OS Avoidance

Which OS?

Heavyweight

Lightweight

Heavyweight OS Features

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

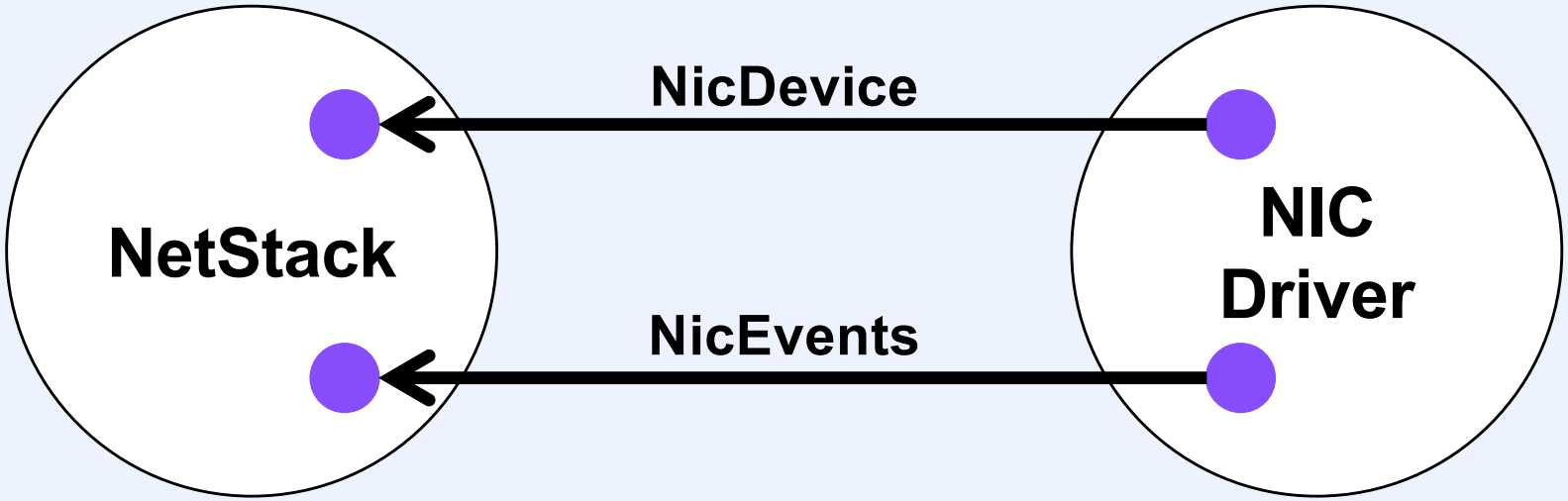
Weight

	API call	Thread yield	Message ping/pong	Process creation
Singularity	80	365	1,040	388,000
FreeBSD	878	911	13,300	1,030,000
Linux	437	906	5,800	719,000
Windows	627	753	6,340	5,380,000

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

Channels Between Network Driver and Network Stack



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;
...
}
}
```

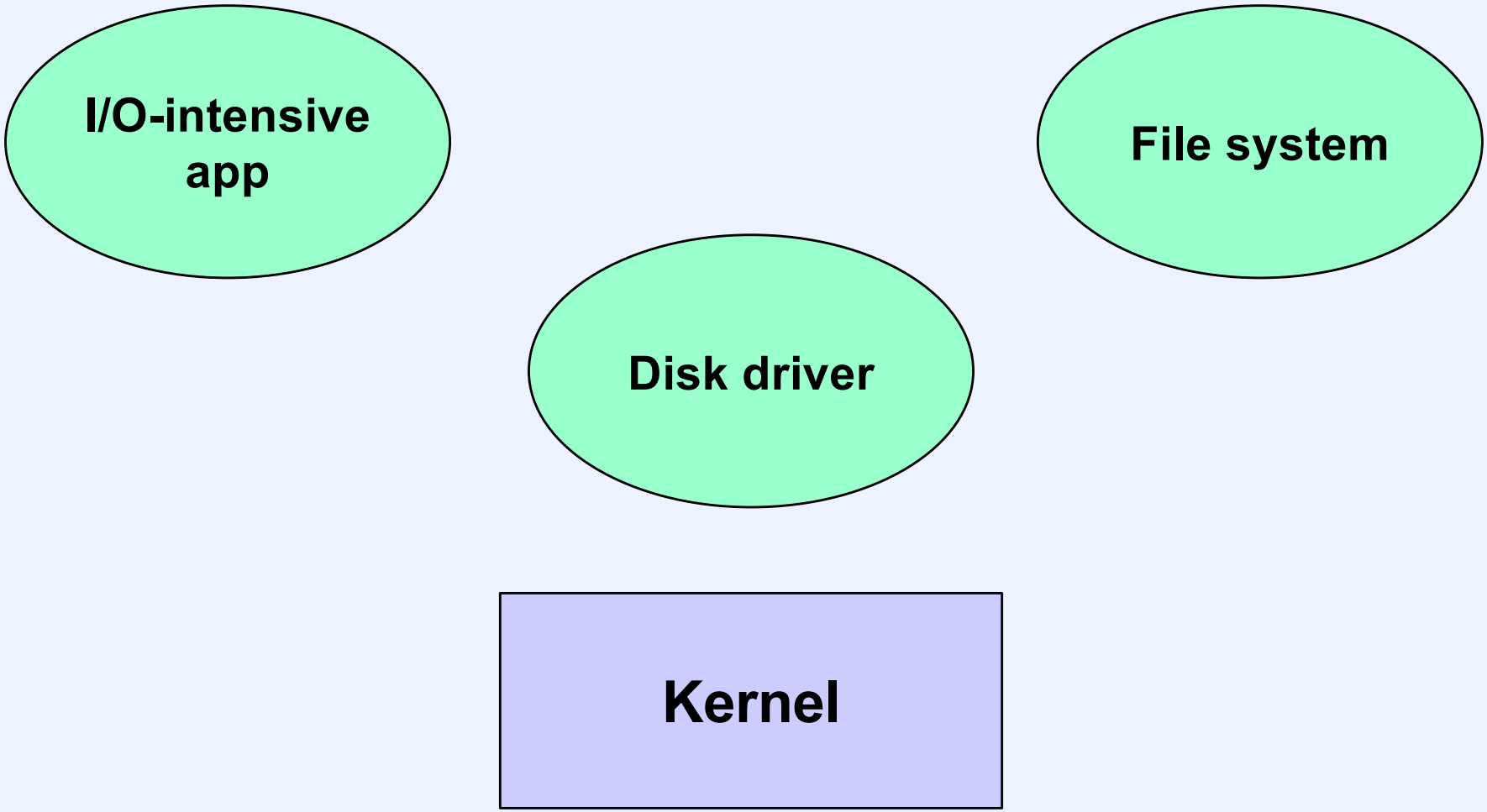
NIC Device Events Contract

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

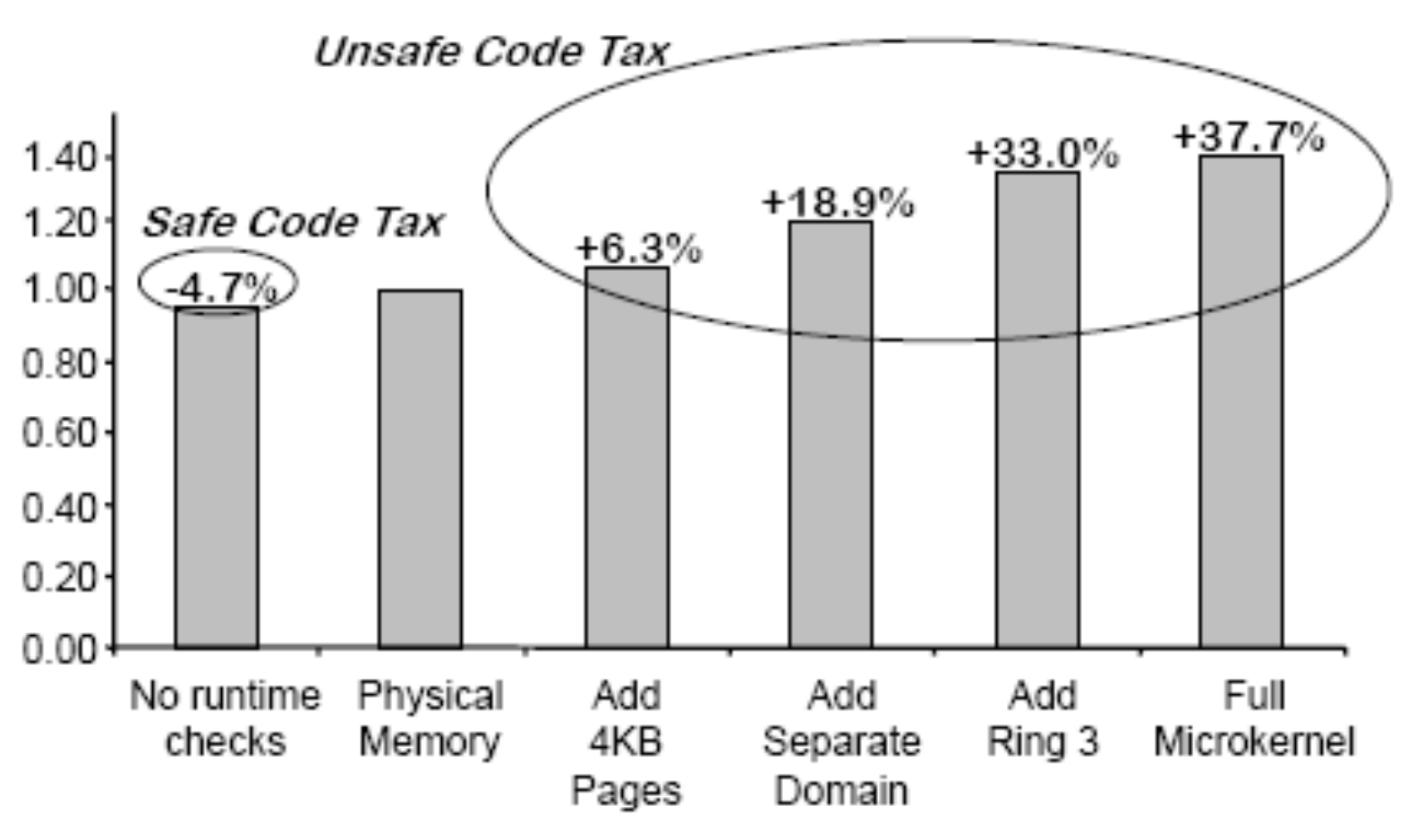
Manifest

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

Scenario



Costs



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!