EECE 494
Real-Time Systems Design

Sathish Gopalakrishnan
University of British Columbia

Course information

- Course website: http://courses.ece.ubc.ca/494/
- Announcements will be posted on the website - you should check the website frequently
- For some announcements, a course mailing list will be used
  - Instructions to sign up for the mailing list are on the website

- Instructor's office: 4045 Kaiser Building (Phone: 604-827-4343)
- Instructor's office hours
  - Mondays (2 p.m. - 3 p.m.) & Thursdays (11 a.m. to 12 noon)
  - Office hours start from the week of January 11

- Teaching Assistant: Bader Na'eem Al-Ahmad

Reading material

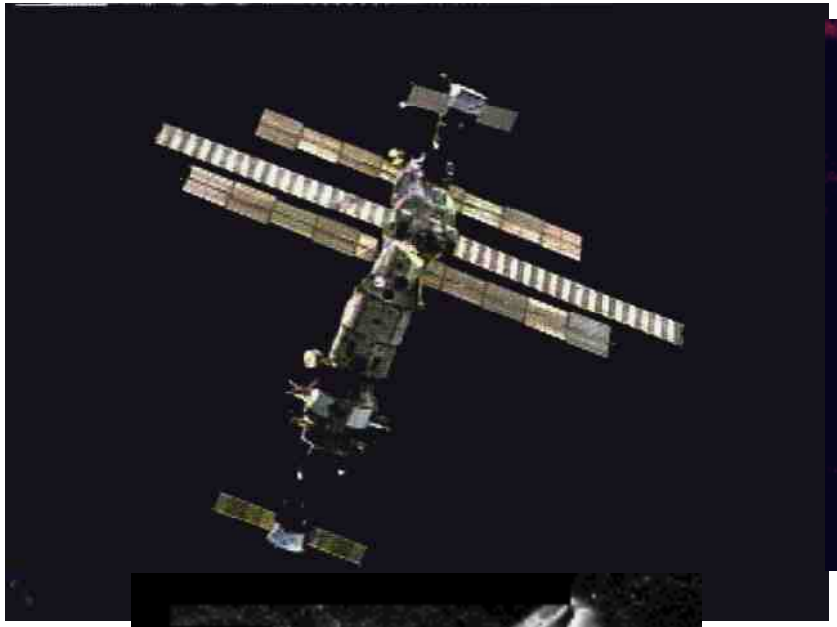
- No required textbook
- All required reading material will be available on the course website
  - You will need to use the UBC VPN to access (almost all) the readings

- Recommended material
  - Hard Real-Time Computing Systems, Giorgio Buttazzo, Springer, 2nd edition, 2006
  - Real-Time Systems, Jane Liu, Prentice Hall, 2000
  - An Introduction to Real-Time Systems, Raymond Buhr & Donald Bailey, Prentice Hall, 1998
  - Programming with POSIX Threads, David R. Butenhof, Addison Wesley, 1997
  - Computers as Components, Wayne Wolf, Morgan Kaufman, 2005

What are real-time systems?

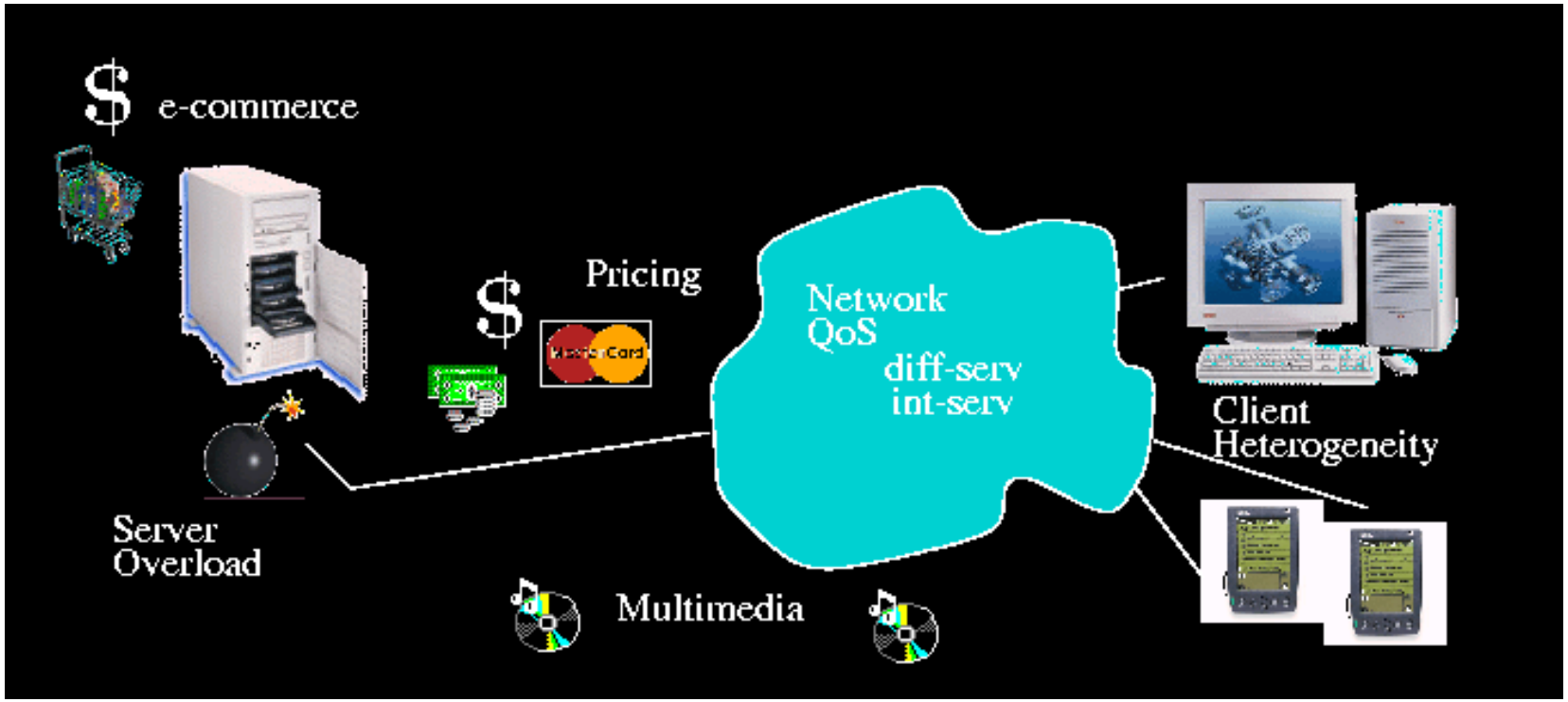
- Study of systems where the correctness of computation depends on the **timing** of the results

What are real-time systems?



Classic applications

More real-time systems

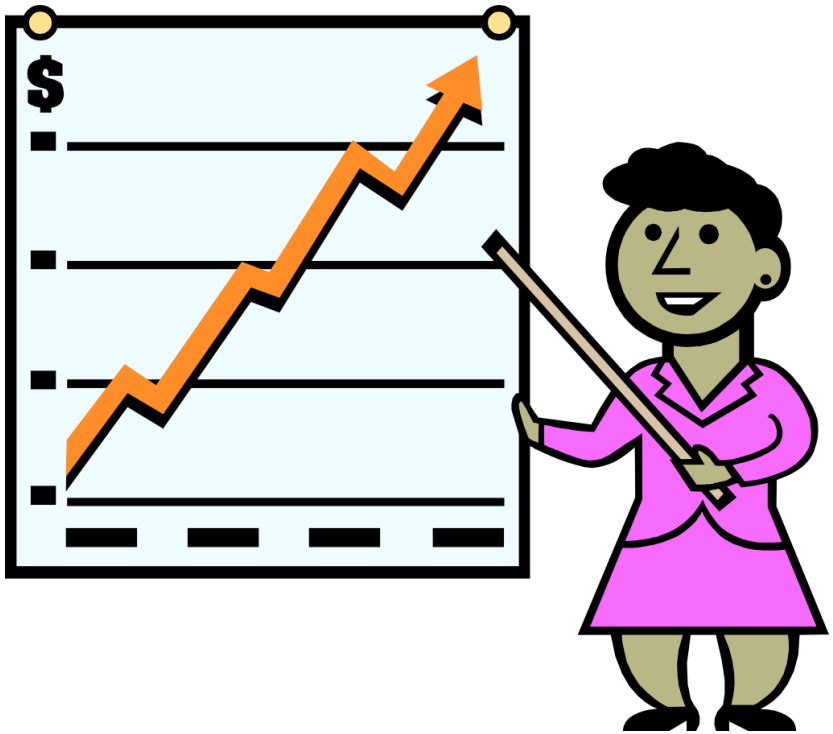


Applications of the 90s

Real-time computing today?

- Where is it going?

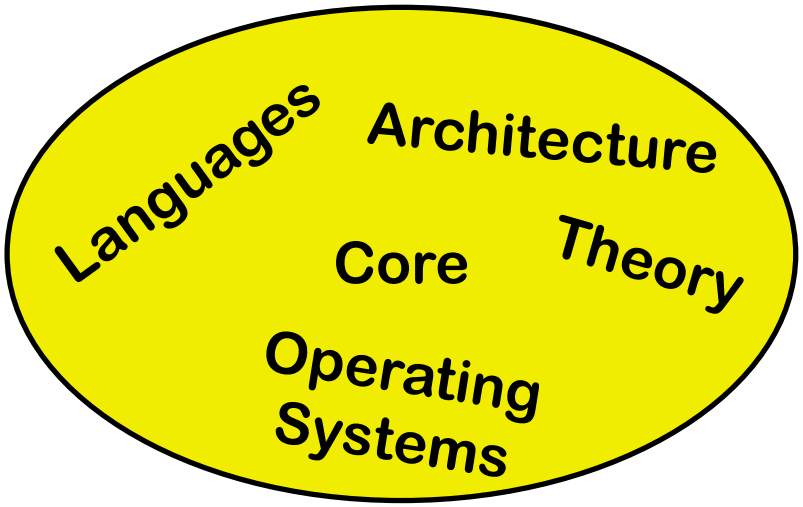
or



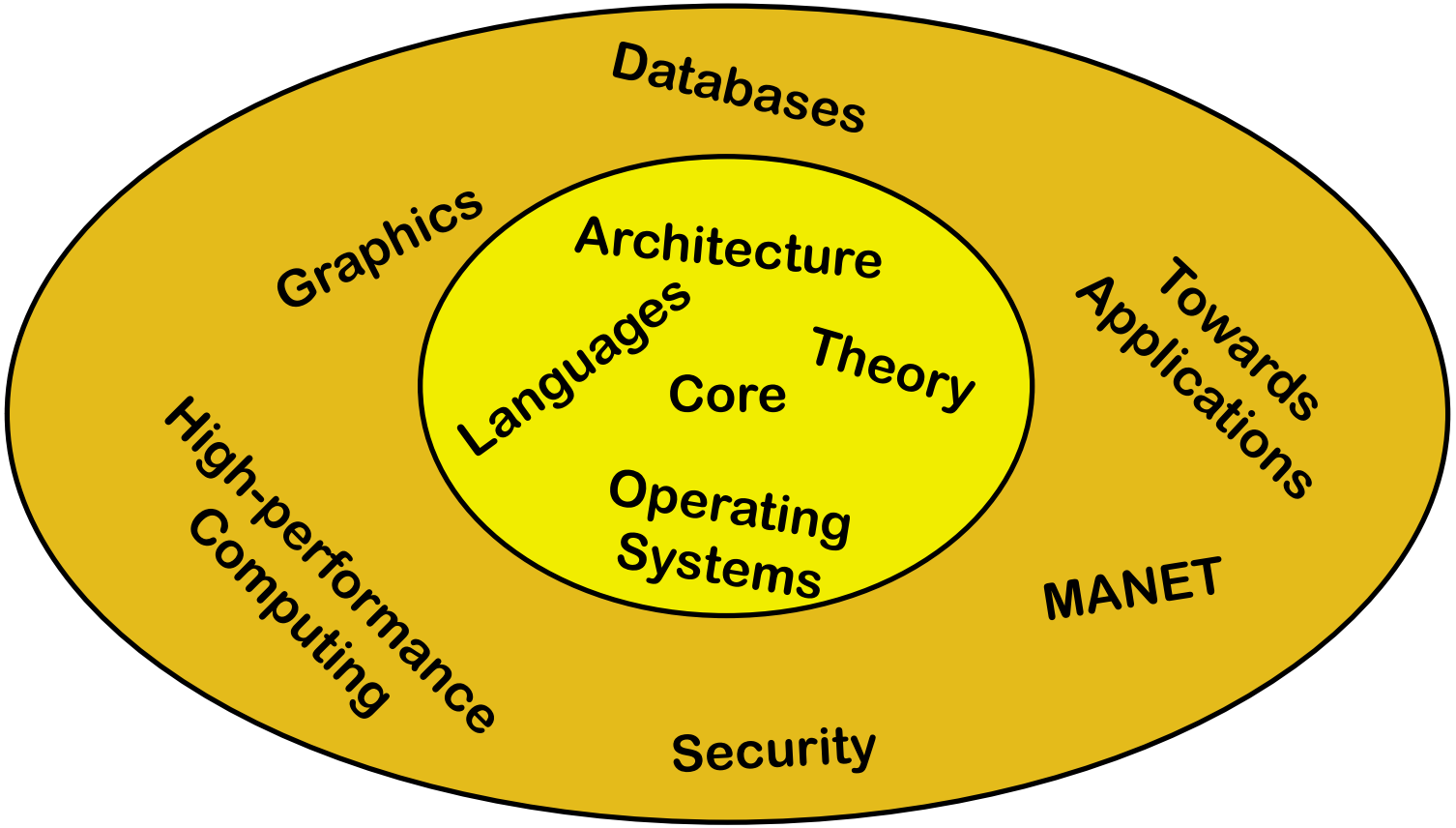
?

Where are computer systems going?

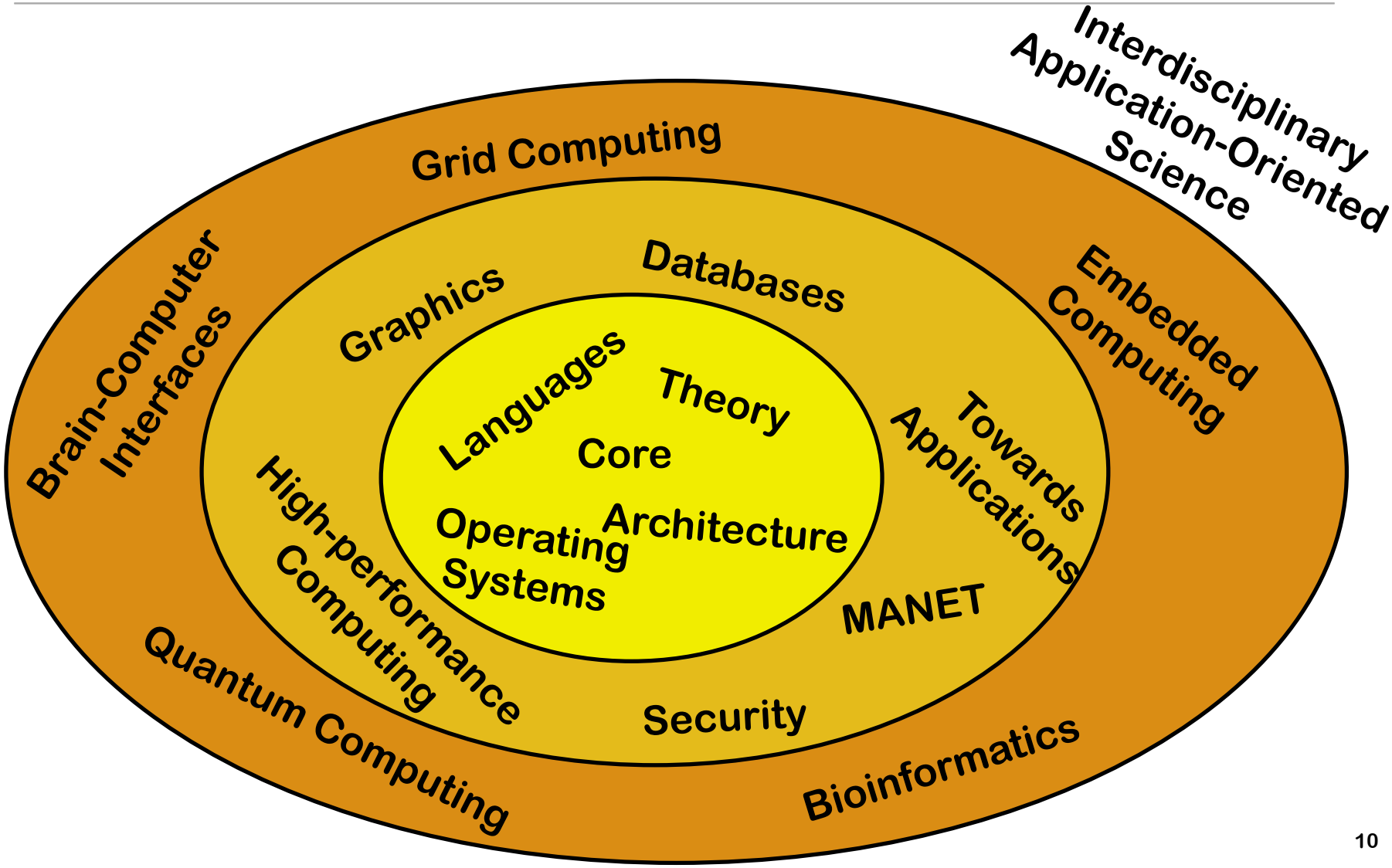
The beginning



Where are computer systems going?

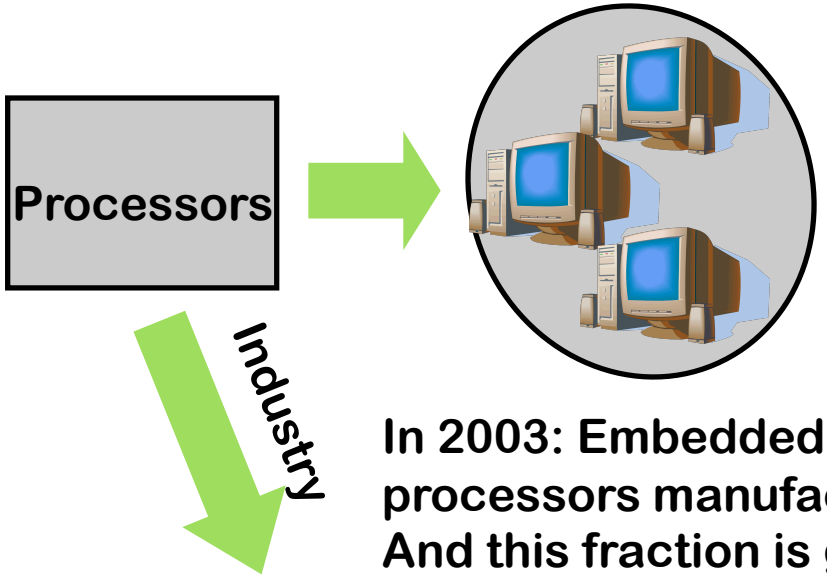


Where are computer systems going?



Real-time and embedded systems: The next frontier

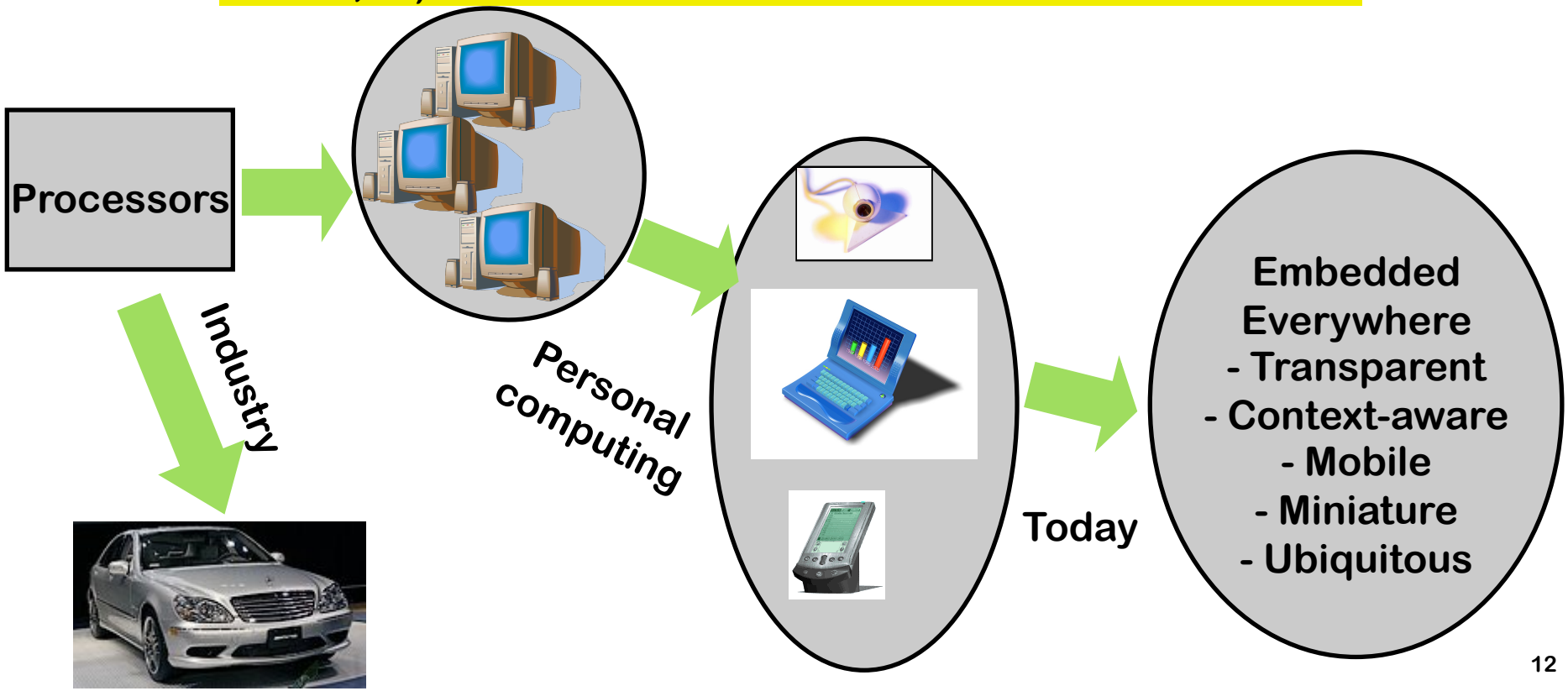
Why embedded computing?



In 2003: Embedded processors were 98% of all processors manufactured.
And this fraction is growing...

Real-time and embedded systems: The next frontier

Trend 1:
Invisible (embedded) computing, implicit interfaces
Context-aware computing (new sensors, new effectors)
Ubiquitous (instrument what we use most: attire, personal effects, …)



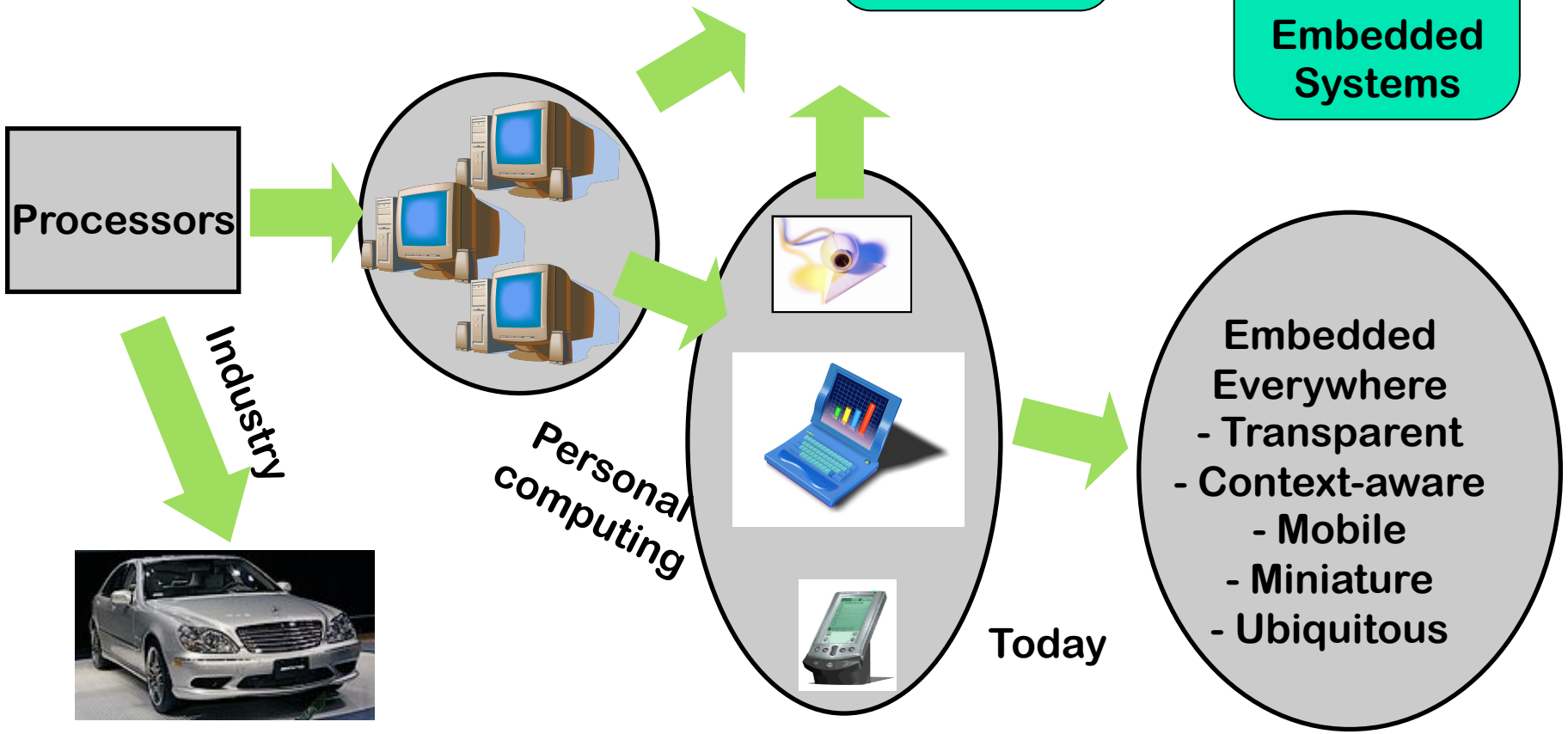
Real-time and embedded systems: The next frontier

Trend 2:
Human population/input is finite;
Future data sources are increasingly embedded devices with autonomy

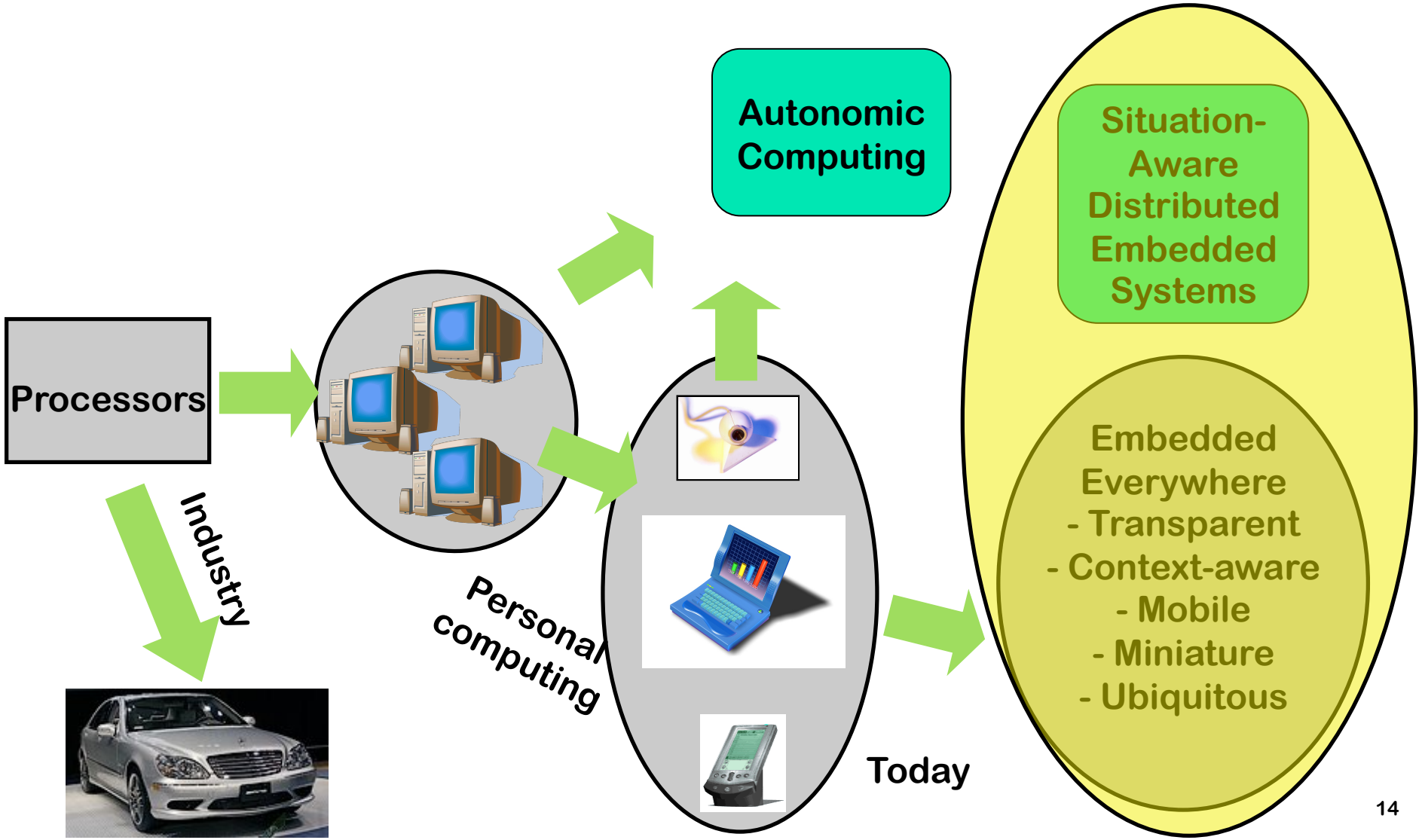
Emergency Response

Autonomic Computing

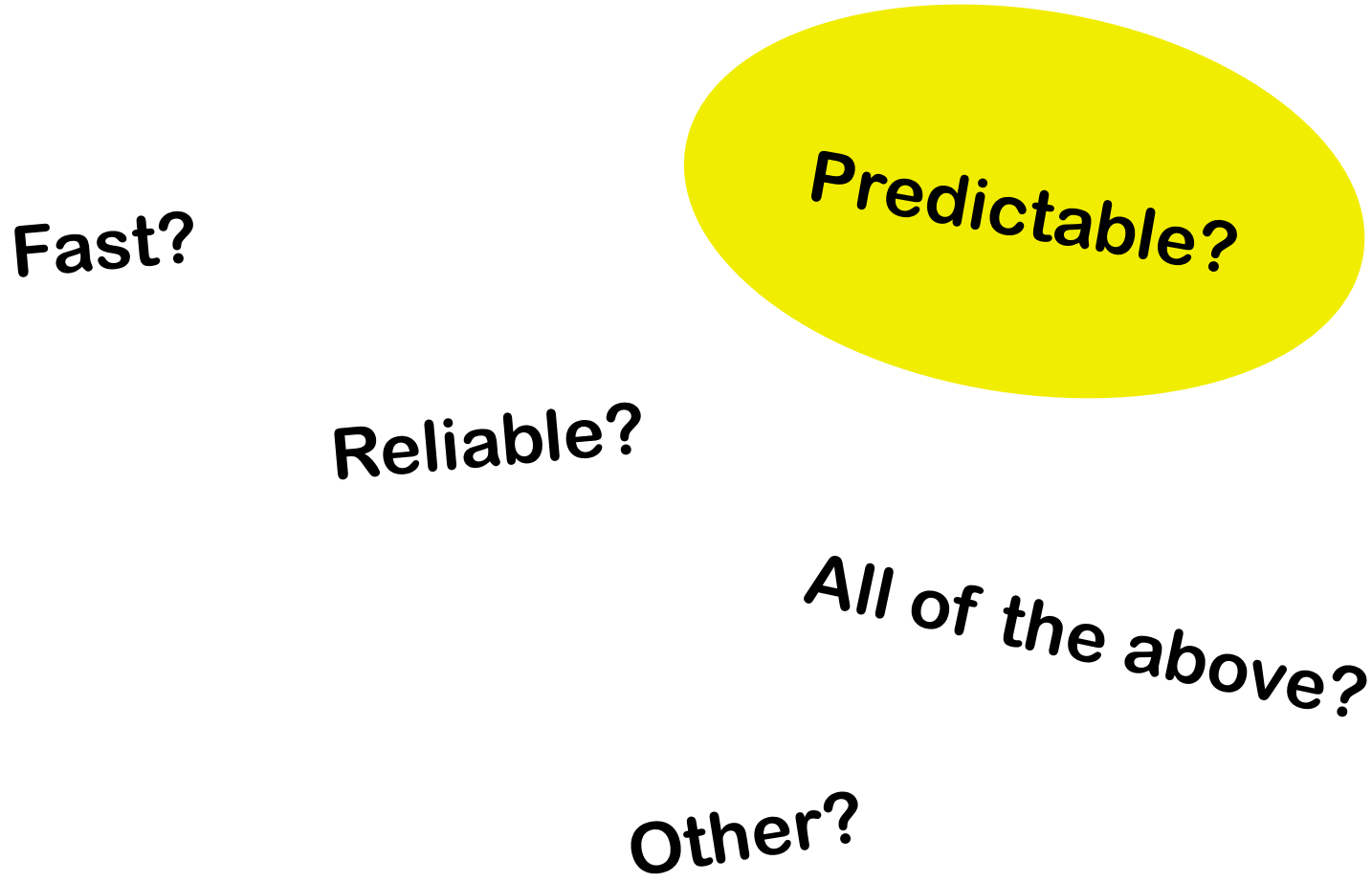
Situation-Aware Distributed Embedded Systems



Real-time and embedded systems: The next frontier



So, what does “real-time” mean?



Why predictability?

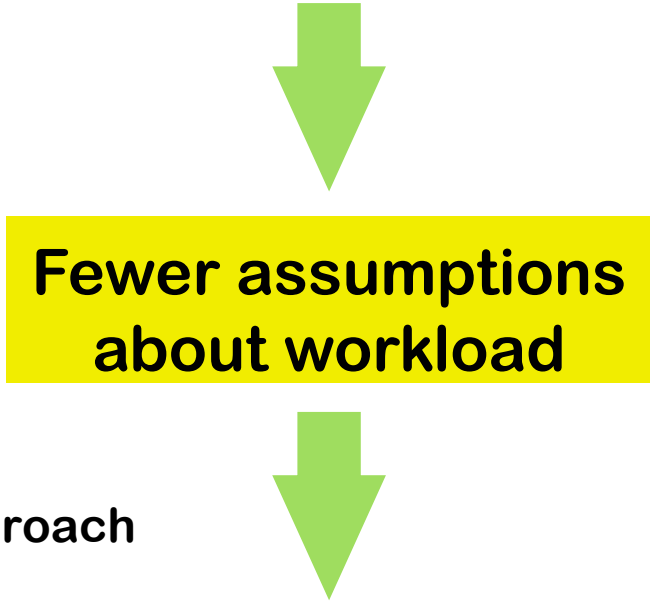
Example: Going to the airport
Which route would you choose?

Route 1: 15 min ($1 Toll)
Route 2: 5 min - 45 min, with 15 min average (free)

You pay for predictability.

The real-time computing roadmap

- Scheduling!
- Milestone 1 (1960s): Cyclic executive scheduling
  - Fixed task set
  - Known arrivals
  - Known resources
- Milestone 2 (1970s): Rate monotonic analysis
  - Tasks could be added or removed
- Milestone 3 (1980s): The Spring scheduling approach
- Milestone 4 (1990s): Quality of service
- Milestone 5 (2000s): Feedback-controlled scheduling

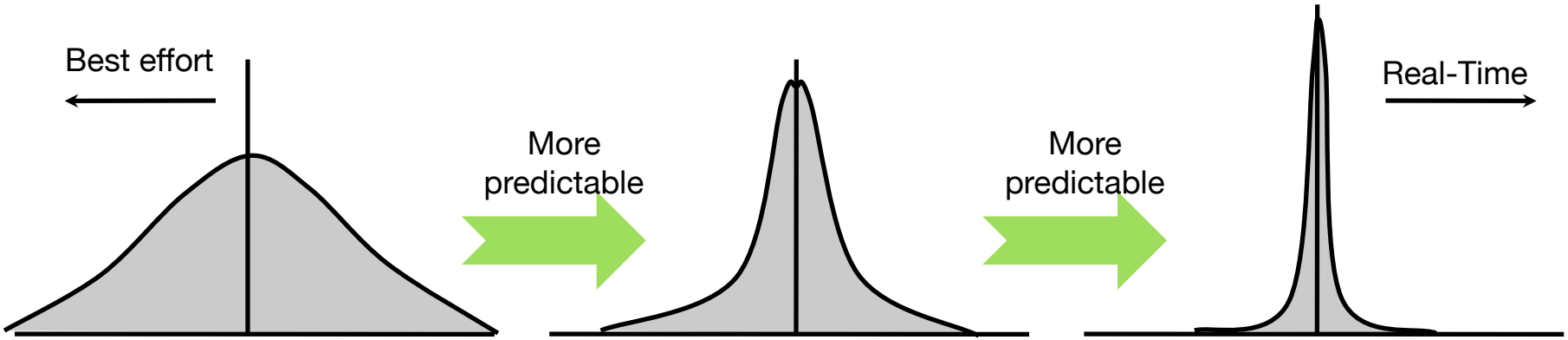


The changing scope of real-time computing

- Classical view
- Definition: correctness of computation depends on the time at which results are generated.
- Applications: embedded control systems
  - Perfect knowledge of resources
  - No conflicts of interest
- Assumptions: hard real-time
  - Meeting deadlines is very important for correctness
  - Deadlines are inflexible

New real-time definitions?

- **Systems with low variance in performance measurements**



- **Systems with guaranteed metrics in which time is either in the numerator or in the denominator**
  - **Response time = Service time + Queuing time**
  - **Utilization = Used resource units/time**
  - **Service throughput = Produced data units/time**

New assumptions

- Unknown resource requirements
- Elastic time constraints
  - Flexible periods (if any)
  - Flexible deadlines
- Adaptation capability
  - Changing algorithm version
  - Adaptive data quality
- Steady-state versus transient metrics

What we will cover in this course

- Basic concepts (tasks, threads, blocking, priorities, importance, resource partitioning, QoS, etc.)
- Periodic versus aperiodic task models and their implications
- Optimality results in real-time scheduling
- Fixed-priority and dynamic-priority aperiodic-task servers
- Blocking, synchronization, and resource access protocols
- Overload, quality of service, and system design for graceful degradation
- Hardware/software co-design for real-time embedded systems
- Engineering methodology
- Reliability and fault tolerance

Course organization

- **Assignments [55%]**
  - Programming assignments: 40%
  - Problem sheets: 10%
  - One-pagers: 5% (individual work)
- **Assignments to be completed in groups**
  - Groups (of 3 or 4 students) will be assigned - by the end of this week
  - All assignments to be submitted on the due date via electronic handin
- **Examinations [40%]**
  - Final exam: 40%
- **Class participation [5%]**
  - Attendance, completion of surveys, questions in class, …

Expected background

- Understanding of operating systems
- Interrupt handling
- Multithreaded programming
- C/C++ programming skills

- Useful habits
  - Regular reading
  - Early start on assignments
  - Learn from manuals

That's it for today!

Questions?