# Countering IPC Threats in Multiserver Operating Systems

*Jorrit N. Herder*
*Vrije Universiteit Amsterdam*

14th Pacific Rim Symposium on Dependable Computing
Taipei, Taiwan December 15-17, 2008

# Multiserver Operating Systems

- Potential to improve dependability
  - Each module run as independent process
    - Robustness via address-space separation
    - Fine-grained control over privileges
    - ...
- More complex IPC model required
  - Direct function calls no longer possible
    - Instead, pass messages between modules


Jorrit N. Herder <jnherder@cs.vu.nl>

# IPC Example: I/O Request

- Driver builds message in its memory

| m_source | m_type | message arguments | |
|---|---|---|---|
| PRINTER | DEVIO | Port 0xAB | |

- Driver sends message to kernel
- Kernel does I/O and returns results

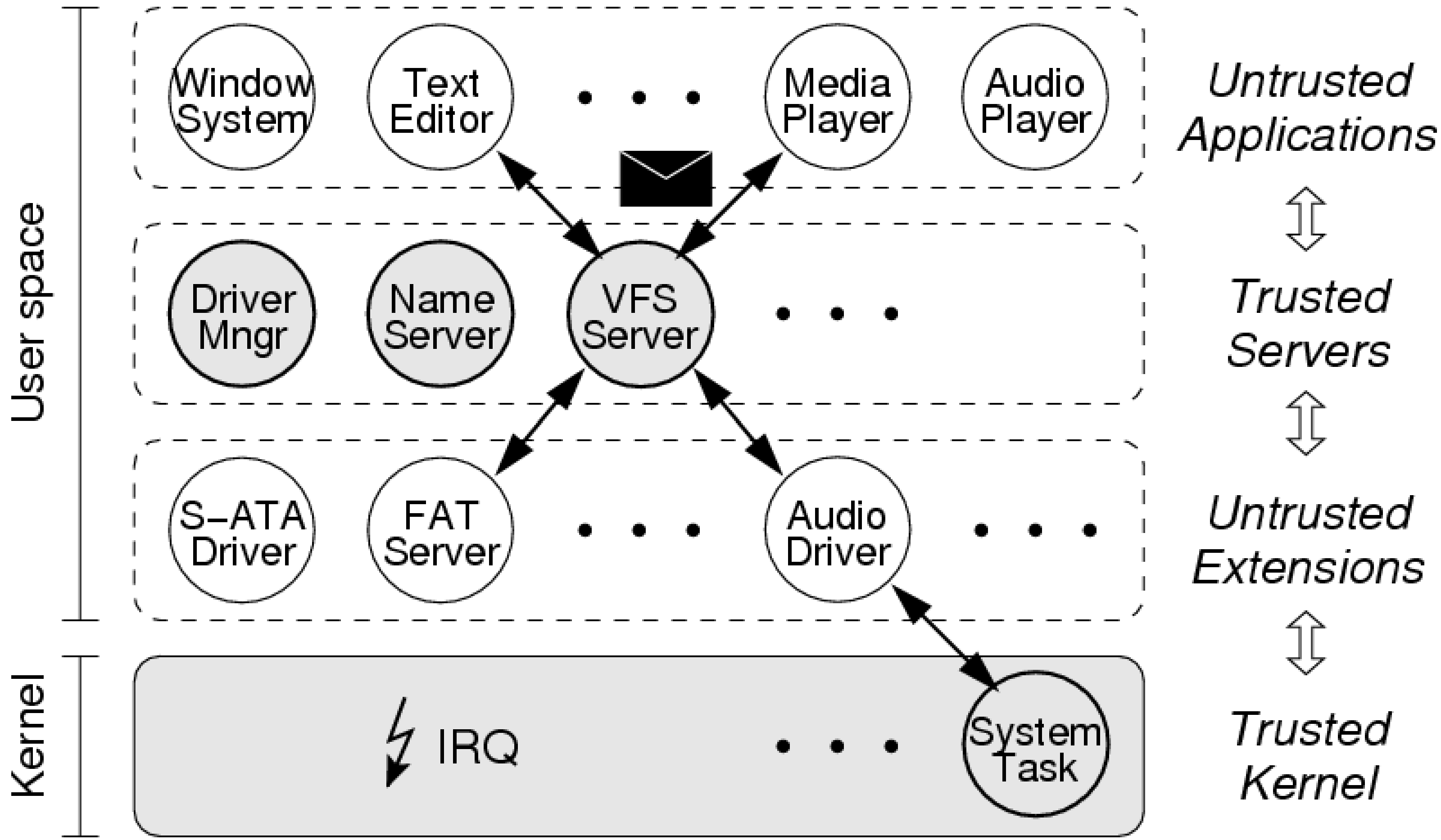
| | | | |
|---|---|---|---|
| SYSTEM | OK | Port 0xAB | Result CD |

# Potential Source of Problems

- Very complex IPC patterns can exist
  - Booting MacOS X: 102,885 Mach IPC calls
  - MINIX 3 POSIX read/write: >25 messages
  - Background activity: 476 messages/sec
- Not all processes can be trusted
  - E.g., device drivers may contain bugs


Jorrit N. Herder <jnherder@cs.vu.nl>

# Unreliable IPC to/from Drivers





Jorrit N. Herder <jnherder@cs.vu.nl>

# Agenda for Today

- IPC threat model
- MINIX 3 IPC infrastructure
- IPC defense mechanisms
- Fault-injection testing
- Time for questions


Jorrit N. Herder <jnherder@cs.vu.nl>

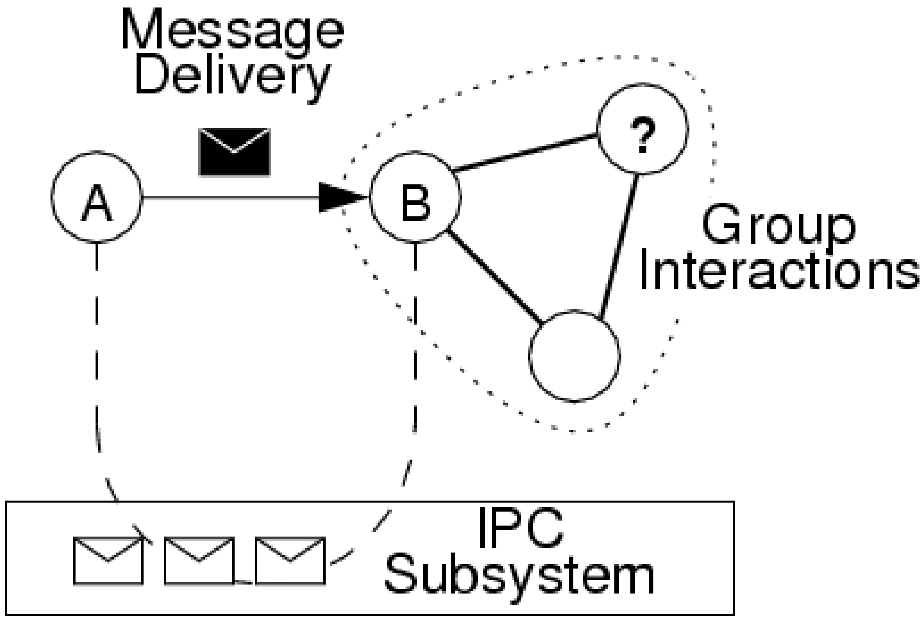
# IPC Infrastructure Assumptions

- IPC implementation is easy to get correct
  - IPC calls are atomic operations
  - Messages cannot get lost
  - IPC endpoints cannot be forged
  - Message integrity preserved
  - Isolation between IPC calls
  - Confidentiality of IPC traffic

# Still Many IPC Threats Exist

- IPC threats due to protocol violations
  - 3 orthogonal threat classes

- Focus is on drivers
  - typically 3rd party
  - up to 70% of code
  - 3-7x more buggy
  - 85% of crashes





Jorrit N. Herder <jnherder@cs.vu.nl>

# Threats (a) IPC Subsystem

- Call parameters
  - Bad IPC primitive
  - Nonexisting endpoint
  - Illegal message buffer
- Global resources
  - Memory exhaustion
  - CPU exhaustion

# Threats (b)  Message Delivery

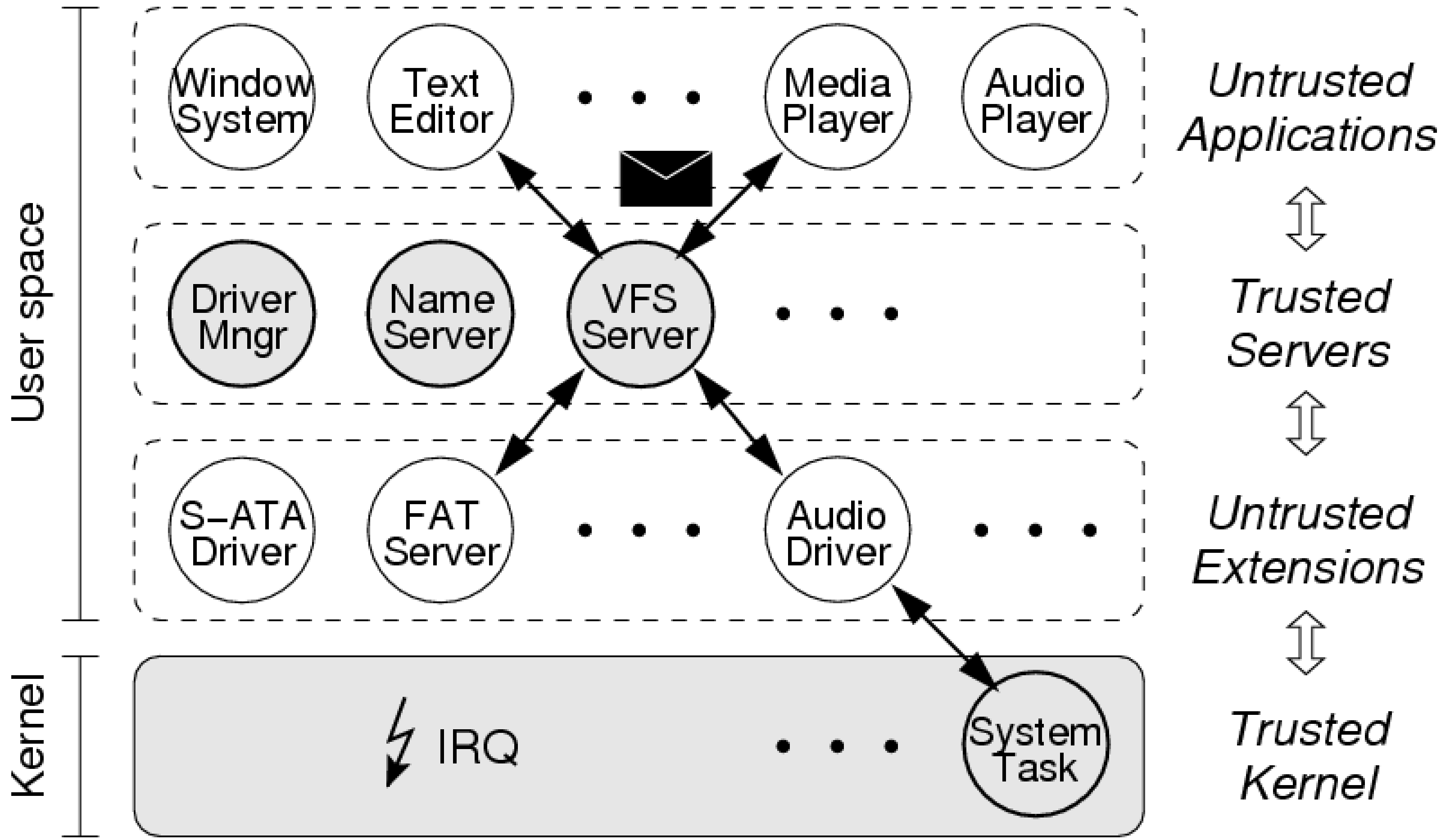
- Addressing
  - Unauthorized IPC target
  - Unintended IPC target
  - Spoofing (if endpoints are dynamic)
- Message contents
  - Oversized message
  - Bad message contents

# Threats (c) Group Interactions

- Flow control
  - Scheduling order
  - Denial of service
- Caller blockage
  - Deadlocks
  - Asymmetric trust


Jorrit N. Herder <jnherder@cs.vu.nl>

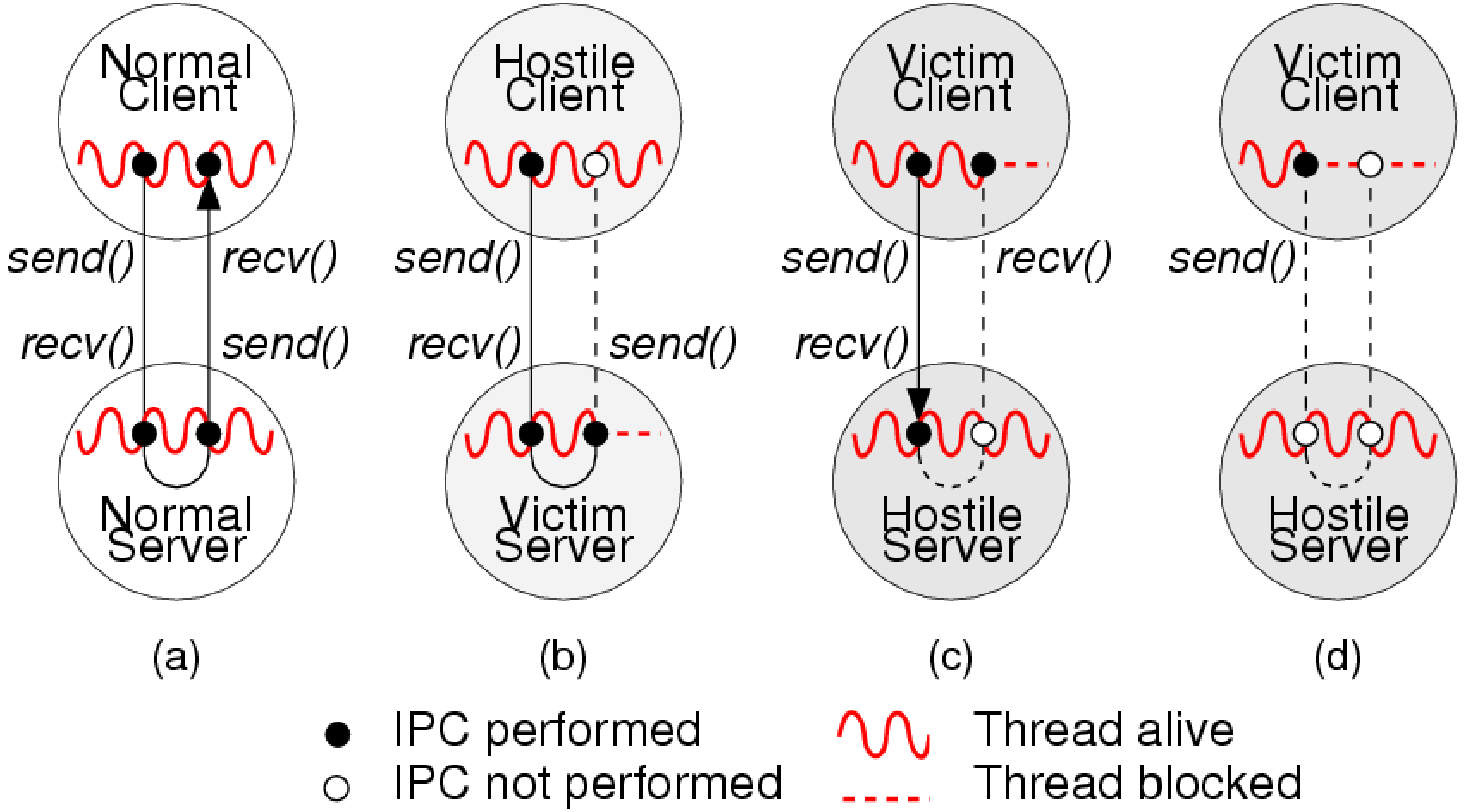
# Trust in OS is Asymmetric





Jorrit N. Herder <jnherder@cs.vu.nl>

# Extended Asymmetric Trust Model
## (for synchronous IPC)




 Jorrit N. Herder <jnherder@cs.vu.nl>

# Design Choices and Trade-offs

- Asymmetric trust most influential threat
  - Other threats (often) simple to counter
- Several IPC defenses possible
  - Language support
  - Timeouts to abort failed IPC
  - One thread per untrusted party
  - Asynchronous and nonblocking IPC

# MINIX 3 Infrastructure

- IPC calls (primitives)
  - Synchronous SEND, SENDREC, RECEIVE
  - Nonblocking NBSEND
  - Asynchronous ASEND, NOTIFY
- Small, fixed-length messages
- Temporally unique IPC endpoints
- Restrictions on IPC calls and destinations

# Defenses (a) IPC Subsystem

- Call parameters
  - Switch upon IPC primitive; default is error
  - Verify endpoint is listed in process table
  - Verify message buffer is in address space
- Global resources
  - No dynamic resource allocation for messages
  - MLFQ scheduler prevents CPU exhaustion

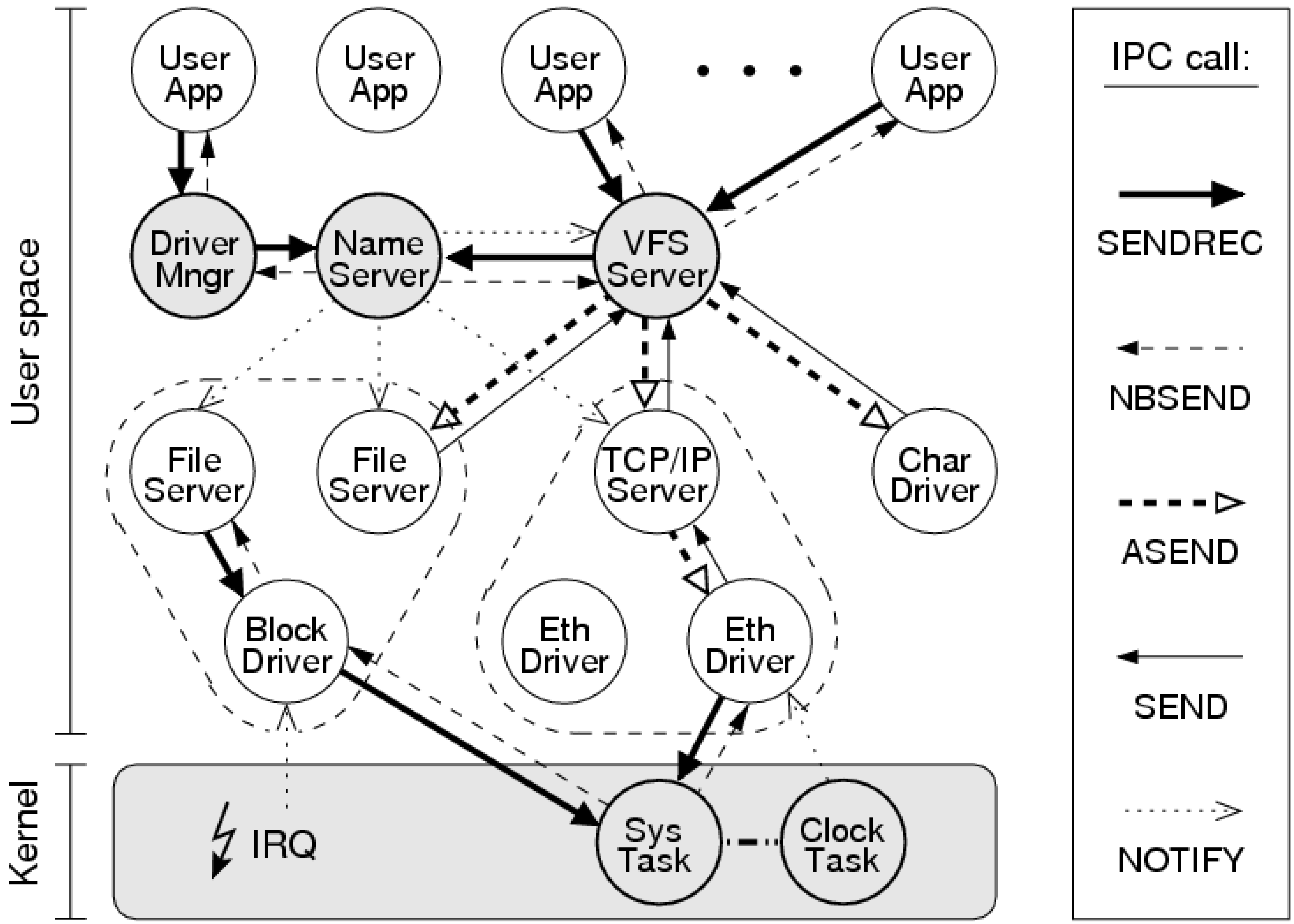
# Defenses (b) Message Delivery

- Addressing
  - Driver policy restricts possible destinations
  - Name server maps endpoints onto services
- Message contents
  - Kernel validates pointer to message buffer
  - Fixed size prevent memory corruption
  - Receiver must check message contents

# Defenses (c) Group Interactions

- Flow control
  - IPC subsystem uses FIFO queuing
  - MLFQ scheduler prevents denial of service
- Caller blockage
  - IPC protocol with safe message ordering
  - Asynchronous and nonblocking IPC
    - For all IPC to untrusted processes
    - Dominates resulting IPC architecture

# MINIX 3  IPC Interactions





Jorrit N. Herder <jnherder@cs.vu.nl>

# Fault-injection Testing

- Inject faults in driver binary at run-time
- Faults mimic OS programming errors
- Testing was done in an iterative process
  - Various bug fixes and a major design change
    - Invalid IPC endpoint caused kernel panic
    - Nonblocking flag on SENDREC not detected
    - Unauthorized IPC due to bad driver policy
    - Asymmetric trust issues led to design change

# Fault-injection Results
## (For 1,000,000 randomly selected faults)

- IPC call denied by IPC subsystem
  - bad IPC primitive 915
  - bad IPC endpoint 14,542
  - bad message buffer 202,223
  - unauthorized IPC primitive 0
  - unauthorized IPC endpoint 1,260
- Bad message contents detected
  - illegal kernel requests 1,524,516


Jorrit N. Herder <jnherder@cs.vu.nl>

# Summary & Conclusion

- Classification of IPC threats
- Extended asymmetric trust model
- IPC design choices and trade-offs
- MINIX 3's dependable IPC architecture
- Results of fault-injection testing


Jorrit N. Herder <jnherder@cs.vu.nl>

# Time for Questions

- Acknowledgements
  - Herbert Bos
  - Ben Gras
  - Philip Homburg
  - Andy Tanenbaum
- More information
  - www.minix3.org


Jorrit N. Herder <jnherder@cs.vu.nl>