The MINIX3 Service Layer

Recent Past, Current Status, and Near-Term Future

MINIXCon 2016

David van Moolenbroek
david@minix3.org

Talk outline

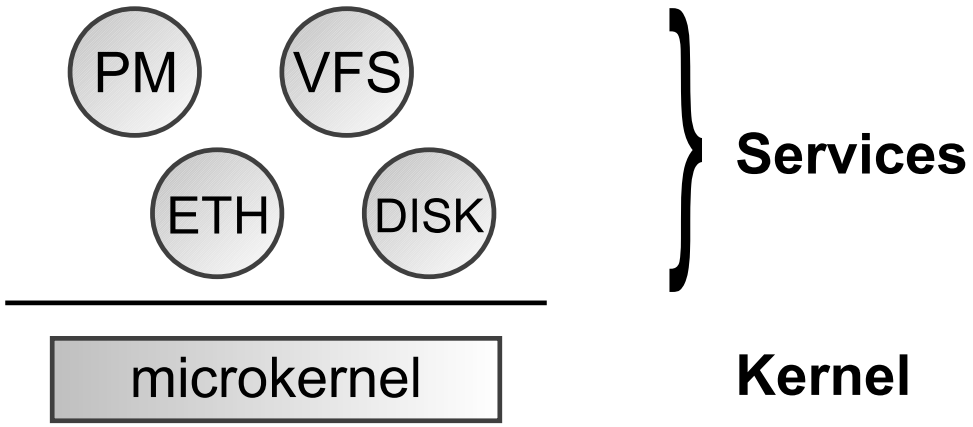
- **Terminology**
- Service layer **evolution**
- Recent and ongoing **projects**
  - A new information service
  - Library-based file systems
  - Network stack redesign
- **Conclusion**
- How to **contribute**

Terminology

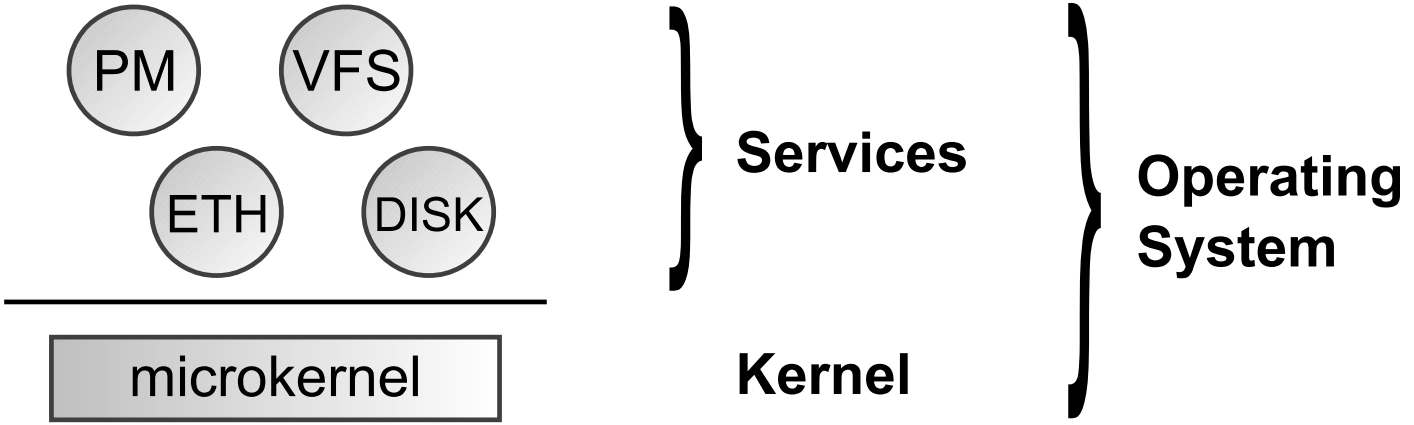
microkernel

Kernel

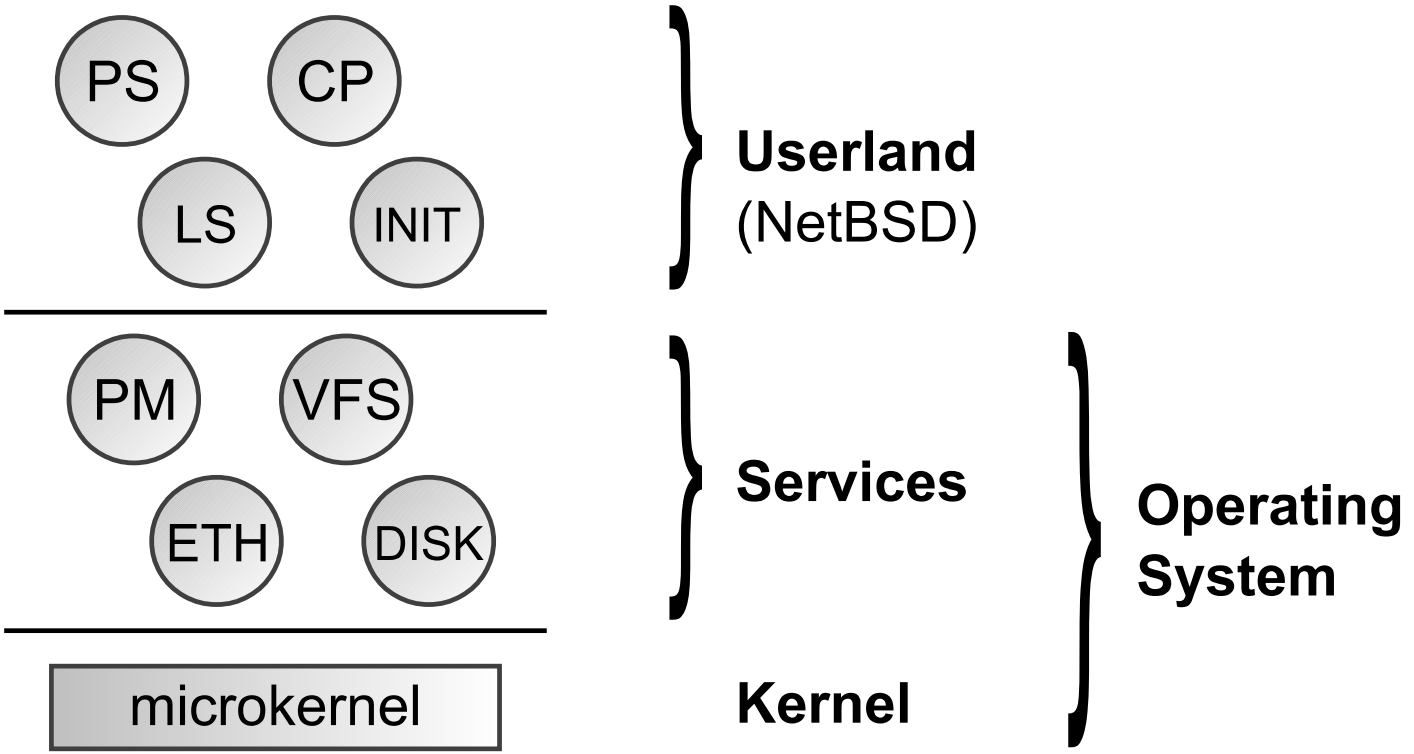
Terminology



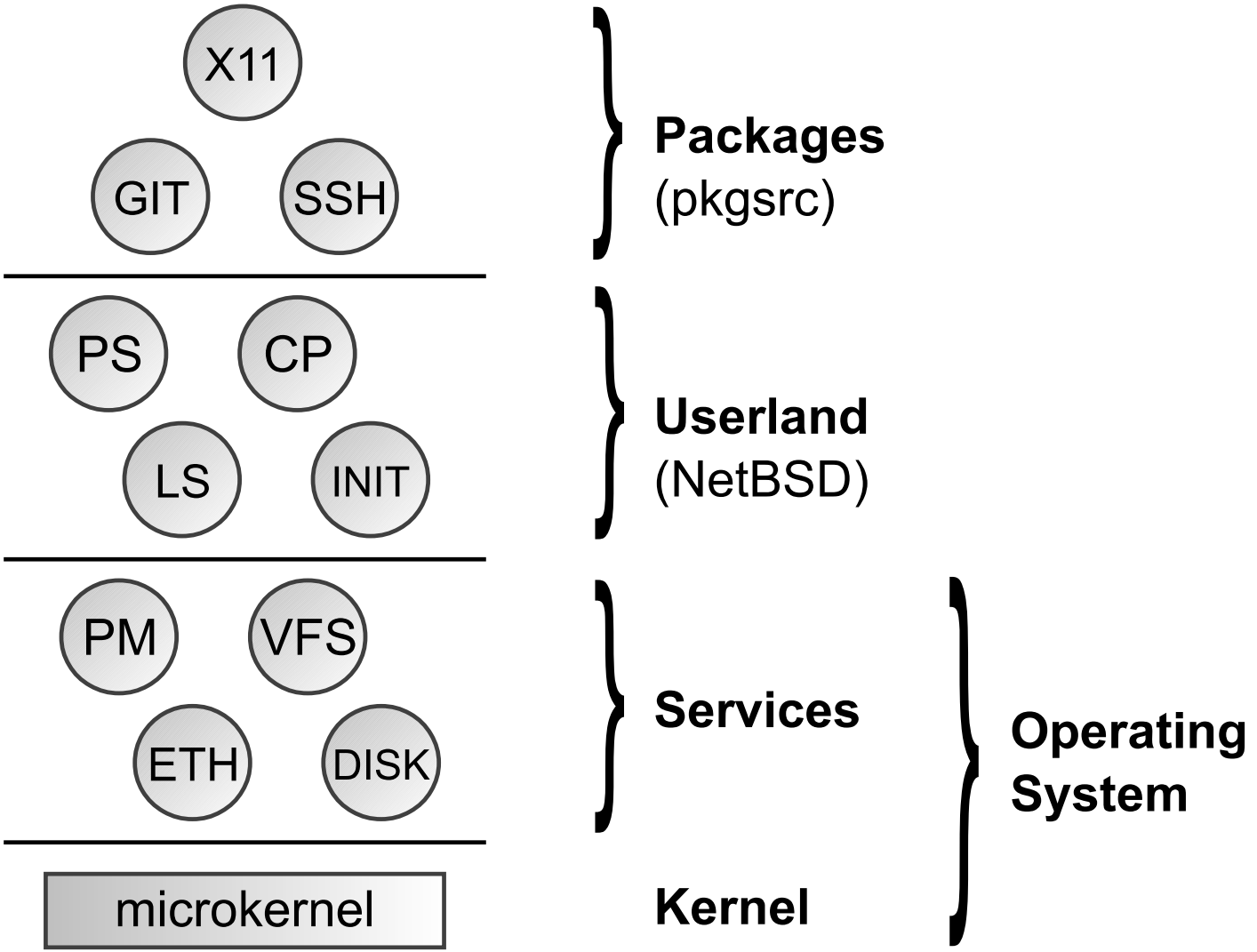
Terminology



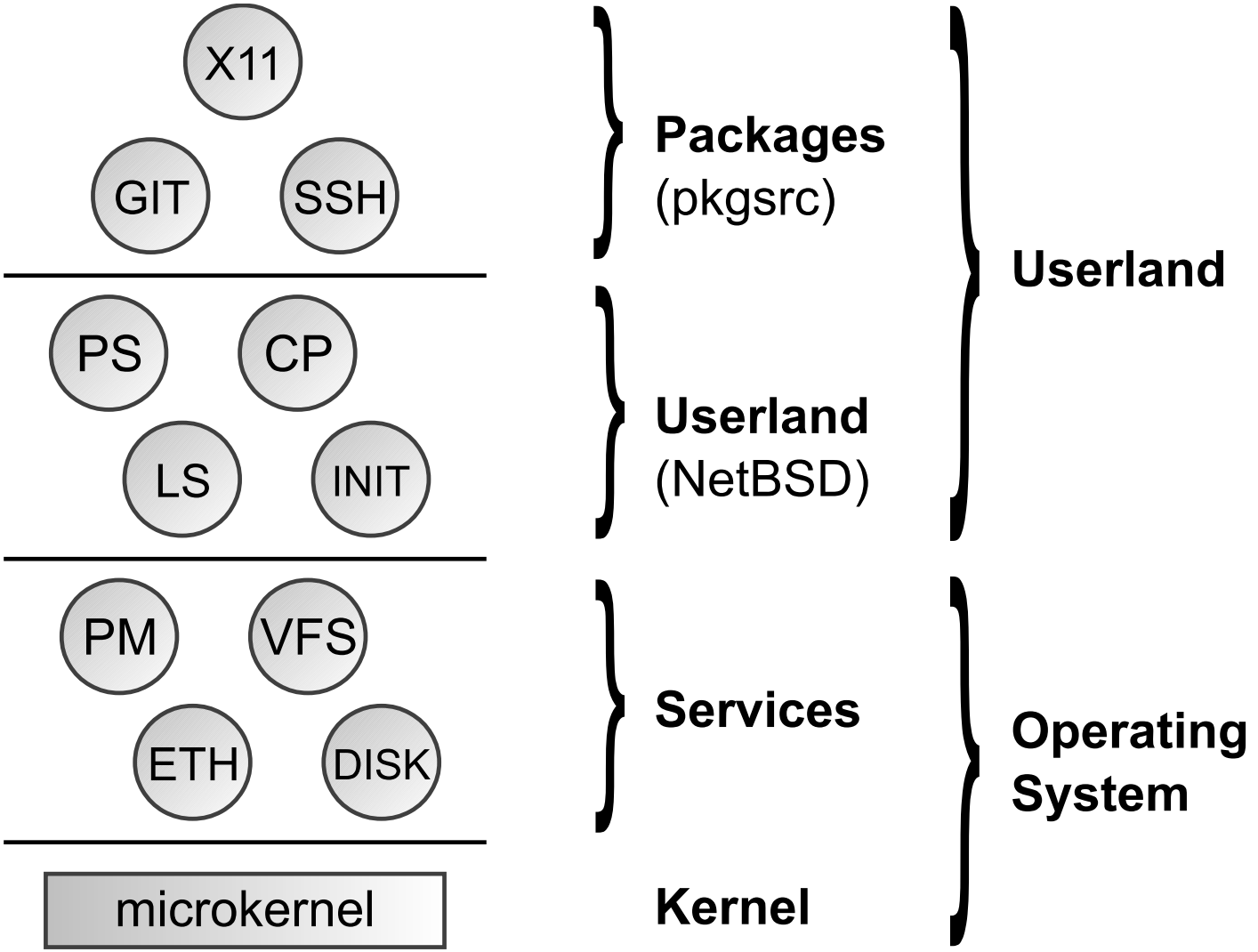
Terminology



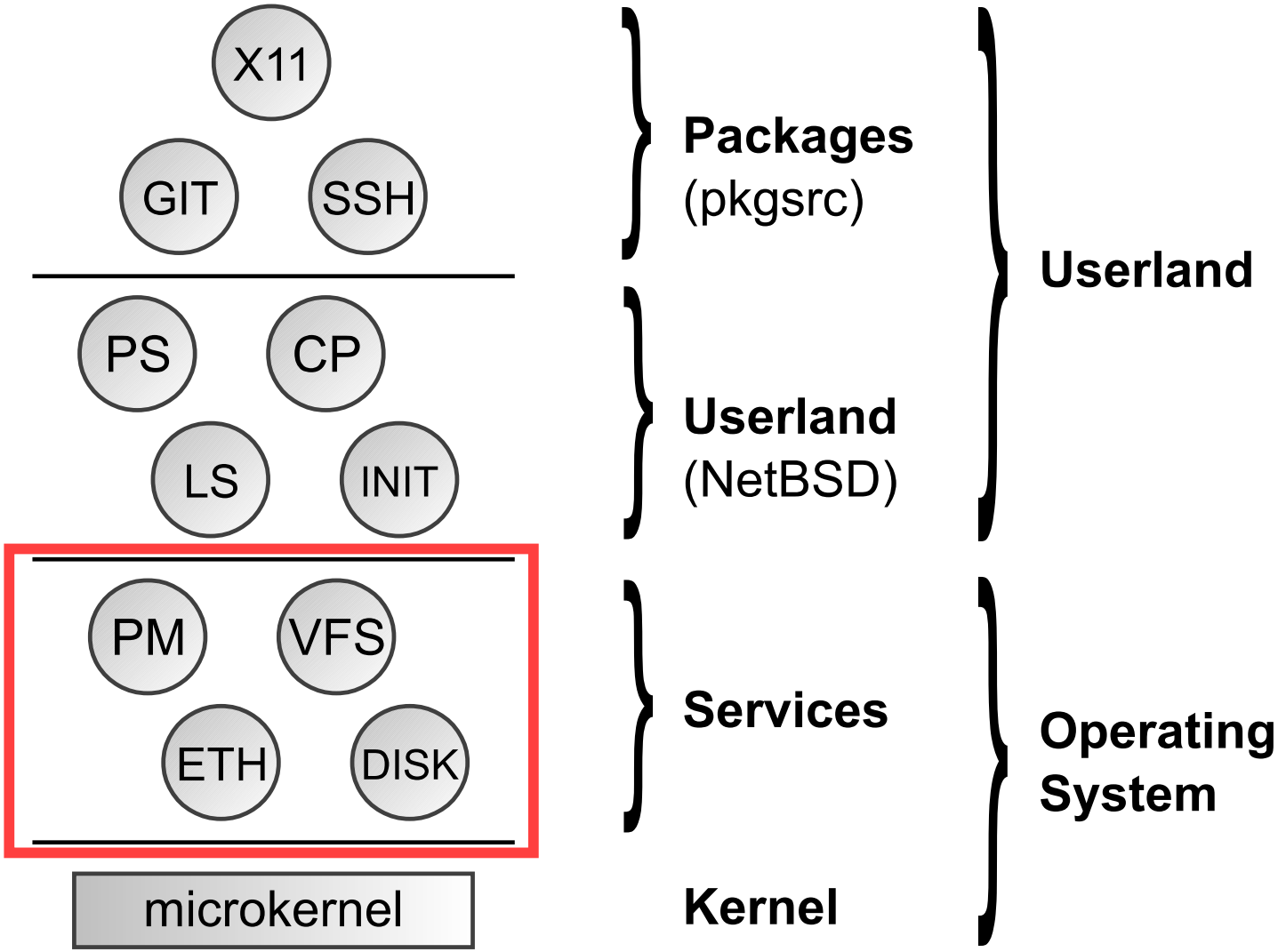
Terminology



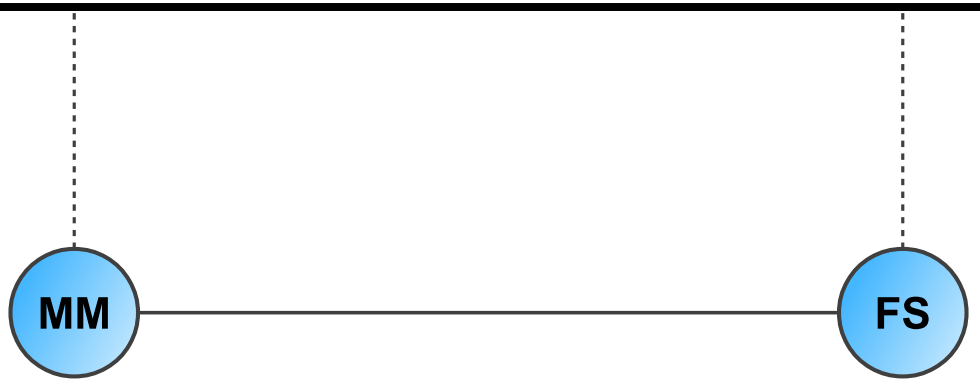
Terminology



Terminology



Service layer evolution

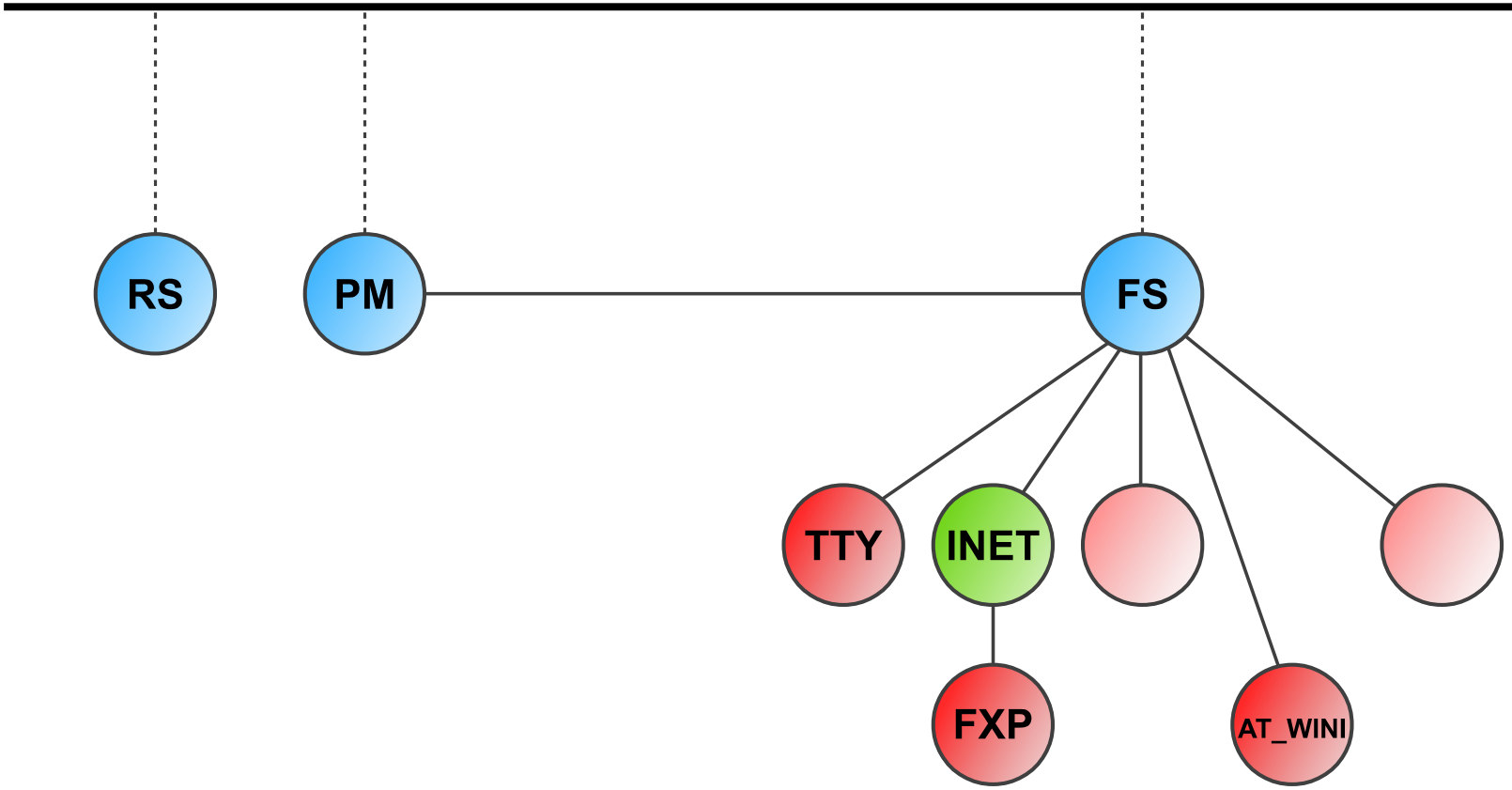


MINIX 1.0.0 (1987)

Service layer evolution

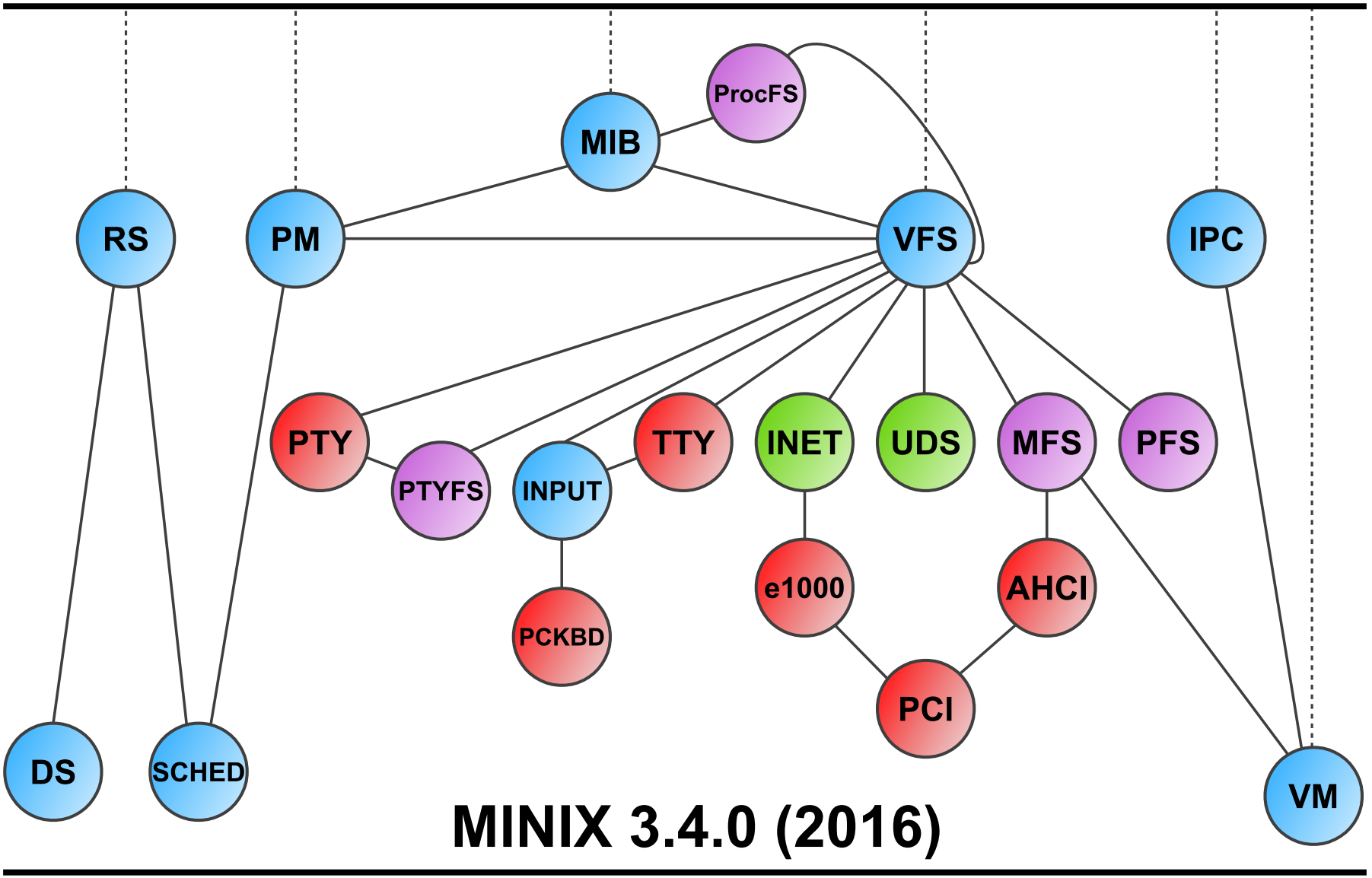
MINIX 2.0.0 (1996)

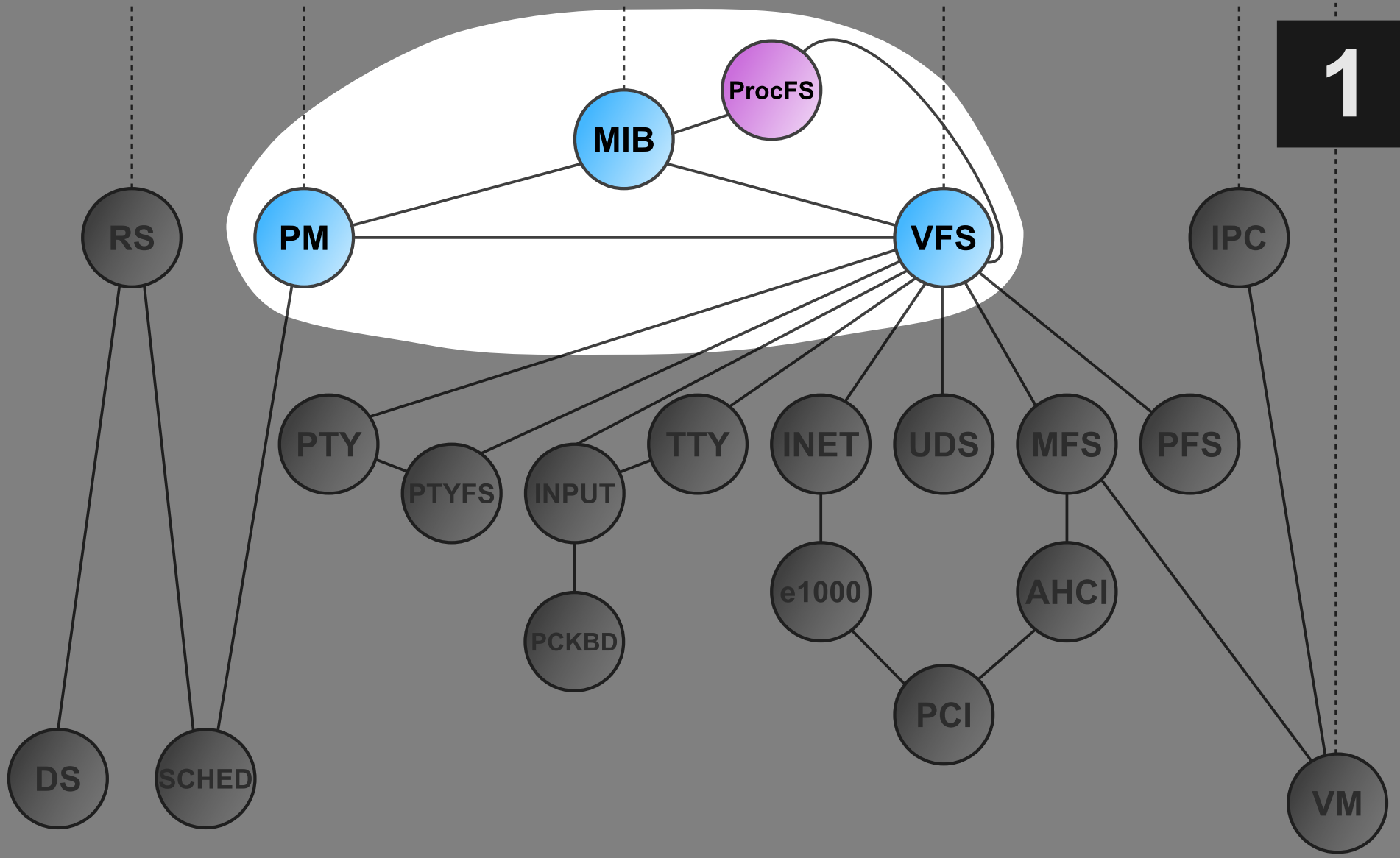
Service layer evolution



MINIX 3.1.0 (2005)

Service layer evolution





A new information service

System information

1

- How to get system info and adjust settings?
  - Largely (inherently) not portable

System information

1

- How to get system info and adjust settings?
  - Largely (inherently) not portable
- Use case: the **ps** utility
  - Show a list of current processes

System information

1

- How to get system info and adjust settings?
  - Largely (inherently) not portable
- Use case: the **ps** utility
  - Show a list of current processes
- In MINIX, ps needs info from *process tables*
  - Distributed across PM, VFS, and the kernel

System information

- How to get system info and adjust settings?
  - Largely (inherently) not portable
- Use case: the **ps** utility
  - Show a list of current processes
- In MINIX, ps needs info from *process tables*
  - Distributed across PM, VFS, and the kernel
- Traditional ps: get tables directly from services
  - Every system change requires ps recompilation

First attempt: ProcFS

1

- Expose information through **/proc** file system
  - Google Summer of Code 2009 project
  - Loosely based on Linux procfs, sysfs

```
minix$ ls /proc
-1      10      127     20      220     246     41      8       hz      pci
-2      11      130     200     222     251     5       84      ipcvecs service
-3      112     171     204     233     3       51      9       kinfo   uptime
-4      117     18      208     234     30      6       98      loadavg
-5      12      182     217     235     33      7       cpuinfo meminfo
1       120     196     22      236     4       70      dmap    mounts
minix$ _
```

First attempt: ProcFS

1

- Expose information through **/proc** file system
  - Google Summer of Code 2009 project
  - Loosely based on Linux procfs, sysfs

```
minix$ ls /proc
-1      10      127     20      220     246     41      8       hz      pci
-2      11      130     200     222     251     5       84      ipcvecs service
-3      112     171     204     233     3       51      9       kinfo   uptime
-4      117     18      208     234     30      6       98      loadavg
-5      12      182     217     235     33      7       cpuinfo meminfo
1       120     196     22      236     4       70      dmap    mounts
minix$ _
```

- **ps** was changed accordingly – problem solved!

First attempt: ProcFS

1

- Expose information through **/proc** file system
  - Google Summer of Code 2009 project
  - Loosely based on Linux procfs, sysfs

```
minix$ ls /proc
-1      10      127     20      220     246     41      8       hz      pci
-2      11      130     200     222     251     5       84      ipcvecs service
-3      112     171     204     233     3       51      9       kinfo   uptime
-4      117     18      208     234     30      6       98      loadavg
-5      12      182     217     235     33      7       cpuinfo meminfo
1       120     196     22      236     4       70      dmap    mounts
minix$ _
```

- **ps** was changed accordingly – problem solved!
- That is, until we wanted to import **NetBSD ps**...

The *sysctl* interface


```
int sysctl(int mib[], u_int miblen,
           void *oldp, size_t *oldlenp,
           void *newp, size_t newlen);
```

The *sysctl* interface

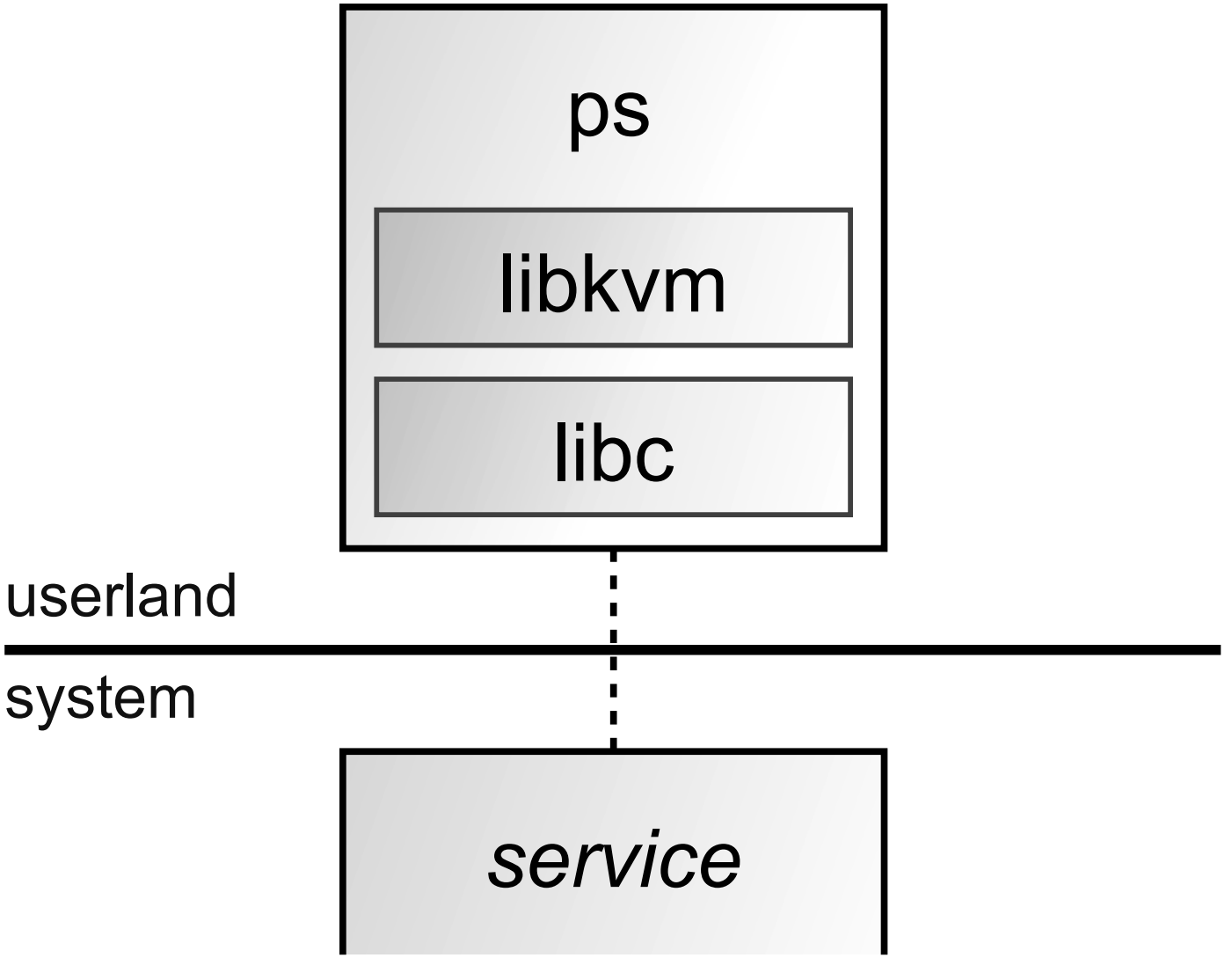
1

```
int sysctl(int mib[], u_int miblen,
           void *oldp, size_t *oldlenp,
           void *newp, size_t newlen);
```

- Access to a hierarchical **key-value store**
- **Key**: an array of numbers, one per tree level
  - *Management Information Base* style
  - For example: 1.4.2.13.5
  - Symbolic names: “kern.hostname” instead of “1.10”
- **Value**: integer, string, or structure
- Not persistent; most keys managed by system

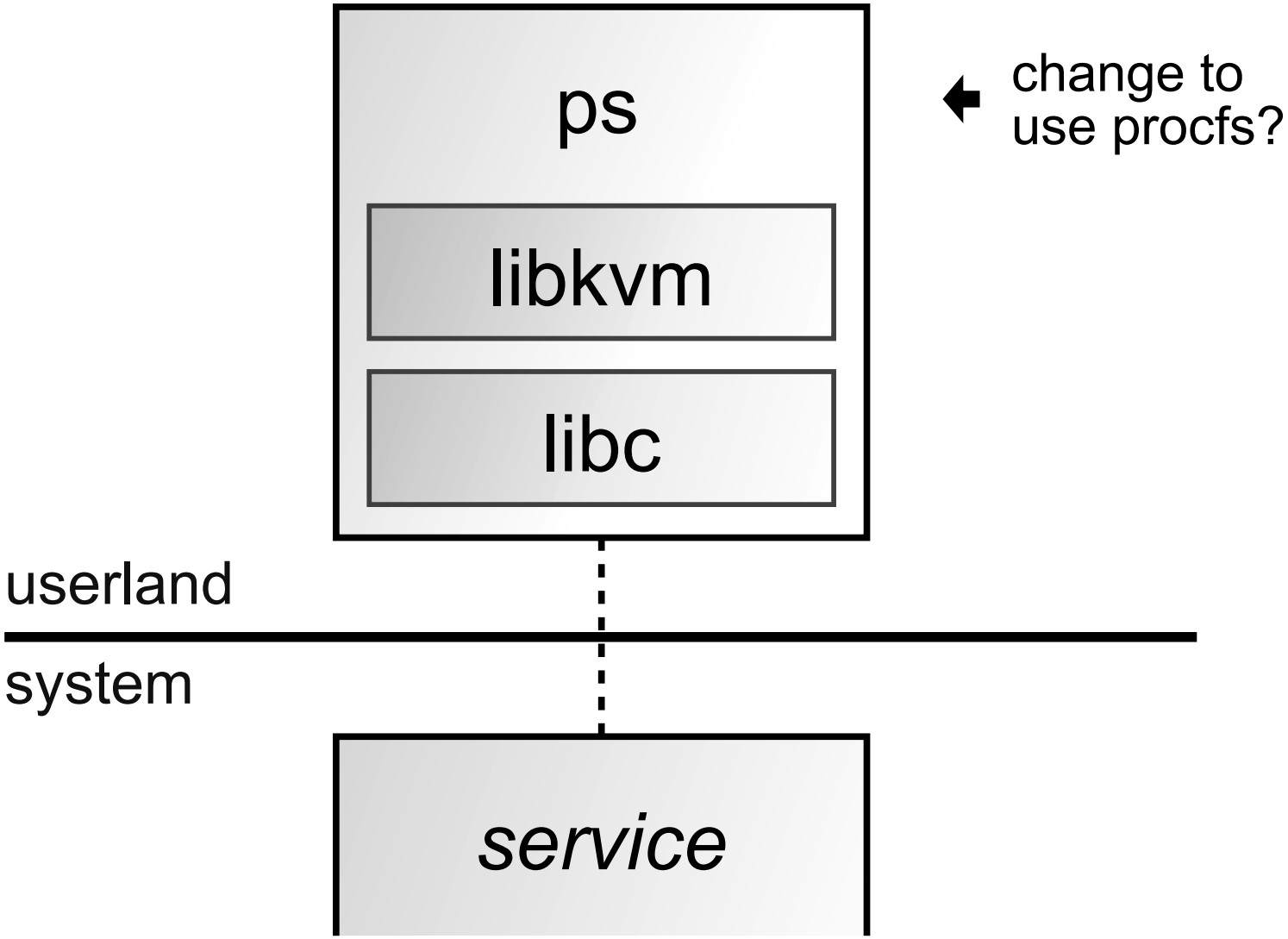
Importing NetBSD ps

1

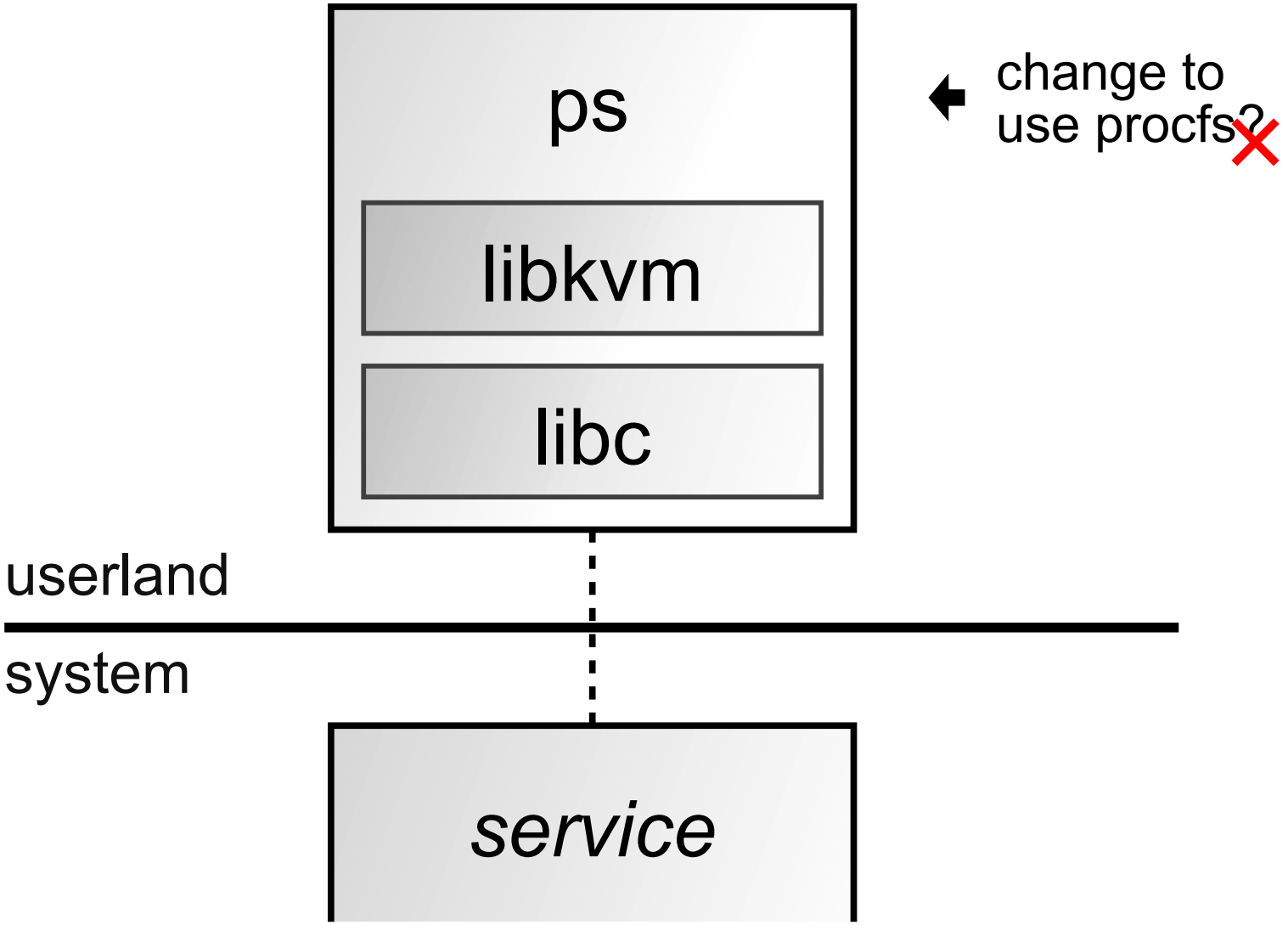


Importing NetBSD ps

1

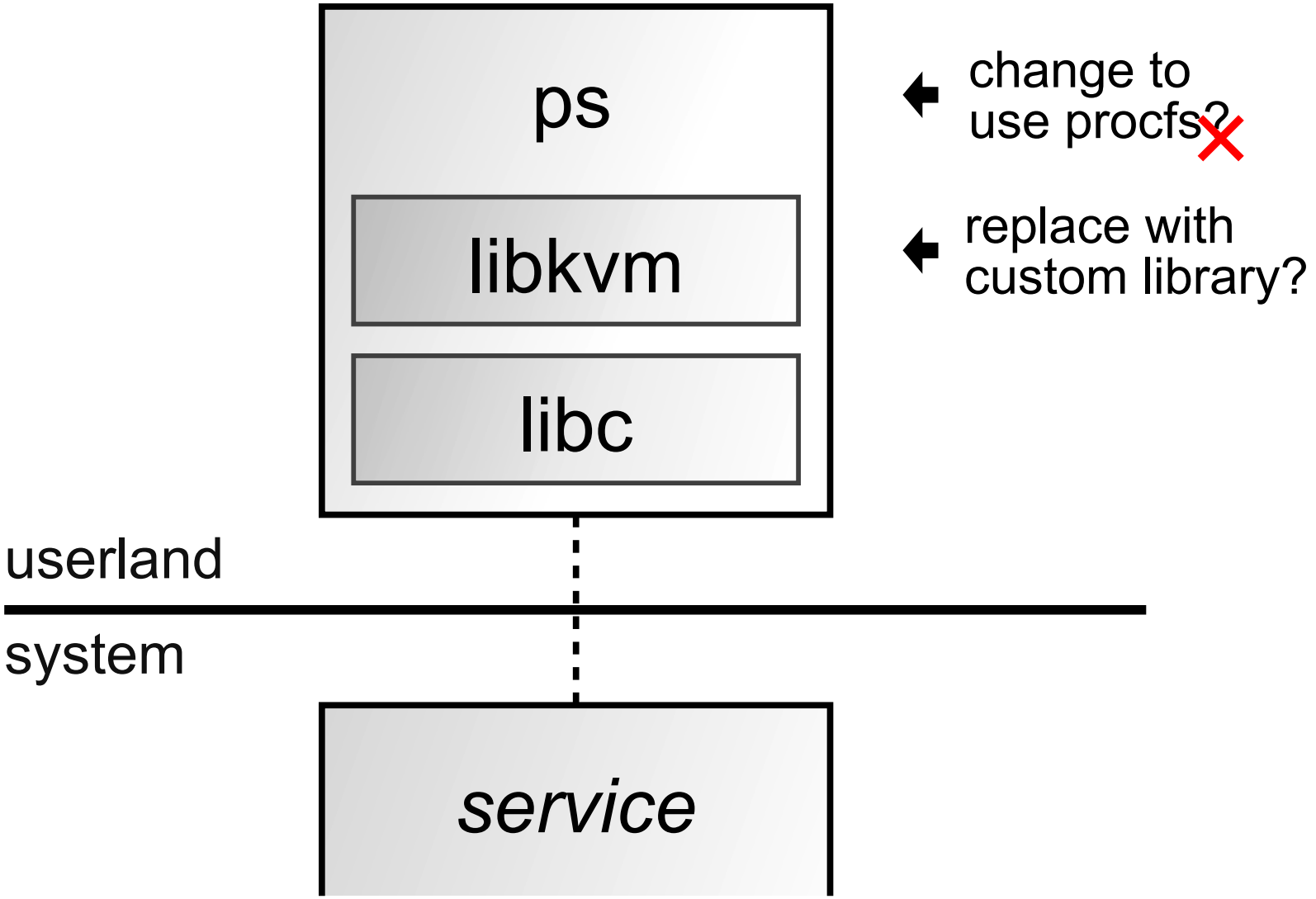


Importing NetBSD ps



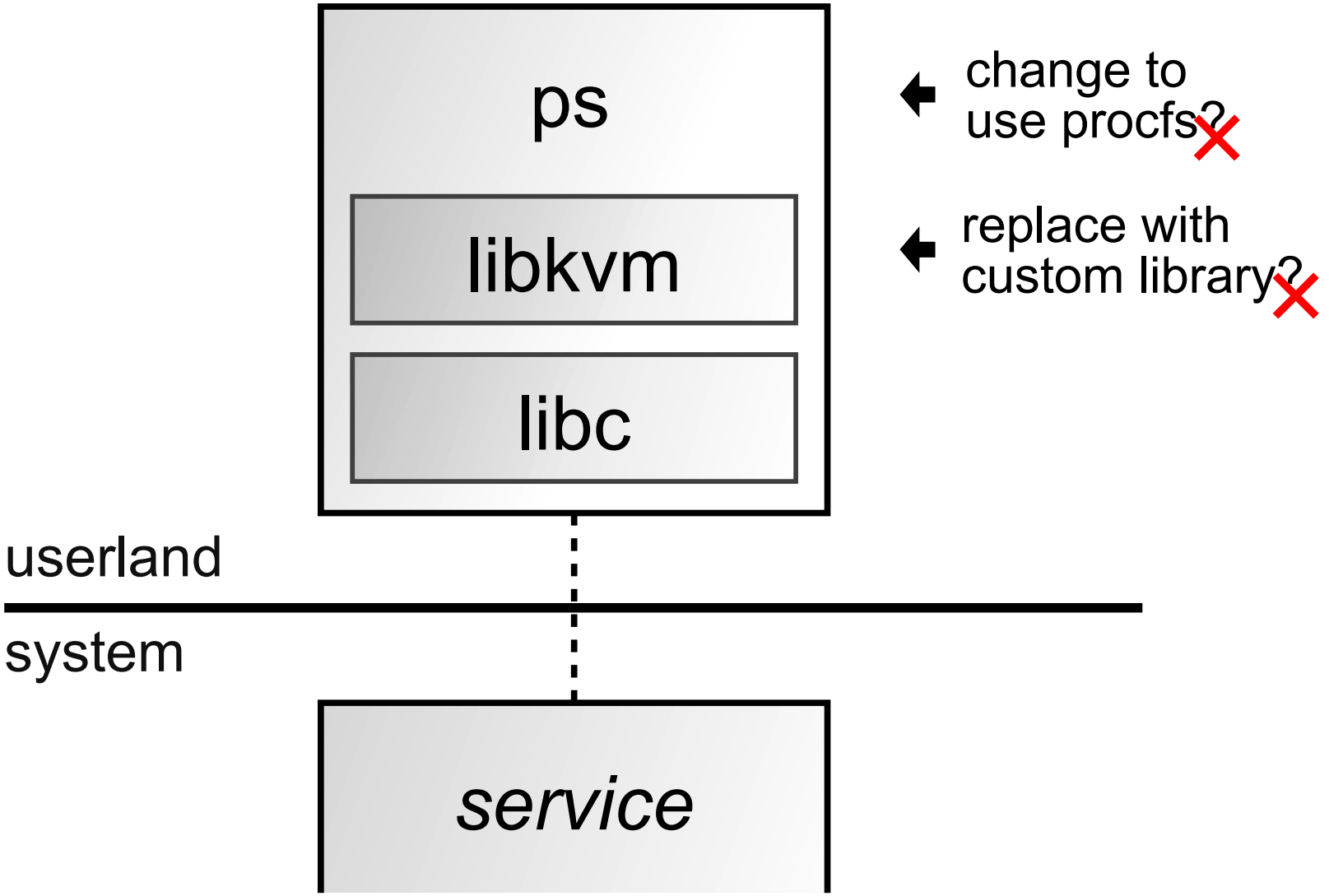
Importing NetBSD ps

1



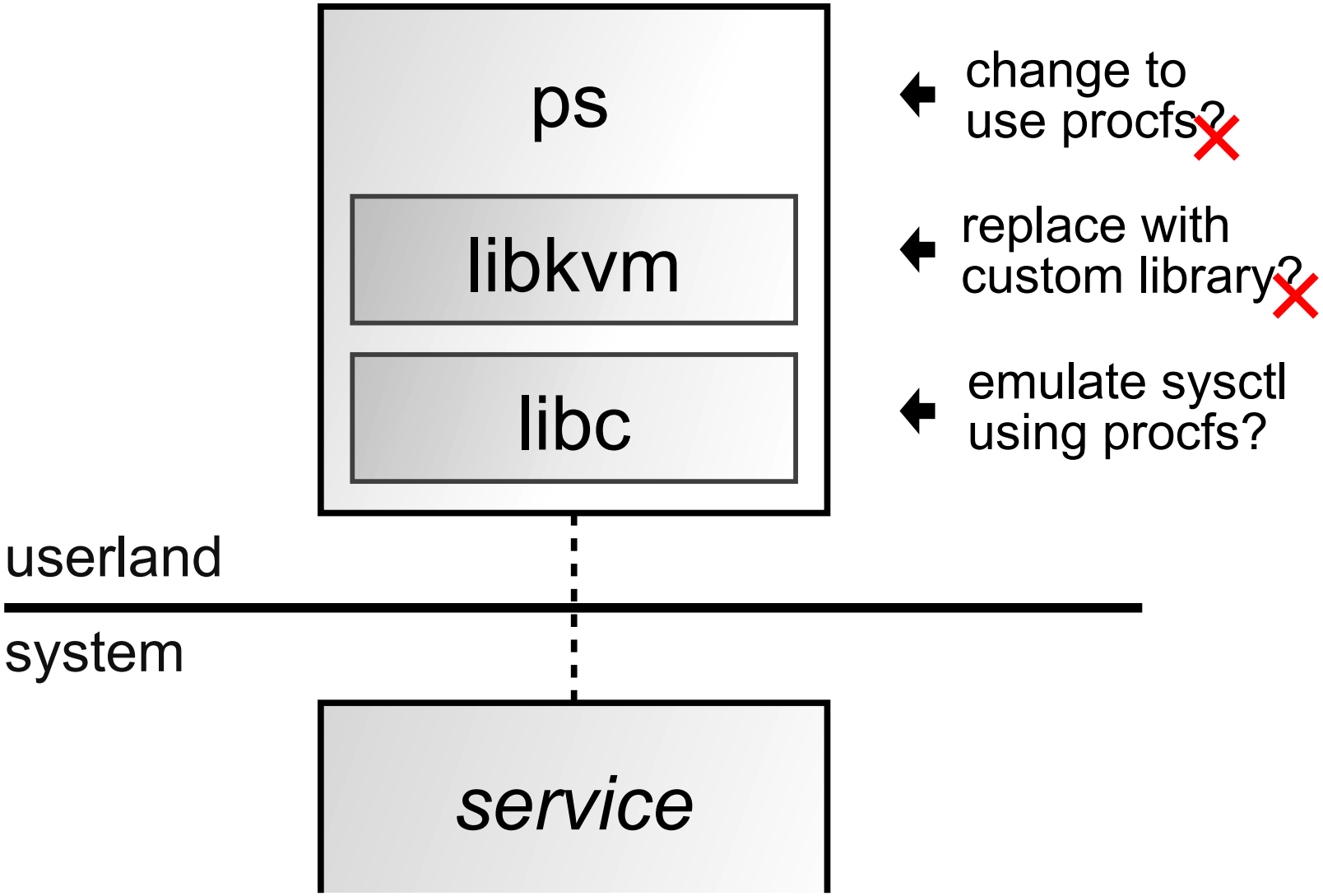
Importing NetBSD ps

1



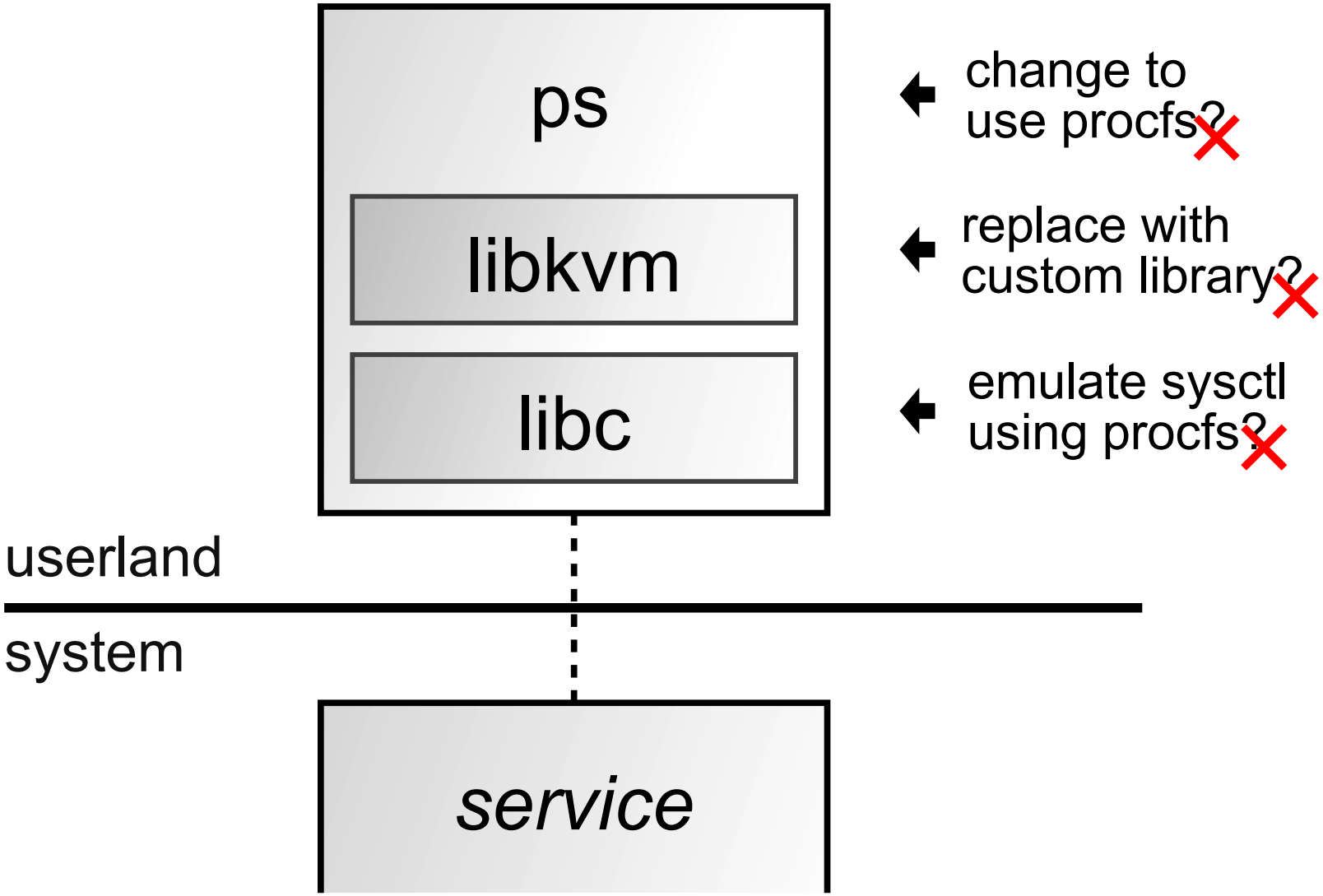
Importing NetBSD ps

1



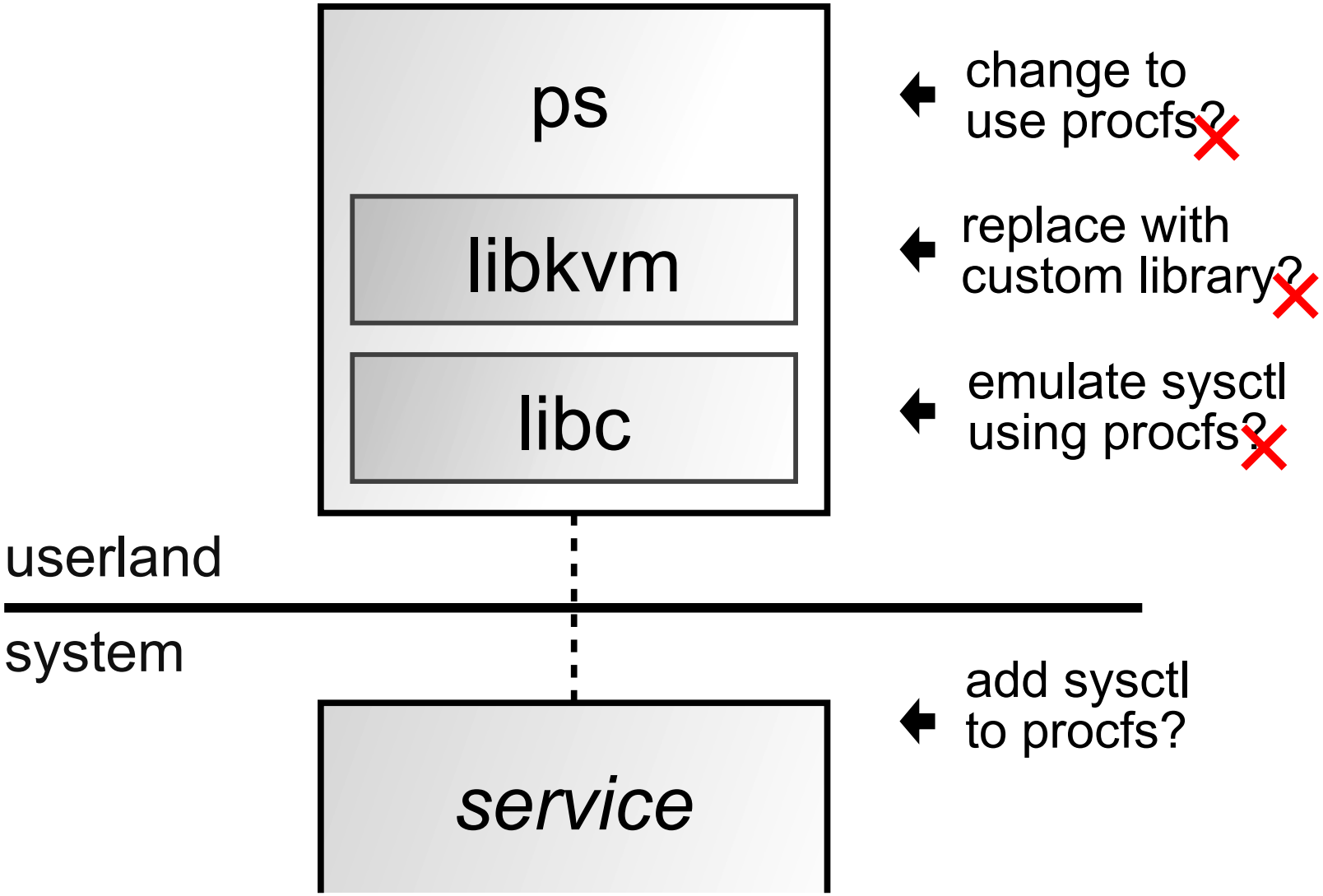
Importing NetBSD ps

1



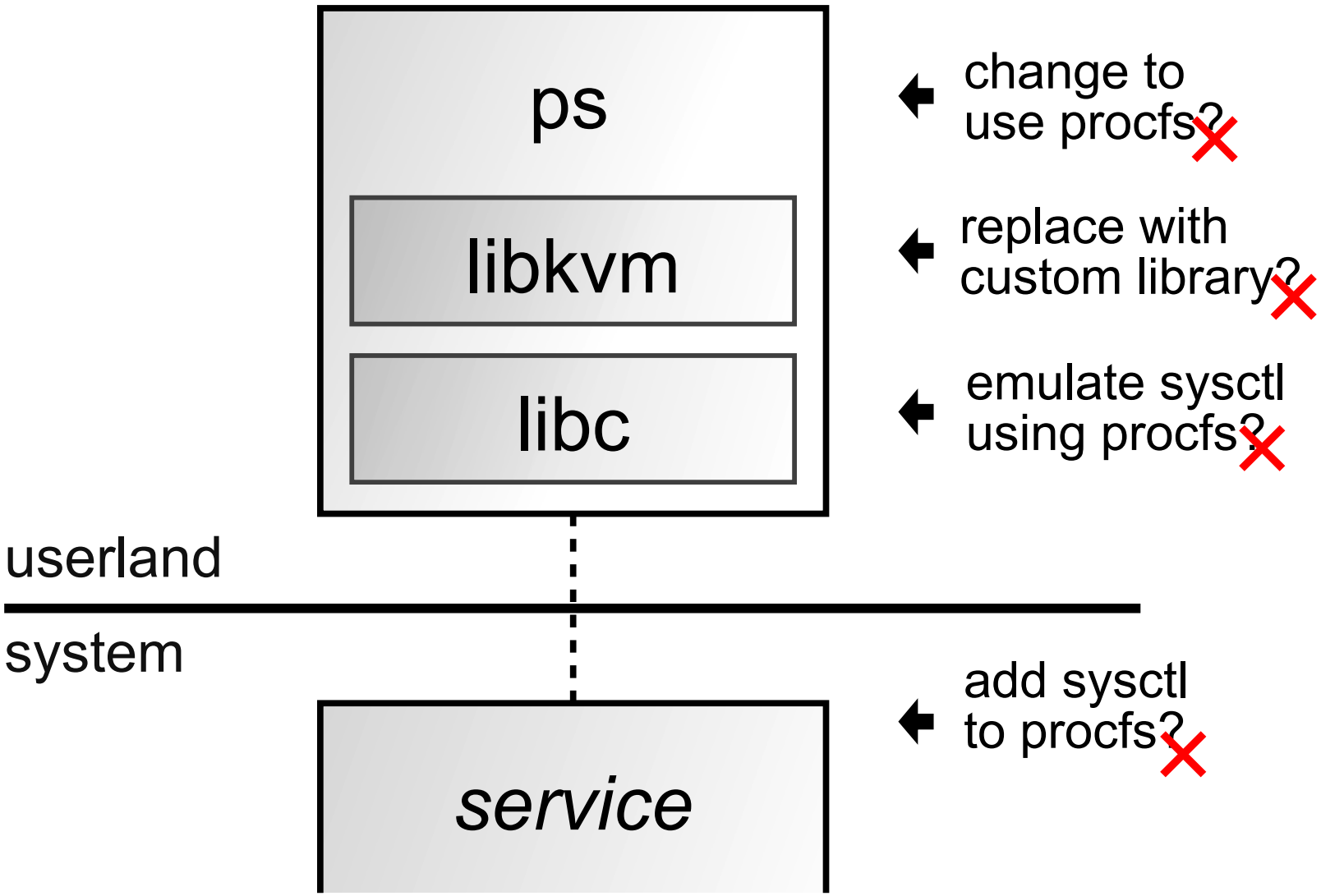
Importing NetBSD ps

1



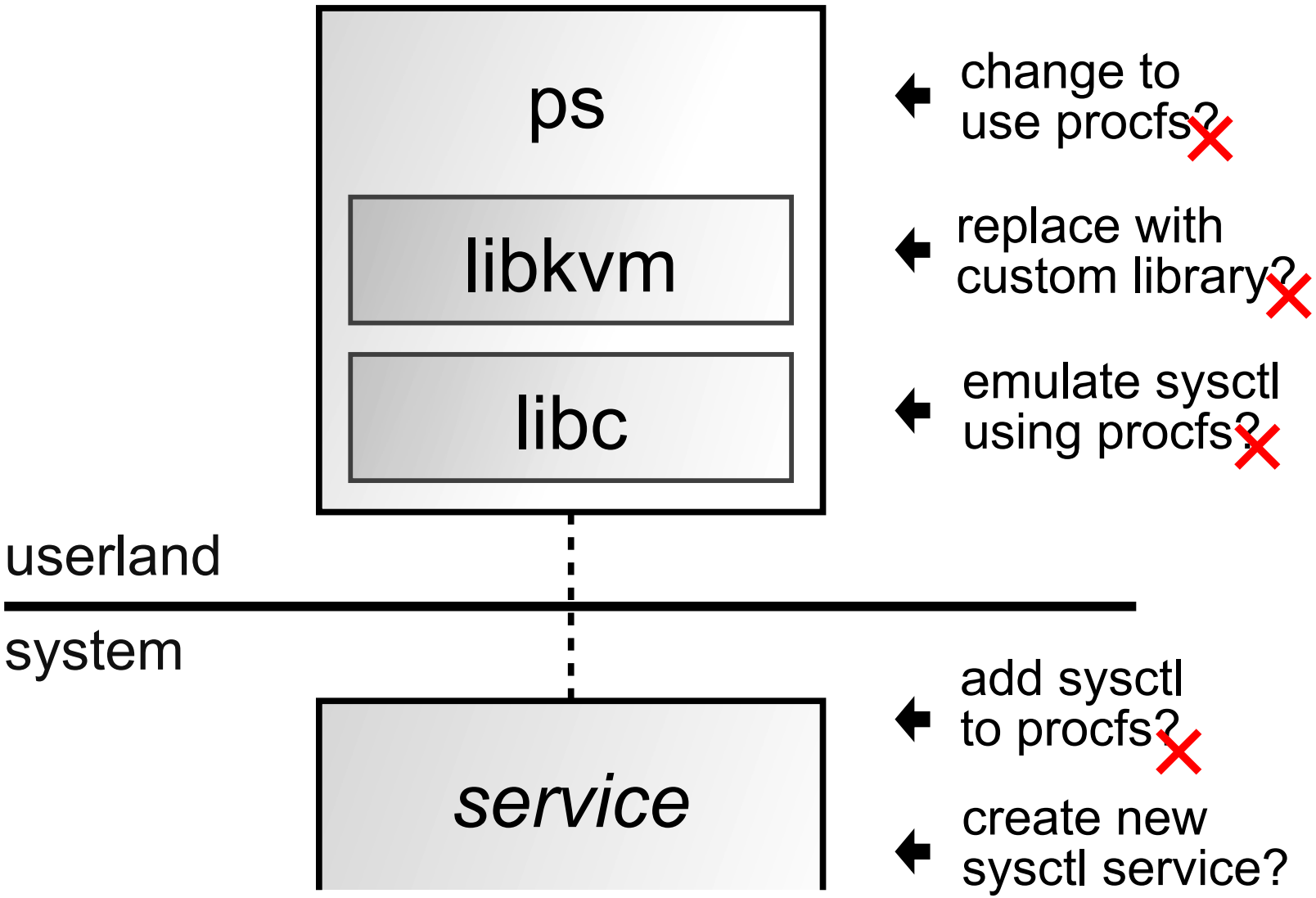
Importing NetBSD ps

1



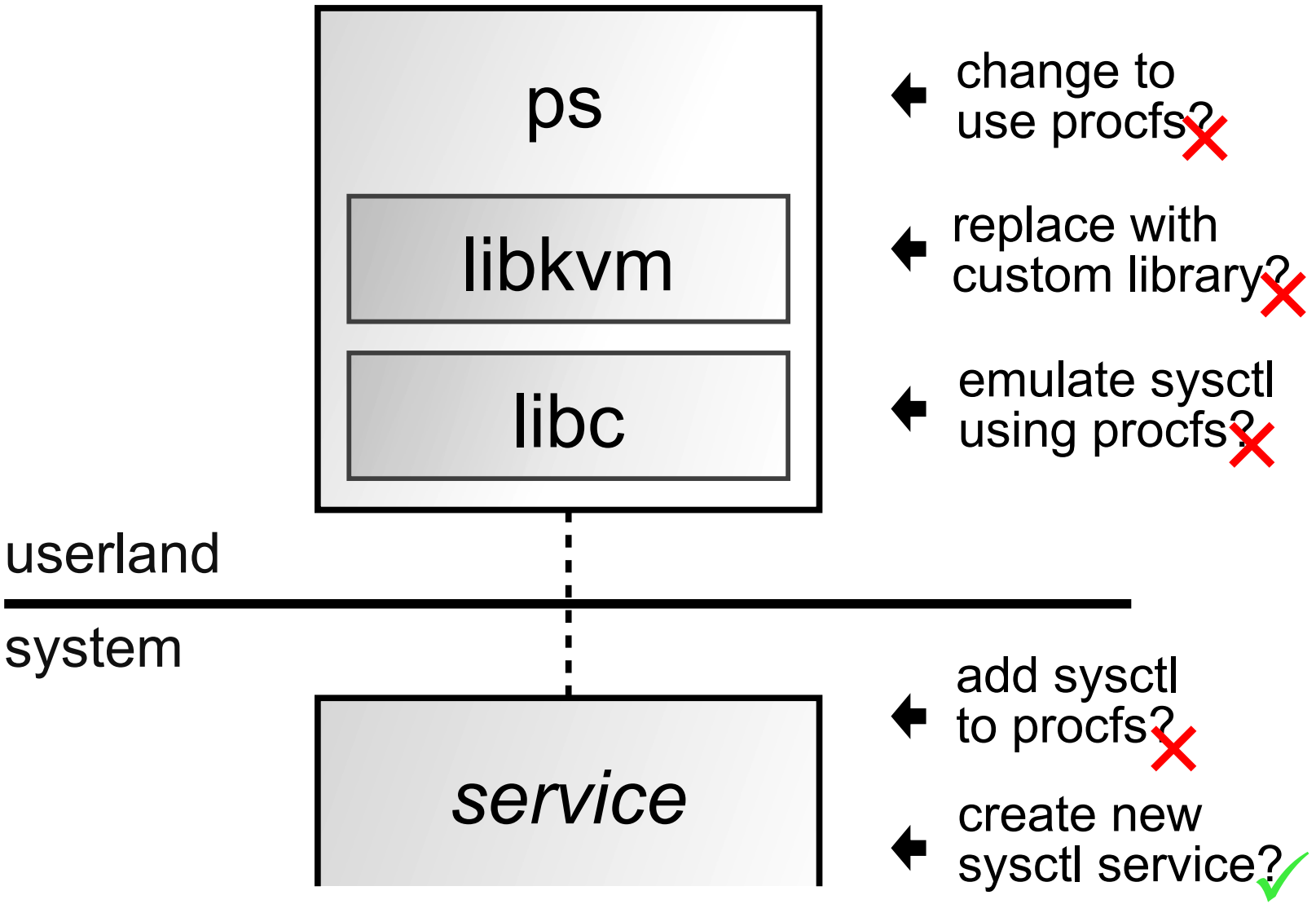
Importing NetBSD ps

1



Importing NetBSD ps

1



Current status and future

1

- Management Information Base (MIB) service
  - **Merged**, available in 3.4.0!
  - About **85** of NetBSD's ~800 **keys** implemented
  - **4,146 LoC** (Lines of Code)
  - ProcFS now calls into the MIB service
  - Imported **userland**: ps, sysctl, top, ipcs, ipcrm

Current status and future

1

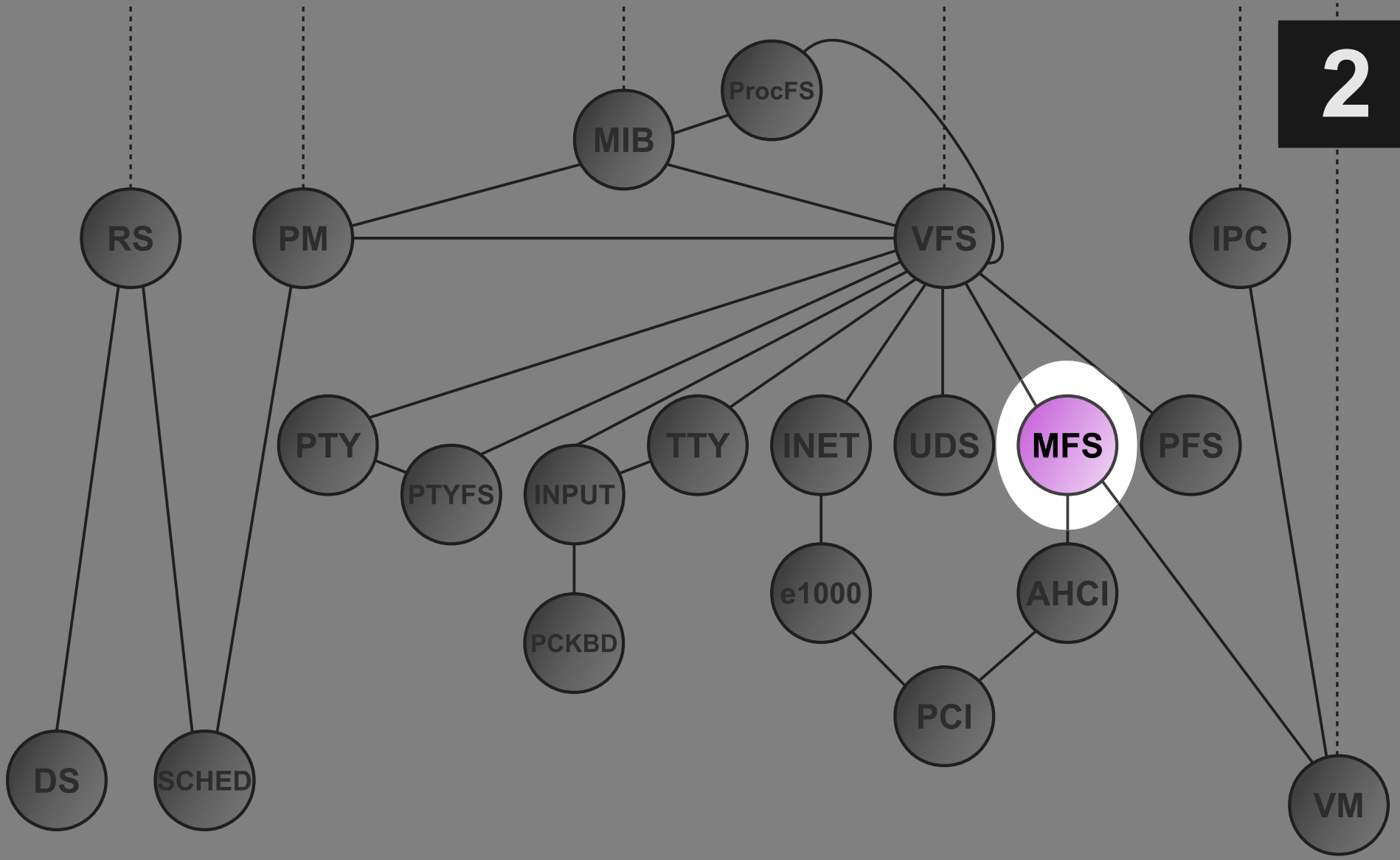
- Management Information Base (MIB) service
  - **Merged**, available in 3.4.0!
  - About **85** of NetBSD's ~800 **keys** implemented
  - **4,146 LoC** (Lines of Code)
  - ProcFS now calls into the MIB service
  - Imported **userland**: ps, sysctl, top, ipcs, ipcrm

- A path forward for importing other NetBSD tools

Try it: *sysctl -a*

1

```
minix$ sysctl -a
kern.ostype = Minix
kern.osrelease = 3.4.0
kern.osrevision = 304000000
kern.version = Minix 3.4.0 (GENERIC)
kern.maxvnodes = 1024
kern.maxproc = 256
kern.maxfiles = 1024
kern.argmax = 262144
kern.securelevel = -1
kern.hostname = minix
kern.hostid = 0
kern.clockrate: tick = 16666, tickadj = 16666, hz = 60, profhz = 60, stathz = 60
kern.posix1version = 200112
...
vm.loadavg: 0.00 0.00 0.00
vm.maxslp = 20
vm.uspace = 0
hw.machine = i386
hw.ncpu = 1
hw.byteorder = 1234
hw.physmem = 1073213440
hw.usermem = 1069870560
hw.pagesize = 4096
:_
```



Library-based file systems

Disk-backed file systems

- Not much progress
  - 2006: “FS” service split into VFS and MFS
  - 2016: two more disk-backed FSes (ext2, isofs)

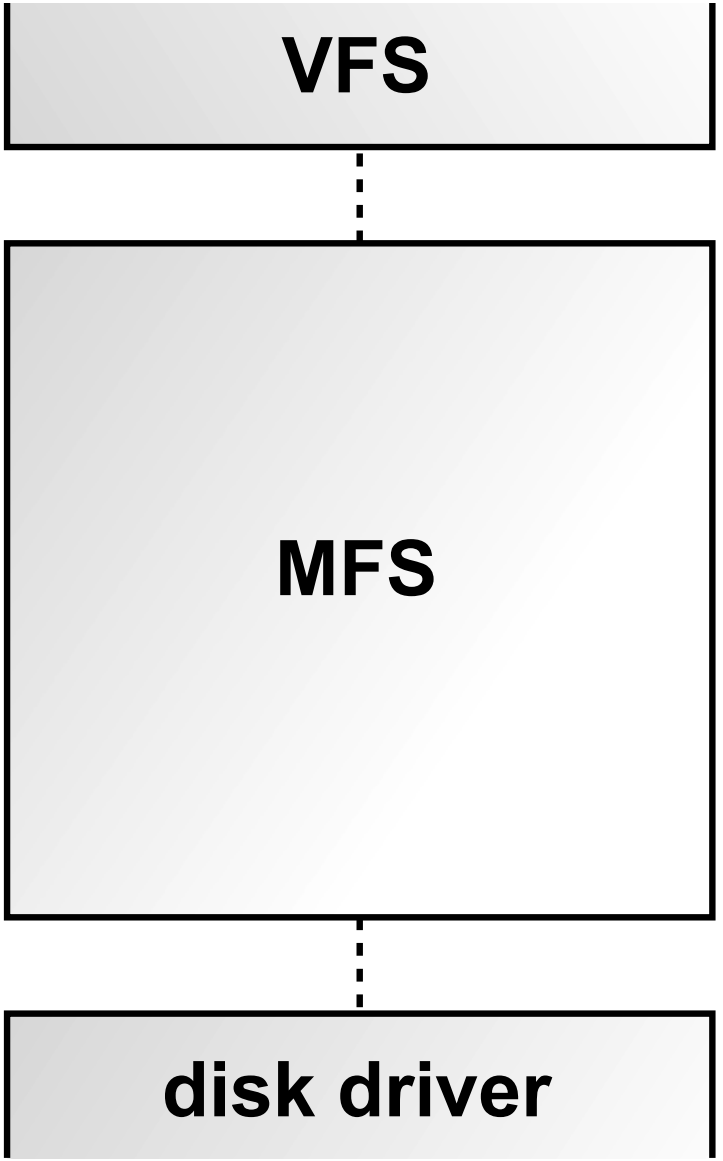
Disk-backed file systems

- Not much progress
  - 2006: “FS” service split into VFS and MFS
  - 2016: two more disk-backed FSes (ext2, isofs)
- It is *pretty hard* to write a file system service
  - Implementing e.g. **rename** is seriously difficult
  - ext2 and isofs started as copies of MFS
  - Maintenance nightmare!

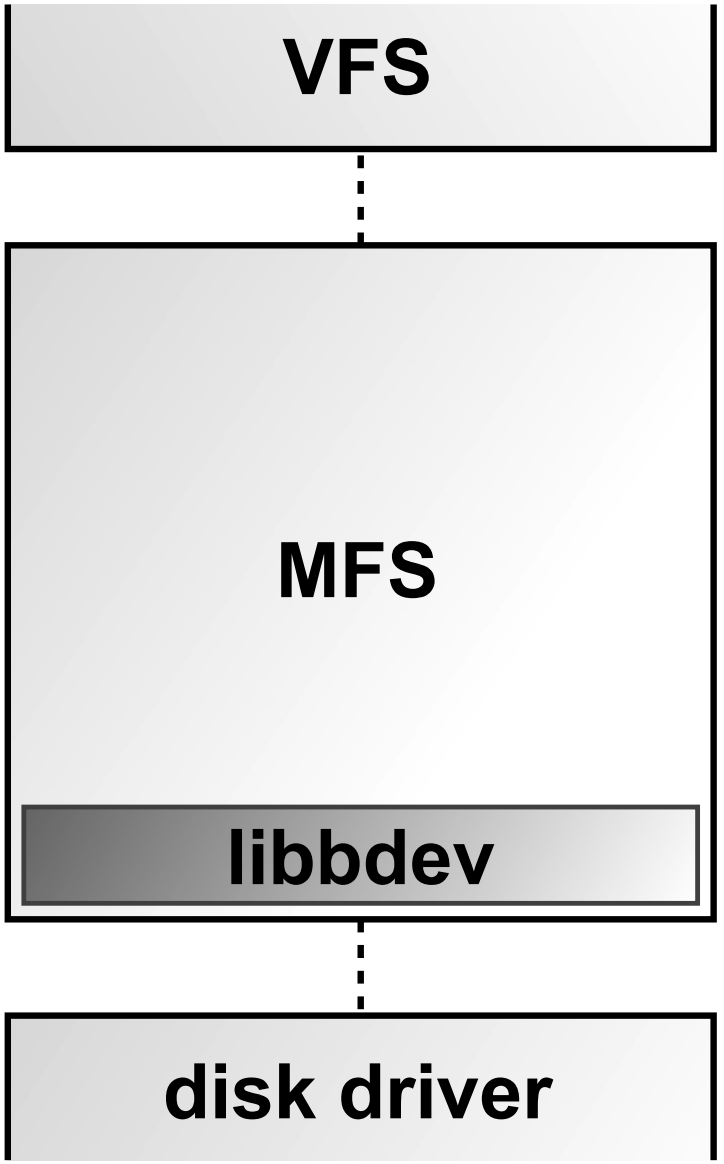
Disk-backed file systems

- Not much progress
  - 2006: “FS” service split into VFS and MFS
  - 2016: two more disk-backed FSes (ext2, isofs)
- It is *pretty hard* to write a file system service
  - Implementing e.g. **rename** is seriously difficult
  - ext2 and isofs started as copies of MFS
  - Maintenance nightmare!
- What can we do to improve on this?

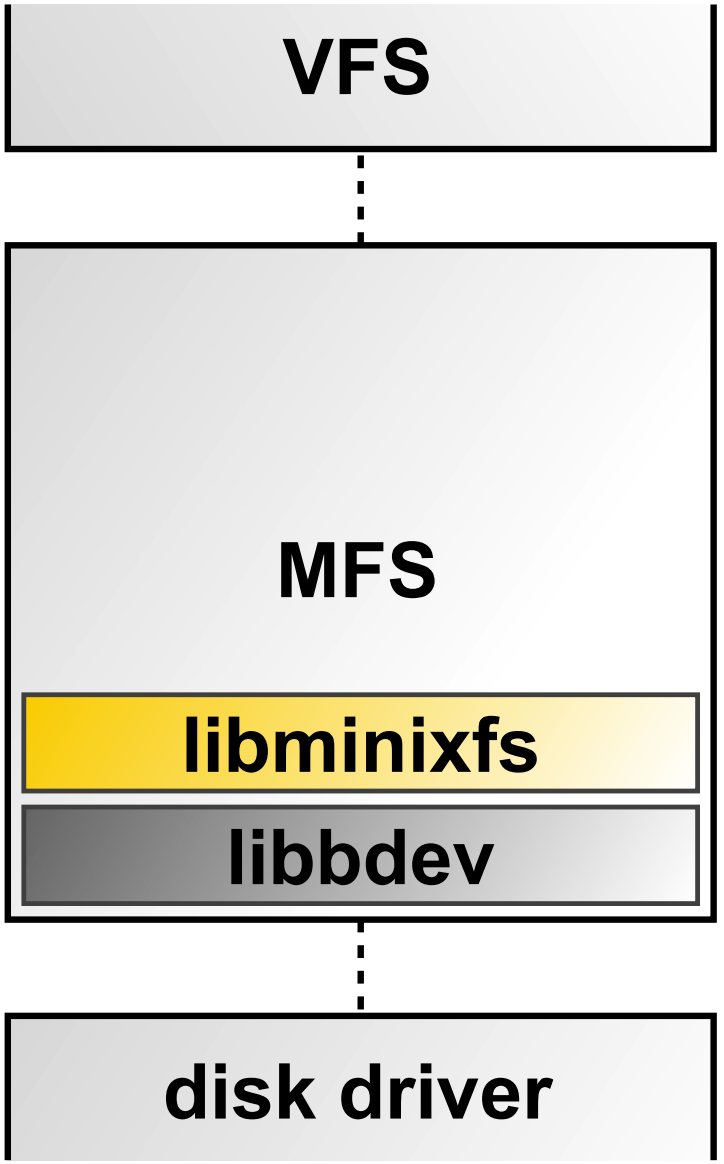
Some first steps



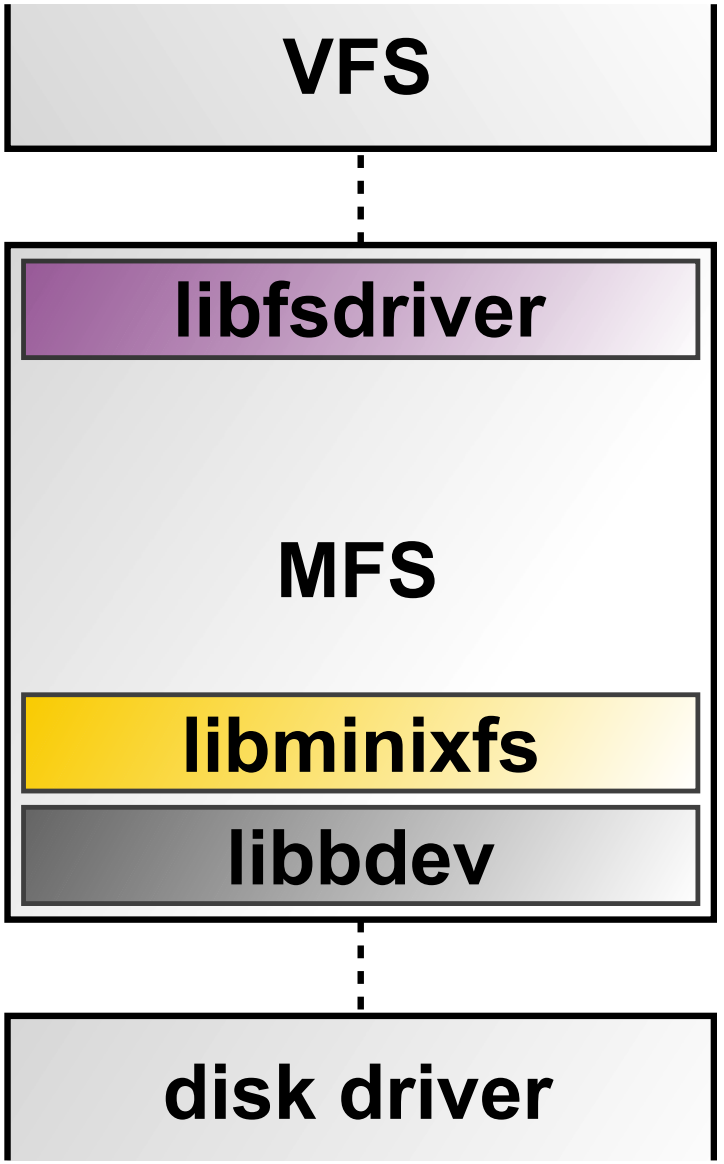
Some first steps



Some first steps



Some first steps



Towards decomposition

- File systems are **graphs**

Towards decomposition 2

- File systems are **graphs**
- Most file system layouts have UNIX origins
  - The graph nodes are *inodes*
  - The graph edges are *directory entries*
  - FS operations: series of node and edge operations

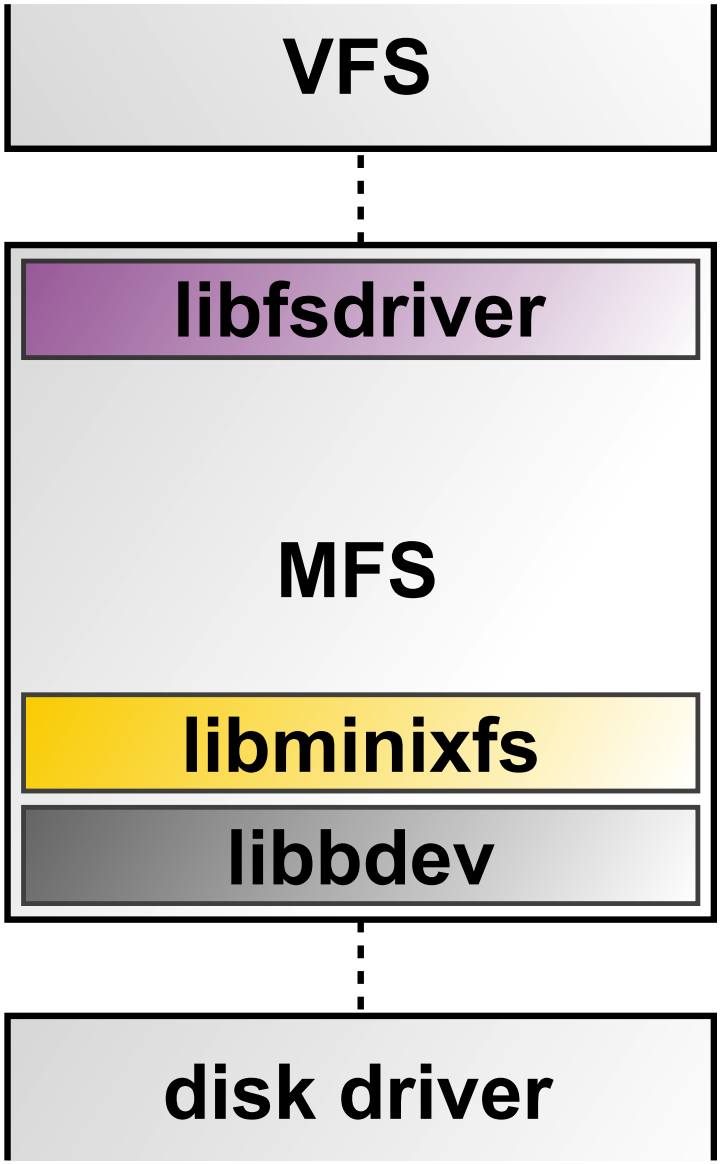
Towards decomposition 2

- File systems are **graphs**
- Most file system layouts have UNIX origins
  - The graph nodes are *inodes*
  - The graph edges are *directory entries*
  - FS operations: series of node and edge operations
- To **create** a file, one needs to..
  - Allocate a new inode
  - Create a directory entry for that inode

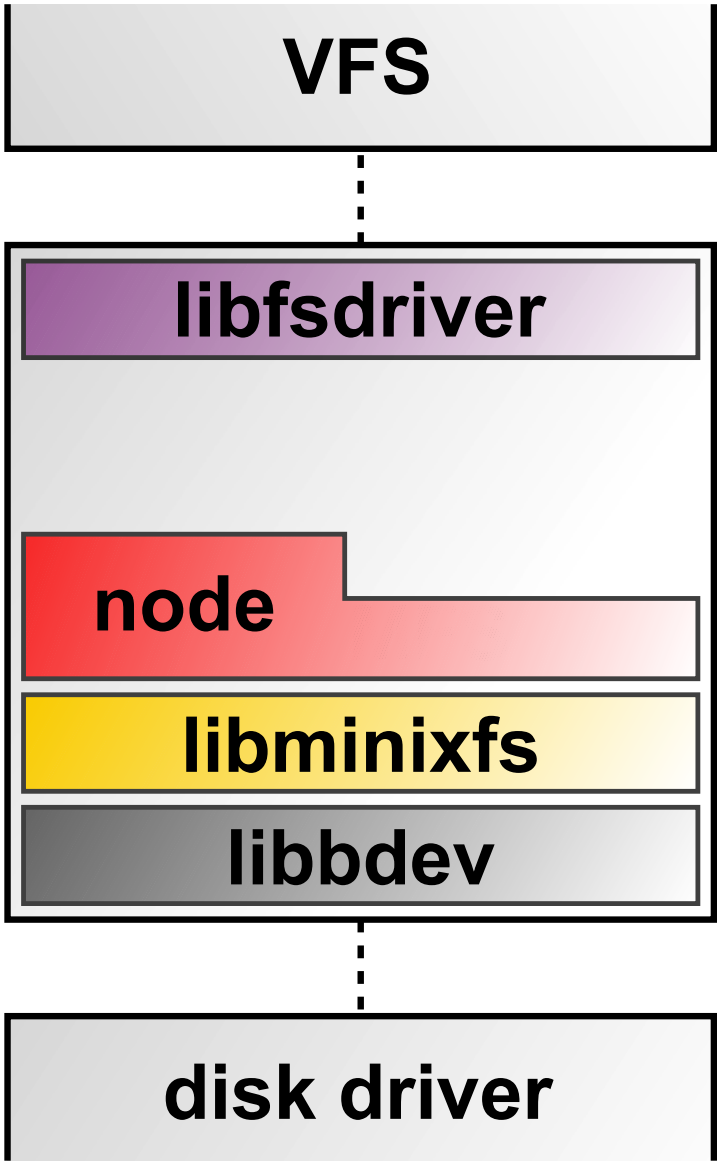
Towards decomposition 2

- File systems are **graphs**
- Most file system layouts have UNIX origins
  - The graph nodes are *inodes*
  - The graph edges are *directory entries*
  - FS operations: series of node and edge operations
- To **create** a file, one needs to..
  - Allocate a new inode
  - Create a directory entry for that inode
- This **graph-level logic** is (mostly) generic!
  - That means we can **reuse** that part

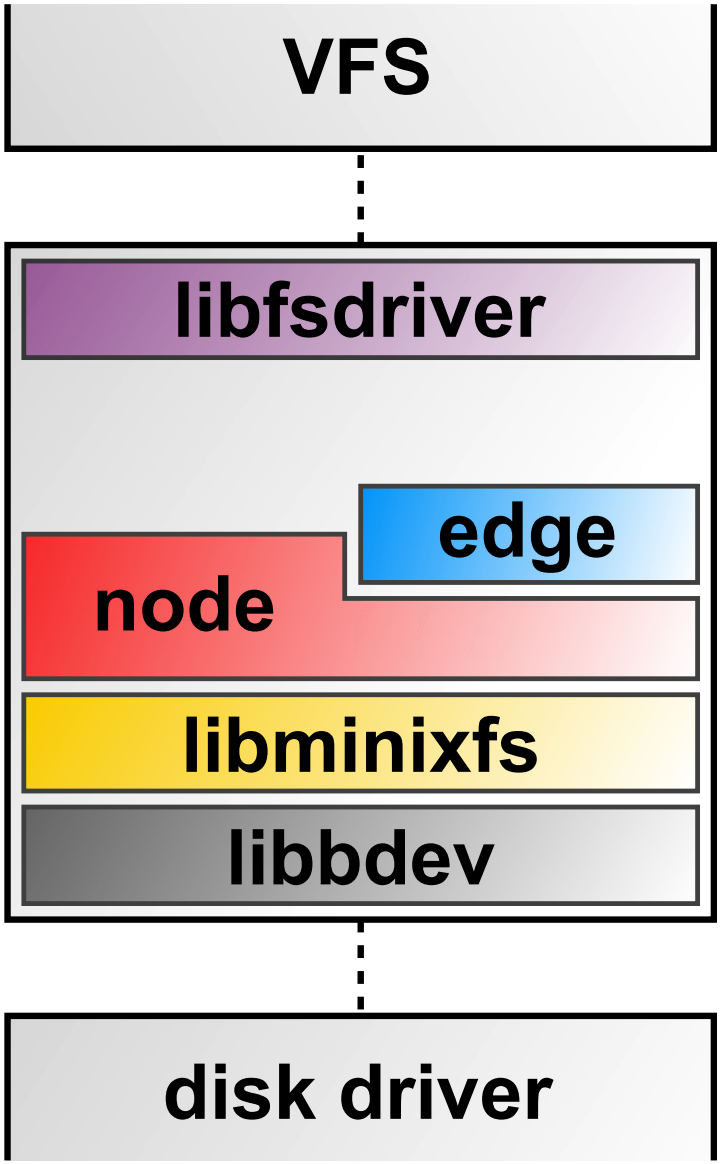
Splitting up the file system 2



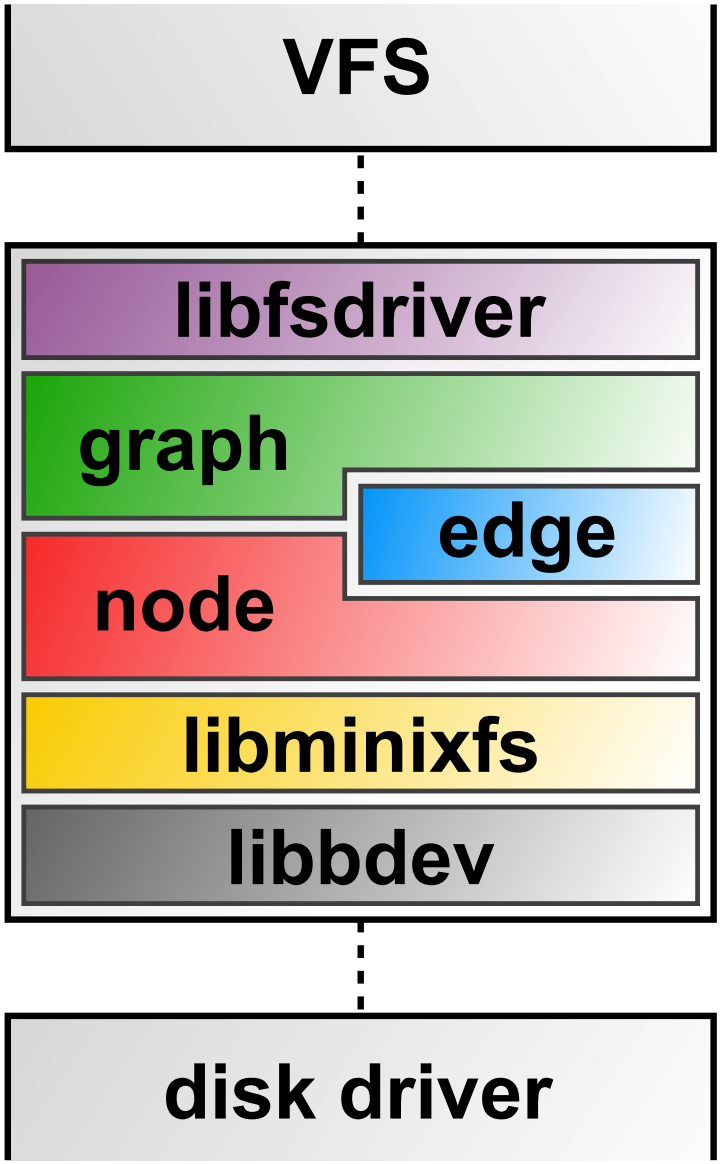
Splitting up the file system 2



Splitting up the file system 2



Splitting up the file system 2



Current status and future

- **TwinFS**
  - A new crash-consistent file system layout

Current status and future

- **TwinFS**
  - A new crash-consistent file system layout
- **Prototype**
  - Graph layer: 2,847 LoC (reusable!)
  - Edge layer: 678 LoC
  - Node layer: 3,010 LoC

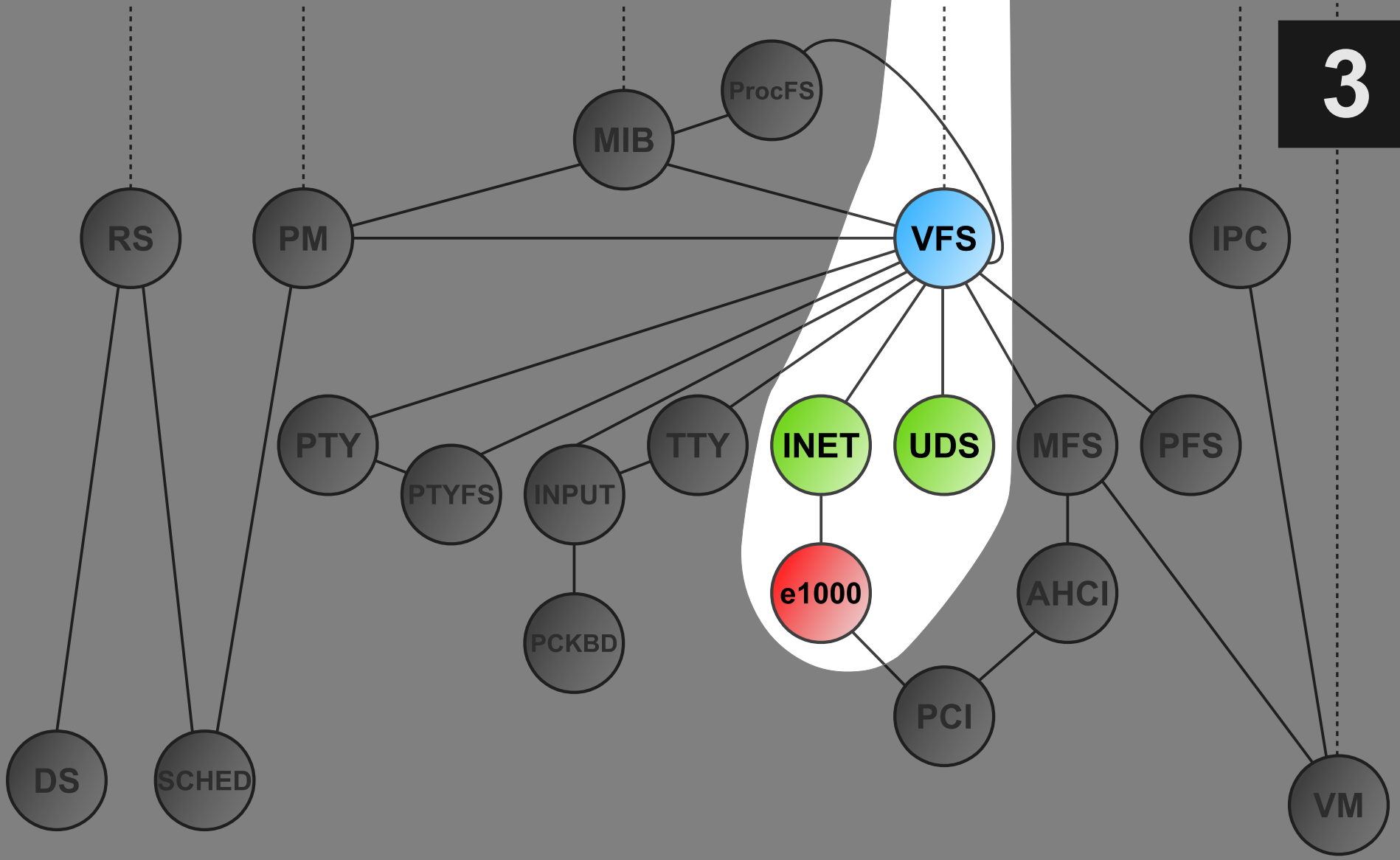
Current status and future

- **TwinFS**
  - A new crash-consistent file system layout
- **Prototype**
  - Graph layer: 2,847 LoC (reusable!)
  - Edge layer: 678 LoC
  - Node layer: 3,010 LoC
- Not merged yet – needs testing

Current status and future

- **TwinFS**
  - A new crash-consistent file system layout
- **Prototype**
  - Graph layer: 2,847 LoC (reusable!)
  - Edge layer: 678 LoC
  - Node layer: 3,010 LoC
- Not merged yet – needs testing

- A way forward to *simplify* writing FS services



Network stack redesign

A tale of scope creep

- **Original goal**: IPv6 support

A tale of scope creep

- **Original goal**: IPv6 support
- “Hmm, while I'm here..”

A tale of scope creep

- **Original goal**: IPv6 support
- “Hmm, while I'm here..”
- **Now**: three subprojects
  - Native BSD socket API
  - Replacing the TCP/IP stack
  - Revisiting the packet level

Native BSD socket API

- INET and UDS are **character drivers**
  - BSD socket API (*socket*, *bind*, *connect*..) in **libc**
  - This approach has several **problems**

Native BSD socket API

- INET and UDS are **character drivers**
  - BSD socket API (*socket*, *bind*, *connect*..) in **libc**
  - This approach has several **problems**
- **Solution**: turn BSD socket calls into syscalls
  - Turn INET and UDS into “socket drivers”
  - VFS forwards socket calls to right socket drivers

Native BSD socket API

- INET and UDS are **character drivers**
  - BSD socket API (*socket*, *bind*, *connect*..) in **libc**
  - This approach has several **problems**
- **Solution**: turn BSD socket calls into syscalls
  - Turn INET and UDS into “socket drivers”
  - VFS forwards socket calls to right socket drivers
- **Current status**
  - All infrastructure complete (but untested)
  - Almost done converting UDS to socket driver

Replacing the TCP/IP stack 3

- **Requirements** for a new TCP/IP stack:
  - Comes with IPv6 support
  - Interface-compatible with NetBSD
  - Maintainable

Replacing the TCP/IP stack 3

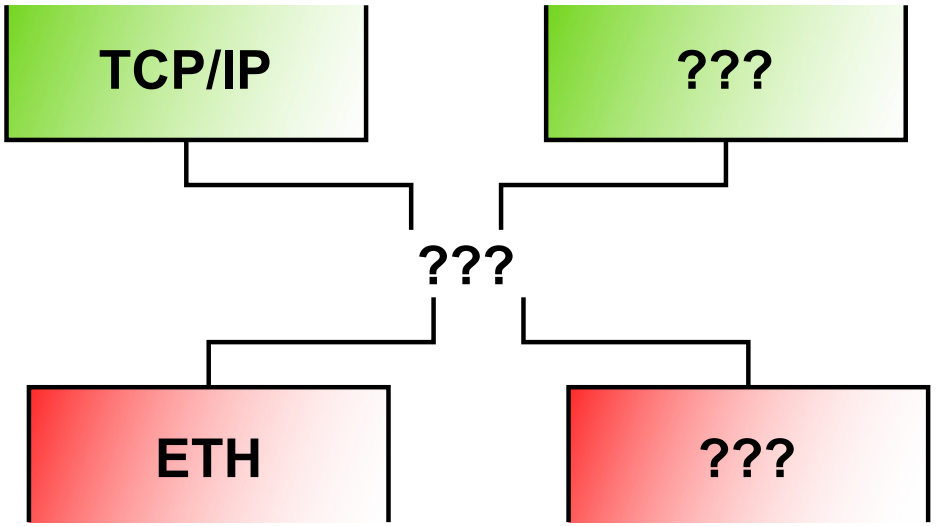
- **Requirements** for a new TCP/IP stack:
  - Comes with IPv6 support
  - Interface-compatible with NetBSD
  - Maintainable
- Internal discussions yielded three **options**:
  1. Import a more recent INET from minix-vmd
  2. Reimplement the wrapper around lwIP
  3. Use RUMP to extract NetBSD's TCP/IP stack

Replacing the TCP/IP stack 3

- **Requirements** for a new TCP/IP stack:
  - Comes with IPv6 support
  - Interface-compatible with NetBSD
  - Maintainable
- Internal discussions yielded three **options**:
  1. Import a more recent INET from minix-vmd
  2. Reimplement the wrapper around lwIP
  3. Use RUMP to extract NetBSD's TCP/IP stack
- Leaning heavily towards option **2**

Revisiting the packet level

- Preparing for the **future**
  - Support for other protocol families?
  - Support for non-ethernet devices?
  - Support for a firewall?



Conclusion

- The service layer is evolving rapidly
  - MINIX is growing up and catching up
- That calls for proper software engineering
  - Not just **adding** functionality
  - But also **restructuring** what is already there
  - And **reducing** redundancy
- The main concern is **maintenance**!

How to contribute (1)

- **We could use your help!**
- Code development...
  - Porting more of NetBSD userland
  - Writing a device driver
  - Filling in missing functionality
- ...and other activities
  - Documentation
  - PR work
  - Testing

How to contribute (2)

- **Information and wishlists**
  - Wiki: http://wiki.minix3.org
- **Source code, bug reports, submitting code**
  - GitHub: https://github.com/Stichting-MINIX-Research-Foundation/minix
- **Getting support**
  - Newsgroup: http://groups.google.com/group/minix3
  - IRC: FreeNode #minix-dev

End of talk

david@minix3.org