

# **Duplex: A Reusable Fault Tolerance Extension Framework for Network Access Devices**

Srikant, Jiawu, Wei,  
Kartik, Tzi-cker

**SUNY Stony Brook**

**Rether Networks Inc**



# Outline

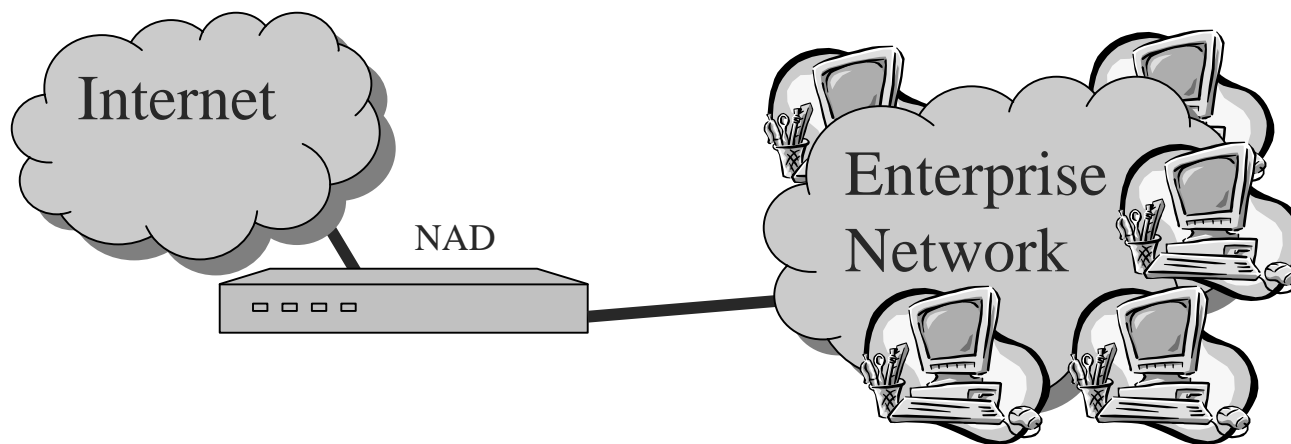
---

- Introduction
- Duplex framework
- Implementation details
- Performance measurement with an instance of Duplex - ISMD
- Issues with the Duplex framework

# Introduction

---

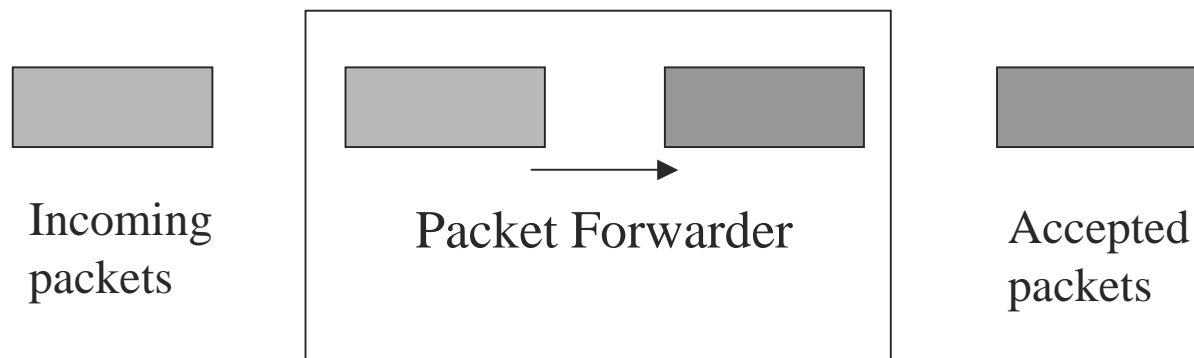
- Network Access Devices
  - Connect an enterprise to the Internet
  - Perform packet filtering activities
    - Firewalls, traffic shapers, NAT, virus/email scanners, etc.
  - Reliability of NADs is crucial
- Duplex provides a fault-tolerance extension to NADs
  - Minimal changes to existing implementations/logic



# NAD Features

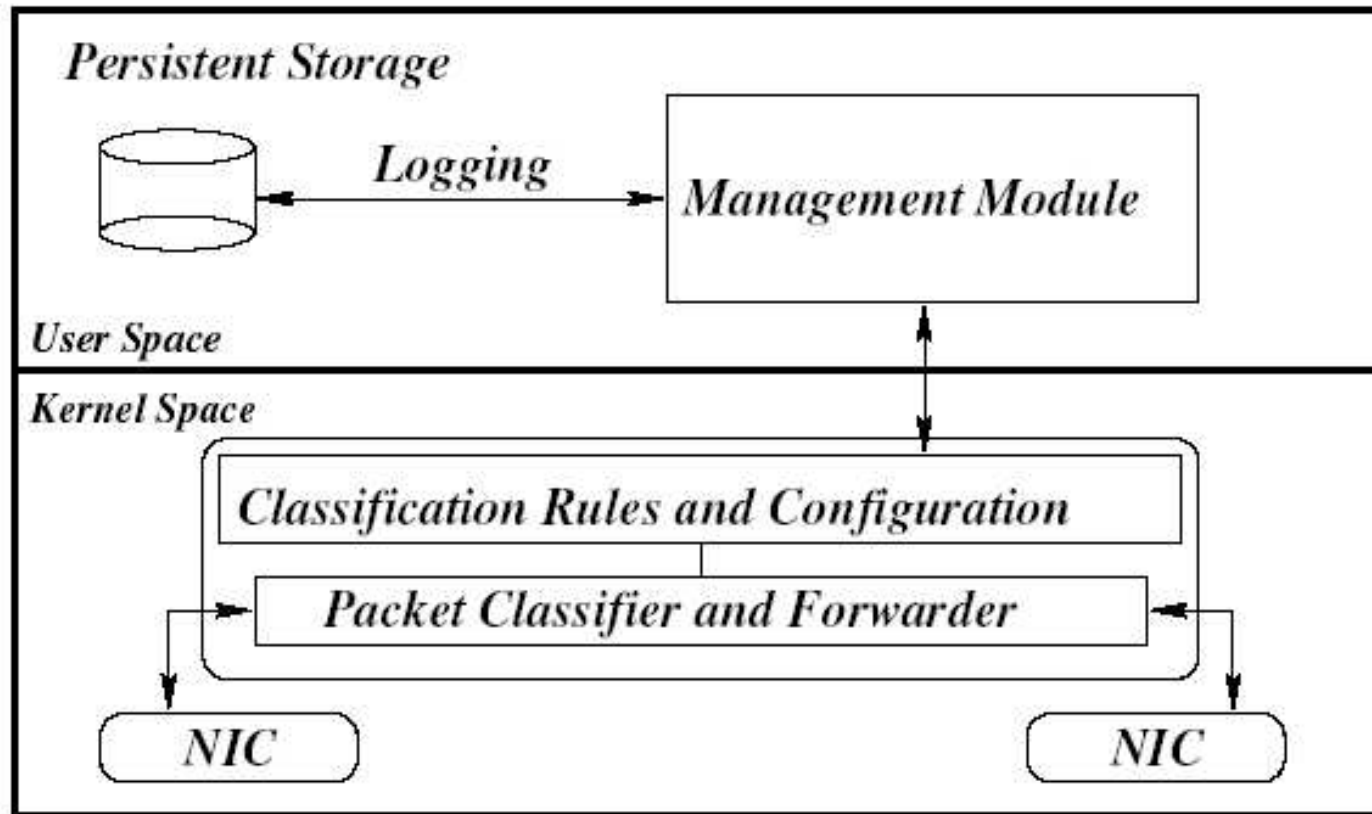
---

- Most NADs share 3 common features
  1. Minimal persistent state.
  2. A small amount of packet loss is acceptable
  3. Internet Connectivity is paramount
- All NADs are similar except for the core functionality



# NAD Components

---



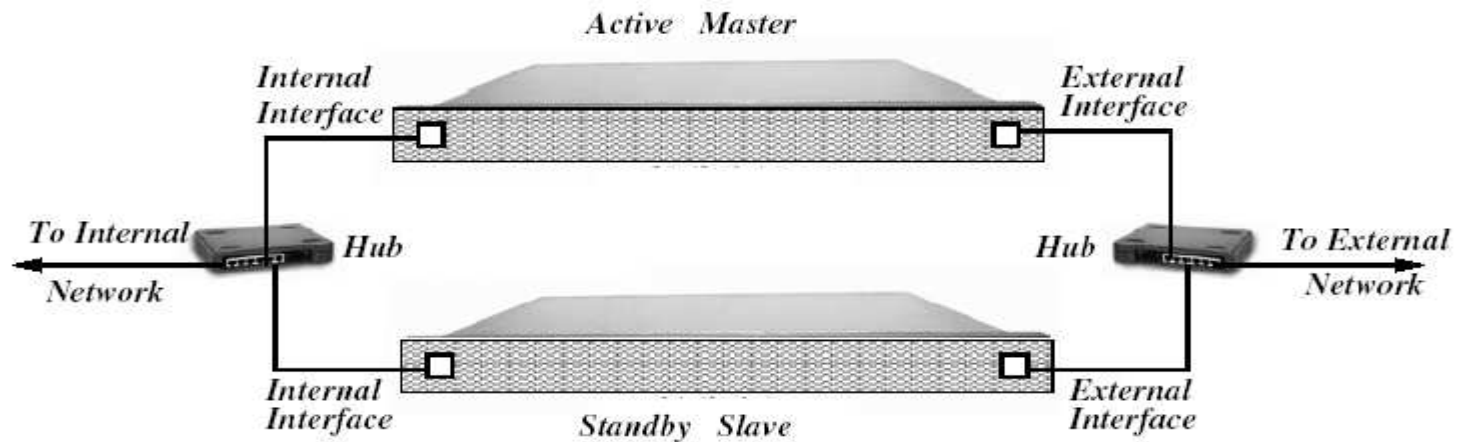
Kernel engine: packet processing

User management: configuration and management

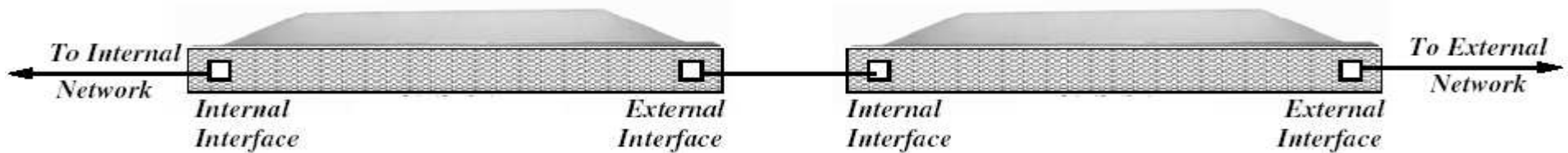
Log/Recovery module: optional

# HW system setup

---

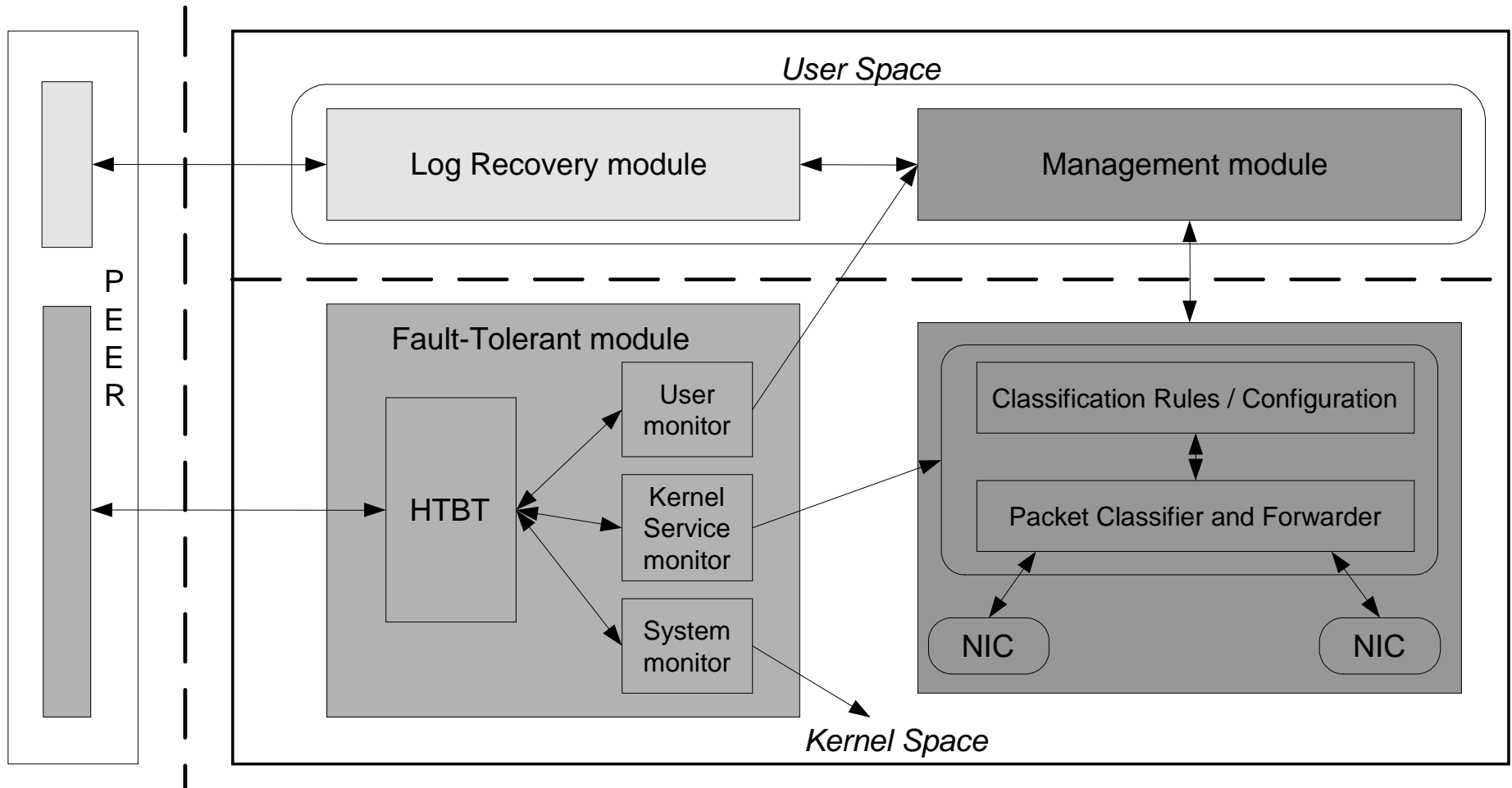


(a) Parallel Mode of Operation



(b) Serial Mode of Operation

# SW modules



# Service Programming Interface

---

- Configuration setup
  - Processes to be monitored
  - Action specification in the event of process status change
  - For each kernel service, specify actions to be taken in case of failure
- Startup
  - Bootstrap and register user processes as specified in configuration file

# Service Programming Interface

---

- User-level liveness check
  - Registering PID list
- Kernel service module liveness check
  - `iam_alive(&var)`
- State logging and recovery
  - `Log_async()`: local disk, Slave
  - `Log_notify()`: for Slave, callback for updated local log
  - `Log_recover()`: recover from a failure
- Export of ***devicestate*** (Master/Slave/...)

# Failure Detection

---

- Implemented in Timer ISR
  - Rely on WDT hardware in case of ISR failure
  - Bypass converts NAD into a passive cable
- Local System Monitoring
  - Kernel service : `iam_alive()` interface
  - User management processes: PIDs
  - User log/recovery process: PID
  - System liveness: system call entry, scheduler entry

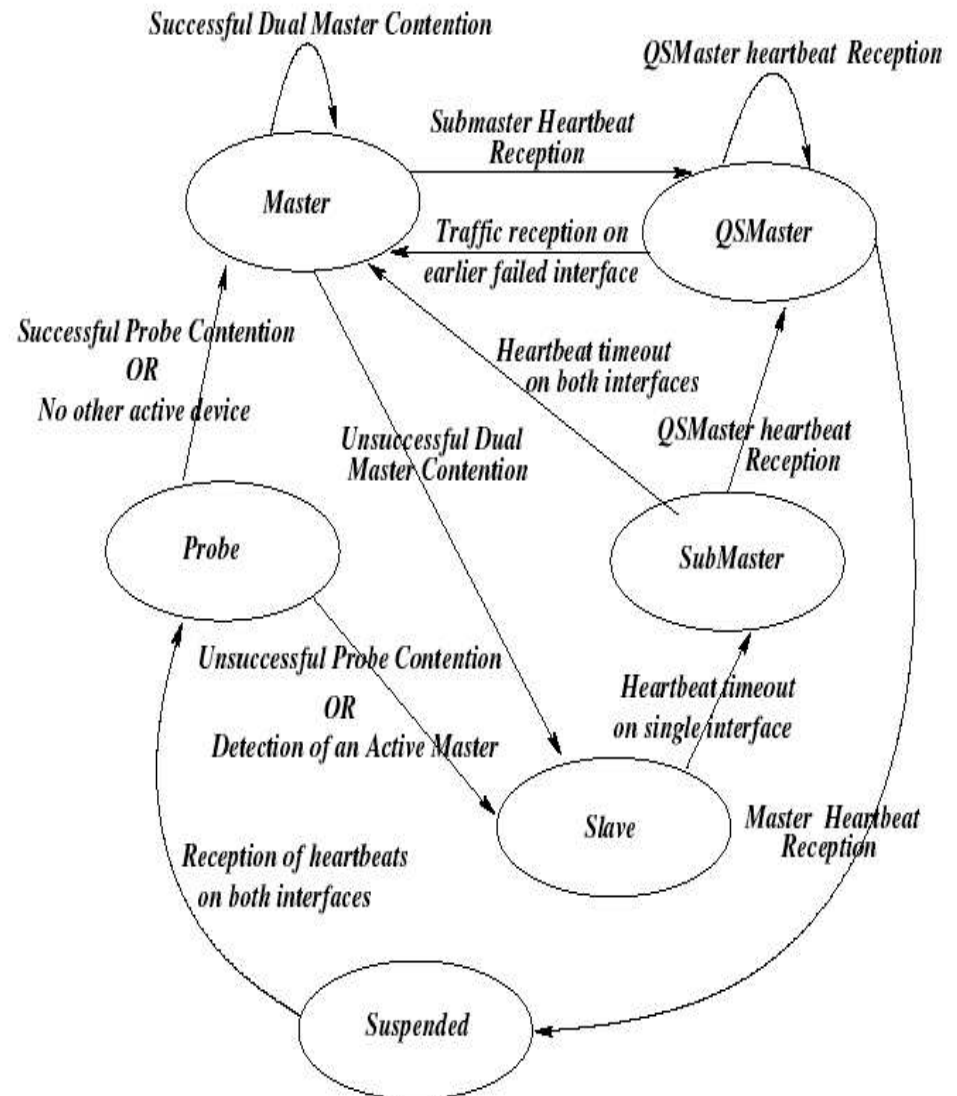
# Failure Detection

---

- Failure detection of peer device
  1. Adaptive Heartbeat mechanism over existing NICs
  2. Two-level timeout mechanism to distinguish HTBT lost and device failure
    - $Hard\_timeout = soft\_timeout + current\_load / 2^{load\_factor}$
    - Soft timeout: increase probe rate (eliminate congestion factor)
  3. No HTBT sent out in case of local failures

# Parallel-dual mode Device States

- Probe
  - Initial state
- Master & Slave
  - Active & hot-standby
- QuasiMaster & SubMaster
  - Intermediate states
- Suspend
  - After failure detection



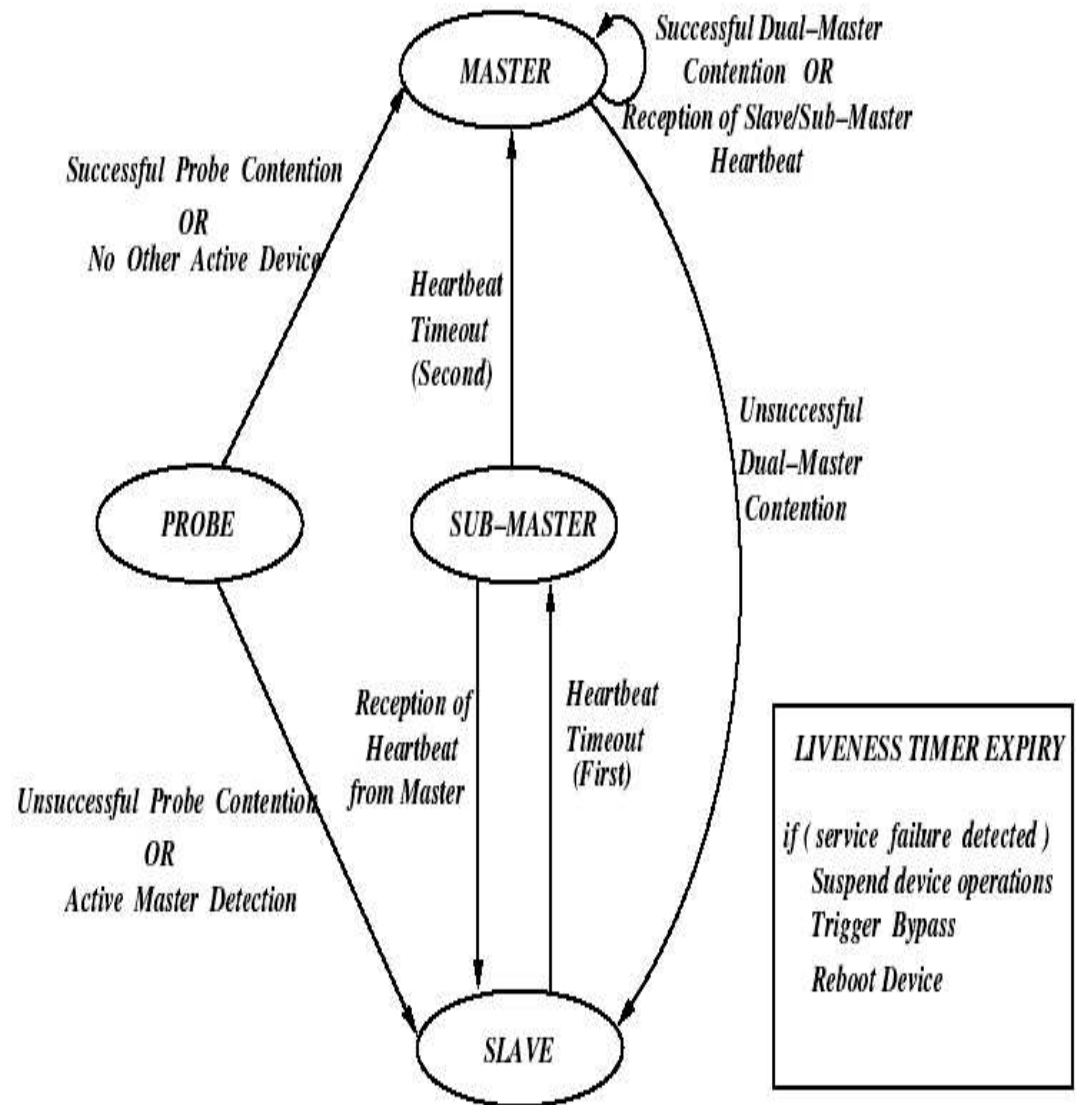
# Parallel-dual mode

---

- State transition triggered by
  - Reception of HTBT packet
  - Absence of HTBT within a certain period of time
- Pessimistic assumption
  - Both devices may enter Master state
    - Transient failure: dual Master contention
    - One device is isolated from the network, so it doesn't matter

# Serial-dual mode Device States

- Probe
  - Initial state
- Master & Slave
  - Active & hot-standby
- Submaster
  - Intermediate failure state



# Serial-dual mode

---

- Interface failure detection
  - Missing traffic on an Interface
  - Response : Bypass triggering
- Local failure detection
  - In the event of local failures, bypass is triggered
  - recovery according to the configuration file
  - System crashes trigger WDT automatically
- A link is always available

# Fast Log Recovery mechanism

---

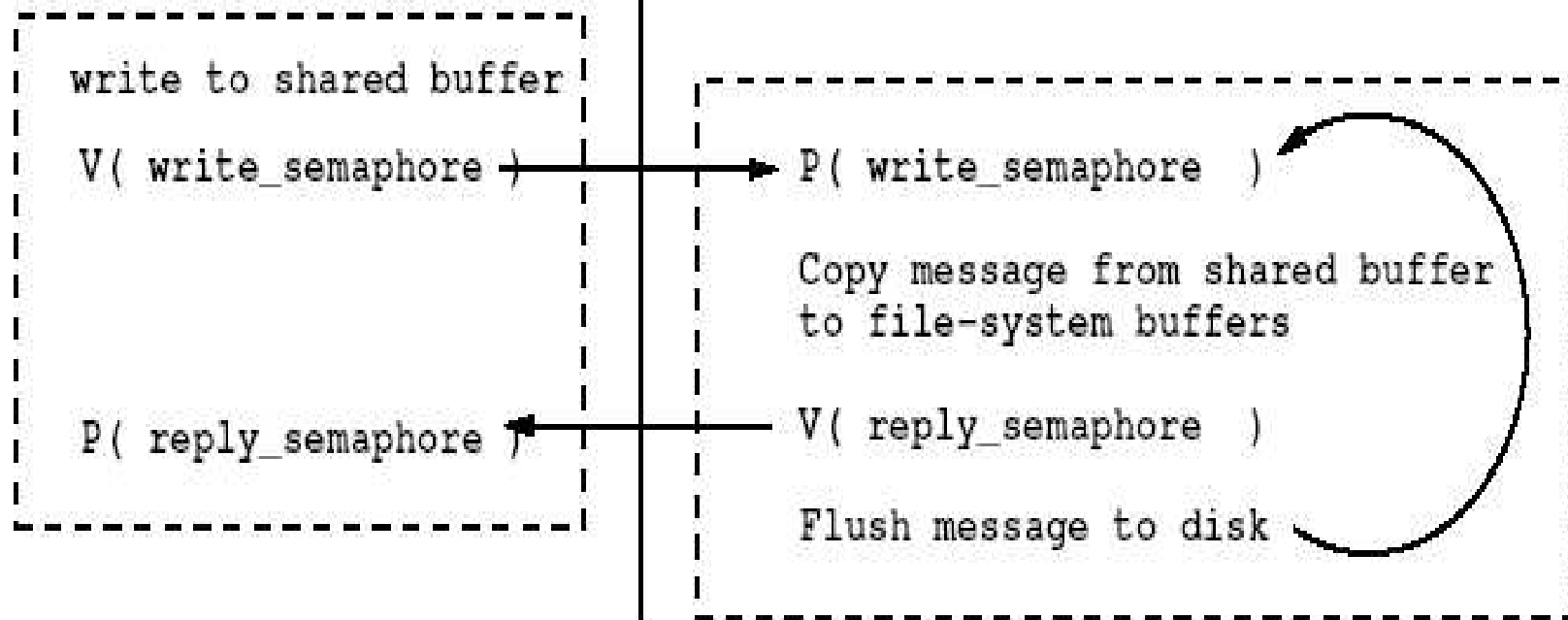
- Synchronous logging is essential for synchronization
- Incremental logging
- User-directed logging (log only when necessary)
- Forked logging: overlapping logging and service since logging is disk-I/O intensive
- Overhead: a memory copy and a context switch
- Higher priority of logging process:
  - Raise with `sys_nice()`
  - Scheduling assumption: service process is always running
- Synchronization between Master/Slave device

# Logging process

---



`log_async(message)`



# **An instance of Duplex: ISMD system**

---

- **Components:**
  - Internet Service Management Device: NAD
  - AS, GUI: web-based management
- **Kernel engine:**
  - forwarder loop
  - Packet scheduling, forwarding, load balancing, NAT, multi-homing
- **User manager daemon:**
  - Communicate with AS-GUI
  - Start kernel kernel engine
  - Log/recovery

# ISMD - Customization

---

- User:
  - Configuration file
  - Insert log/recovery points
- Kernel:
  - Liveness check targets: forwarding loop, system
  - State based forwarding loop: Master/Slave

# Performance: Fail-over and packet loss

---

Traffic (Mbps)	Fail-over time (ms)		Lost Packets	
	Parallel	Serial	Parallel	Serial
0	101	106	0	0
2	129	126	58	1594
5	159	160	189	3721
10	231	220	514	7246
20	346	247	1632	13899
30	457	460	3370	17424
40	588	591	6270	28511
50	717	713	8705	29477
60	833	836	12239	36385
70	952	955	26365	42221
80	1073	1089	32494	48579
90	1174	1197	51225	58902
100	1287	1267	68705	73728

# Analysis

---

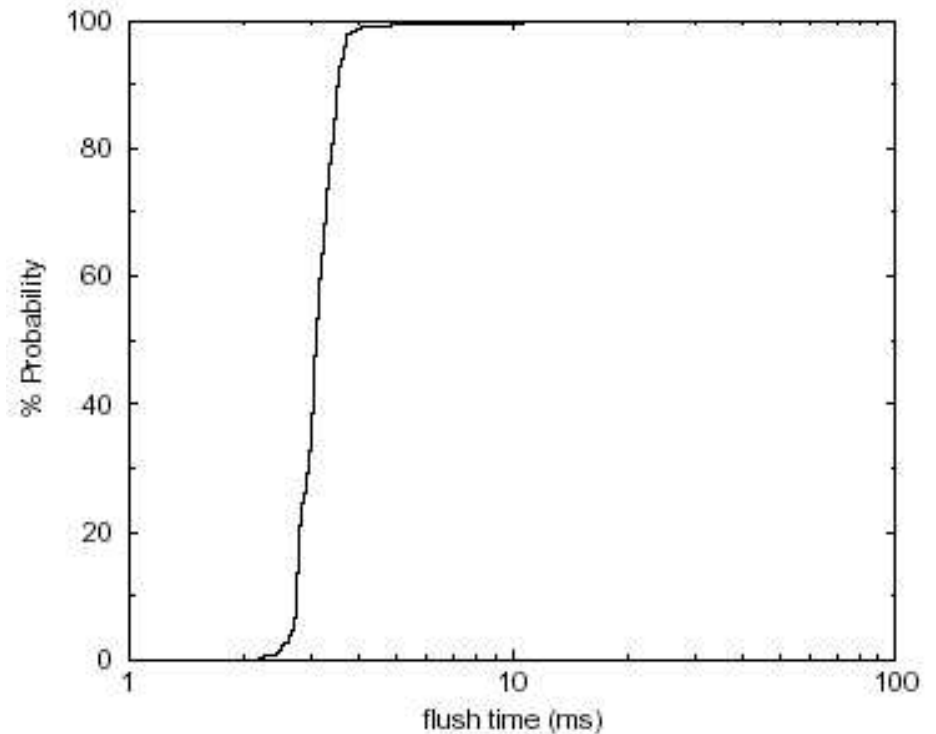
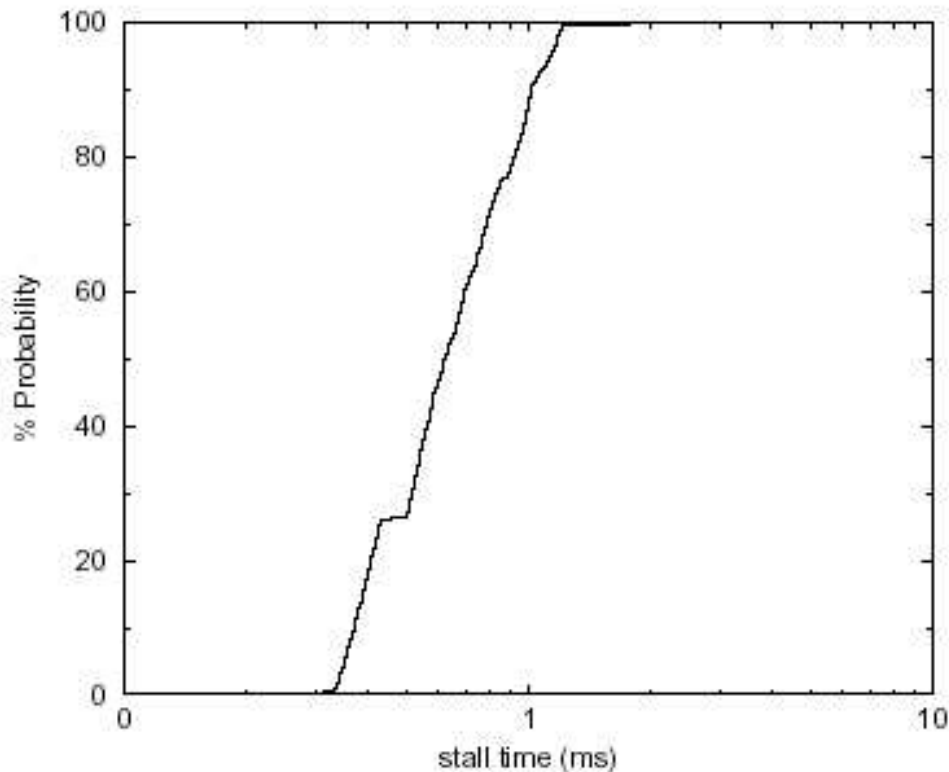
- Packet loss
  - Parallel-dual < serial-dual mode
  - For < 40Mbps, the difference keep increasing.
  - In parallel mode packets are buffered at slave
- The difference is relatively constant between 40Mbps and 80Mbps.
  - (Buffer is full and the parallel dual mode could not benefit more for the buffering mechanism)
- For > 80Mbps, collision dominates the loss

# Analysis

---

- Around 1 second fail-over time
  - minimal impact on the operation of hosting NAD, since its internal buffer memory can easily compensate a service stall time of this magnitude (13MB at 100Mbps rate)
- Availability =  $MTTF / (MTTF + MTTR)$ 
  - $MTTF > 1.16$  days with  $MTTR = 1$  sec for FT system requirement (99.999%)

# Test Result – Logging (WD200 5400rpm)

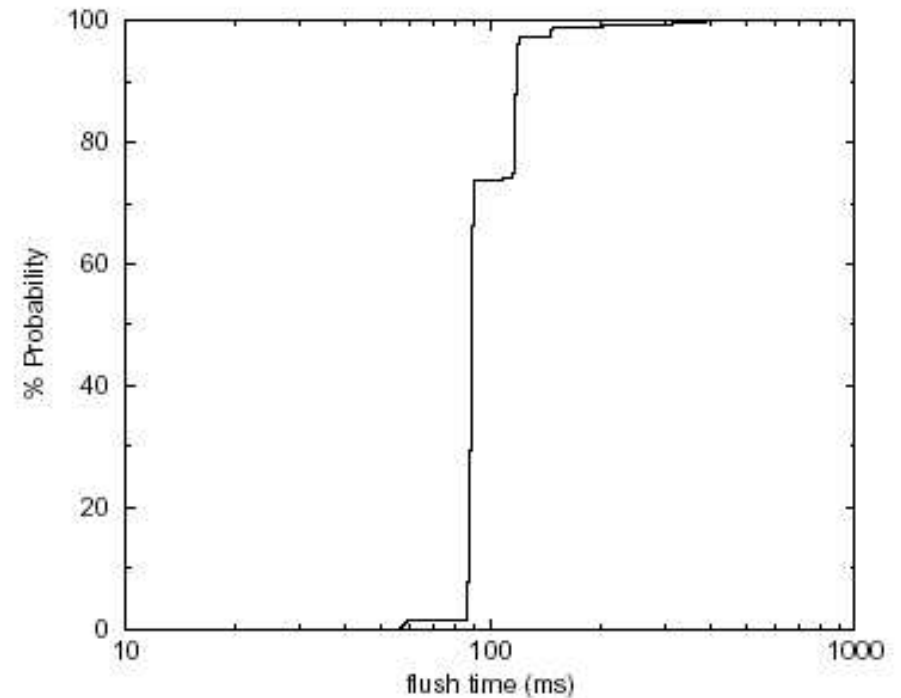
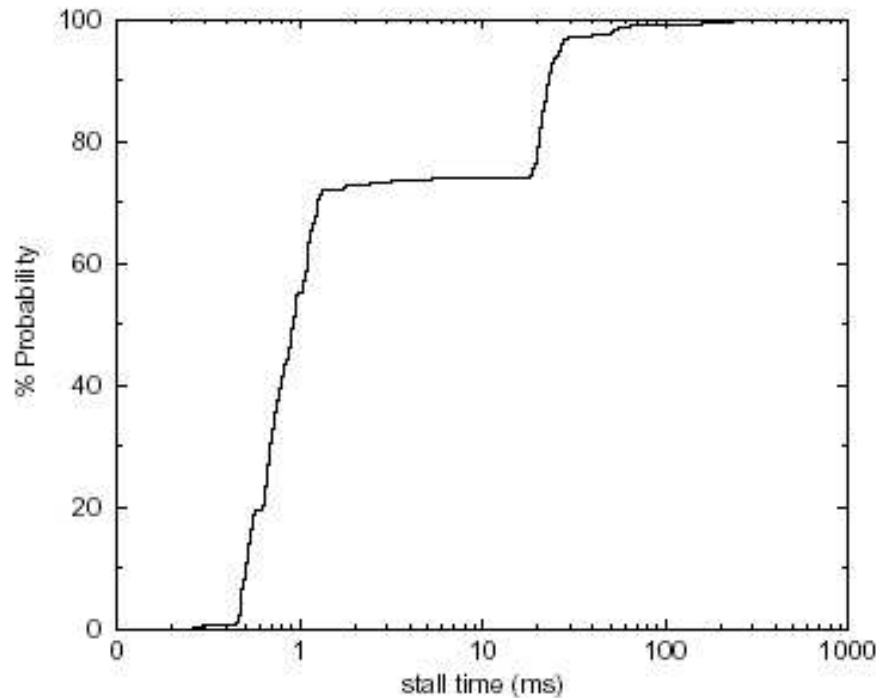


1ms stall time 90% of the time;

3ms flush time 90% of the time

# Test Result – Logging (DOM or Flash)

---



- < 1ms stall time 70% of the time; 1-10ms stall time 90% of the time
- 100 ms flush time for almost all writes

# Issues with Duplex Framework

---

- Scalability (more than 2 nodes)
  - Parallel: BC HTBT and modified State Transition logic
  - Serial: additional HTBT communication paths are required
- Link availability is not enough
  - NAT-like functionality dependent service
  - No QoS guarantee for normal forwarding device
  - Duplex provides a very high availability
- Weak in State based SPI
- HTBT partitioning problem
  - Redundant HTBT path for better failure location