- **Exokernel: an operating system architecture for application-level resource management. Dawson R. Engler, M. Frans Kaashoek, and James O'Toole Jr, SOSP '95**
  - Introduces a new kernel which outsources policy *and* some mechanisms to applications
  - Applications are made up of OS libraries (that interact with a non-portable exokernel). OS does not trust OS libraries. Hence, applications can trust the application libraries.

OS structure

- Monolithic kernel
- Microkernel
- Exokernel

- Question: What is the problem being solved here?
  - Why do applications want to control the interface?

Traditional OS

- Traditional Operating Systems manage lower level details and perform resource allocation. They are designed to be portable. Hence, they define interfaces (e.g. system calls). Abstractions add overhead and so the OS is general purpose but not optimized for a specific scenario
  - Data bases - need access to hard drive
  - Movie player – need access to display, DVD drive, CPU
  - Games – need access to CPU, display
- Standard interfaces are good for general purpose
  - Games/database … developers work closely with OS developer to create custom tuned versions

Exokernel

- OS provides
  - secure binding of resources – only the OS can assign resources, h/w enforces that you do not access what you should have access to
    - What about page tables?
    - What about filter code to separate network packets?
    - Can you use network interface as a IPV4 & IPV6 host?
      - Can two libraries use the h/w in mutually incompatible ways?
  - Visible resource revocation
    - Work with the library for TLB miss notification, page fault etc.
    - Forcefully react to libraries that do not respond fast
  - Abort protocol

Exokernel

- What is the tradeoff?
  - Benefits
  - Overhead?

- How does this compare to
  - Unstructured OS like DOS
  - Embedded systems
  - VM
  - VM with heavy hardware assist

Role of people on OS development

- Hardware developers (Intel)
- OS developers (MS, Apple, Linus et al.)
- Application developers (You, EA)

- What role does each play in:
  - Monolithic kernel
  - Microkernel
  - Exokernel

- For a given application, which approach is better
  - Given a choice, which one would you choose?

Deployability of OS research ideas

- The authors argue that many of the OS research topics (that we will cover later in the semester) are never deployed. We haven't seen exokernels in general purpose OS's either.
  - Which is more deployable, monolithic, microkernel, exokernel?
  - What about portability?

Performance

- Which abstraction do you expect will give good performance: monolithic kernel, micro kernel, exokernel?
  - For null benchmarks
  - For SPEC benchmarks
  - For your applications
    - Note the parallels between these and RISC/CISC

Safety?

- What about the safety primitives provided by the OS for the entire user. How do these react?
  - Safety for user means that others code should not affect them and their own code should fail gracefully, if at all

Future

- How would exokernels behave with multicore?

- Browser based OS?
  - MS Singularity
  - Google Gears (local storage) + Chromium (to execute rendering code) + Google cloud (remote services)?