What Could Microkernels Learn from Monolithic Kernels (and Vice Versa)

http://d3s.mff.cuni.cz

Martin Děcký

decky@d3s.mff.cuni.cz

CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Department of Distributed and Dependable Systems

HelenOS

What a Long Title ...

Just barely missed the prize for the longest talk title of the devroom ...

What a Long Title ...

Just barely missed the prize for the longest talk title of the devroom ...

Still, the title is not saying enough.

What a Long Title ...

HelenOS

Just barely missed the prize for the longest talk title of the devroom ...

Still, the title is not saying enough.

This is an opinion piece. Feel free to disagree and let's discuss that.

Department of Distributed and Dependable Systems

Introduction

- **Martin Děcký**
  - Computer science researcher
    - Operating systems
    - Charles University in Prague
  - Co-author of **HelenOS** (since 2004)
    - Portable general-purpose microkernel multiserver operating system designed and implemented from scratch
  - User of **GNU/Linux** (since 1998)
    - Also occasional contributor

HelenOS 0.6.0

- **Released on December 21st 2014**
  - Culmination of more than 2 years of development (including GSoC '12, GSoC '14, ESA SOCIS '13)
    - GUI
    - Support for BeagleBoard, BeagleBone, Raspberry Pi, MIPS Malta, LEON3
    - ext4 as default root file system, UDF support
    - IPv6 support, auto-configuration
    - Audio support (including Intel HD Audio)
    - Miscellaneous (guard pages, device drivers, telnet, VNC)

About the Premise ...

Could the microkernel systems really learn something from the monolithic systems?

Microkernels vs. Monolithic Kernels

GDI

NOD

Microkernels vs. Monolithic Kernels

Microkernels

Monolithic Kernels

Department of Distributed and Dependable Systems

Microkernels vs. Monolithic Kernels

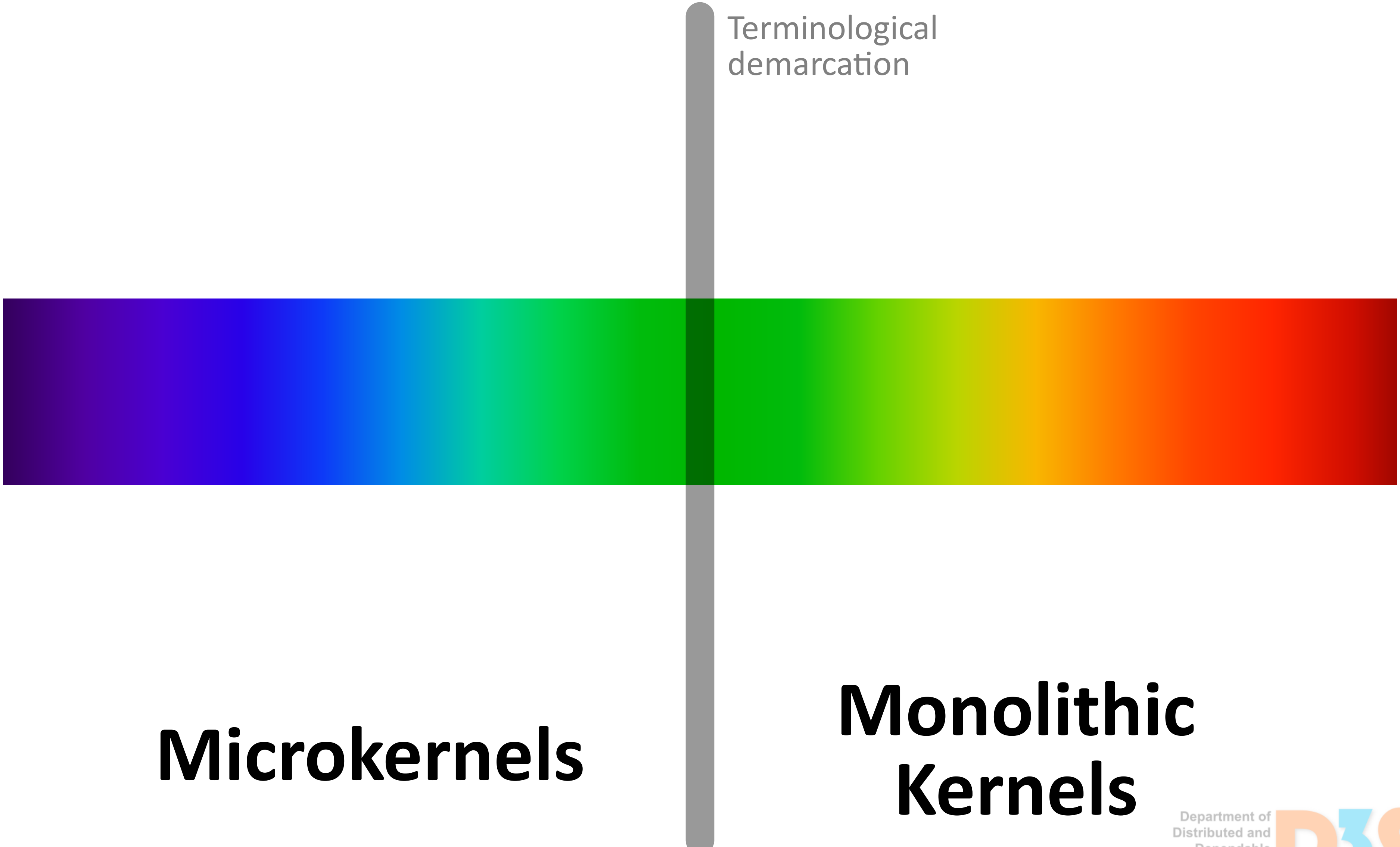
The Iron Curtain

Microkernels

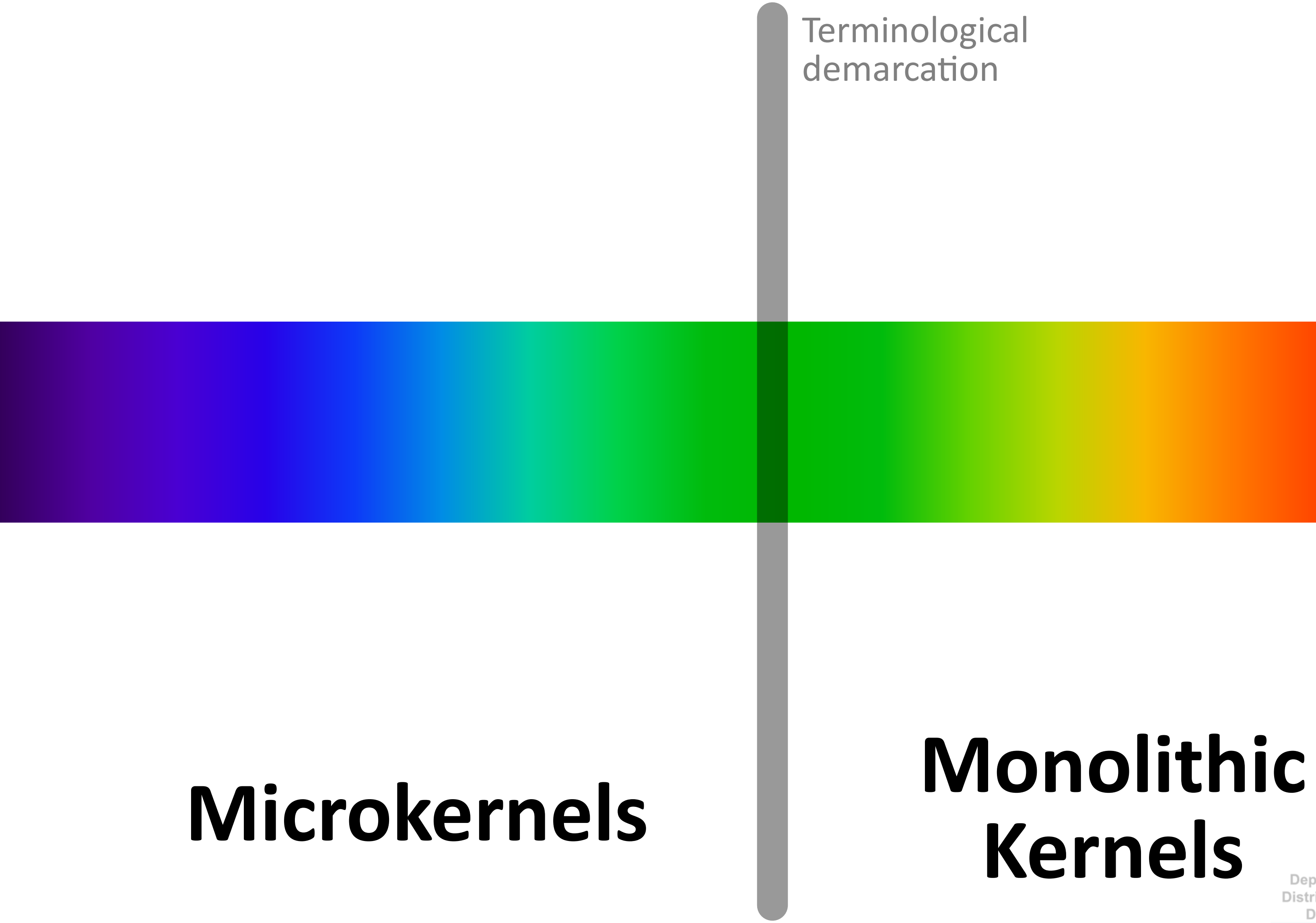
Monolithic Kernels

Department of Distributed and Dependable Systems

Microkernels & Monolithic Kernels

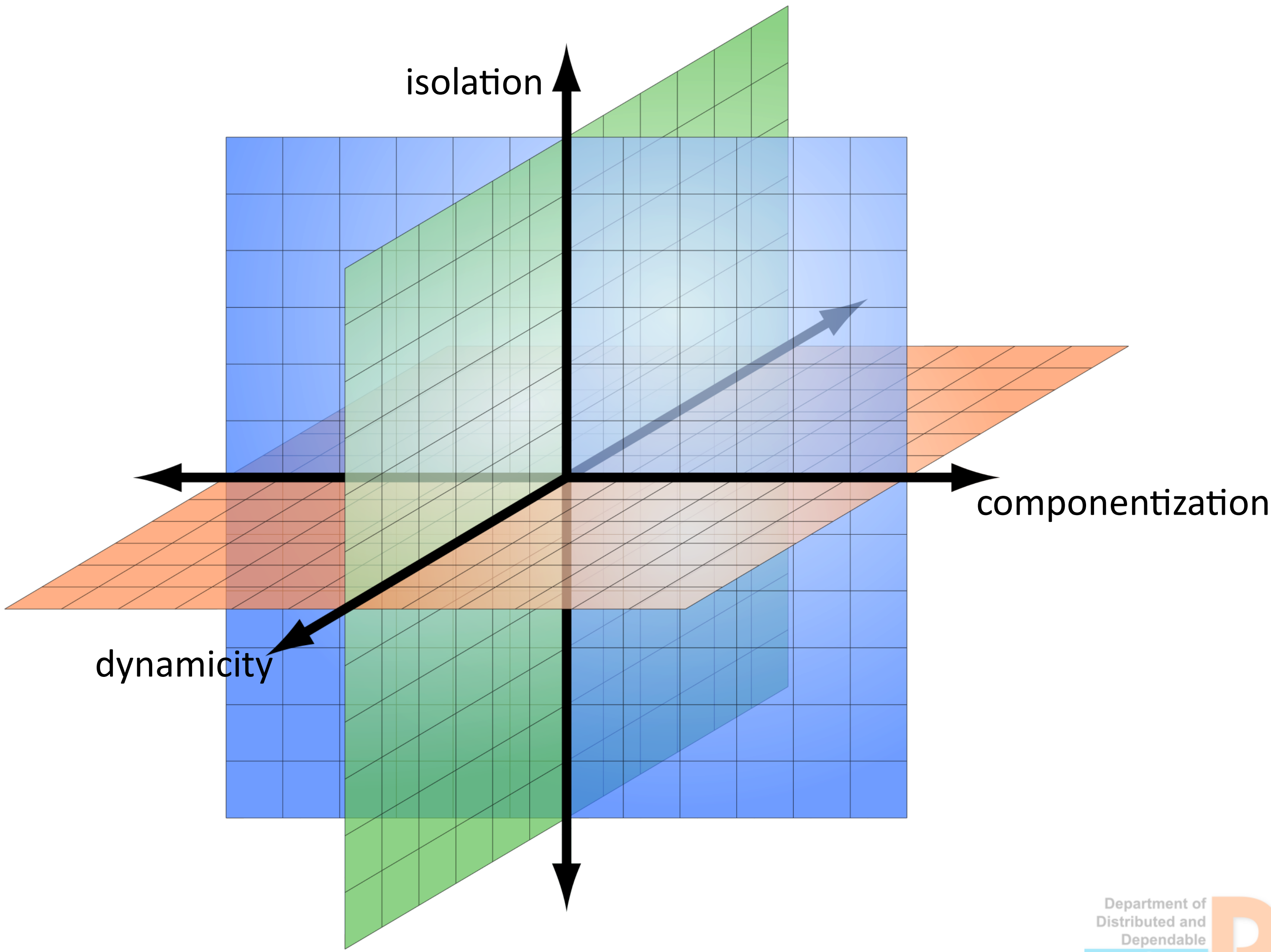


Microkernels & Monolithic Kernels



Department of Distributed and Dependable Systems

Microkernels & Monolithic Kernels



Meet the New Kid on the Block

- **OS^v**
  - Operating system for virtual machines
  - Only drivers for virtual and paravirtual devices
    - Real devices can be supported via rump kernels
  - Slim API necessary to run POSIX applications and a JVM
    - In kernel mode
    - Single user, single process, single image, single address space


Atlantis
United States
NASA

Microkernels → Monolithic Kernels

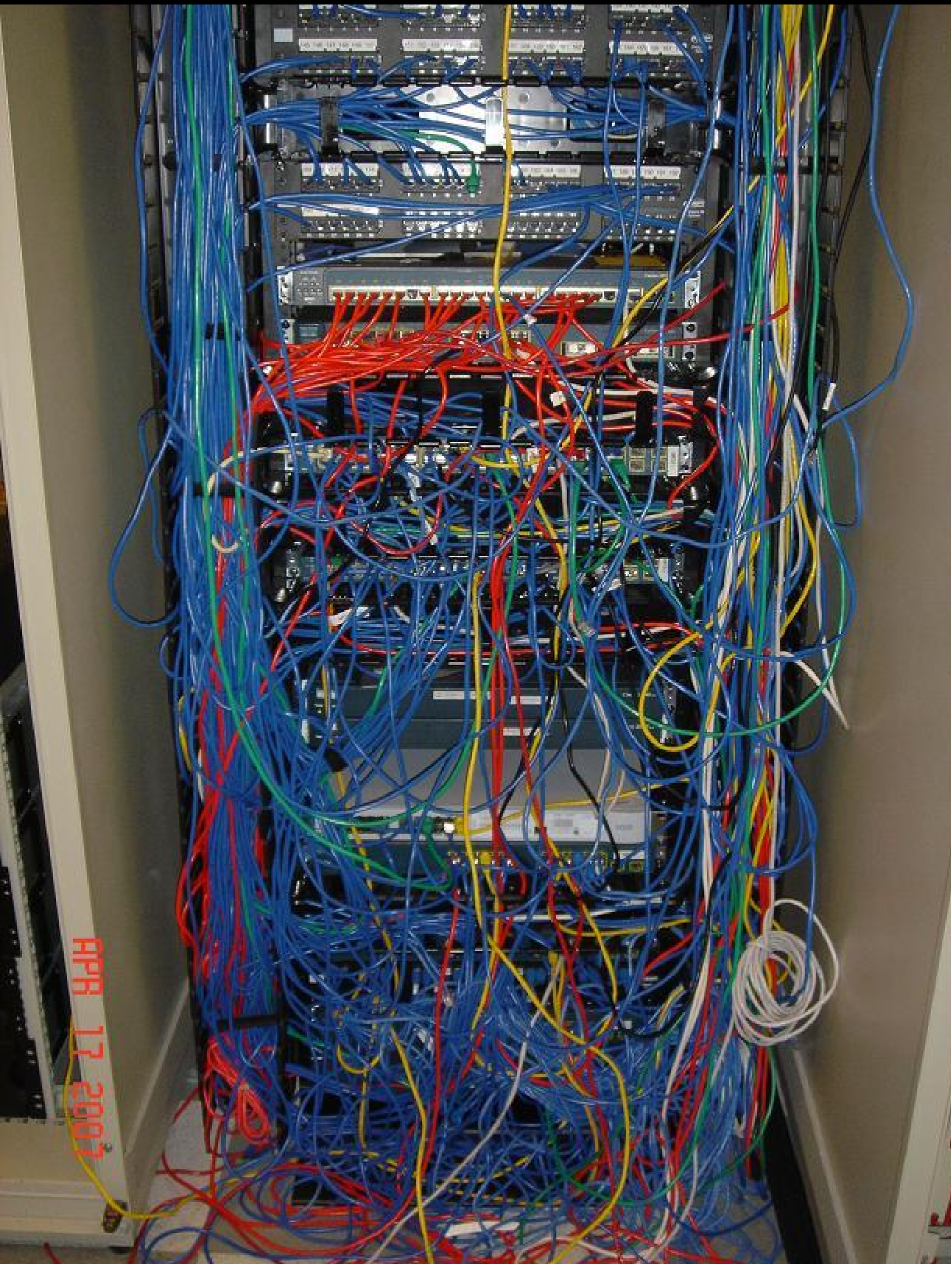
- More power for user space
  - Gradually moving some drivers to user space (where it makes sense)
    - FUSE, libusb, networking stack, etc.
    - Performance might not be actually very problematic
      - The only trouble is memory copying, but we can avoid it
      - Caching can be still done in a “monolithic” way
  - We can see the opposing trend in many cases
    - KMS/DRM
    - Due to removing duplicities, not due to the technical limitations

Microkernels → Monolithic Kernels (2)

- **Explicit architecture**
  - Software architecture (components) can be explicitly visible in the code
    - Compared to metadata (naming conventions, etc.) that usually disappear after compilation
    - Common objection: *We don't want to restrict the code*
      - Don't we really?
      - Passing pointers anywhere is just not necessary
      - Pointers are not important, the data are

Microkernels → Monolithic Kernels (3)

- **Bootstrap is not run-time execution**
  - An operating system is **not** an “algorithm”
    - It has been running forever
    - It will be running forever
    - The “initial state” is indistinguishable from the “idle state”
    - There is no “terminal state”
  - Designing the same code paths for both the bootstrap, termination and run-time execution harms all of them


APR 17 2007


MAY 6 2007

Monolithic Kernels → Microkernels

- **Smart algorithms and data structures**
  - Surprisingly enough, groundbreaking ideas are usually implemented and evaluated in the monolithic kernels first
    - Copy-on-Write
    - Object allocator
    - Read-Copy-Update
    - Namespaces
    - Global resources (single-system image)

Monolithic Kernels → Microkernels

- **Smart algorithms and data structures (cont.)**
  - Advanced scheduling
    - Earliest Deadline First
    - Multi-level scheduling
  - Dynamic tracing and instrumentation
  - Support for Hardware Transactional Memory
  - Security features
    - Address space layout randomization
    - Extended Fault Isolation (XFI)

Monolithic Kernels → Microkernels (2)

- **Scalability**
  - Monolithic systems are shown to scale to thousands of CPUs
  - Surprisingly, many microkernel systems still target only uniprocessor machines
  - Surprisingly enough, monolithic systems have been successfully scaled down for embedded devices
    - Sure, a monolith is not necessarily a huge object

Monolithic Kernels → Microkernels (3)

HelenOS

- **Portability**
  - Most monolithic systems are (surprisingly) portable
    - Even with respect to the execution environment
  - Many microkernel systems are (surprisingly) hard to port
    - Usually a proper hardware abstraction layer is missing
    - This leads us to ...

Department of Distributed and Dependable Systems

Microkernel Non-Goals

Restarting

- **Dependability through restarting servers**
  - A demonstration of the powerfulness of the isolation
  - Microkernel design by itself does almost nothing for managing the internal state of the servers
    - Servers are rarely stateless
    - The logical state is rarely limited to a single server
    - Restarting of a server rarely solves the root cause of the failure

Binary Size as a Measure of Quality

HelenOS

- **Micro means small, right?**
  - Is a microkernel with a size of 50 KB a better kernel than a microkernel with a size of 150 KB?
  - What about 49 KB?
  - Measuring things such as cyclomatic complexity might be more reasonable, but at best there is a **correlation** (not causation) between the value and the probability of bugs

Department of Distributed and Dependable Systems

Brain-Dead Microkernel

HelenOS

- **Using trivial algorithms as a safeguard**
  - Again, there is a correlation **at best**
  - Trivial bugs in trivial code (with non-trivial consequences)
  - Simplicity is desirable, but not without consideration
  - We know many sophisticated ways how to make sure a complex piece of code is correct (e.g. formal verification)

Department of Distributed and Dependable Systems

Conclusion

- **Remember the Amdahl Law**
  - *Optimize for the common case*
  - The common case is context-dependent
  - Acknowledge other people's common cases
- **Avoid black-or-white vision**
- **Acknowledge other people's ideas**
- **Know your goals and non-goals**
  - Avoid misguided goals