Matthew Dillon
DragonFly BSD Project
23 October 2003

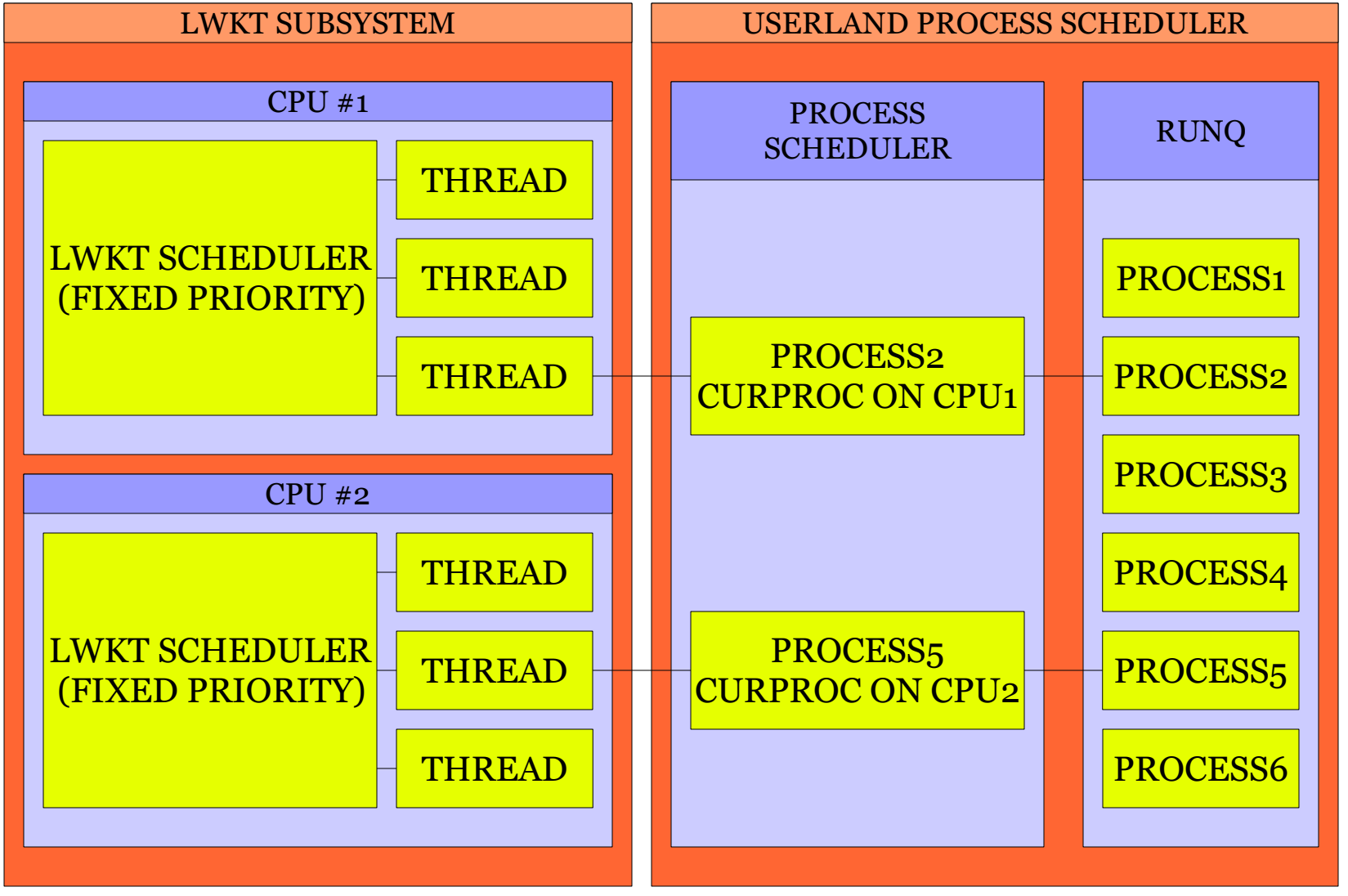
DragonFly Overview

- FreeBSD 4.x and 5.x directions
- Differentiating DragonFly, the basis for a project fork
  - A different, more maintainable user threading API (syscall messaging)
  - A different, more maintainable approach to MP design
    - CPU Isolation by design using IPI messaging rather then by accident w/Mutexes
    - Light Weight Kernel Threading with fewer hacks
- Project Goals
  - Maintaining stability, producing production-capable releases
  - A more consistent and more easily maintained message-based framework
  - UP, MP, SSI Scaleability
  - Userland VFS Development
  - Machine-verified Package Management
- This Presentation
  - Threading And Messaging
  - Our approach to the Big Giant Lock problem. Why not mutexes?
  - Our approach to achieving a Single System Image (SSI)

DragonFly Threading and Messaging Model

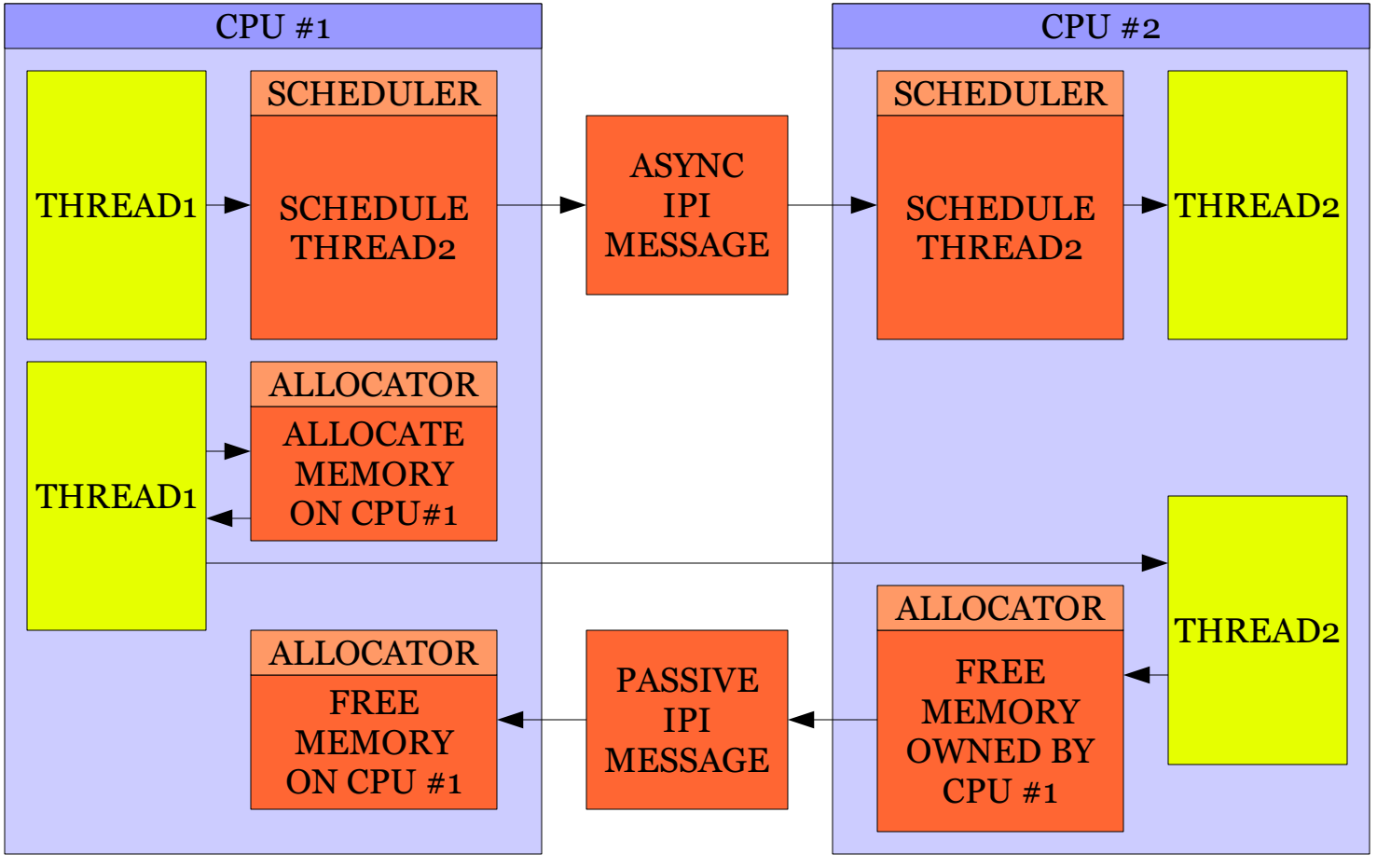
Matthew Dillon
DragonFly BSD Project
23 October 2003

Light Weight Kernel Threading and User Processes



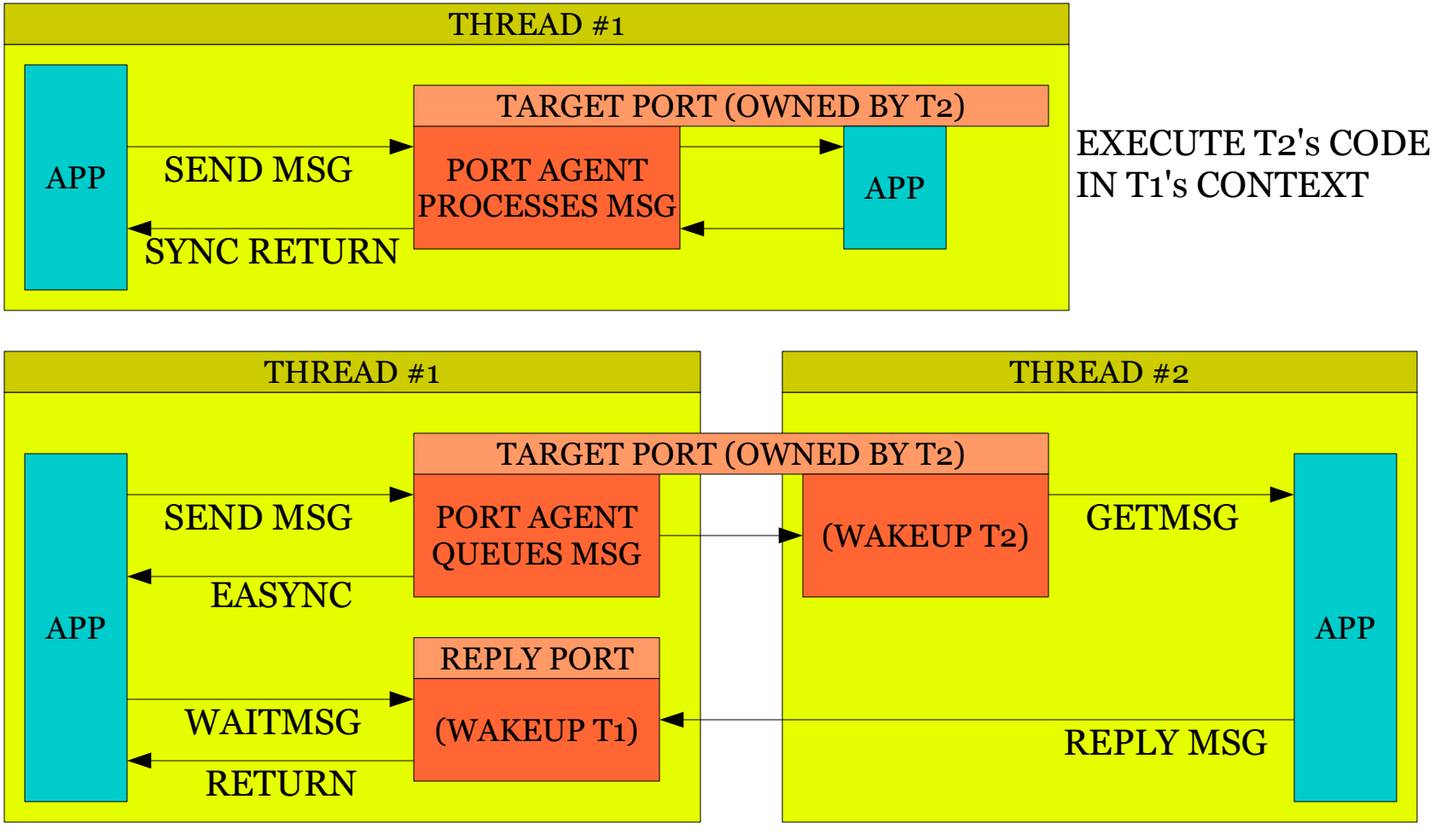
IPI Messaging

- ABSTRACTION PROMOTES CPU ISOLATION
- ASYNCHRONOUS IPI MESSAGING AVOIDS MUTEX OPS
- SIMPLE CRITICAL SECTIONS FOR LOCAL ACCESS
- MANY IPI OPS CAN BE PASSIVE / CONTRAST W/ RCU



Light Weight Kernel Messaging

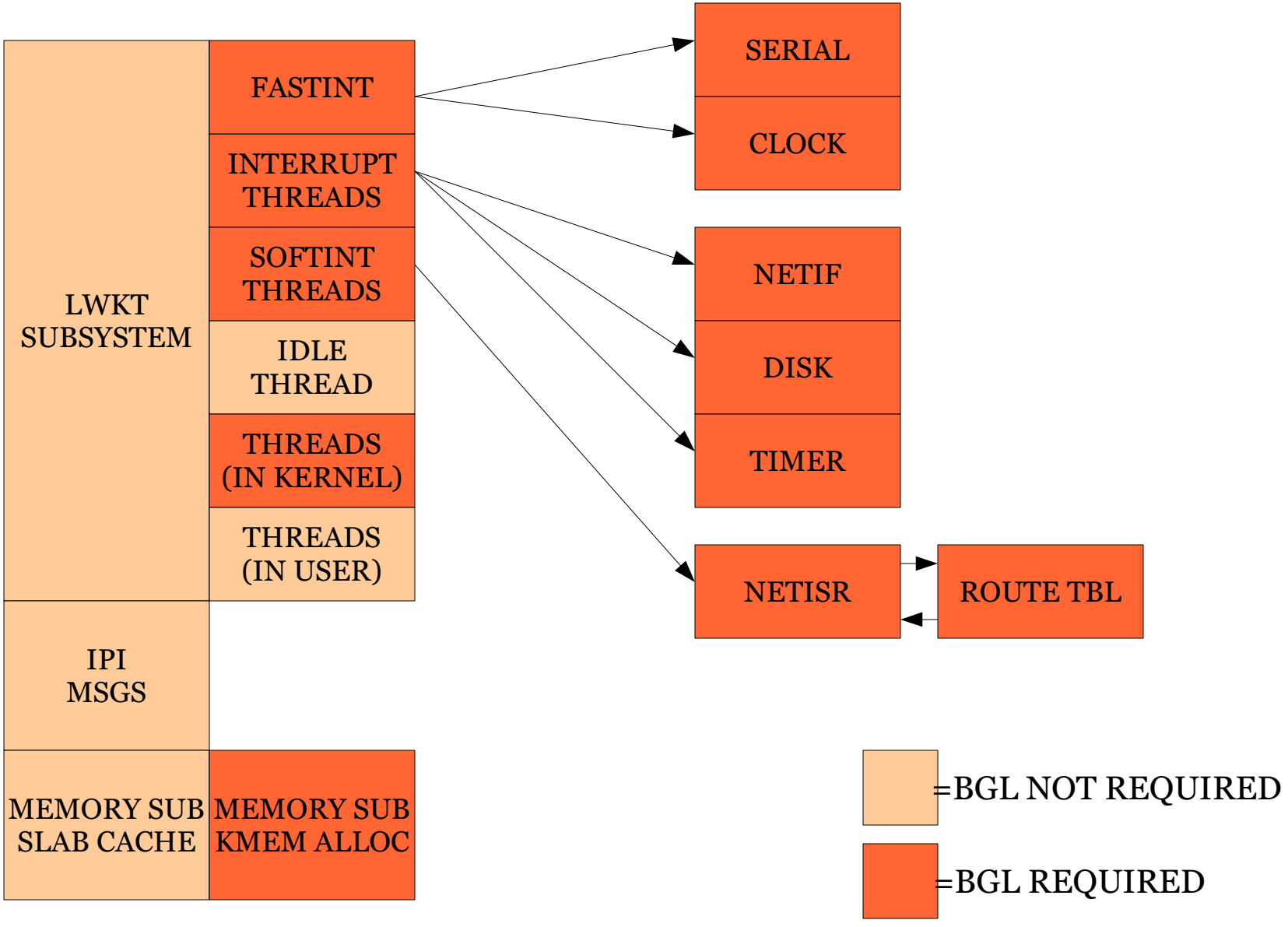
- AMIGA STYLE MESSAGES AND PORTS
- SEMI SYNCHRONOUS / PORT AGENT
- FAST SYNCHRONOUS PATH



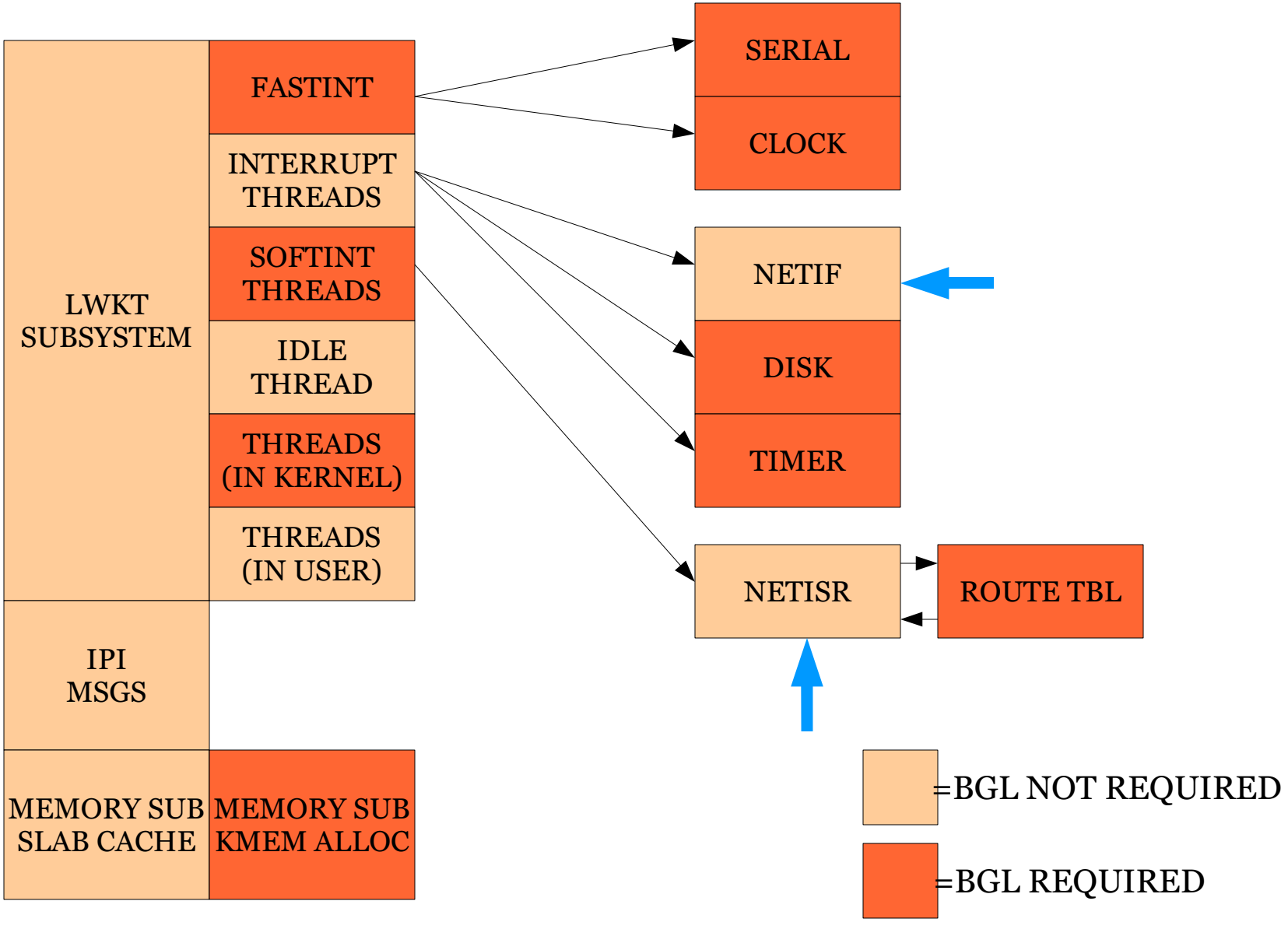
Big Giant Lock Removal

Matthew Dillon
DragonFly BSD Project
23 October 2003

Current BGL Coverage

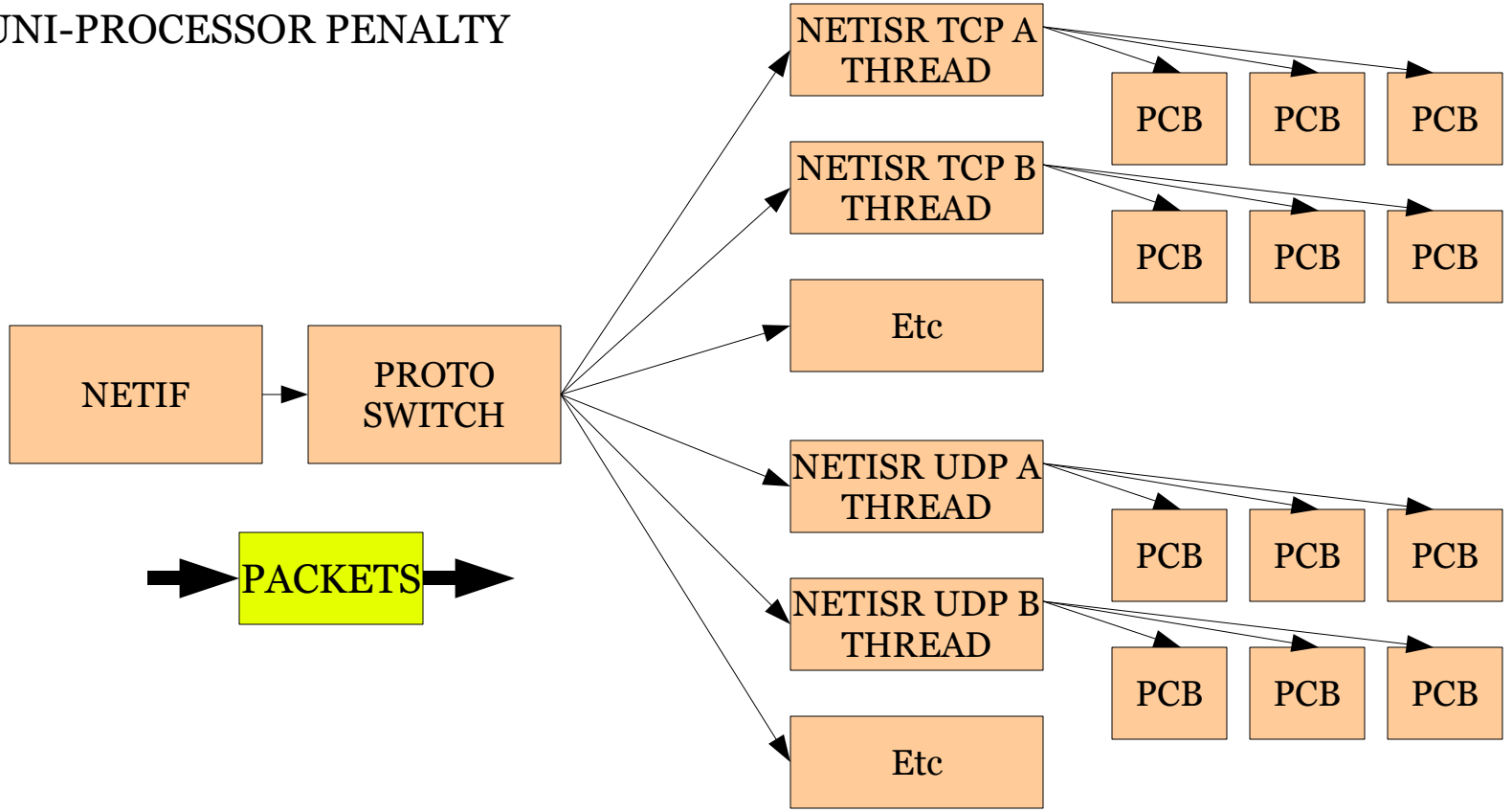


Next Stage BGL Removal



BGL Removal – Network Detail

- WORK BEING DONE BY JEFFREY HSU
- UTILIZES LWKT MSGS AND PORTS
- MULTIPLE THREADS PER PROTOCOL
- NO UNI-PROCESSOR PENALTY



=BGL NOT REQUIRED

Achieving a Single System Image (SSI)

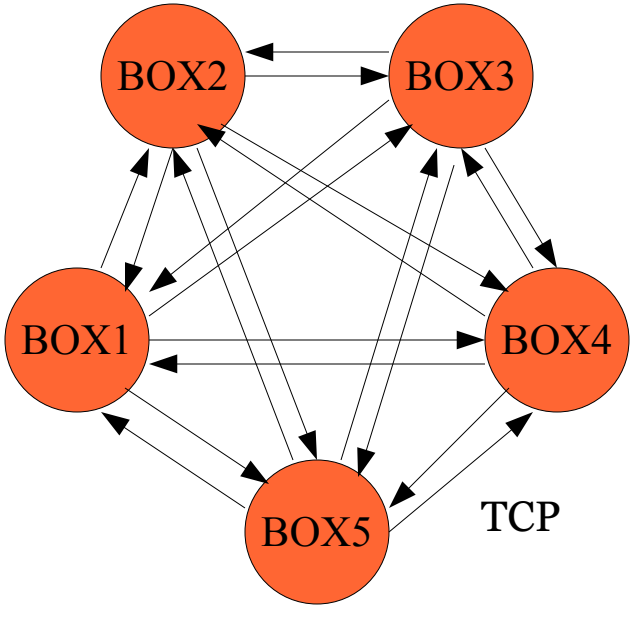
Matthew Dillon
DragonFly BSD Project
23 October 2003

Upcomming SSI Implementation Details

- CLUSTER MULTIPLE BOXES USING STANDARD NETWORK PROTOCOLS
- THE THREAD MESSAGING ABSTRACTION BECOMES VITAL
- THE CPU MESSAGING ABSTRACTION BECOMES VITAL (IPIs vs MUTEXs)
- IMPLEMENT A NETWORKED MESI CACHE COHERENCY MODEL
- PAGE-LEVEL CACHE COHERENCY, RANGE BASED LOCKING
- COPY DATA INDIRECTLY VIA THE CACHE COHERENCY MODEL
- GLOBAL FILE HANDLES, MESSAGING INTERFACE
- WHAT IS THE ULTIMATE TEST? PROCESS MIGRATION IN PIECES
- ADDING ROBUSTNESS (TIMEOUTS, TRANSACTIONS, VOTING)
- CONTRIBUTING RESOURCES TO A CLUSTER

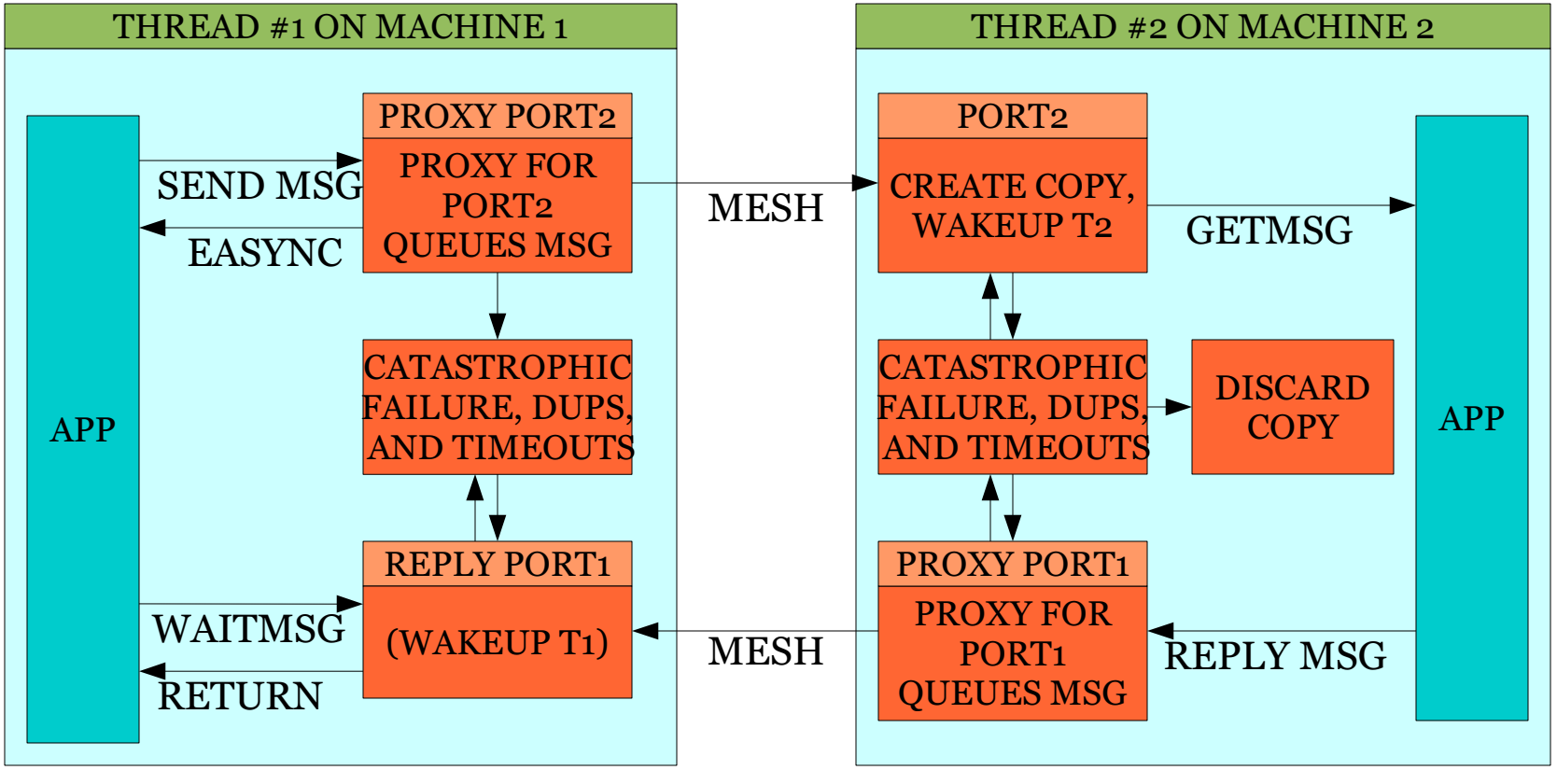
Message Traffic MESH Topology

- ABSTRACTED LOGICAL LAYER TO MAKE MESH PER-CPU
- RESERVE BUFFER SPACE ON PER-CPU BASIS
- FLOW CONTROL MORE EASILY MANAGED USING DIRECT-CONNECT
- DIRECT CONNECT CAN BE ABSTRACTED OVER GENERAL GRAPH WITH ROUTING
- DATA SOURCE, TARGET MAY BE DIFFERENT FROM MESSAGE SOURCE, TARGET
- USE SEPARATE MESH FOR CACHE COHERENCY PROTOCOL AND DATA XFER
- USE TCP (FIRST MAKE IT WORK, THEN MAKE IT FAST, OR NOT AT ALL)



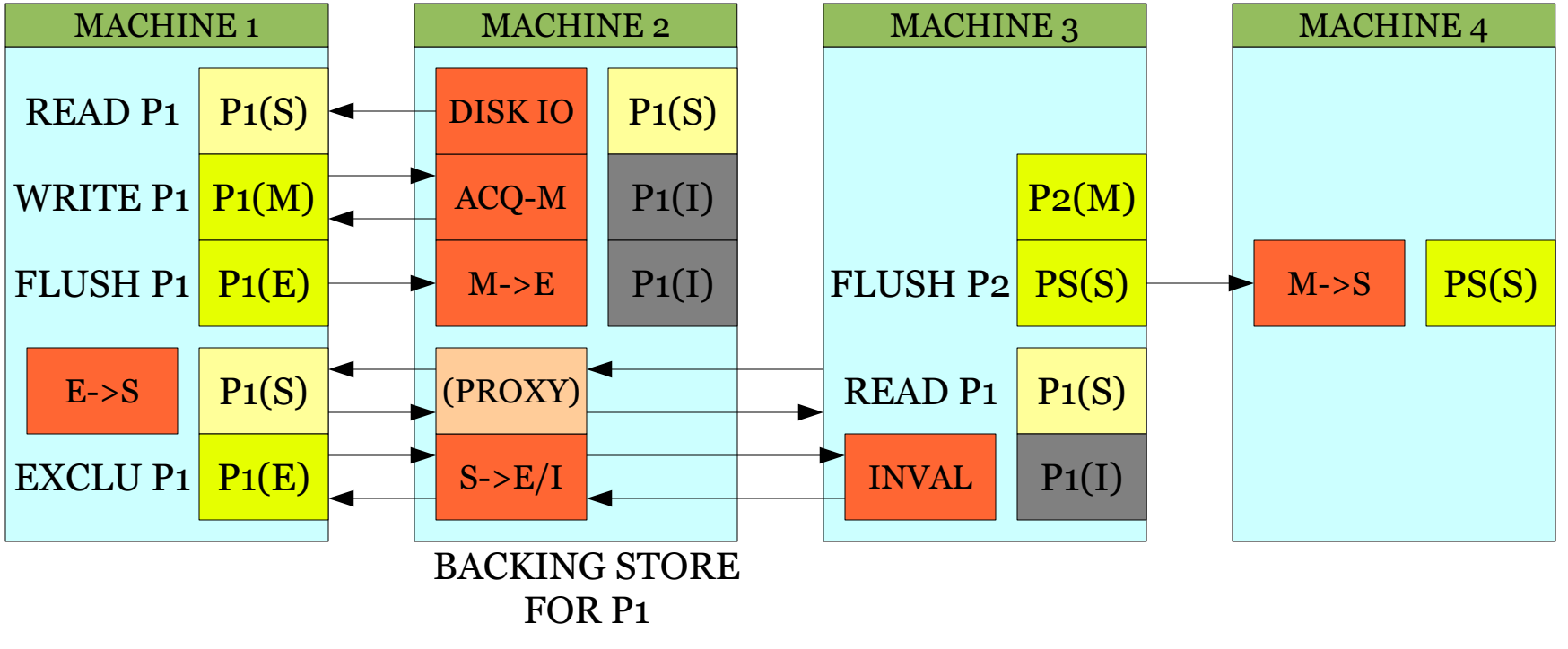
Using Proxy Message Ports

- PROXY PORT REPRESENTS REAL PORT
- LOCAL COPY OF MESSAGE HELD UNTIL REMOTE REPLY OR FAILURE
- PROXY PORT HANDLES MESH FAILURES, TIMEOUTS, AND PROTOCOL ISSUES
- ASSOCIATED DATA HANDLED BY CACHE COHERENCY PROTOCOLS
- PATH FOR ASSOCIATED DATA DICTATED BY CACHE COHERENCY PROTOCOLS
- PATH FOR ASSOCIATED DATA CAN BE OPTIMIZED



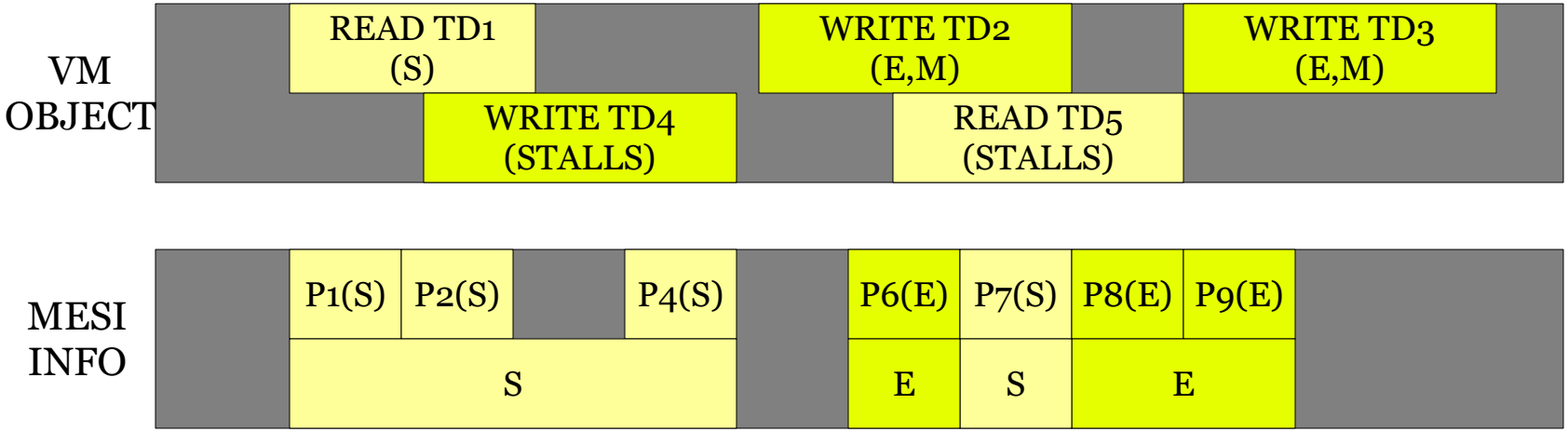
MESI Cache Coherency

- MESI = MODIFIED EXCLUSIVE SHARED INVALID
- DATA SHARING IS VITAL FOR EFFICIENT OPERATION OVER WAN INTERFACES
- CACHE COHERENCY MAKES THE CLUSTER INVISIBLE
- MEI IS EASIER TO IMPLEMENT BUT FAR LESS EFFICIENT
- E->M, M->E TRANSITIONS REQUIRE NO MESSAGE TRAFFIC
- FLUSHING MODIFIED DATA CAN MOVE FROM 'M' TO EITHER 'E' OR 'S'
- MACHINE HOLDING E OR M DECIDES DISPOSITION, ELSE BS DECIDES DISPOSITION
- IF E/M HOLDER IS UNKNOWN, BACKING STORE CAN PROXY REQUEST



Range Locking

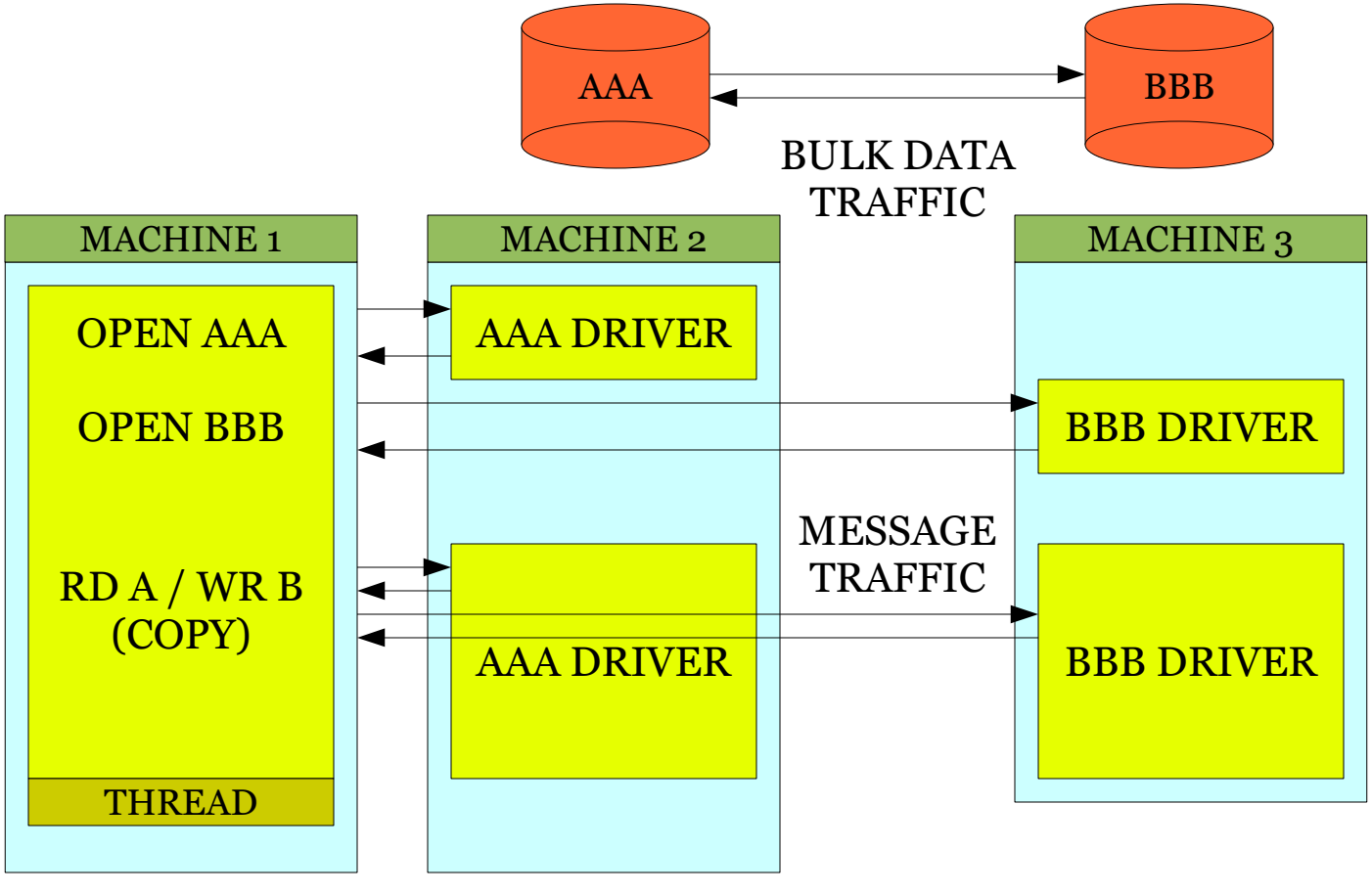
- RESERVE OFFSET RANGE IN OBJECT FOR UPCOMING I/O OPERATION (HEURISTIC)
- PRESERVE UNIX READ/WRITE ATOMICY WITHOUT LIMITATION
- ALLOWS PARALLEL READS, WRITES, AND COMBINATIONS ON THE SAME FILE
- AGGREGATE INTO LARGER GRANULARITIES TO REDUCE MANAGEMENT OVERHEAD
- POTENTIALLY KEEP TRACK OF MESI STATUS IN A FIXED AMOUNT OF RAM
- RESERVE MULTIPLE RANGES TO SUPPORT TRANSACTIONS



- CREATE A SINGLE (S) RECORD FOR P1-P4 BY OBTAINING A SHARED LOCK ON P1-P4
- CREATE A SINGLE (E) RECORD FOR P8-P9
- AGGREGATE OR THROW AWAY RECORDS TO REDUCE MEMORY USE
- ADD SERIAL NUMBER TO VM PAGES TO ALLOW REVALIDATION OF CACHE STATUS
- CAN MANAGE CACHE ON A BYTE RANGE BASIS RATHER THEN ON A PAGE BASIS

Global File Handles

- ACCESSIBLE FROM ANY HOST WITHIN THE CLUSTER
- POTENTIALLY ACCESSIBLE FROM OUTSIDE THE CLUSTER
- ALLOWS DEVICE DRIVERS TO DECIDE WHETHER TO MIGRATE OR NOT
- DATA ASSOCIATED WITH I/O SEPARTELY MANAGED VIA CACHE COHERENCY MODEI



Piecemeal Process Migration

- CACHE COHERENCY MODEL ALLOWS ADDRESS-SPACE SHARING ACROSS MACHINES
- DRIVERS FOR FILE DESCRIPTORS CAN MIGRATE ASYNCHRONOUSLY
- SOME DRIVERS MIGHT STAY ON THE MACHINE HODING THE PHYSICAL STORAGE
- TTYS AND PIPES CAN MIGRATE COMPLETELY OVER
- SOCKETS ARE MORE COMPLEX, BUT MIGRATION IS STILL POSSIBLE