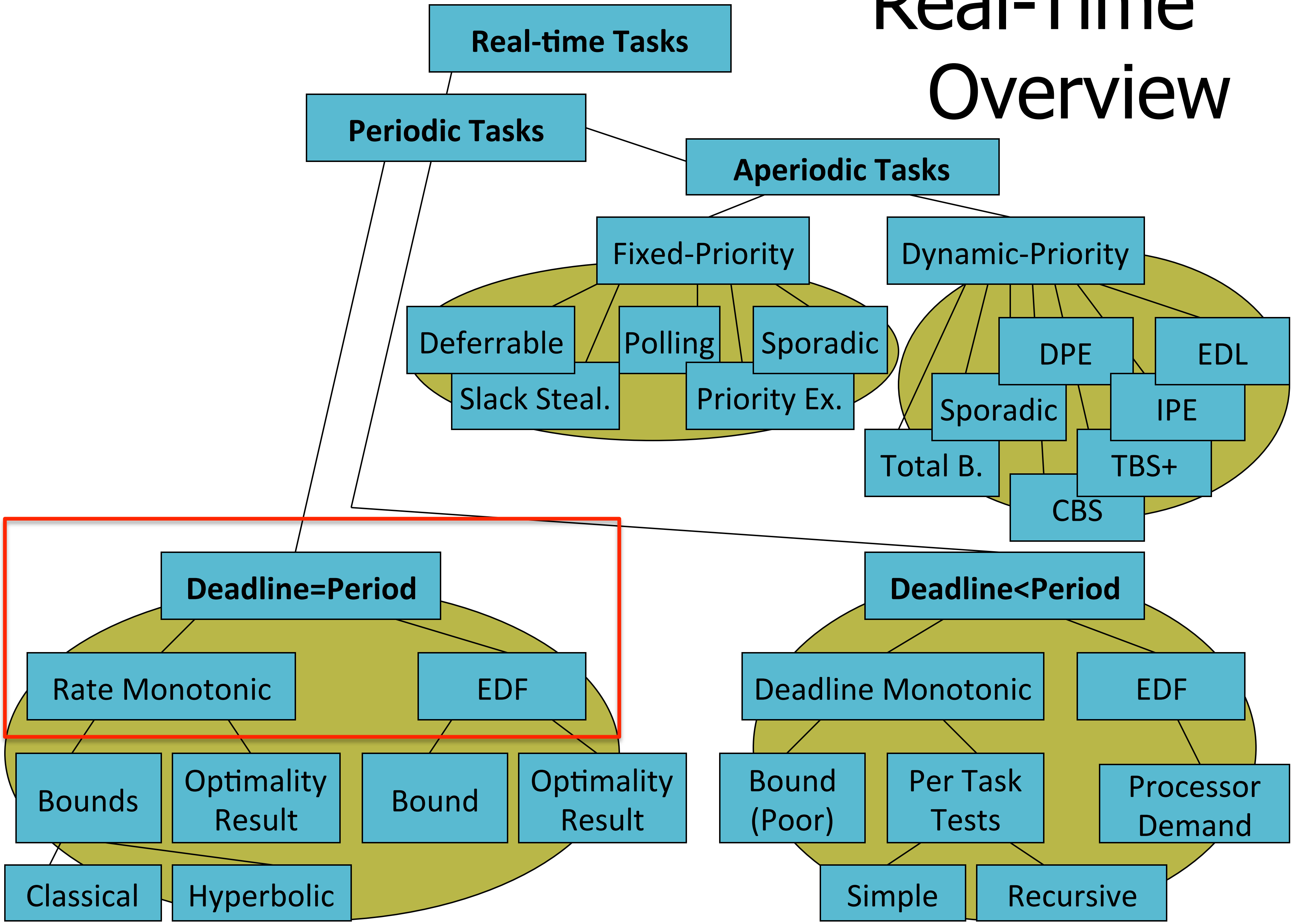
Diagnosis:
A Development-Time Data Mining Example

- Run system multiple times
- Log all observable interactions (messages exchanged, resources allocated, etc)
- Label execution as "correct" (no observable problems) or "incorrect" (problems observed)
- Separate logs into "good" data set and "bad" data set
- Look for sequences of events in the "good" pile but not the "bad" pile and vice versa

Two Classical Challenges

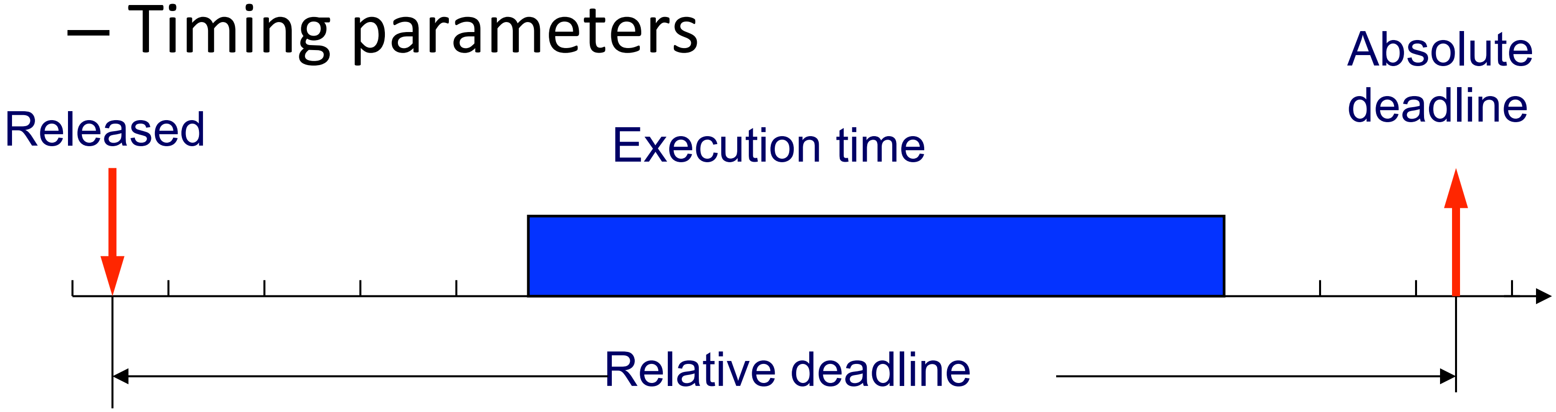
- ***Establish Functional Correctness:*** How to build functionally correct systems from possibly flawed components?
- ***Establish Temporal Correctness:*** What are the analytic foundation for robust timing guarantees in highly dynamic, time-critical software systems?

Real-Time Overview



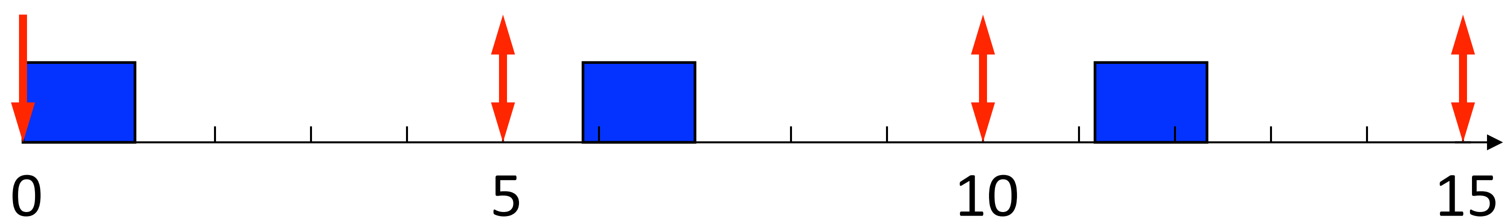
Real-Time Workload

- Job (unit of work)
  - a computation, a file read, a message transmission, etc
- Attributes
  - Resources required to make progress
  - Timing parameters



Real-Time Task

- Task : a sequence of similar jobs
  - Periodic task (*p,e*)
    - Its jobs repeat regularly
    - Period *p* = inter-release time (0 < *p*)
    - Execution time *e* = maximum execution time (0 < *e* < *p*)
    - Utilization U = e/p



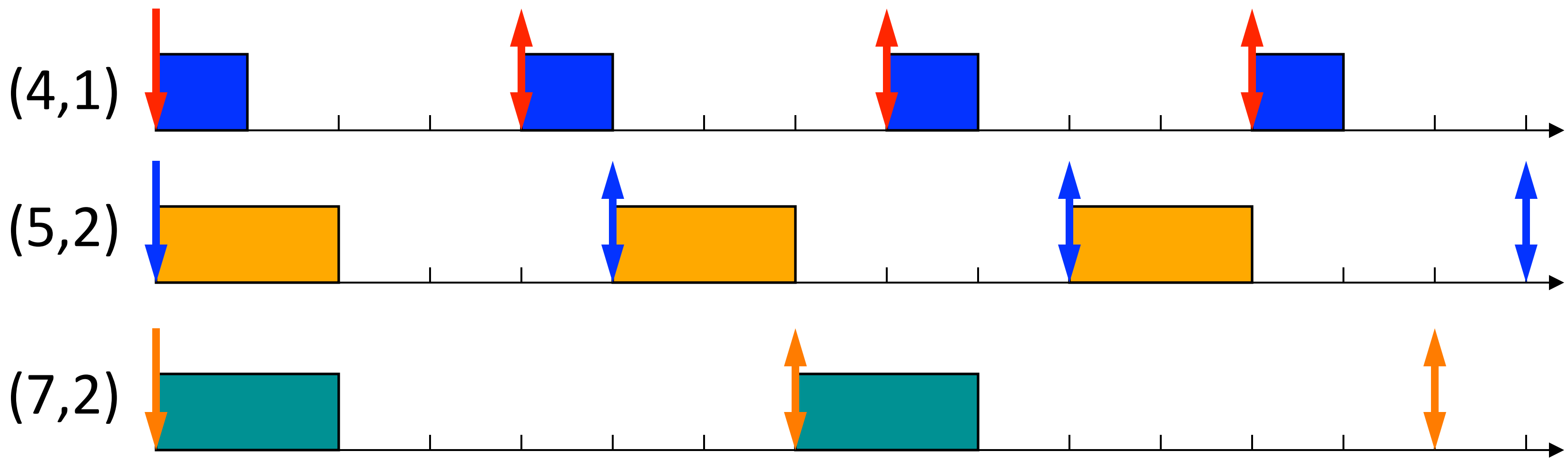
Deadlines: Hard vs. Soft

- Hard deadline
  - Disastrous or very serious consequences may occur if the deadline is missed
  - Validation is essential : can all the deadlines be met, even under worst-case scenario?
  - Deterministic guarantees

- Soft deadline
  - Ideally, the deadline should be met for maximum performance. The performance degrades in case of deadline misses.
  - Best effort approaches / statistical guarantees

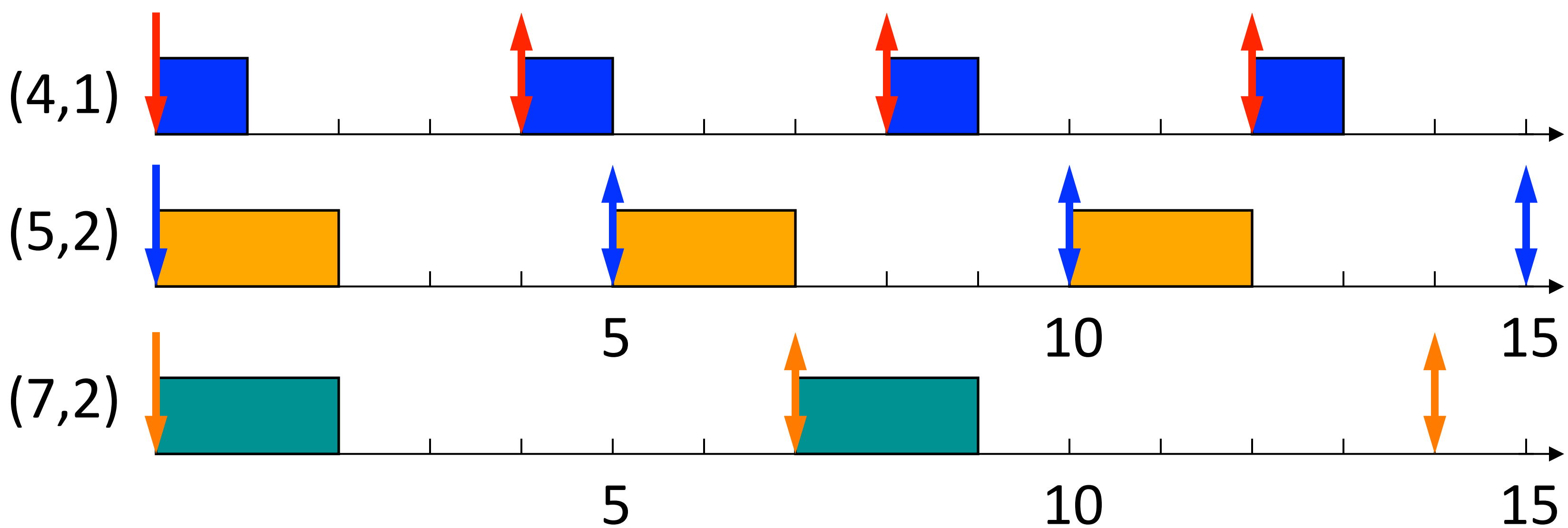
Schedulability

- Property indicating whether a real-time system (a set of real-time tasks) can meet their deadlines



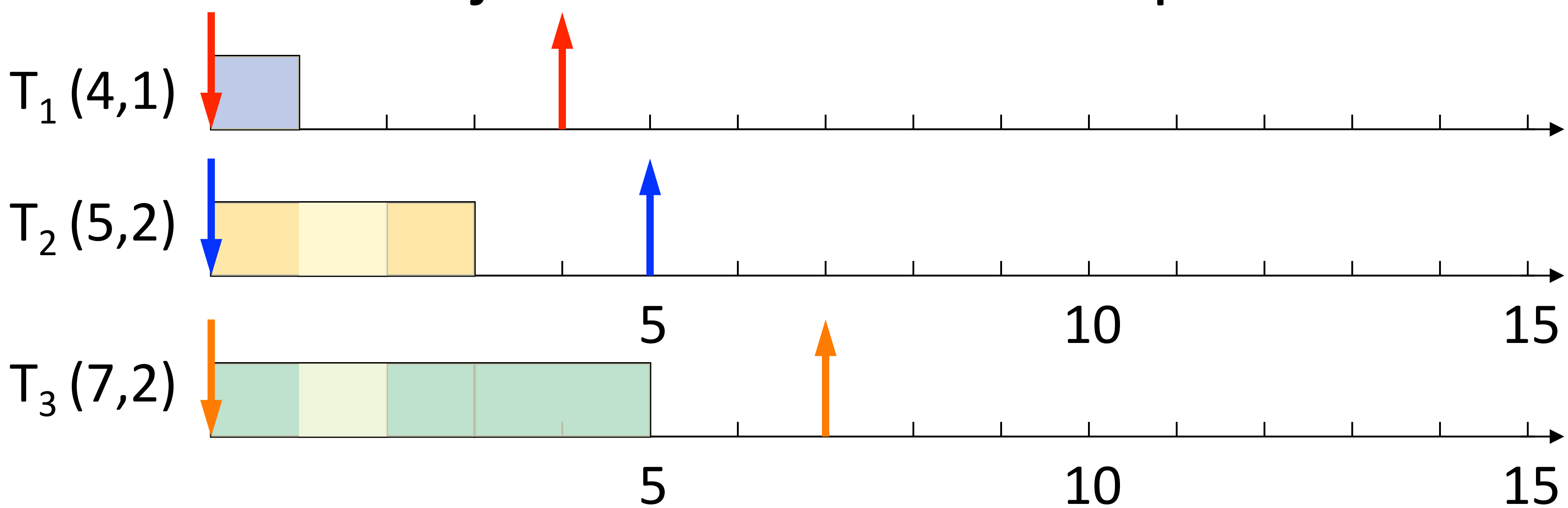
Real-Time Scheduling

- Determines the order of real-time task executions
- Static-priority scheduling
- Dynamic-priority scheduling



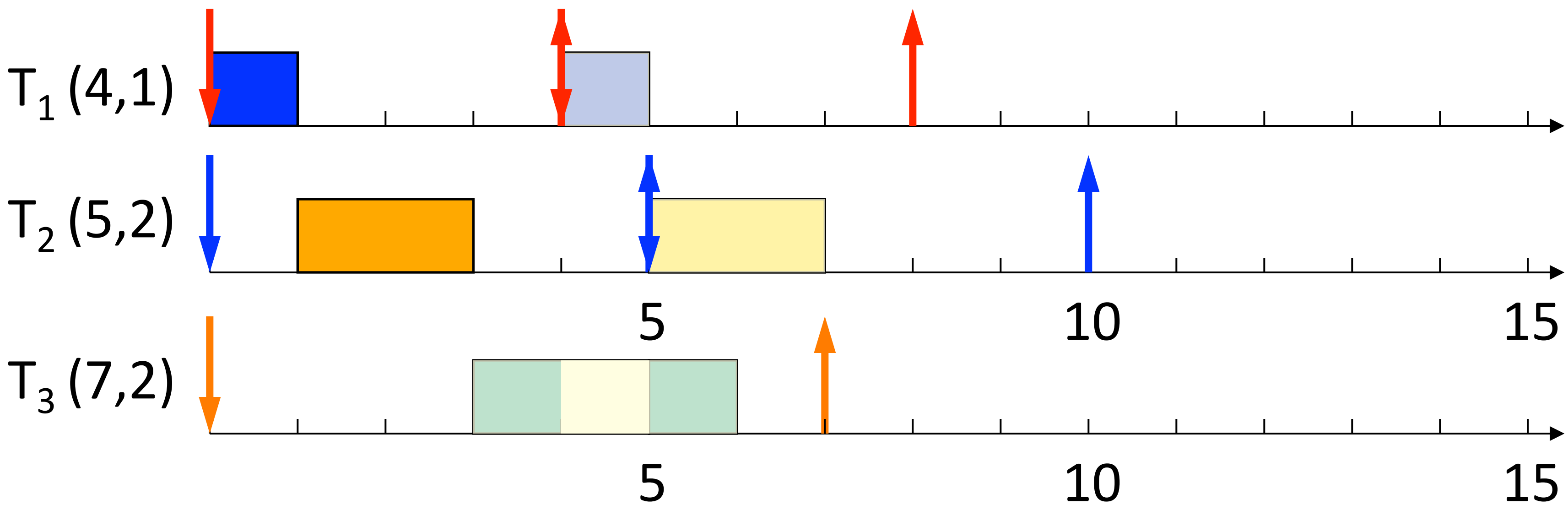
RM (Rate Monotonic)

- Optimal **static-priority** scheduling
- It assigns priority according to period
- A task with a shorter period has a higher priority
- Executes a job with the shortest period



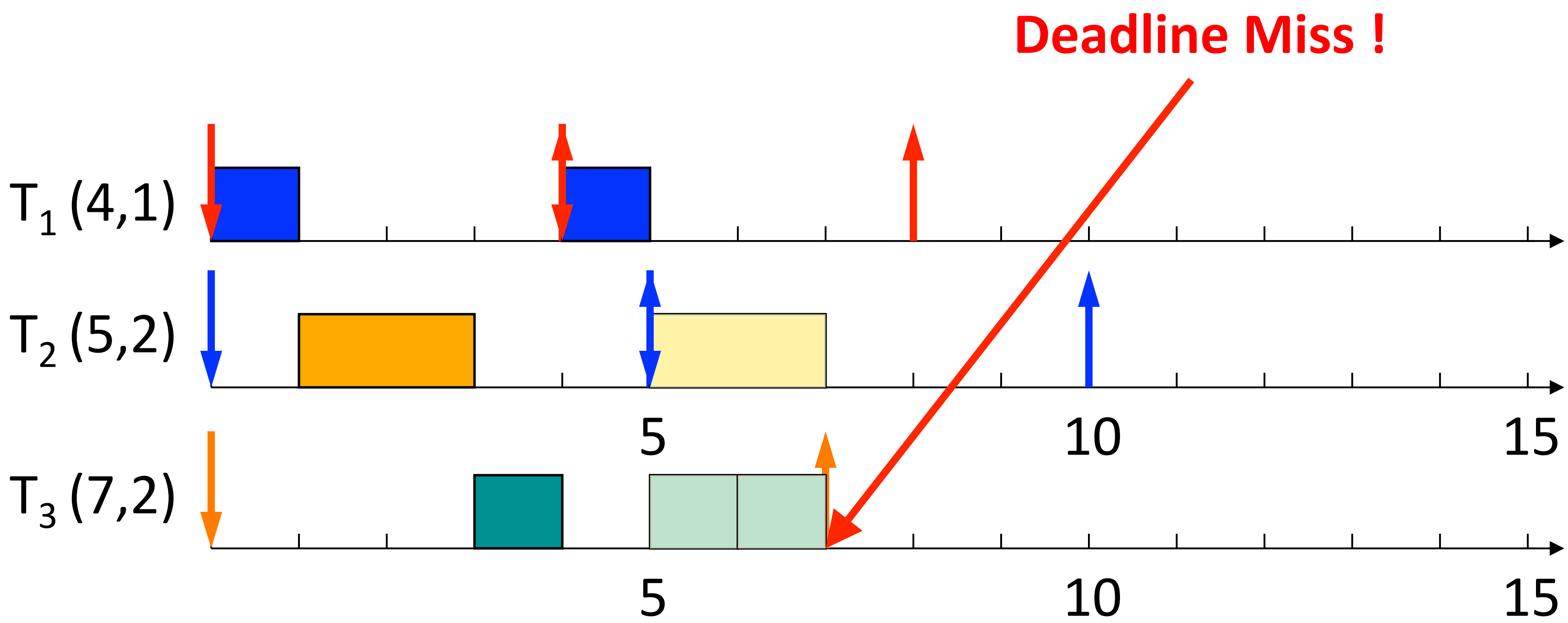
RM (Rate Monotonic)

- Executes a job with the shortest period



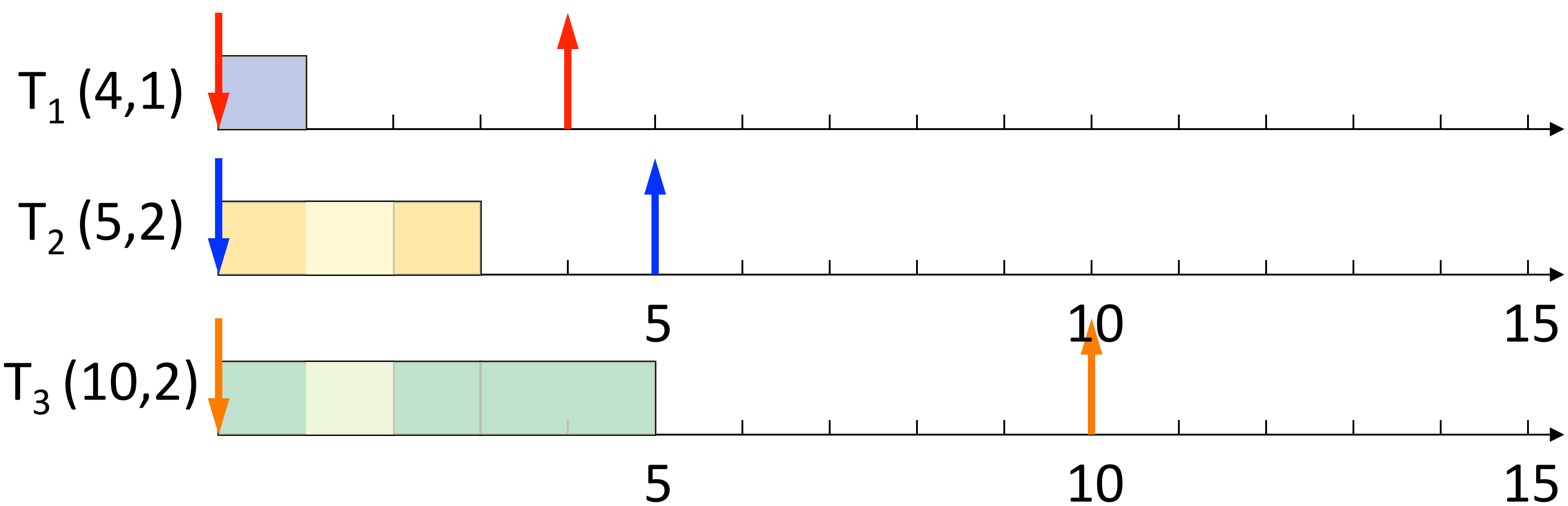
RM (Rate Monotonic)

- Executes a job with the shortest period



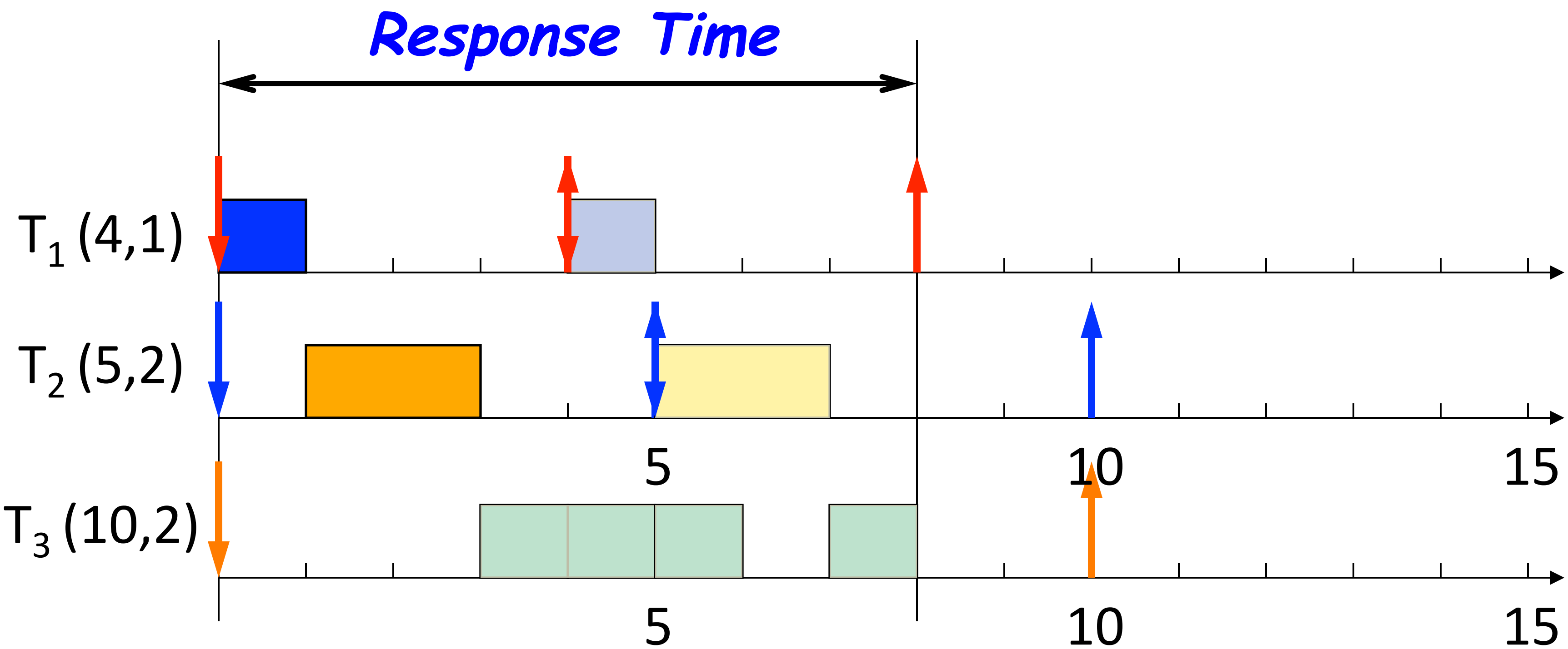
Response Time

- Response time
  - Duration from released time to finish time



Response Time

- Response time
  - Duration from released time to finish time

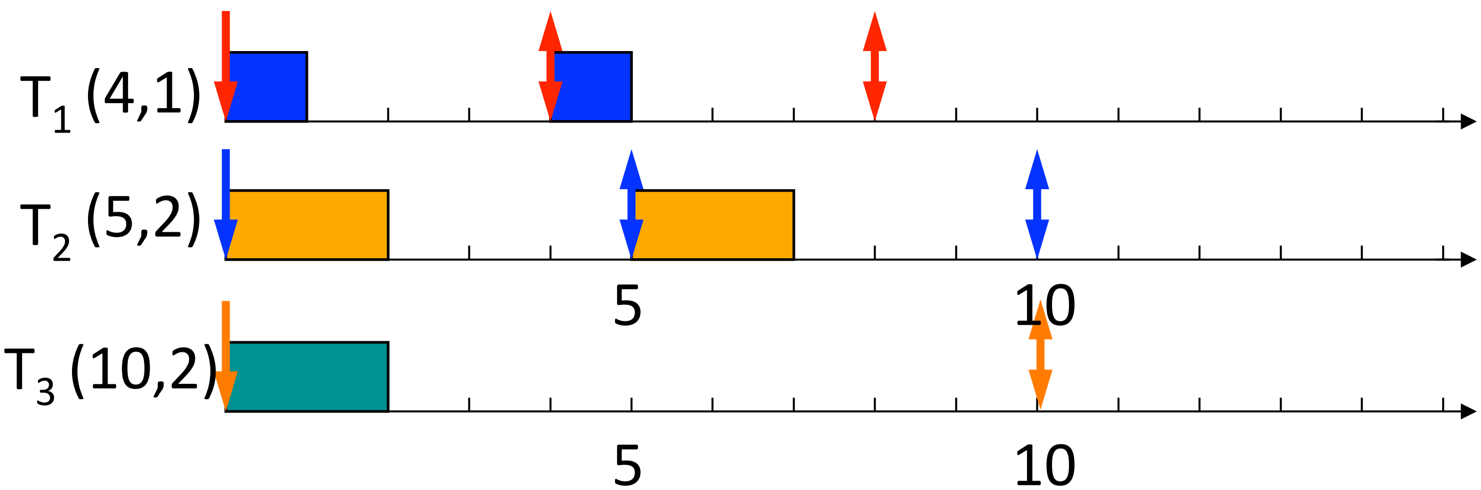


Response Time

- Response Time (***r$_i$***) [Audsley et al., 1993]

$$r_i = e_i + \sum_{T_k \in HP(T_i)} \left\lceil \frac{r_i}{p_k} \right\rceil \cdot e_k$$

- HP(T_i) : a set of higher-priority tasks than T_i



RM - Schedulability Analysis

- Real-time system is schedulable under RM if and only if $r_i \leq p_i$ for all task $T_i(p_i,e_i)$

Joseph & Pandya,

"Finding response times in a real-time system",

The Computer Journal, 1986.

RM – Utilization Bound

- Real-time system is schedulable under RM if

$$\sum U_i \leq n\ (2^{1/n}-1)$$

Liu & Layland,

"Scheduling algorithms for multi-programming in a hard-real-time environment", Journal of ACM, 1973.

RM – Utilization Bound

- Real-time system is schedulable under RM if

  $\sum U_i \leq n\,(2^{1/n}-1)$

- Example: $T_1(4,1)$, $T_2(5,1)$, $T_3(10,1)$,

$$\sum U_i = 1/4 + 1/5 + 1/10$$
$$= 0.55$$
$$3\,(2^{1/3}-1) \approx 0.78$$

Thus, $\{T_1, T_2, T_3\}$ is schedulable under RM.

RM – Utilization Bound

- Real-time system is schedulable under RM if
  $$\sum U_i \le n\,(2^{1/n}-1)$$

- Example: $T_1(4,1)$, $T_2(5,2)$, $T_3(7,2)$,

$$\sum U_i = 1/4 + 2/5 + 2/7$$
$$\approx 0.94$$
$$3\,(2^{1/3}-1) \approx 0.78$$

Thus, $\{T_1, T_2, T_3\}$ is NOT schedulable under RM.

RM – Utilization Bound

- Real-time system is schedulable under RM if

$$\sum U_i \leq n\,(2^{1/n}-1)$$

- Example: $T_1(4,1)$, $T_2(5,2)$, $T_3(10,2)$,

$$\sum U_i = 1/4 + 2/5 + 2/10$$
$$= 0.85$$
$$3\,(2^{1/3}-1) \approx 0.78$$

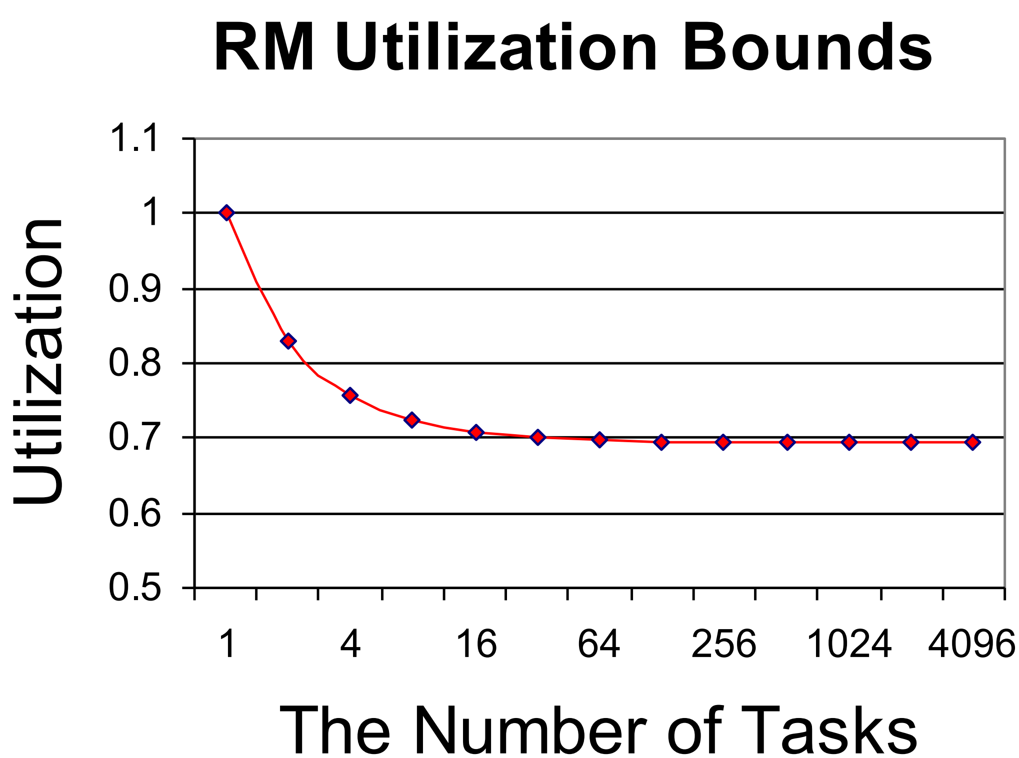
However, $\{T_1, T_2, T_3\}$ is still schedulable under RM (as we just showed) even their total utilization is higher than the bound!

RM – Utilization Bound

- Real-time system is schedulable under RM **if (but not only if)** $\sum U_i \leq n\,(2^{1/n}-1)$
- The above condition is only a **sufficient but not necessary** condition!
  - We only know tasks with utilization lower than the bound is guaranteed to be schedulable under RM.
  - We know nothing about tasks with higher utilization!

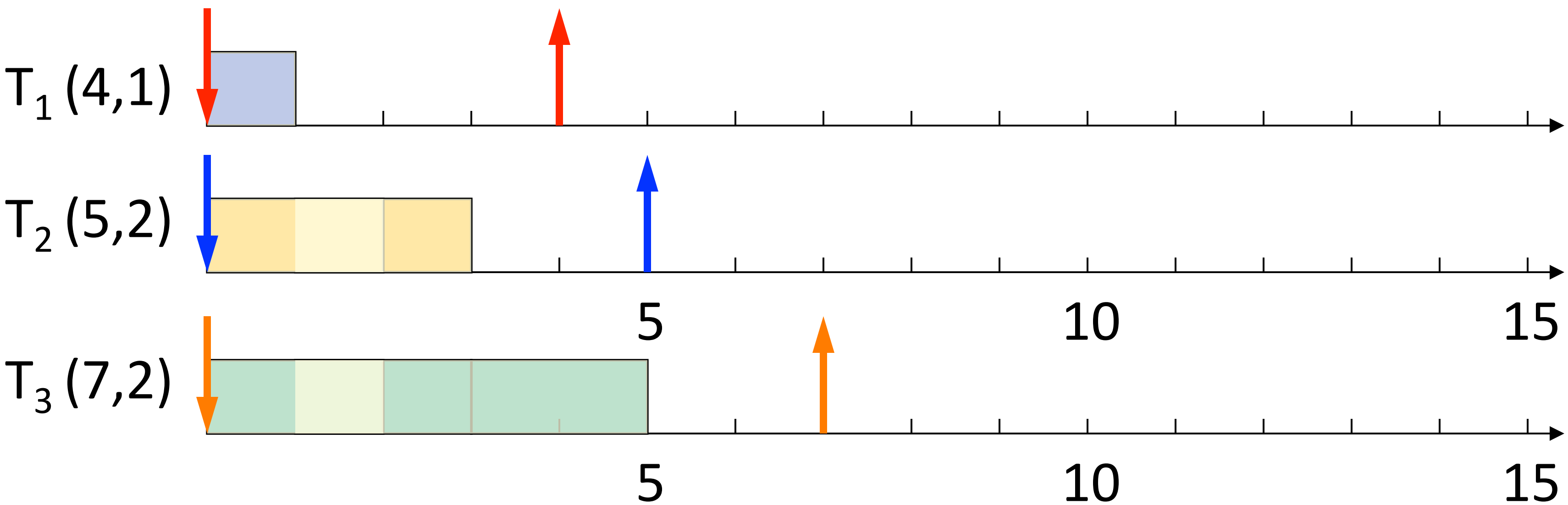
RM – Utilization Bound

- Real-time system is schedulable under RM if
  $\sum U_i \leq n\,(2^{1/n}-1)$



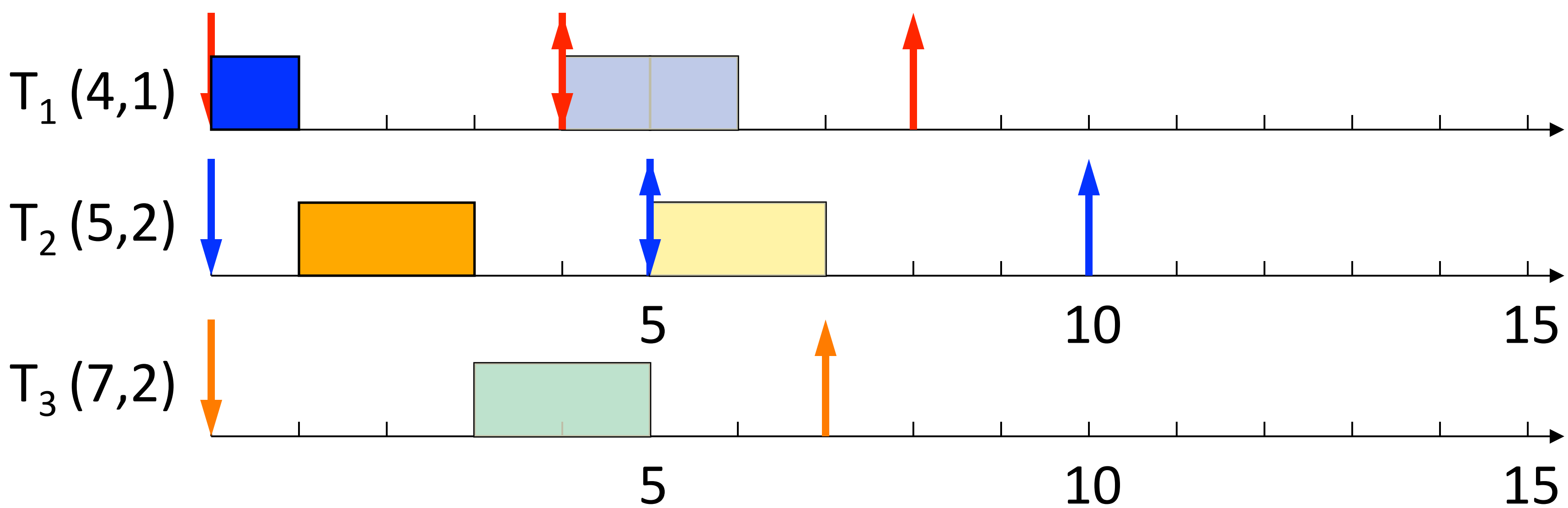
EDF (Earliest Deadline First)

- Optimal **dynamic** priority scheduling
- A task with a shorter deadline has a higher priority
- Executes a job with the earliest deadline



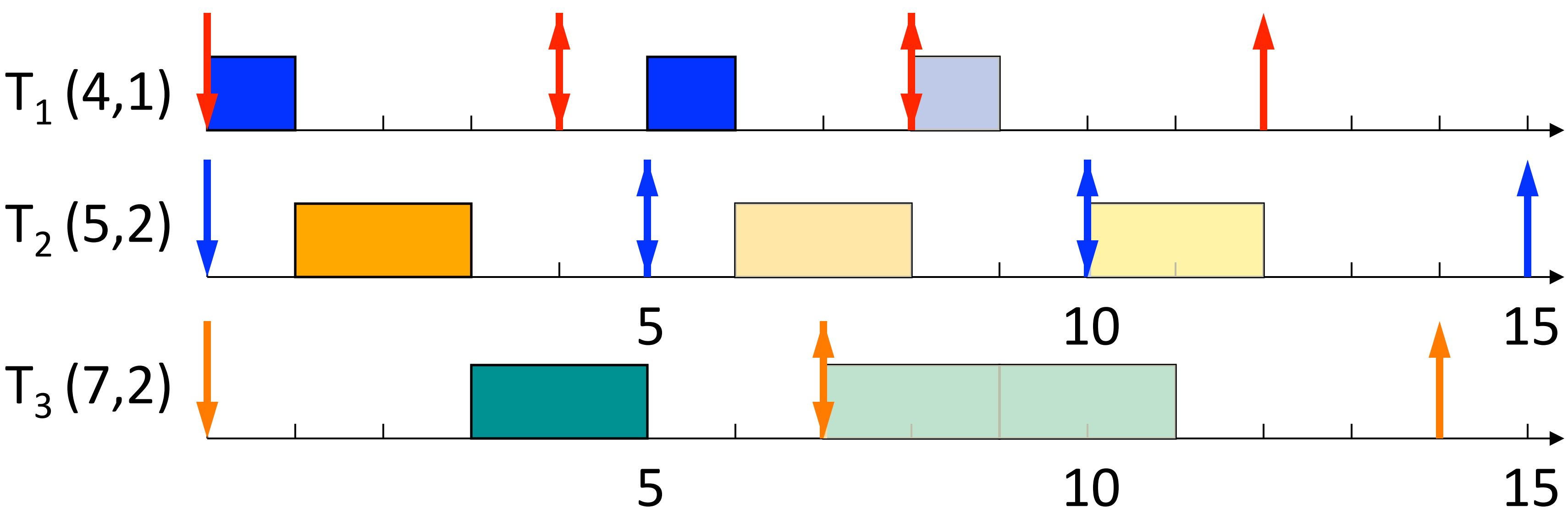
EDF (Earliest Deadline First)

- Executes a job with the earliest deadline



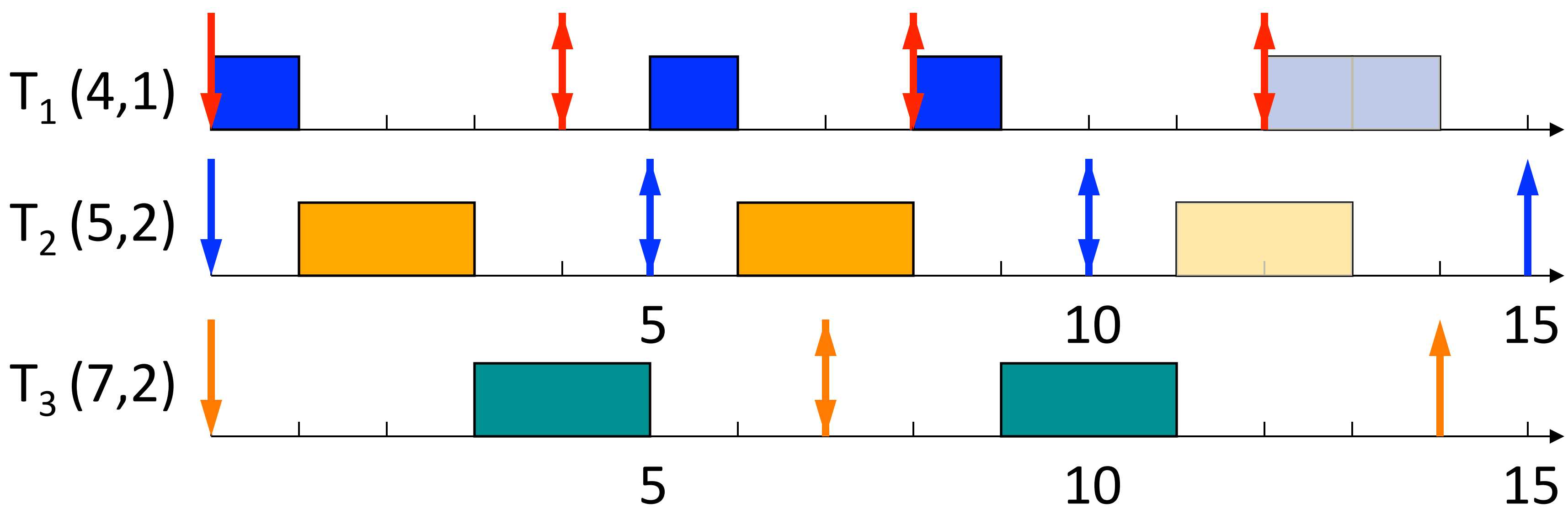
EDF (Earliest Deadline First)

- Executes a job with the earliest deadline



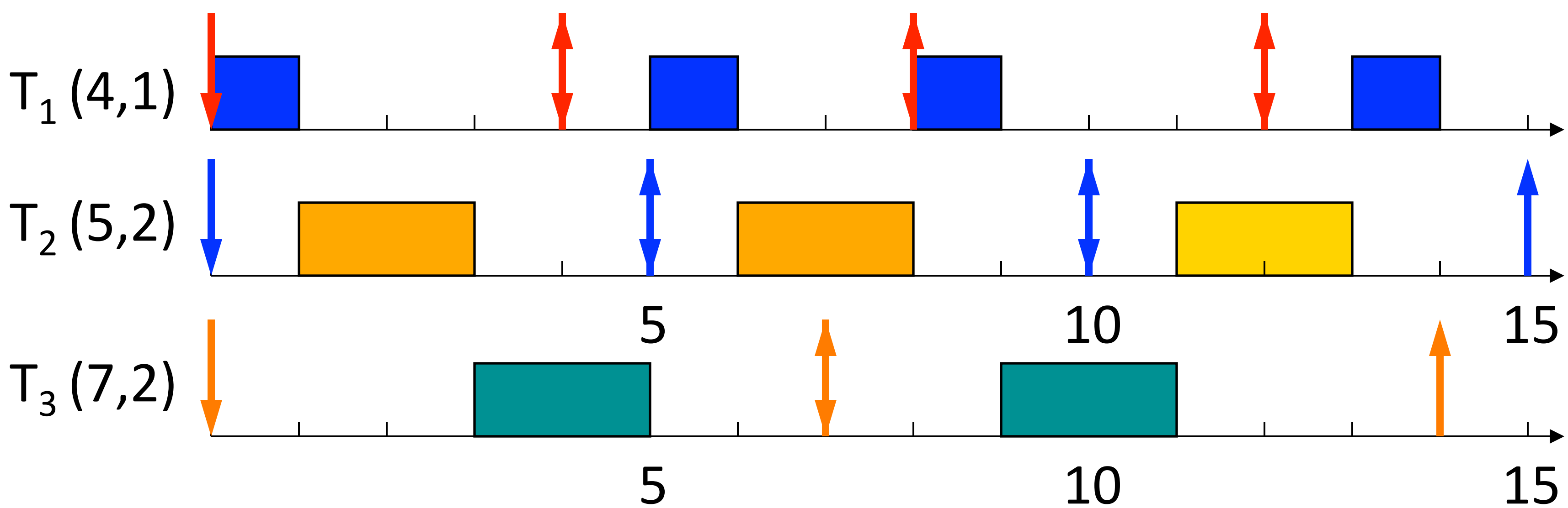
EDF (Earliest Deadline First)

- Executes a job with the earliest deadline



EDF (Earliest Deadline First)

- Optimal scheduling algorithm
  - if there is a schedule for a set of real-time tasks, EDF can schedule it.



EDF – Utilization Bound

- Real-time system is schedulable under EDF **if and only if**

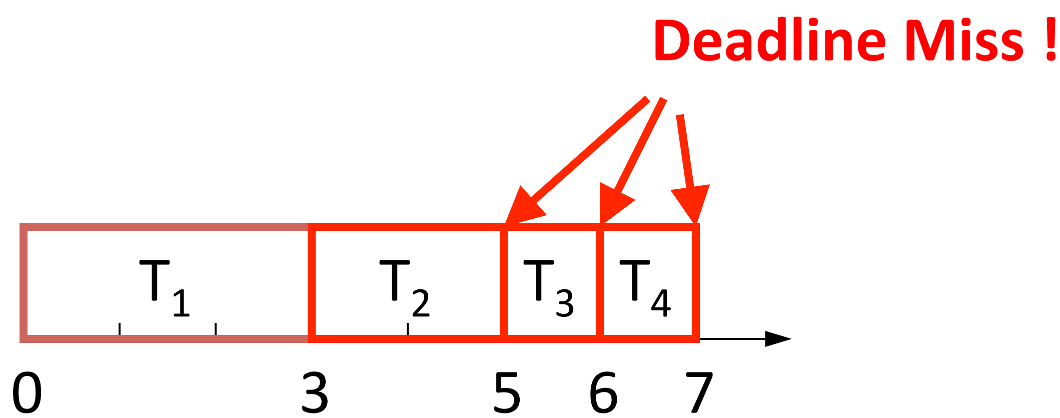
$$\sum U_i \leq 1$$

Liu & Layland,

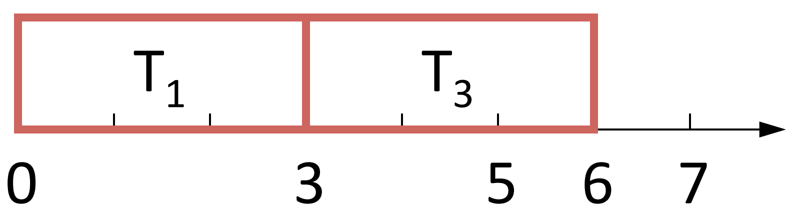
"Scheduling algorithms for multi-programming in a hard-real-time environment", Journal of ACM, 1973.

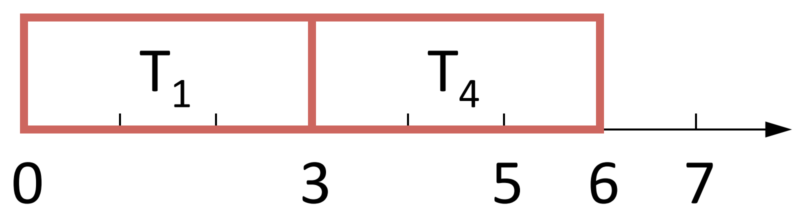
EDF – Overload Conditions

- Domino effect during overload conditions
  - Example: $T_1(4,3)$, $T_2(5,3)$, $T_3(6,3)$, $T_4(7,3)$



Better schedules :

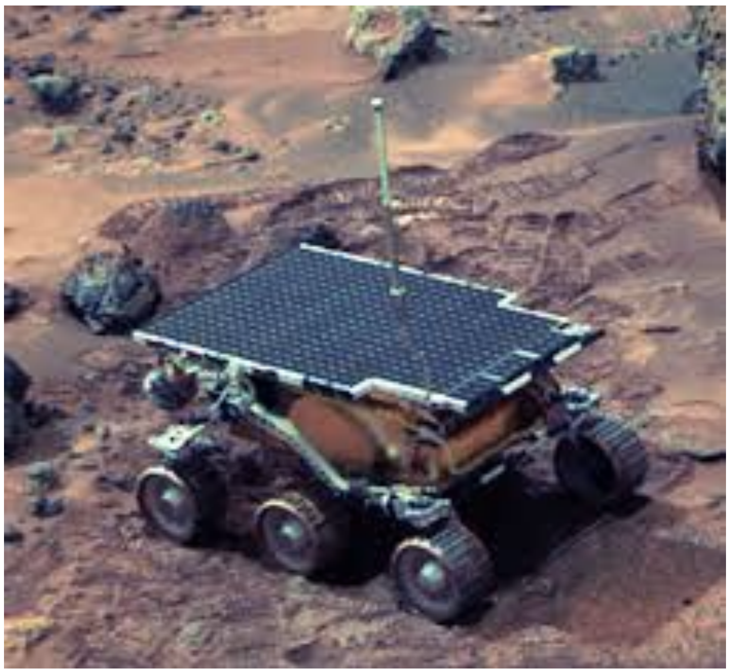




RM vs. EDF

- Rate Monotonic
  - Simpler implementation, even in systems without explicit support for timing constraints (periods, deadlines)
  - Predictability for the highest priority tasks
- EDF
  - Full processor utilization
  - Misbehavior during overload conditions
- For more details: Buttazzo, “Rate monotonic vs. EDF: Judgement Day”, EMSOFT 2003.

Real-world Example: What Happens on Mars?



Prof. Lui Sha, CS, UIUC

Priority Inheritance and Priority Ceiling Protocols

L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization", IEEE Transactions on Computers, Vol. 39, No. 9, Sept. 1990.

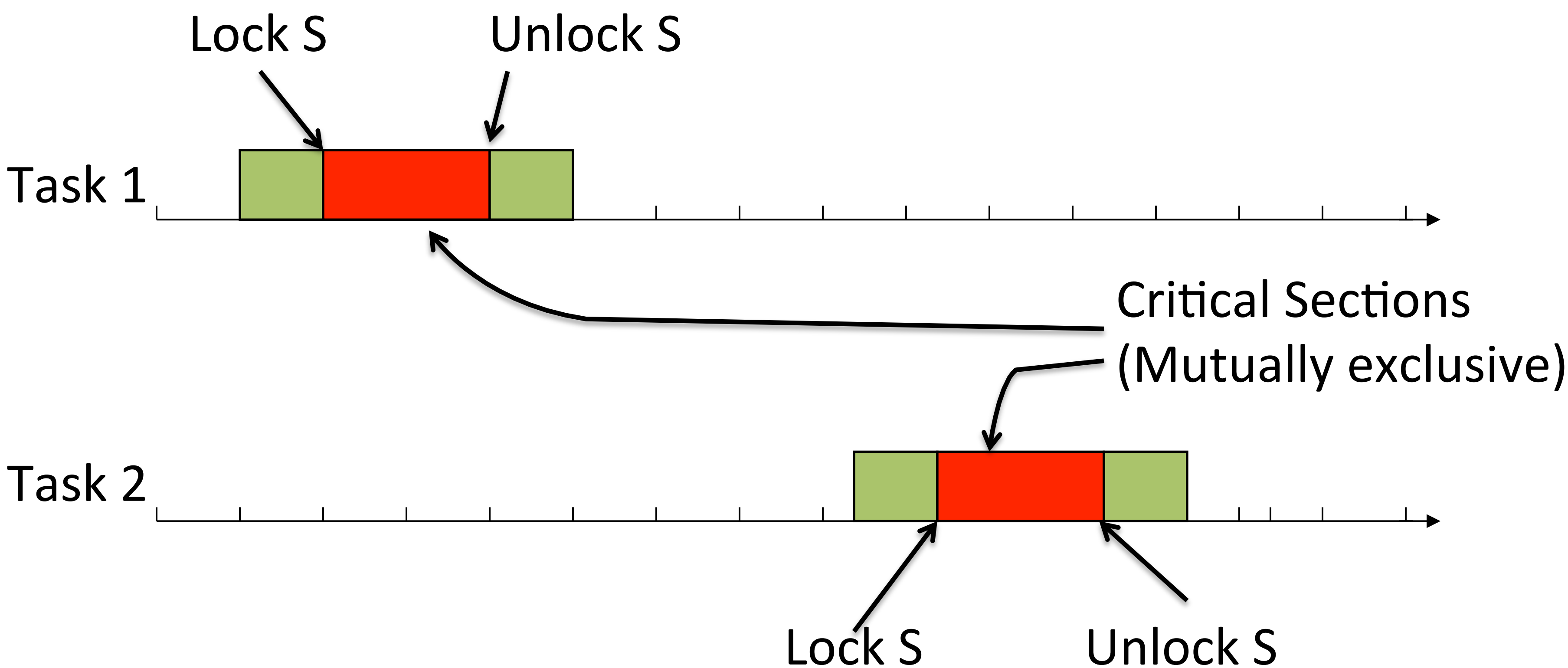
http://research.microsoft.com/en-us/um/people/mbj/Mars_Pathfinder/Mars_Pathfinder.html

The Problem

- Tasks have **synchronization** constraints
  - Semaphores protect critical sections
- Blocking can cause a higher-priority task to wait on an **lower-priority one** to unlock the resource
  - Problem: In all previous scheduling examples, we assumed that a task can only wait for **higher priority tasks** not **lower-priority tasks**

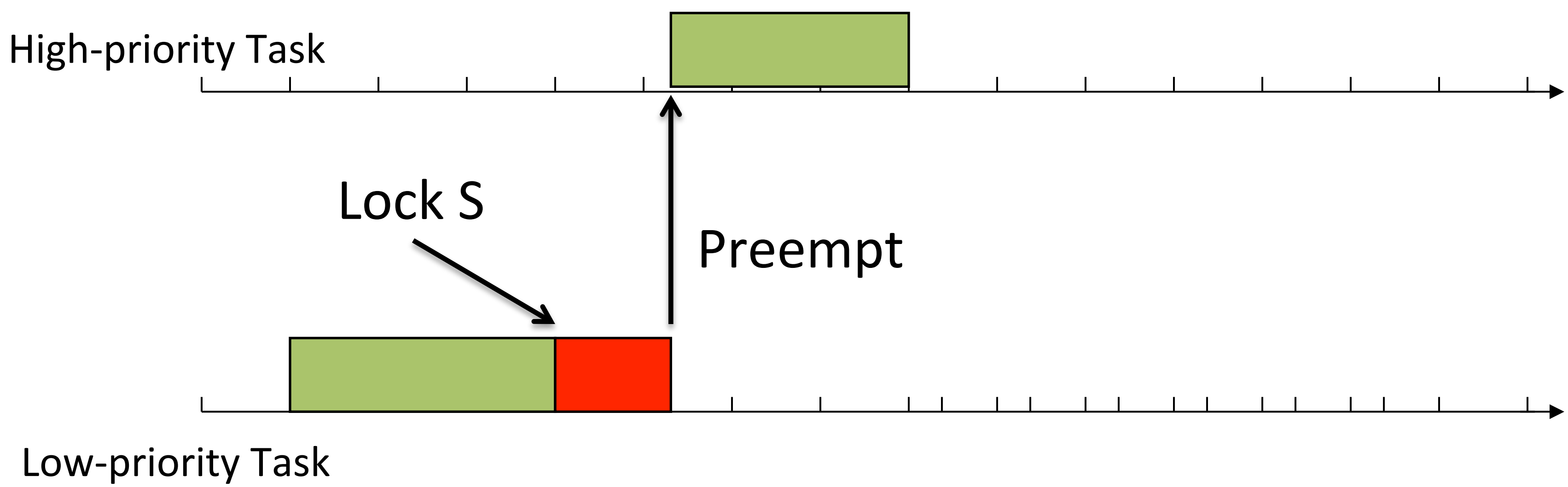
Mutual Exclusion Constraints

- Tasks that lock/unlock the same semaphore are said to have a mutual exclusion constraint



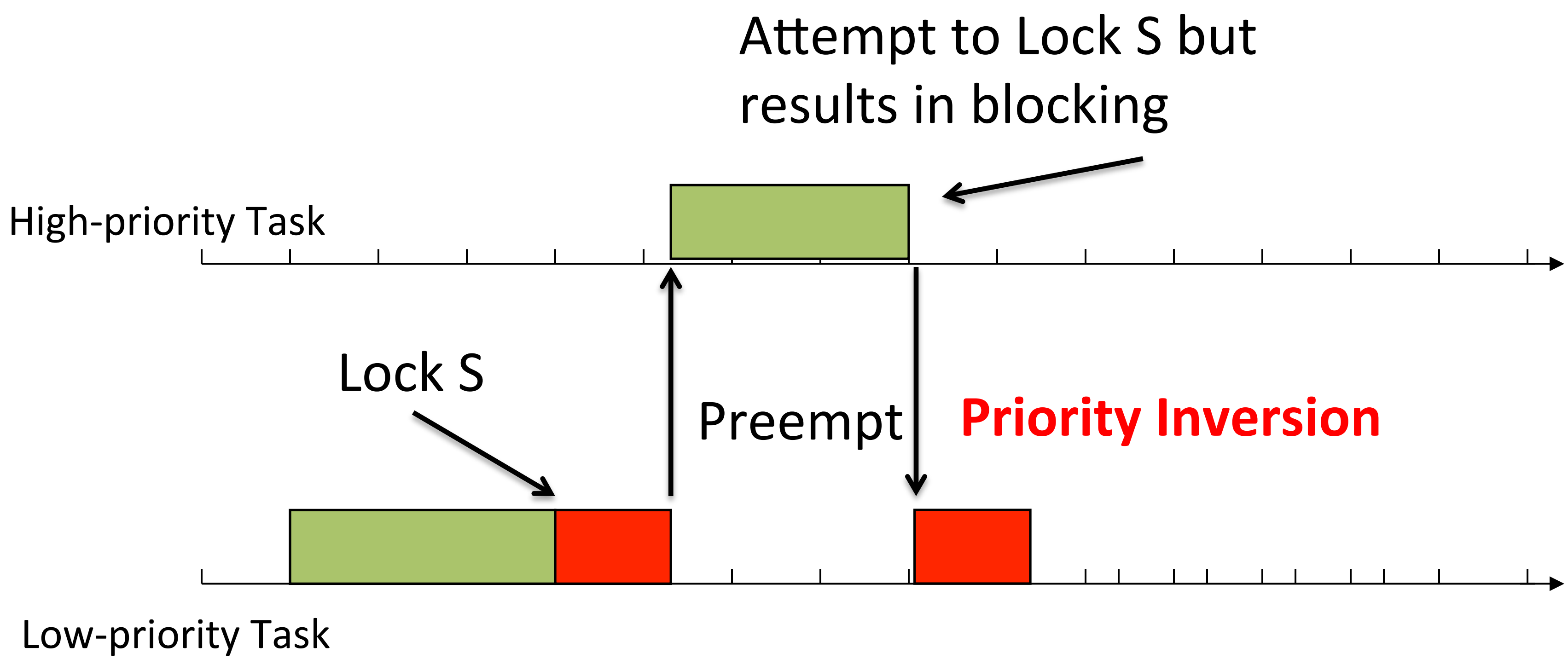
Priority Inversion

- Locks and priorities **may be at odds**. Locking results in priority inversion.



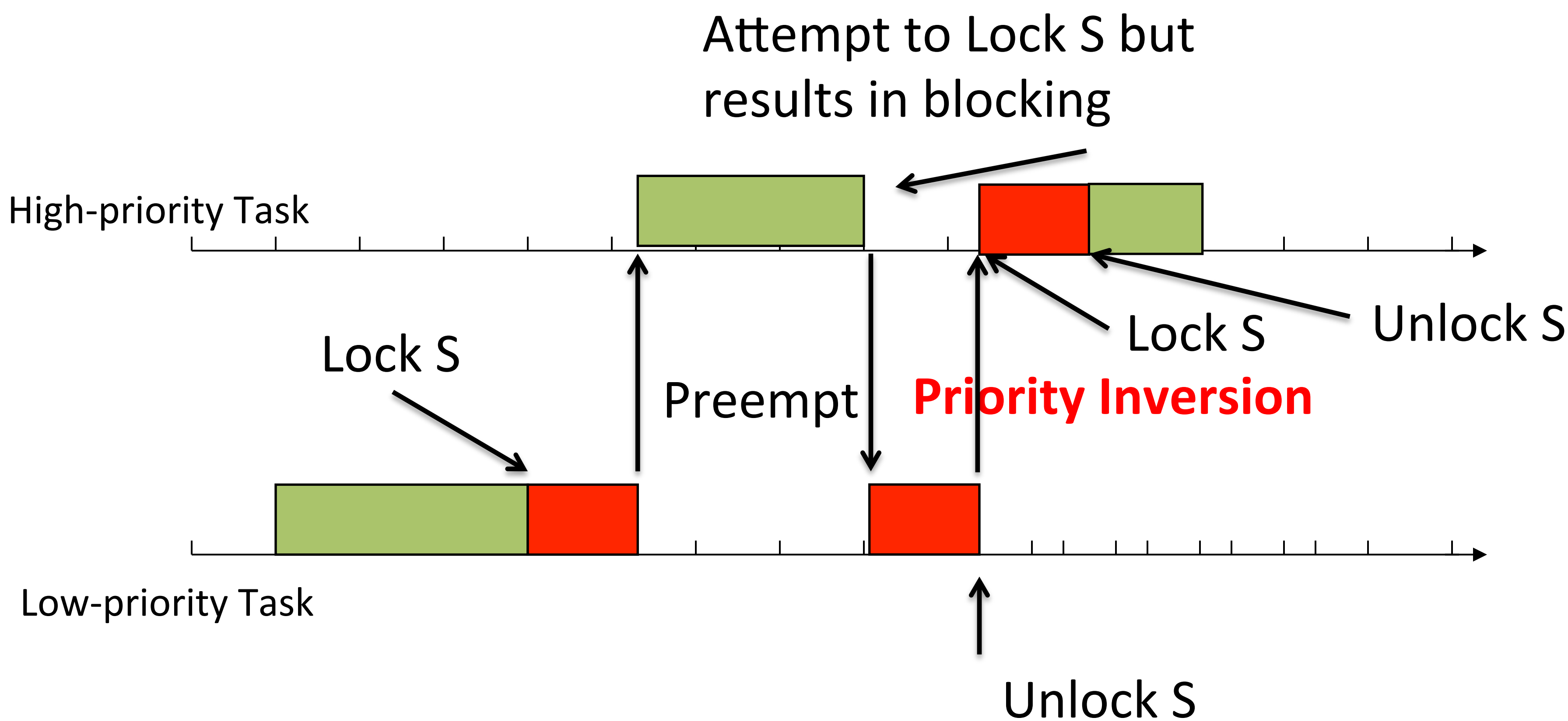
Priority Inversion

- Locks and priorities **may be at odds**. Locking results in priority inversion.



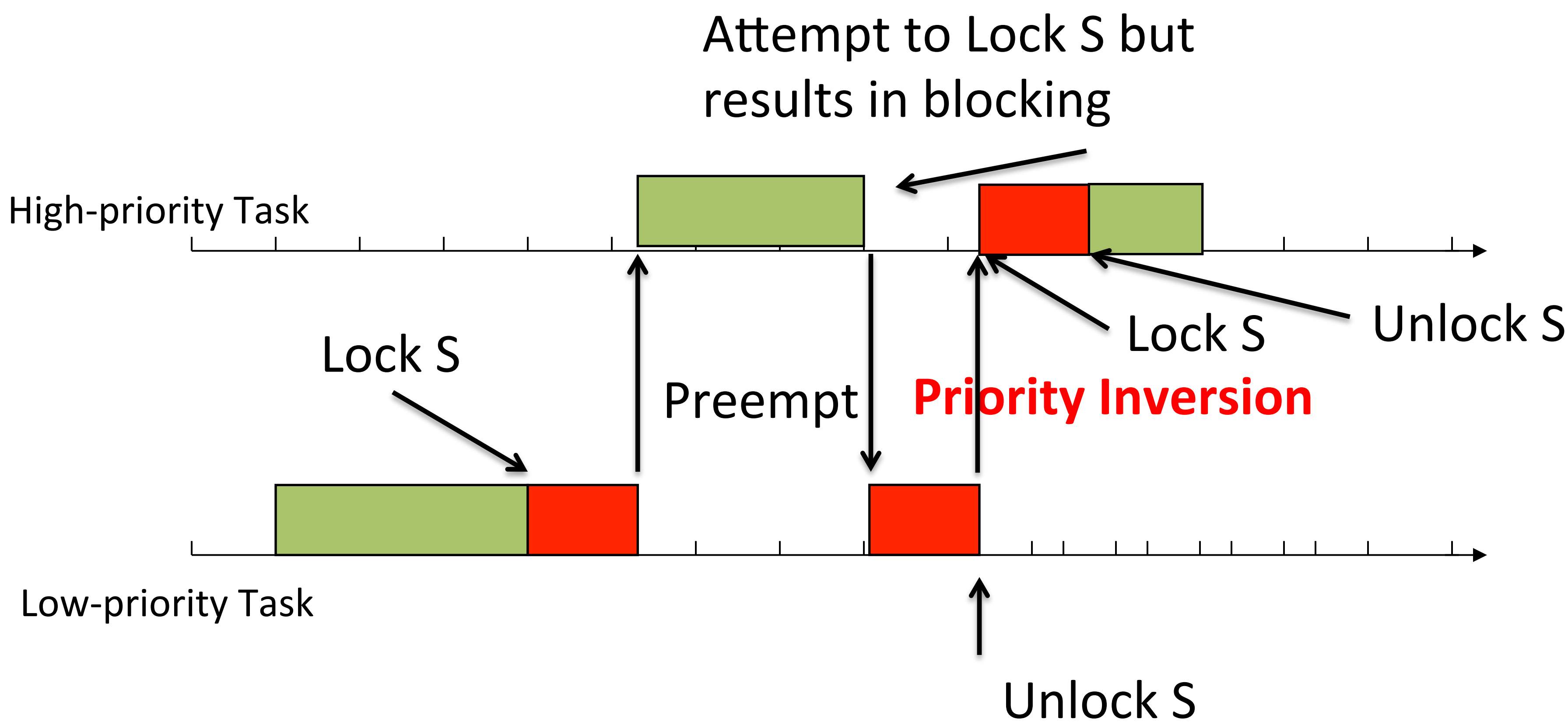
Priority Inversion

- How to account for priority inversion?



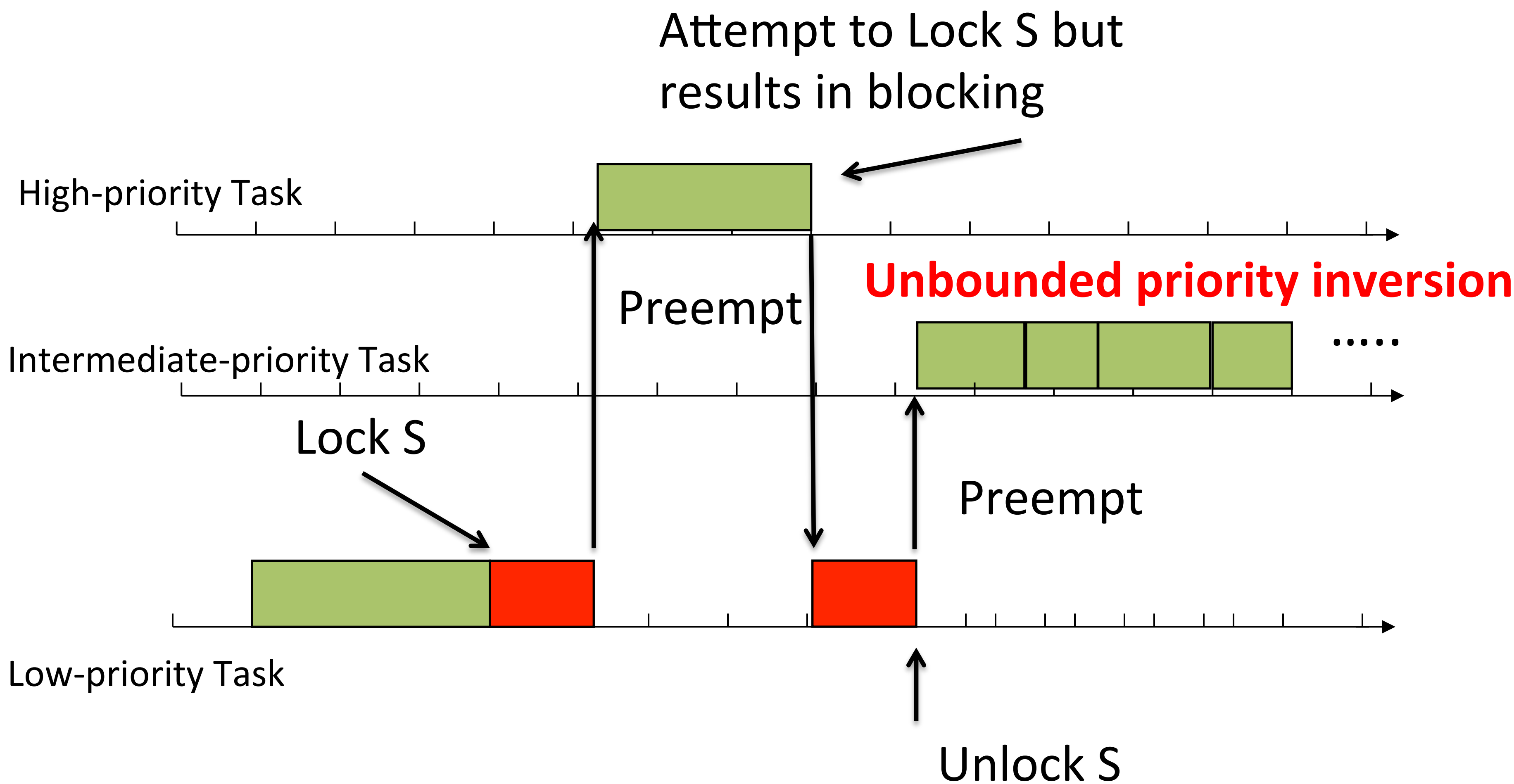
Priority Inversion

- What is the problem of this scheme? Can the high-priority task gets delayed unboundedly?



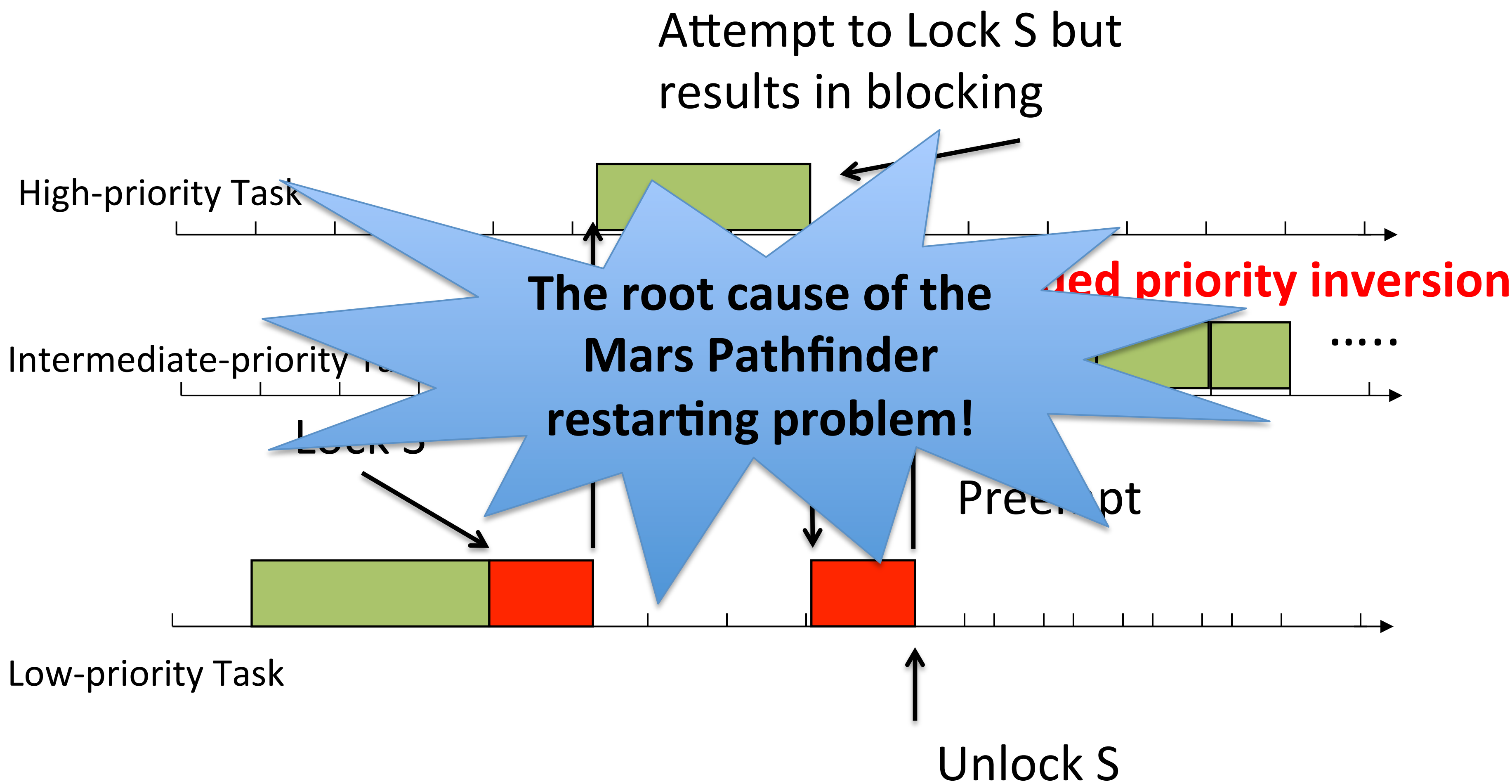
Unbounded Priority Inversion

- Consider the following: a series of intermediate priority tasks is delaying a higher-priority one



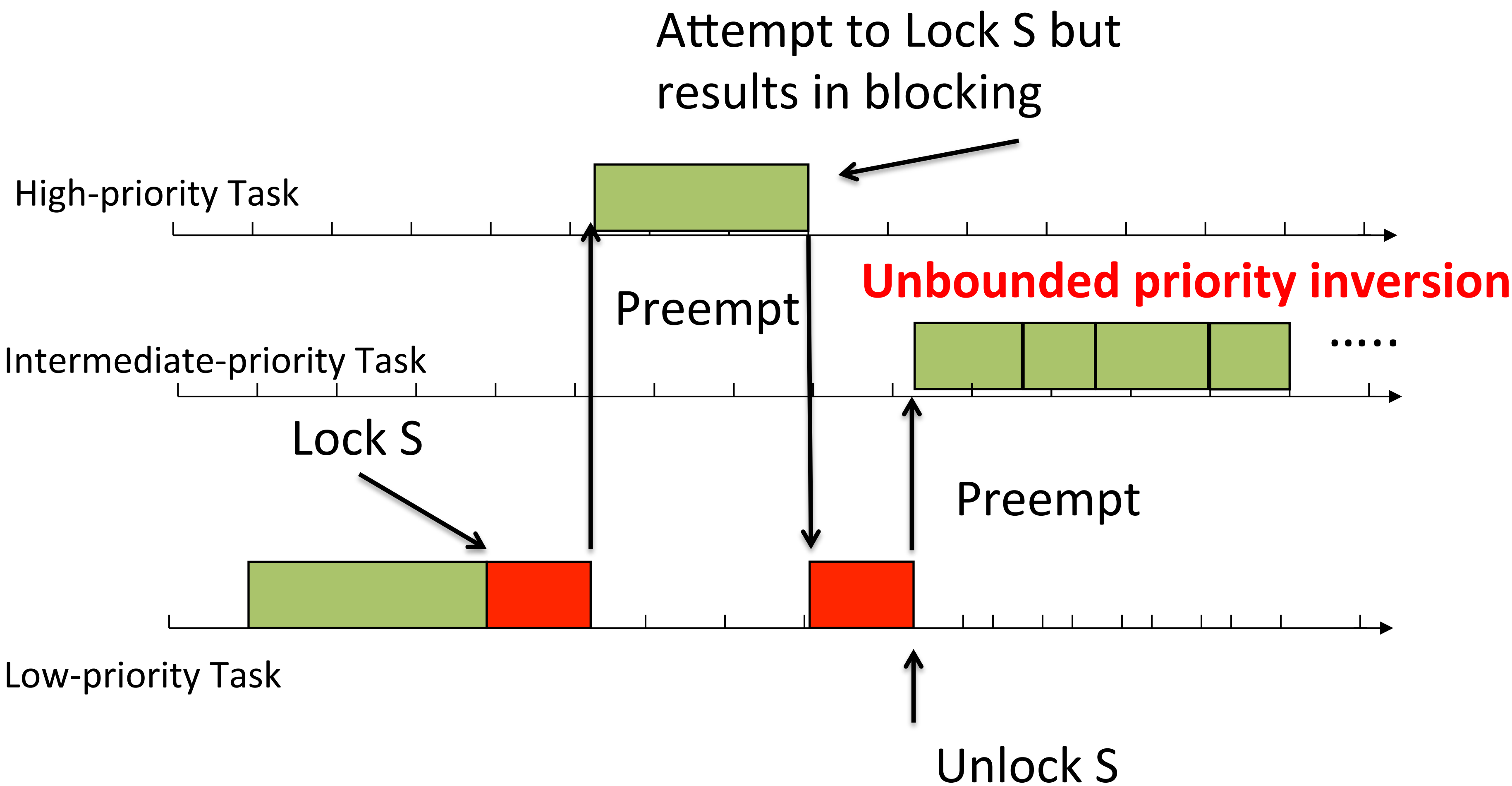
Unbounded Priority Inversion

- Consider the following: a series of intermediate priority tasks is delaying a higher-priority one



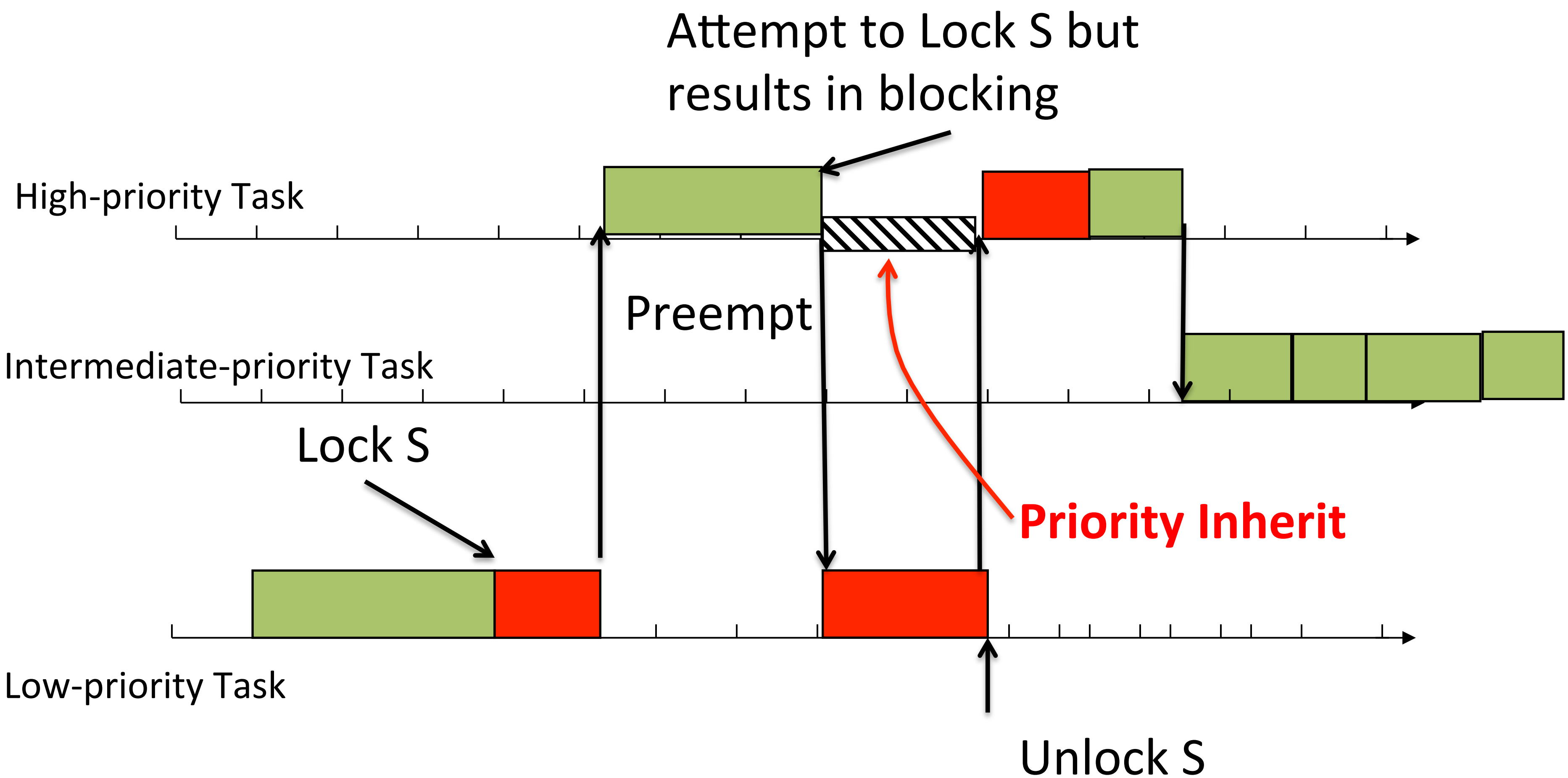
Unbounded Priority Inversion

- How to **prevent** unbounded priority inversion?



Priority Inheritance Protocol

- Let a task inherit the priority of any higher priority task it is blocking

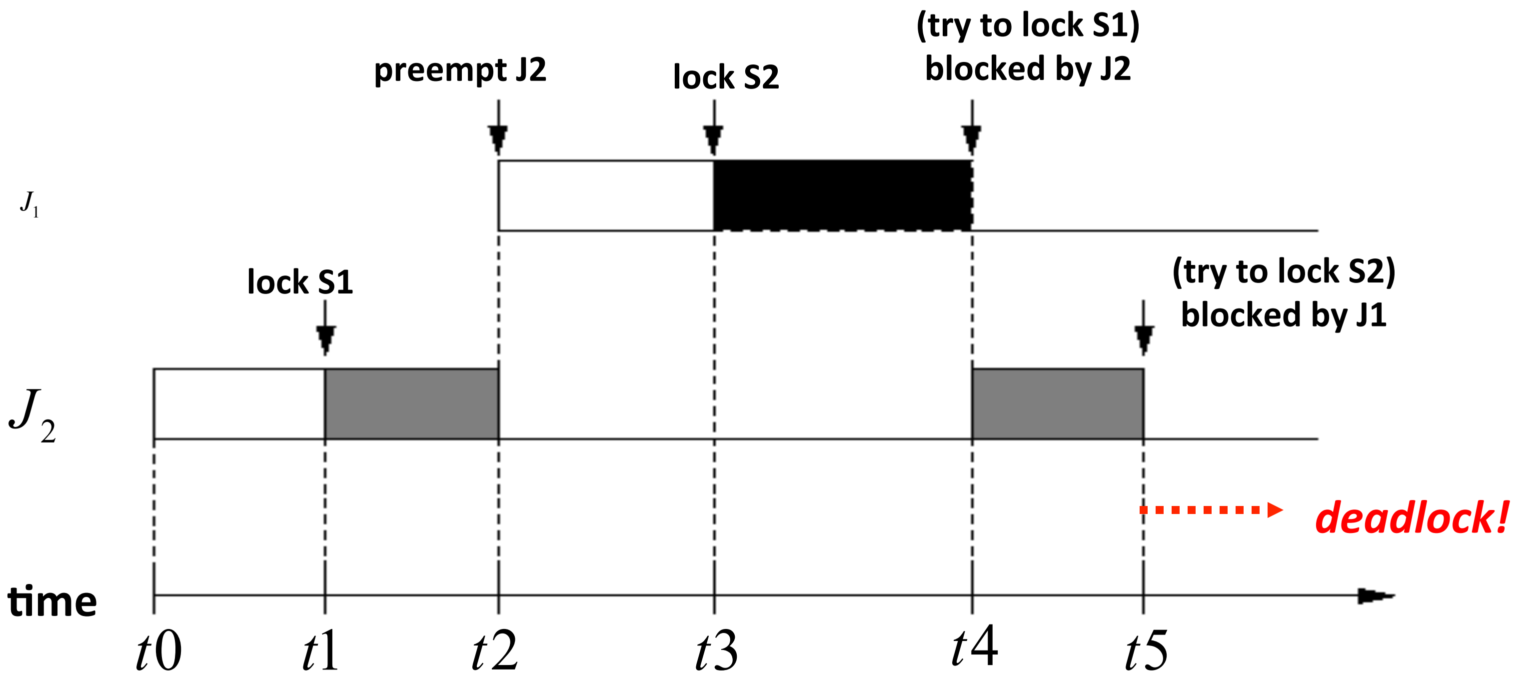


Deadlocks in PIP

$$J_1 = \{\ldots, P(S_2) \ldots, P(S_1), \ldots, V(S_1) \ldots, V(S_2), \ldots\}$$

$$J_2 = \{\ldots, P(S_1) \ldots, P(S_2), \ldots, V(S_2) \ldots, V(S_1), \ldots\}$$

crossing nested semaphores

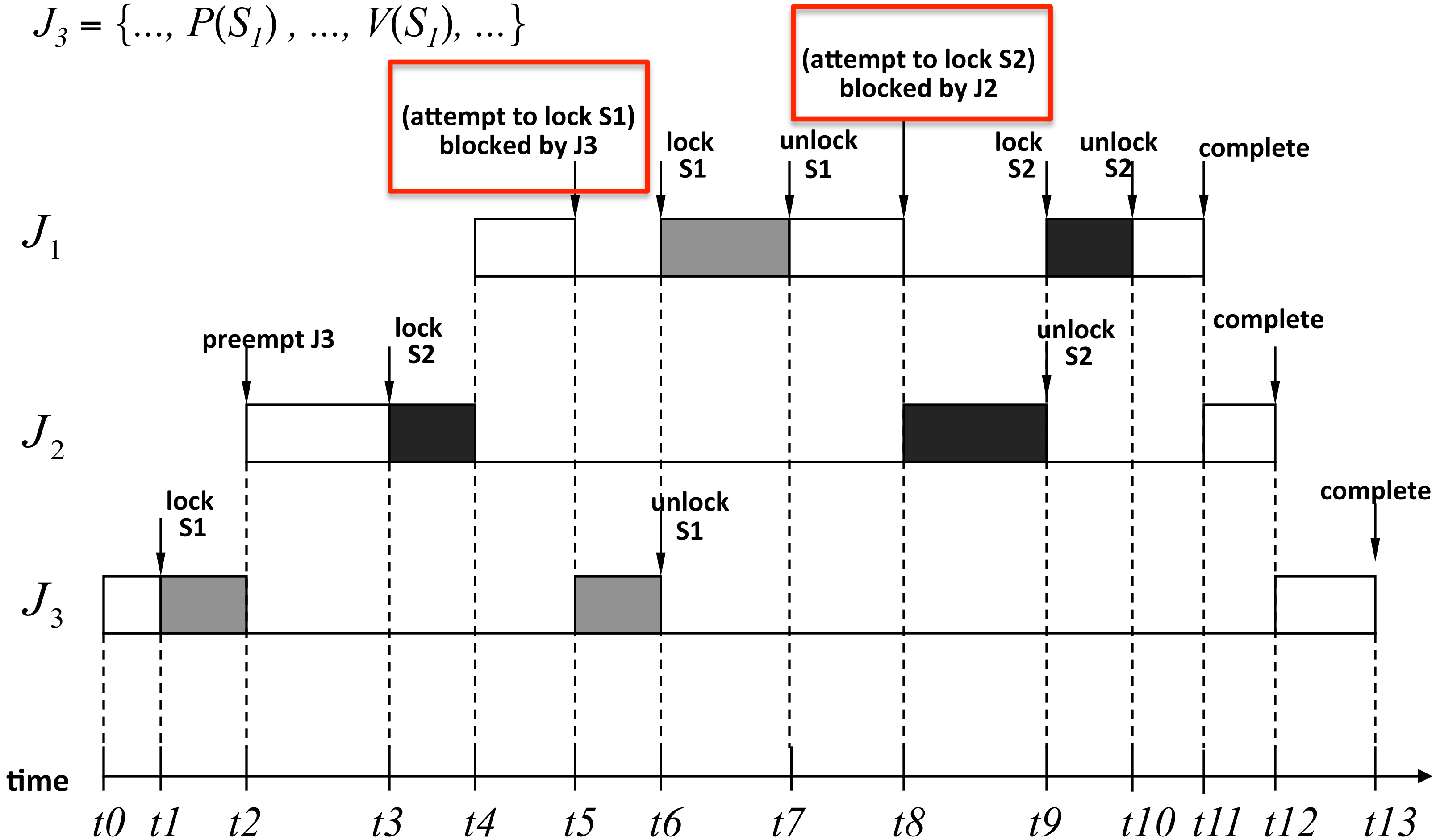


Blocking Chains in PIP

$J_1 = \{\ldots, P(S_1), \ldots, V(S_1), \ldots, P(S_2), \ldots, V(S_2), \ldots\}$

$J_2 = \{\ldots, P(S_2), \ldots, V(S_2), \ldots\}$

$J_3 = \{\ldots, P(S_1), \ldots, V(S_1), \ldots\}$



Priority Ceiling Protocol (PCP)

- Goals:
  - Solve problems of PIP.
    - Prevent deadlocks and blocking chains
- Basic idea:
  - Priority ceiling of a semaphore:
    - The priority of the highest priority task that may use the semaphore
  - Additional condition for allowing a job *J* to start a new critical section
    - **only if J's priority is higher than all priority ceilings of all the semaphores locked by jobs other than J.**

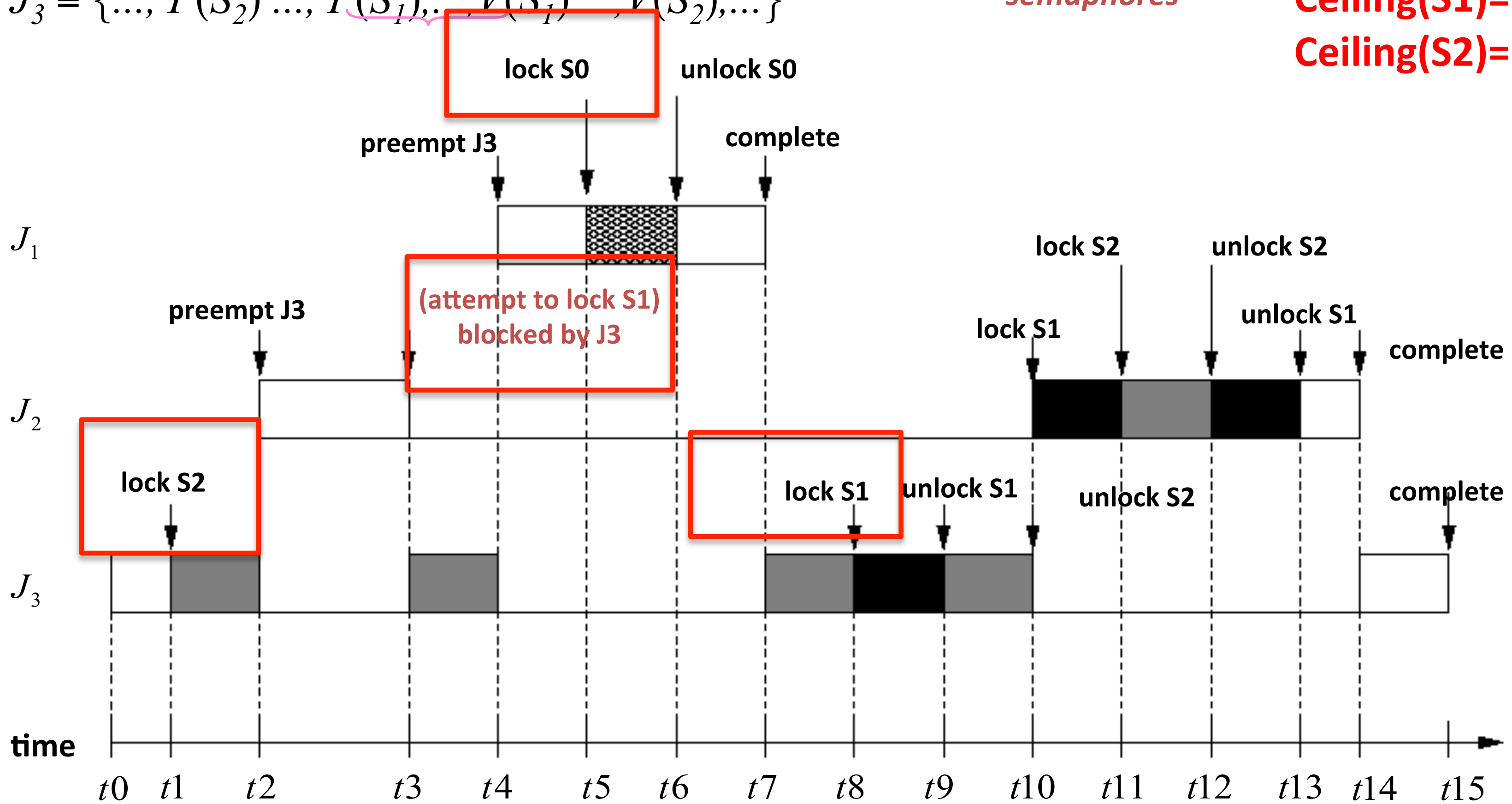
Examples for PCP (1/2)

- Prevent deadlocks

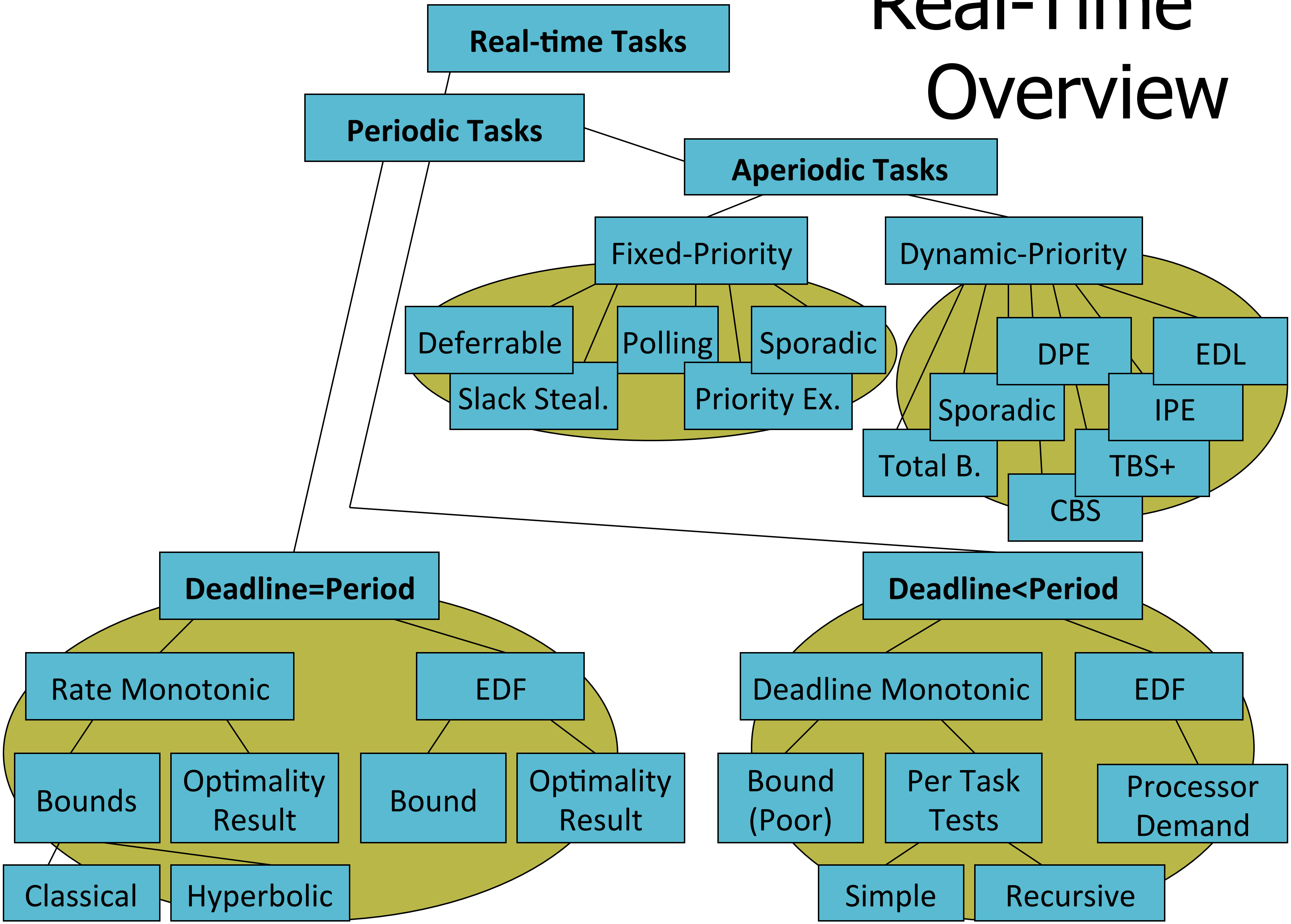
$J_1 = \{\ldots, P(S_0) \ldots, V(S_0), \ldots\}$
$J_2 = \{\ldots, P(S_1) \ldots, P(S_2), \ldots, V(S_2) \ldots, V(S_1), \ldots\}$
$J_3 = \{\ldots, P(S_2) \ldots, P(S_1), \ldots, V(S_1) \ldots, V(S_2), \ldots\}$

nested crossing semaphores

Ceiling(S0)= H
Ceiling(S1)= M
Ceiling(S2)= M

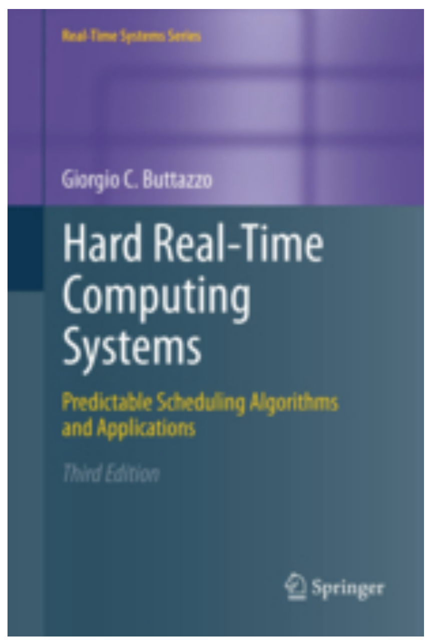


Real-Time Overview

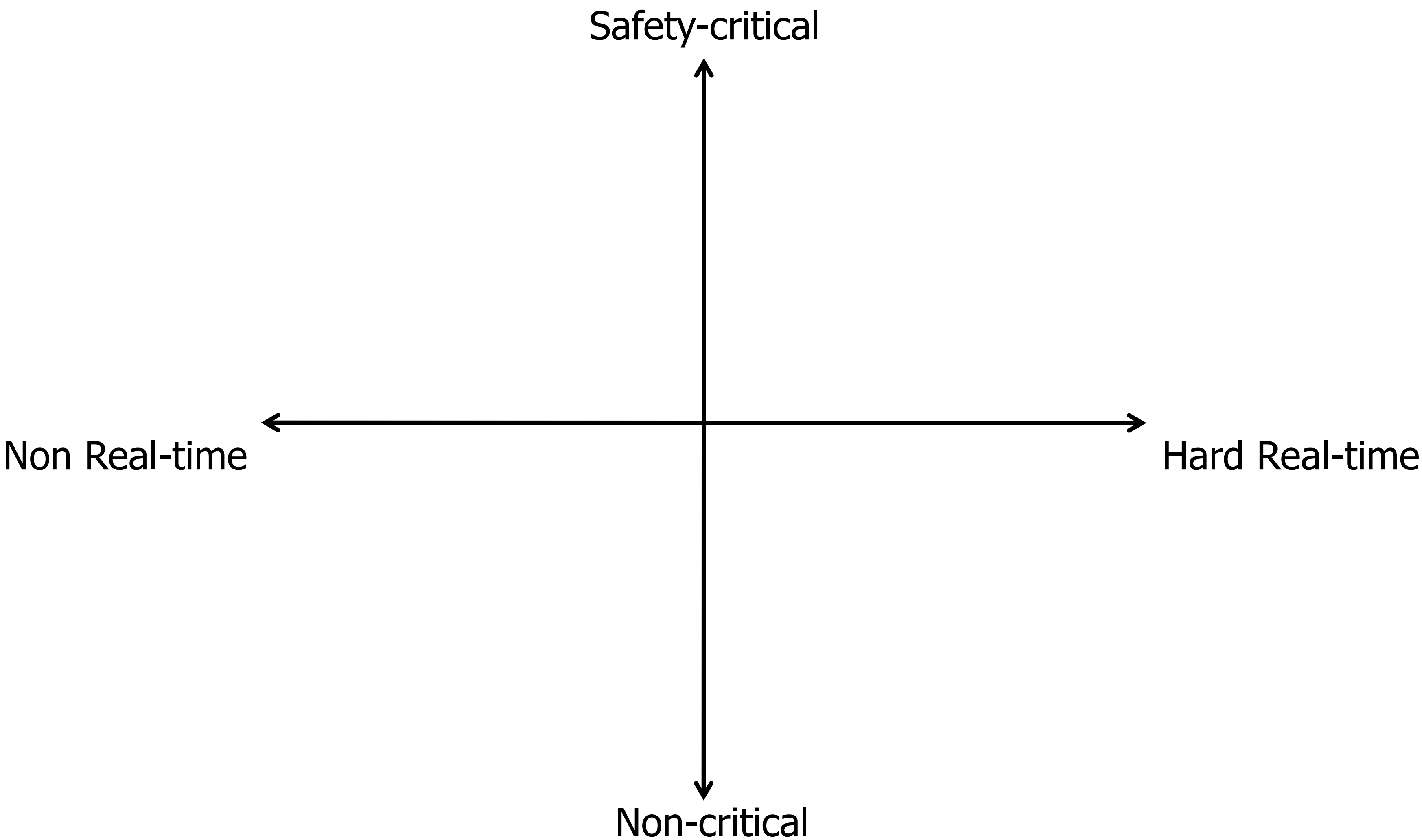


Further Reading

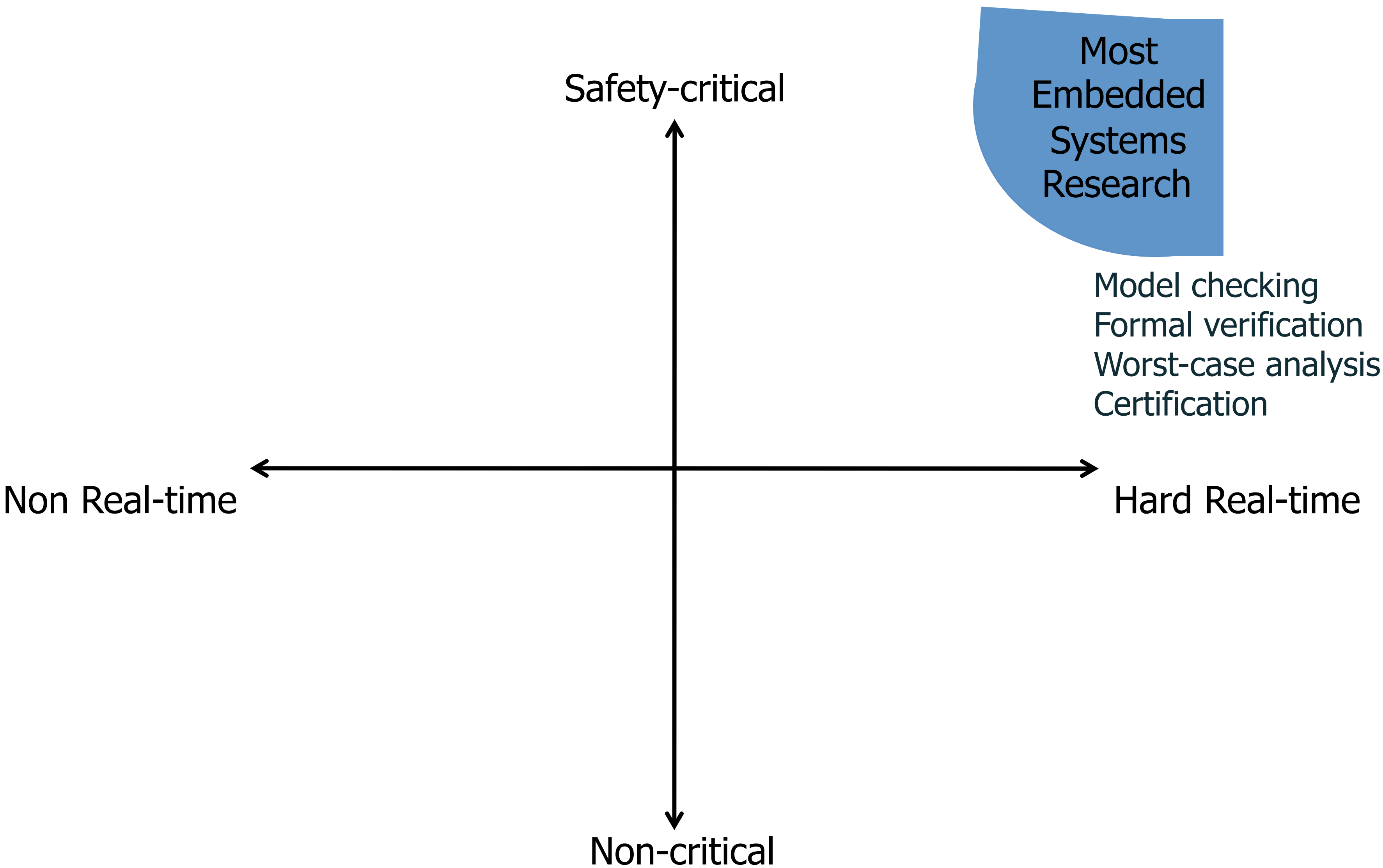
- Buttazzo, Giorgio C. Hard real-time computing systems: predictable scheduling algorithms and applications. Vol. 24. Springer, 2011.



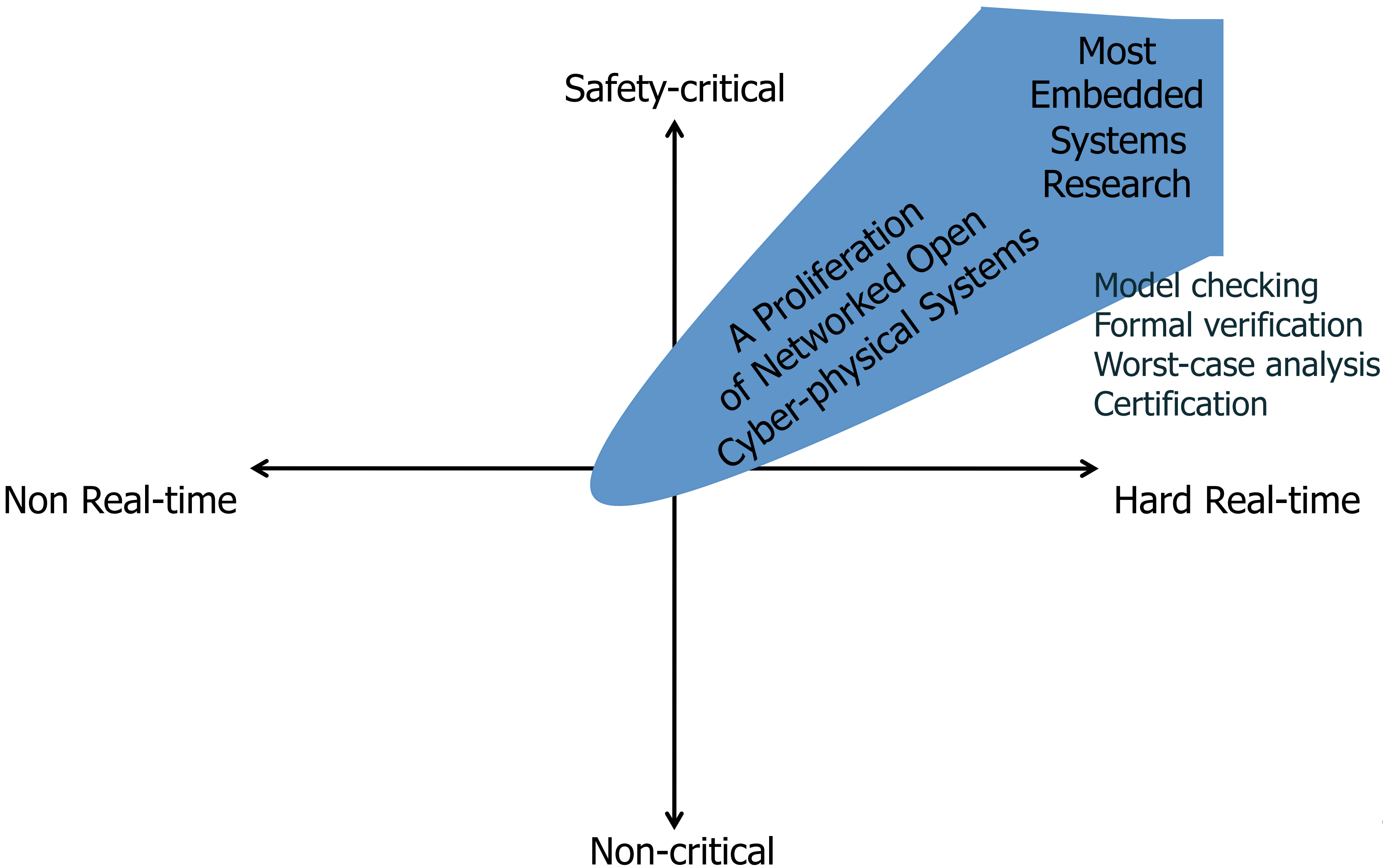
Open Cyber-Physical Systems



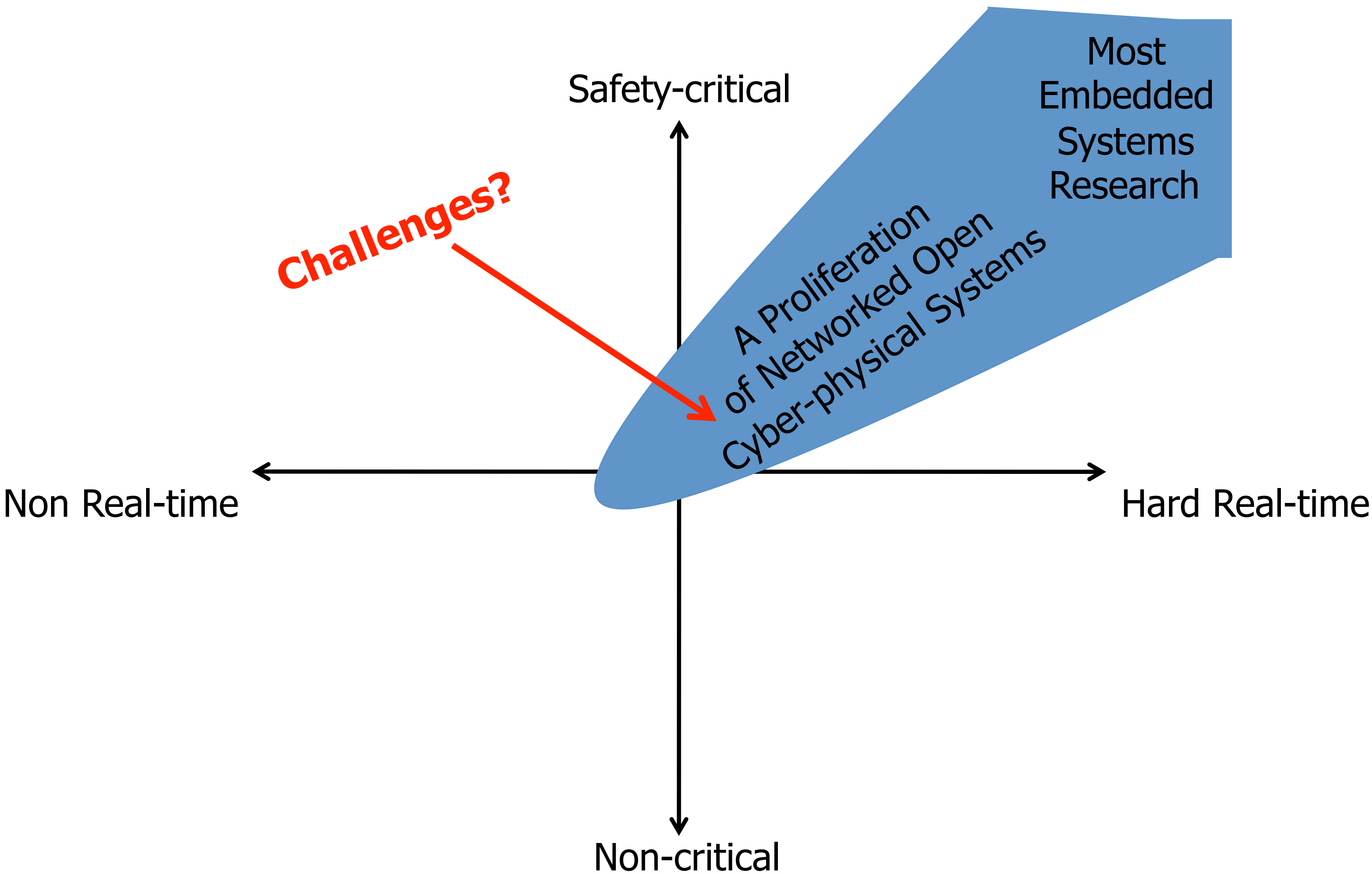
Open Cyber-Physical Systems



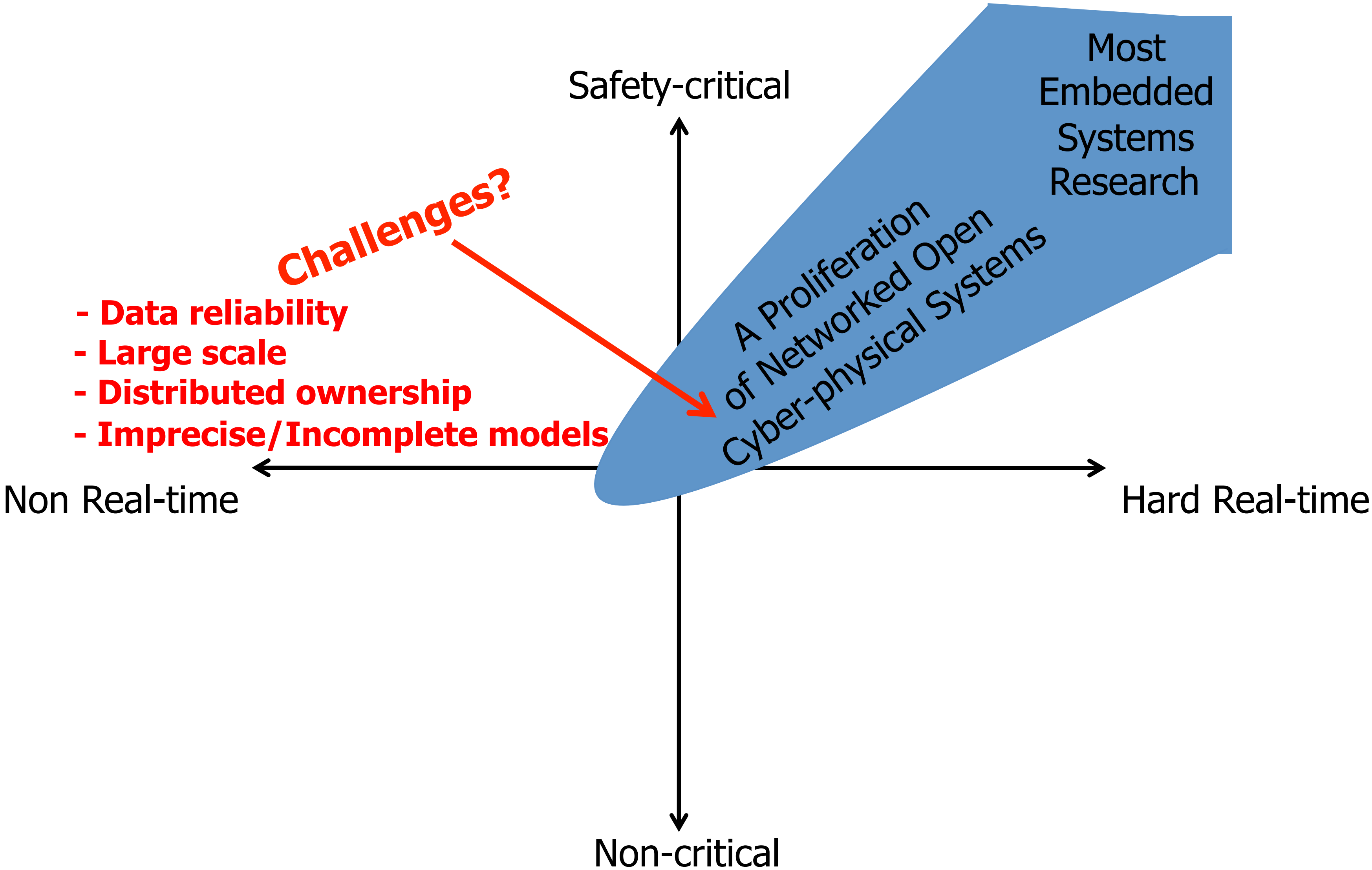
Open Cyber-Physical Systems

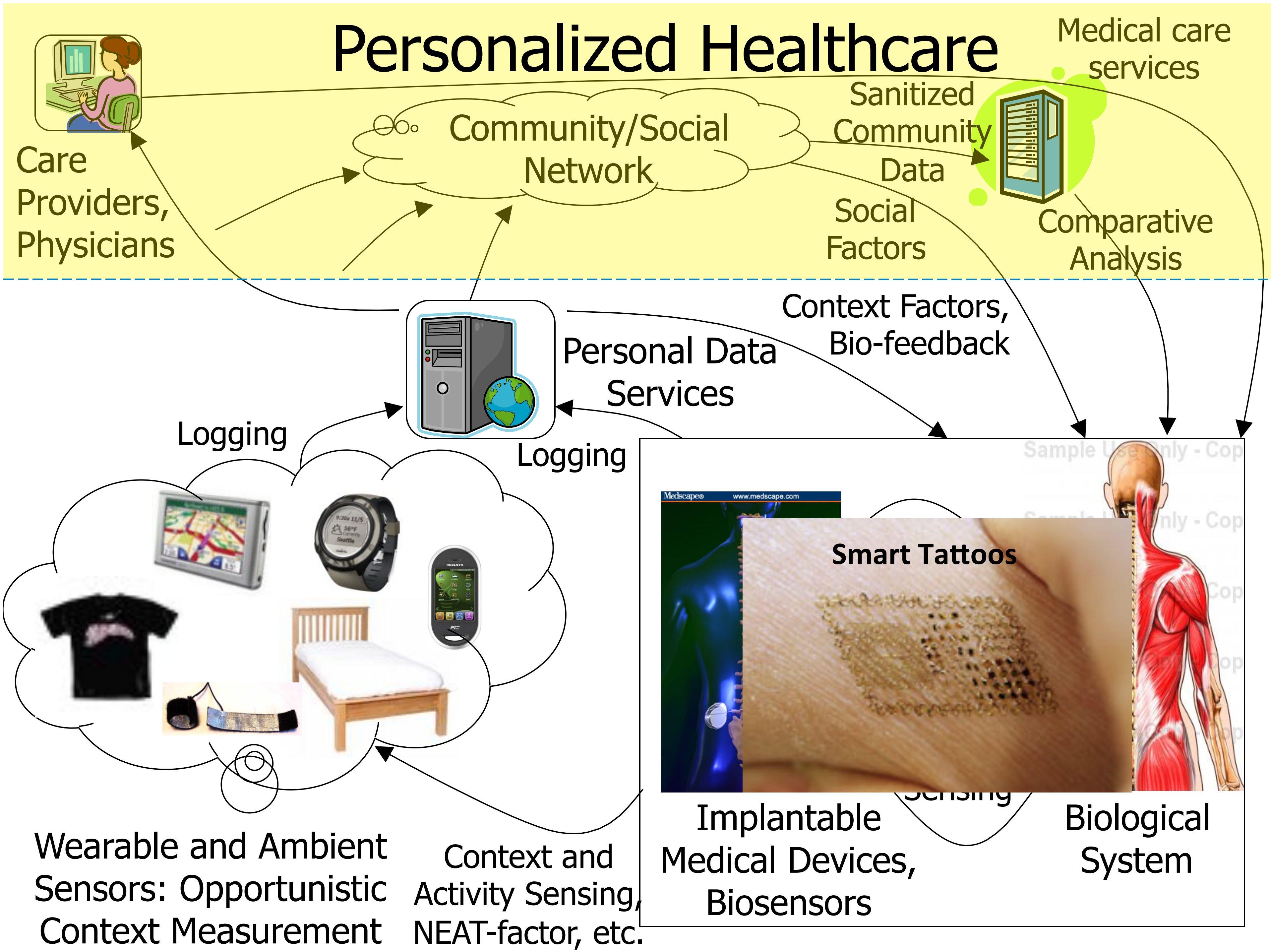


Open Cyber-Physical Systems



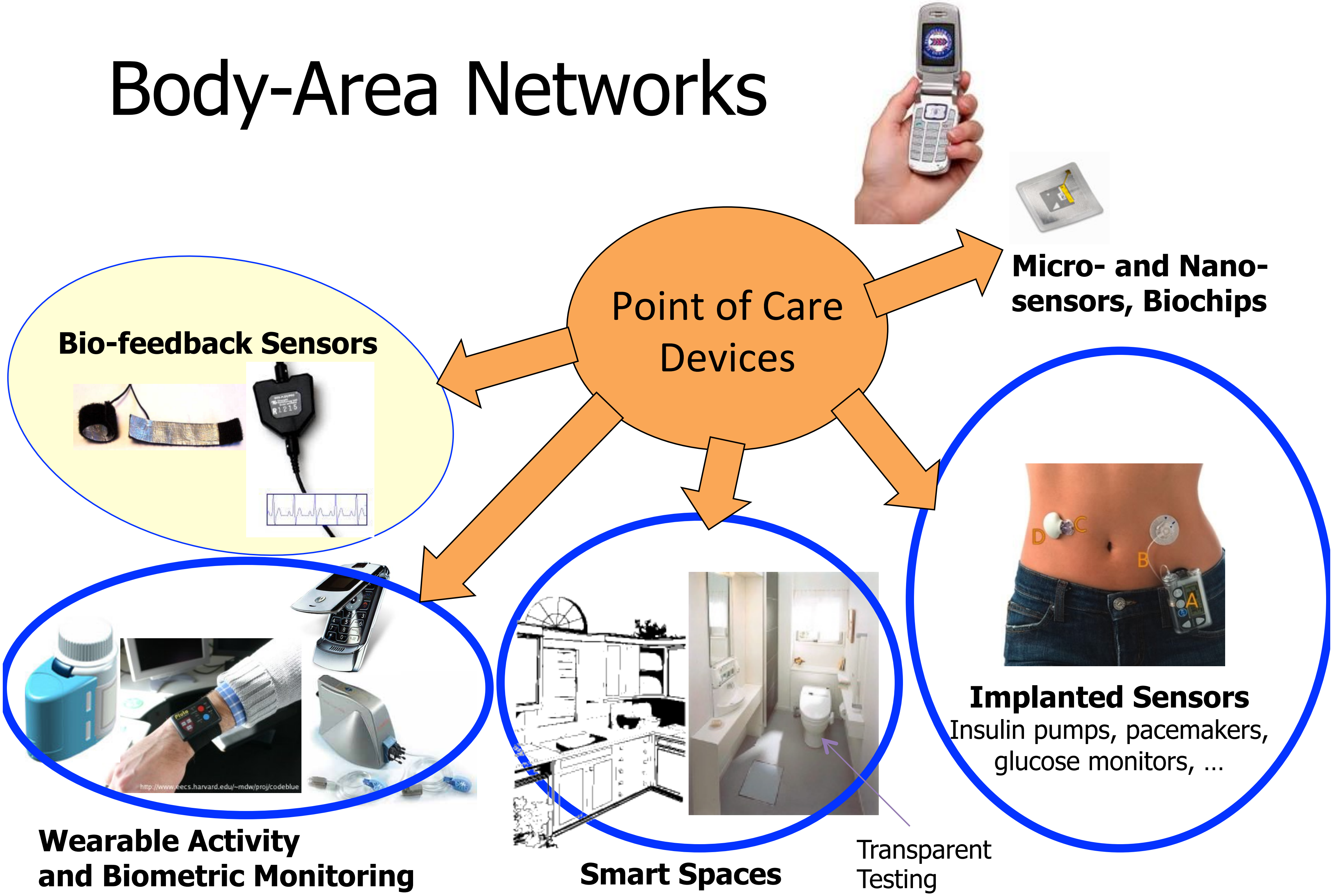
Open Cyber-Physical Systems





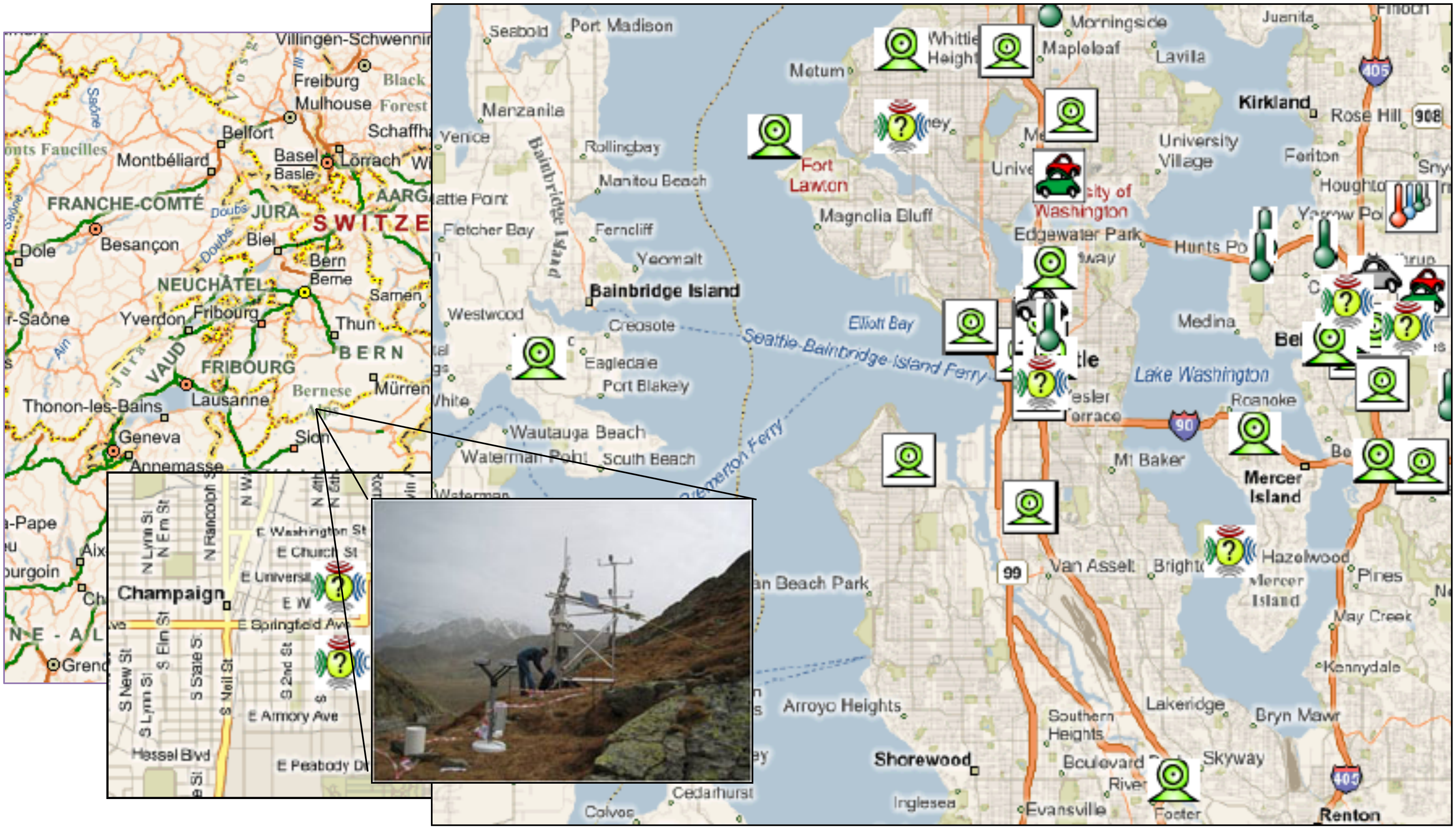
Personalized Healthcare
Medical care services
Care Providers, Physicians
Community/Social Network
Sanitized Community Data
Social Factors
Comparative Analysis
Context Factors, Bio-feedback
Personal Data Services
Logging
Logging
Smart Tattoos
Implantable Medical Devices, Biosensors
Biological System
Wearable and Ambient Sensors: Opportunistic Context Measurement
Context and Activity Sensing, NEAT-factor, etc.

Body-Area Networks

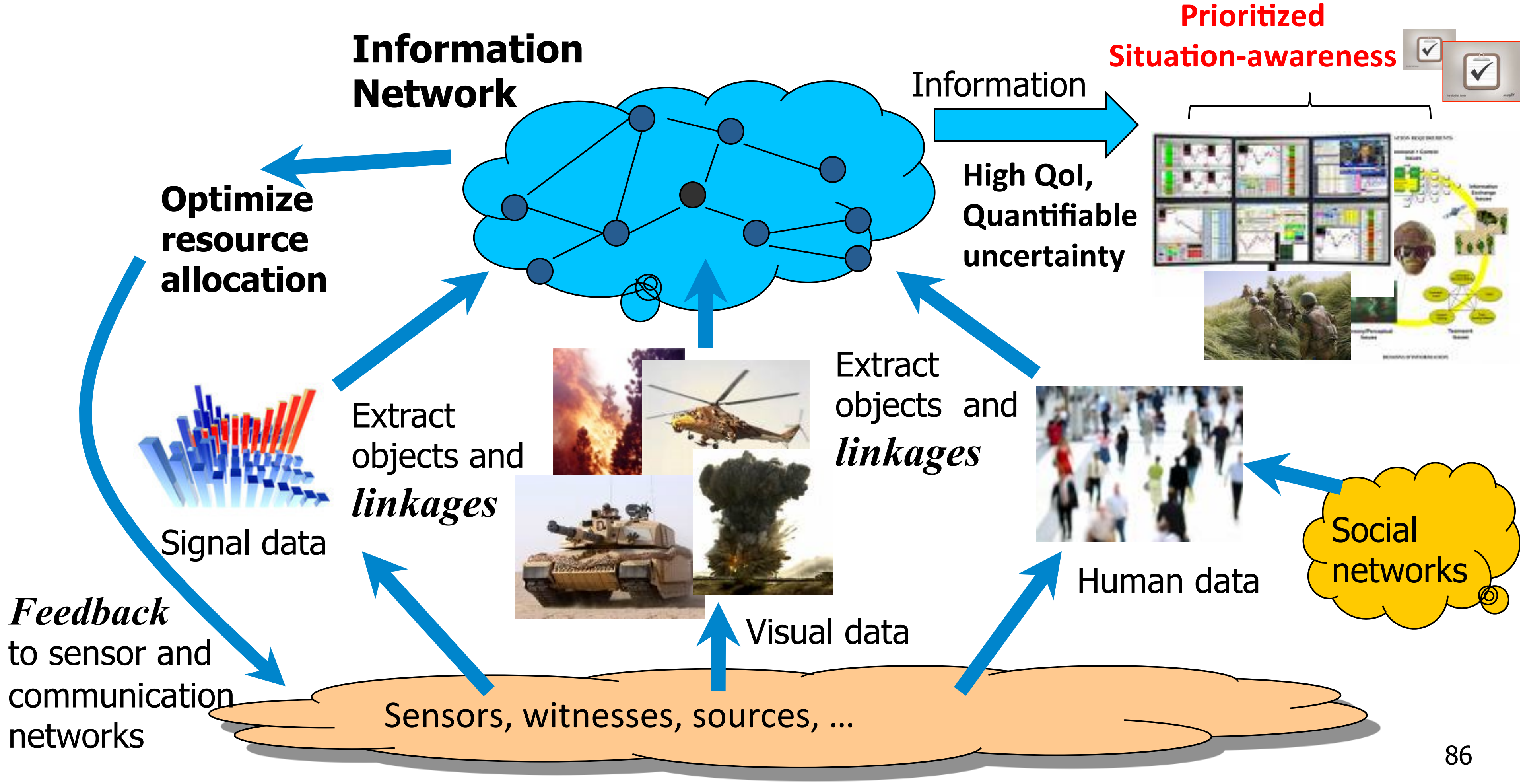


Crowd-sensing
Browsing the Physical World

Feng Zhao http://research.microsoft.com/en-us/projects/senseweb/

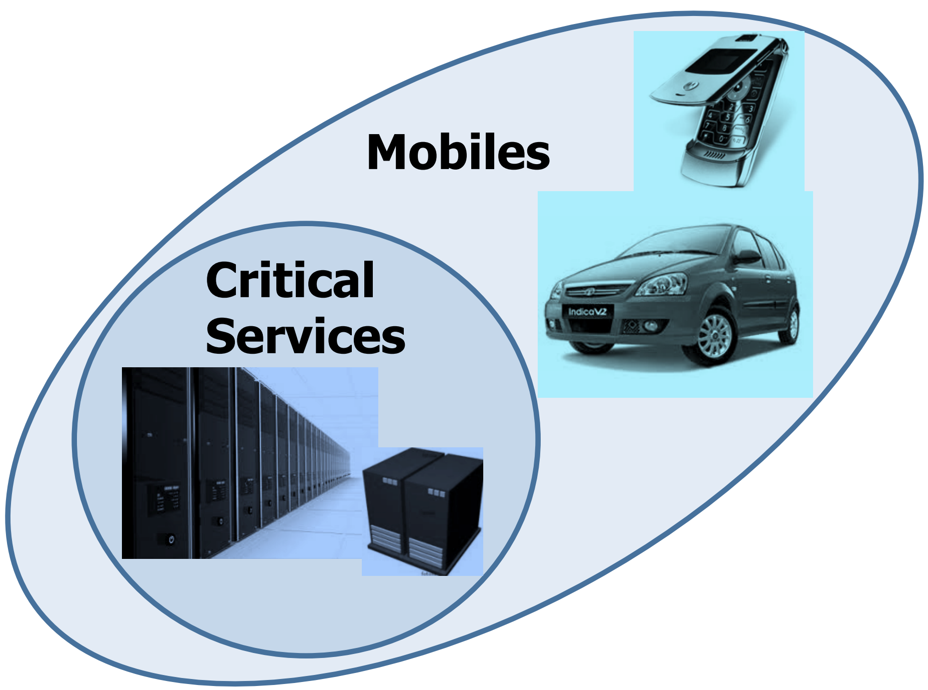


Military Situation Awareness Applications

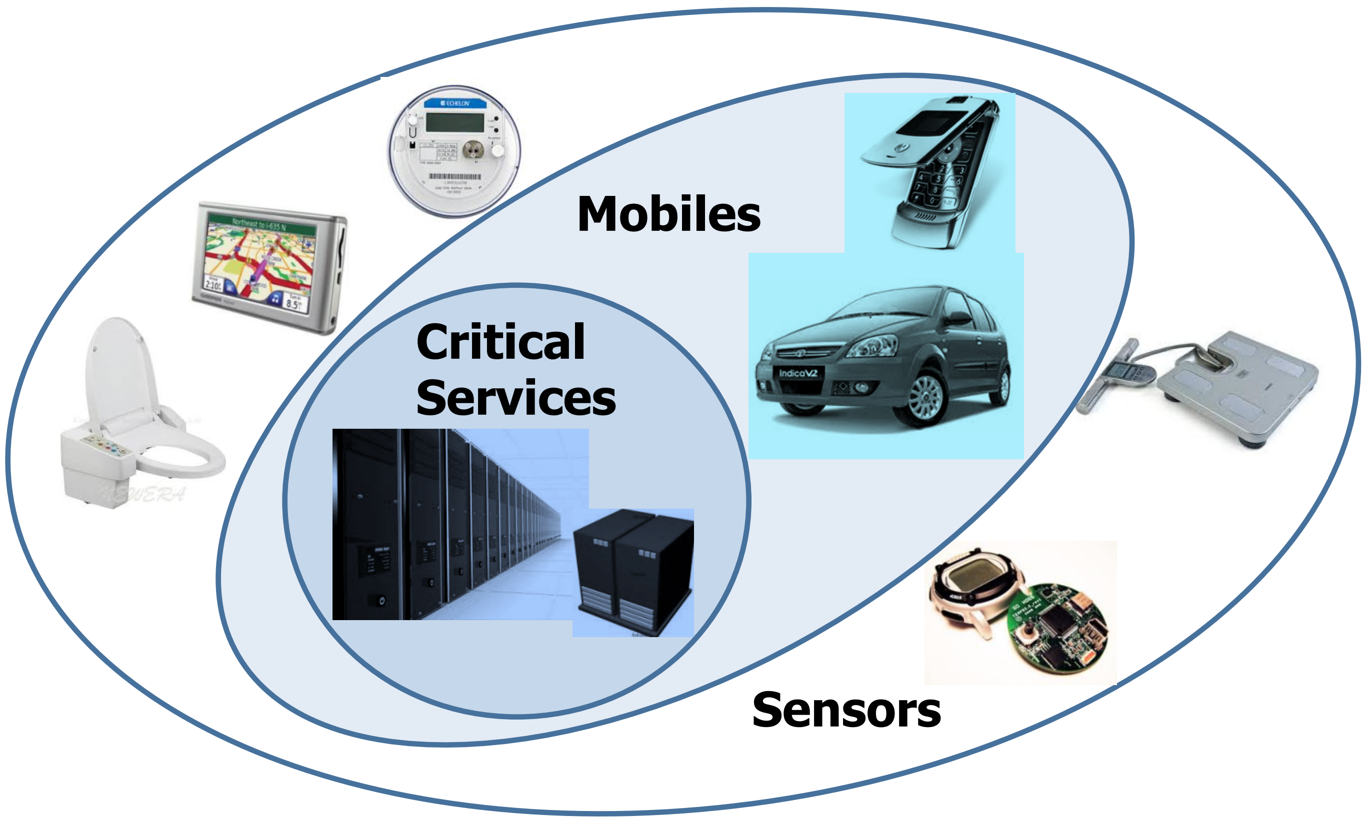


An Architecture for Open CPS

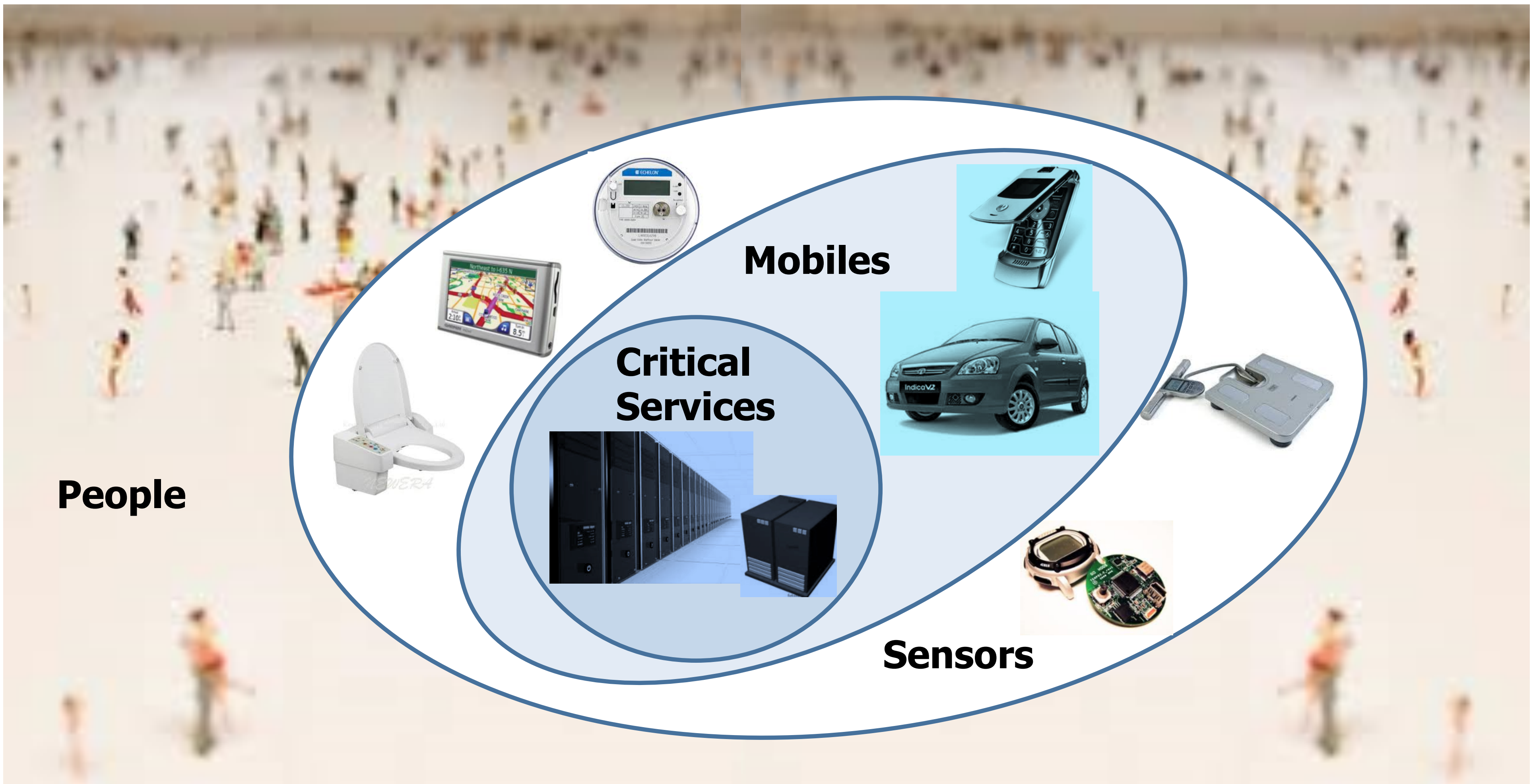
An Architecture for Open CPS



An Architecture for Open CPS



An Architecture for Open CPS



Publication Venues

- Cyber-physical systems
  - International conference on cyber-physical systems (ICCPS)
  - CPSWeek (http://www.cpsweek.org)
- Real-time systems
  - IEEE Real-time systems symposium (IEEE RTSS)
  - IEEE Real-time applications symposium (IEEE RTAS)
  - Springer Journal on Real-time Systems
- Embedded systems
  - International conference on embedded software (EmSoft)
  - ACM Transactions on Embedded Computing Systems (ACM TECS)
- Networked sensing
  - ACM Sensys (http://sensys.acm.org/2014/index.html)
  - ACM/IEEE IPSN (http://ipsn.acm.org/2015/)
  - ACM Transactions on Sensor Networks (ACM ToSN)

Publication Venues (Cont.)

- Data Analytics, Mining and Processing
  - ACM KDD Conference on Knowledge Discovery and Data Mining (http://www.kdd.org/kdd2015/)
  - IEEE International Conference on Data Mining (ICDE) (http://icdm2014.sfu.ca/home.html)
  - ACM International Conference on Web Search and Data Mining (http://www.wsdm-conference.org/2015/)
  - International World Wide Web Conference (WWW) (http://www.www2015.it/)
  - IEEE Transactions on Knowledge and Data Engineering (TKDE) (http://www.computer.org/portal/web/tkde)

Education Venues

- Nano-Tera/Artist Summer School on Embedded System Design
  - http://artist-summer-school.epfl.ch/
- Georgia Tech Summer School on Cyber Physical Systems
  - http://users.ece.gatech.edu/~magnus/CPSschool.html

The CPS Research Landscape
(An Incomplete List, Alphabetic)

- Berkeley (architecture, control, automotive, sensor networks, …):
  - http://chess.eecs.berkeley.edu/
- CMU (real-time, automotive, …):
  - http://users.ece.cmu.edu/~raj/
- Notre Dame (social sensing, CPS in social space)
  - http://www3.nd.edu/~dwang5/
- UIUC (avionics, human-centric, medical, …)
  - http://publish.illinois.edu/cpsintegrationlab/
- U. of Pennsylvania (composability, medical, …)
  - http://precise.seas.upenn.edu/
- University of Virginia (sensor networks, real-time, …)
  - http://www.cs.virginia.edu/~stankovic/rts.html
- Vanderbilt (composition, control, …)
  - http://www.isis.vanderbilt.edu/research/NES

Pointers and Readings

- "Opportunities and Obligations for Physical Computing Systems", *Computer*, Volume 38, Issue 11, November 2005, pages 23-31. (Report produced by a Workshop at the IEEE Real-Time Systems Symposium, December 2003).

  http://repository.upenn.edu/cis_papers/222/
- "High Confidence Medical Device Software and Systems (HCMDSS)" Workshop, June 2 - 3, 2005, Philadelphia, PA.

  http://rtg.cis.upenn.edu/hcmdss/index.php3
- National Workshop on "Aviation Software Systems: Design for Certifiably Dependable Systems", October 5-6, 2006, Alexandria, TX

  http://chess.eecs.berkeley.edu/hcssas/index.html
- NSF Workshop on "Cyber-Physical Systems", October 16-17, 2006, Austin, TX.

  http://varma.ece.cmu.edu/CPS
- National Workshop on "High-Confidence Software Platforms for Cyber-Physical Systems (HCSP-CPS)", November 30 - December 1, 2006, Alexandria, VA.

  http://www.isis.vanderbilt.edu/HCSP-CPS/

Pointers and Readings

- "Joint Workshop On High-Confidence Medical Devices, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability", June 25-27, 2007, Boston, MA.
  http://rtg.cis.upenn.edu/hcmdss07/index.php3
- National Workshop on "Composable and Systems Technologies for High-Confidence Cyber-Physical Systems", July 9-10, 2007, Arlington, VA.
  http://www.isis.vanderbilt.edu/CST-HCCPS/
- National Workshop on "High-Confidence Automotive Cyber-Physical Systems", Apr 3-4, 2008, Troy, MI.
  http://varma.ece.cmu.edu/Auto-CPS/
- CPSWeek, 2008-present
  http://www.cpsweek.org/
- CPS Summit, April 25, 2008, St. Louis, MO, USA.
  http://varma.ece.cmu.edu/Summit
- National Workshop on "Research on Transportation Cyber-Physical Systems: Automotive, Aviation, and Rail", November 18-20, 2008, Washington, DC (USA).
  http://www.ee.washington.edu/research/nsl/aar-cps