OS Avoidance

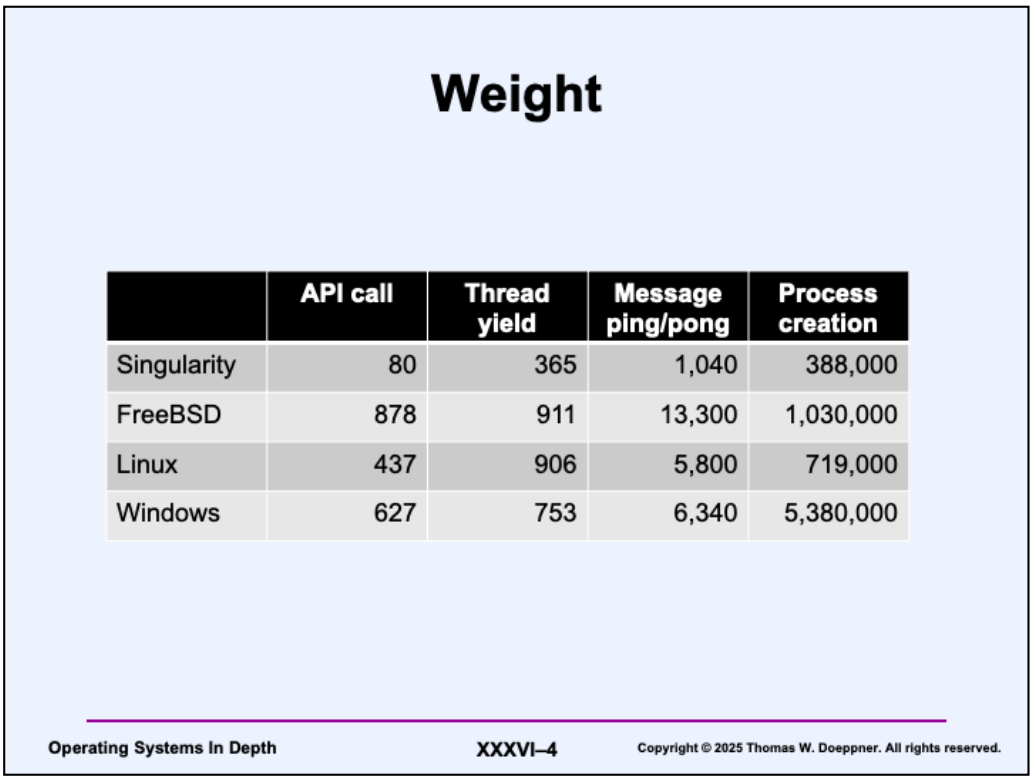
Which OS?

Heavyweight

Lightweight

Heavyweight OS Features

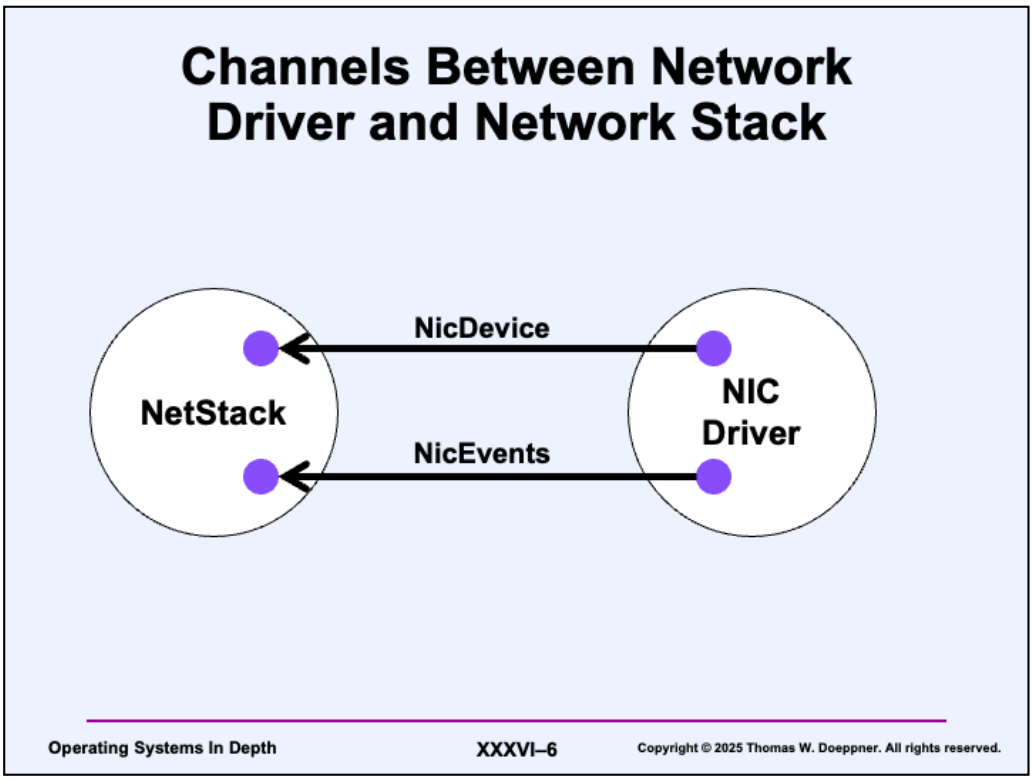
- Separate address spaces
  - virtual memory
- System calls
  - user/privileged-mode distinction



This figure is from "Singularity: Rethinking the Software Stack," by Galen C. Hunt and James R. Larus, both of Microsoft Research. It was published in ACM SIGOPS Operating Systems Review, volume 41, number 2, April 2007. It may be found at https://www.microsoft.com/en-us/research/publication/singularity-rethinking-the-software-stack/. The figures are for the operating systems running on an AMD Athlon 64 3000+ (1.8 GHz) CPU with an NVIDIA nForce4 Ultra chipset. The units are CPU cycles.

Shedding Weight …

- Software-isolated processes (SIPs)
  - use type safety and memory safety to isolate processes
  - all processes run in same address space
  - all run in privileged mode
- IPC via "contract-based channels"
  - bi-directional, reliable message conduits with exactly two endpoints
  - one thread per endpoint
  - formally specified interaction "contract"
  - no other IPC mechanism
  - act as capability mechanism



The arrows indicate channels that are exported by the NIC driver to NetStack.

NIC Driver Contract (1)

```
contract NicDevice {
  out message DeviceInfo(...);
  in message
    RegisterForEvents(
      NicEvents.Exp:READY c);
  in message
    SetParameters(...);
  out message
    InvalidParameters(...);
  out message Success();
  in message StartIO();
  in message ConfigureIO();
  in message
    PacketForReceive(
      byte[] in ExHeap p);
  out message BadPacketSize(
    byte[] in ExHeap p, int m);
  in message
    GetReceivedPacket();
  out message ReceivedPacket(
    Packet * in ExHeap p);
  out message NoPacket();
```

NIC Driver Contract (2)

```
state START: one {
  DeviceInfo! →
    IO_CONFIGURE_BEGIN;
}
state IO_CONFIGURE_BEGIN: one {
  RegisterForEvents? →
    SetParameters? →
      IO_CONFIGURE_ACK;
}
state IO_CONFIGURE_ACK: one {
  InvalidParameters! →
    IO_CONFIGURE_BEGIN;
  Success! → IO_CONFIGURED;
}
```

```
state IO_CONFIGURED: one {
  StartIO? → IO_RUNNING;
  ConfigureIO? →
    IO_CONFIGURE_BEGIN;
}
state IO_RUNNING: one {
  PacketForReceive? →
    (Success!
      or BadPacketSize!) →
        IO_RUNNING;
  GetReceivedPacket? →
    (ReceivedPacket!
      or NoPacket!)
        → IO_RUNNING;
...
}
}
```

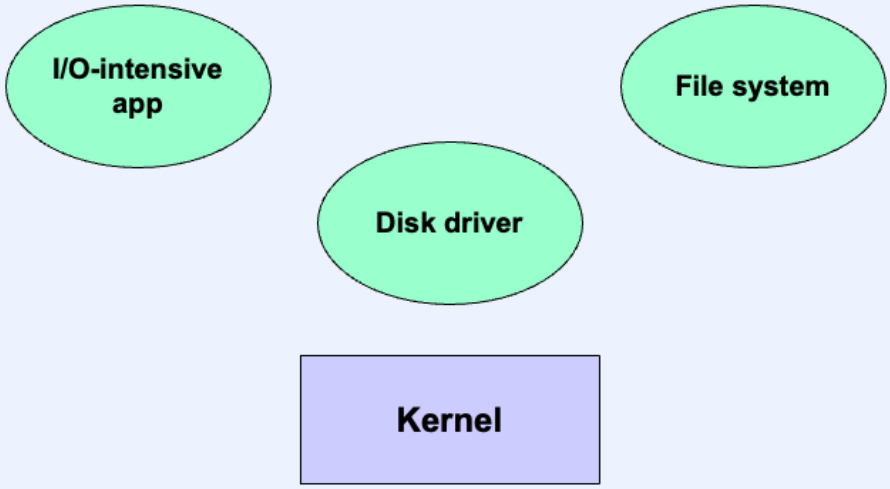
NIC Device Events Contract

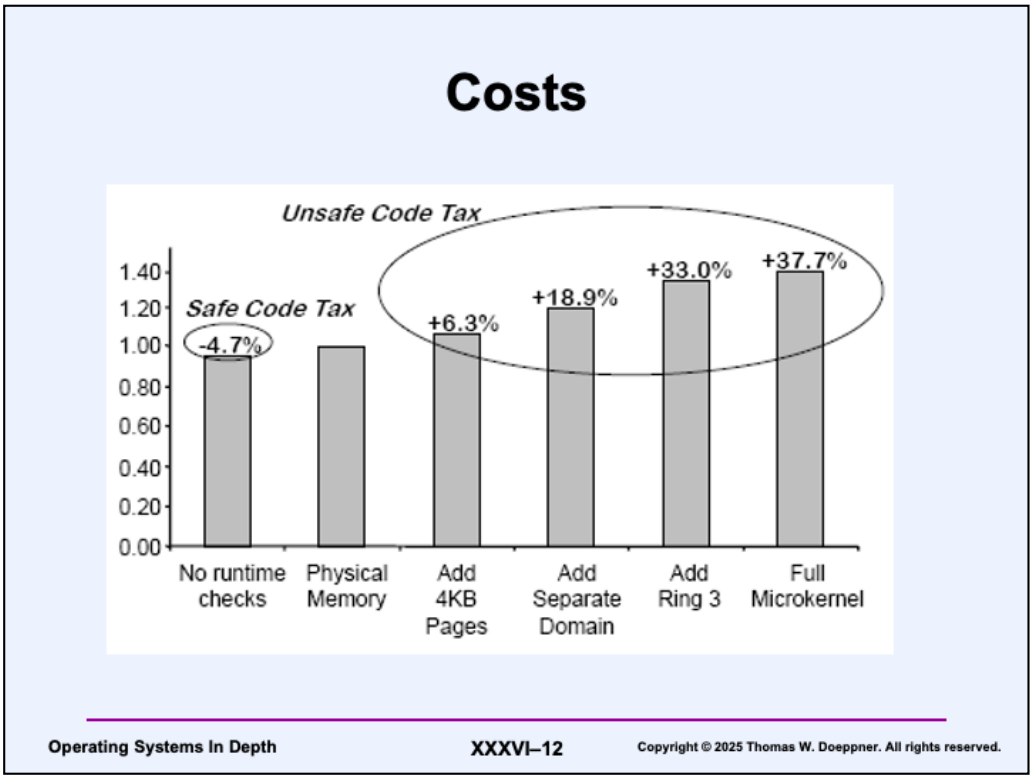
```
contract NicEvents {
  enum NicEventType {
    NoEvent, ReceiveEvent, TransmitEvent,
      LinkEvent
  }
  out message NicEvent(NicEventType e);
  in message AckEvent();
  state READY: one {
      NicEvent! → AckEvent? !READY;
  }
}
```

Manifest

- Each program has a *manifest*
  - details
    - code resources
    - system resources
    - desired capabilities
    - dependencies on other programs

Scenario





The slide, also from Hunt and Larus, shows the costs of various configurations for running the I/O-intensive benchmark sketched in the previous slide. The second bar shows the cost of running the system with address translation turned off—its cost is normalized to one. The first bar shows that it's just 4.7% cheaper without the tests performed in safe code, such as array-bounds checking. The third bar shows the increased cost of turning on address translation, but with all in the same address space. The next bar shows the added cost of putting the app in a separate address space, but still in ring 0. The next bar shows the additional cost for putting the app in ring 3. The last bar shows the cost of moving the other components into separate address spaces in ring 3.

OS's are not Dead

- Linux, MacOS, Windows are flourishing
- The "cloud" consists of
  - virtual machines
  - containers
  - large-scale storage systems
- Security concerns have never been greater
  - isolation is key
    - (controlled) sharing is pretty important too

The End

Not quite:

Homework 4 out today, due April 30

S5FS due May 14

Weenix due May 14

Happy Coding and Good Luck!