Robert Gezelter
SOFTWARE CONSULTANT

http://www.rlgsc.com

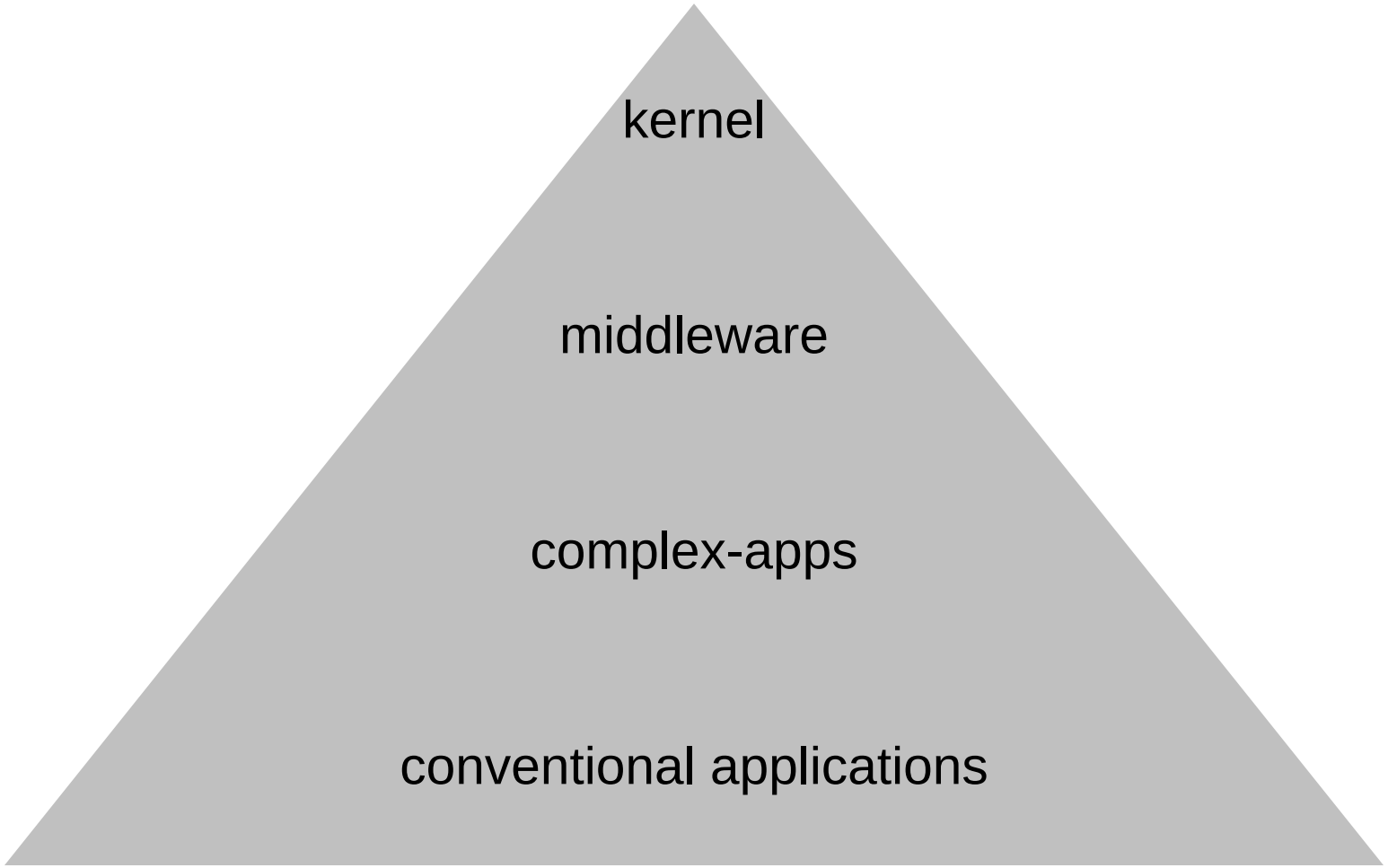
Session 1228

Events and Threads

HPETS 2002
Tuesday, October 8, 2002
11:15 – 12:30
Room 125

Types of Software Components

- compilers, linkers, simple sequential transformations
- real-time applications
- network applications
- middleware (e.g. RMS, network tool kits, real-time toolkits)
- run time libraries
- device drivers and related processes (e.g. XQP)
- operating system kernels



Why Events and Threads?

- outside events not controllable
- multiple tasks
- CPU utilization high; response latency
- need to operate "behind the scenes"

Different Threading Categories

- non-threaded
- non-preemptorially threaded
- collegial preemption
- involuntary preemption

Why Preemption?

- only a single reason – response latency
- other reasons are specious
- preemption has a high cost
- need to operate "behind the scenes"

Let’s look at some basics

- Process Types
- Trade-offs
- Protection Models
- Threading Hazards

Process Types

- Heavyweight
- Lightweight
- Featherweight

Heavyweight Processes

- Separate Register Set
- Separate Stack
- Separately Dispatchable
- Separate Address Space
- Expensive Creation

Lightweight Processes

- Separate Register Set
- Separate Stack
- Separately Dispatchable
- Preemptable
- Low resource consumption

Featherweight Processes

- Shared Register Set
- Shared (nested) Stack
- Separate Address Space
- Extremely inexpensive Creation
- No preemption
- Implicit synchronization
- Extremely inexpensive

Threading Hazards

- synchronization
- complexity
- proper tool?
- debugging
- data structure locking

Synchronization

- if non-premptive – no locking
- if collegial-preemptive – active thread is presumptive lock
- if involuntary – explicit locking mandatory

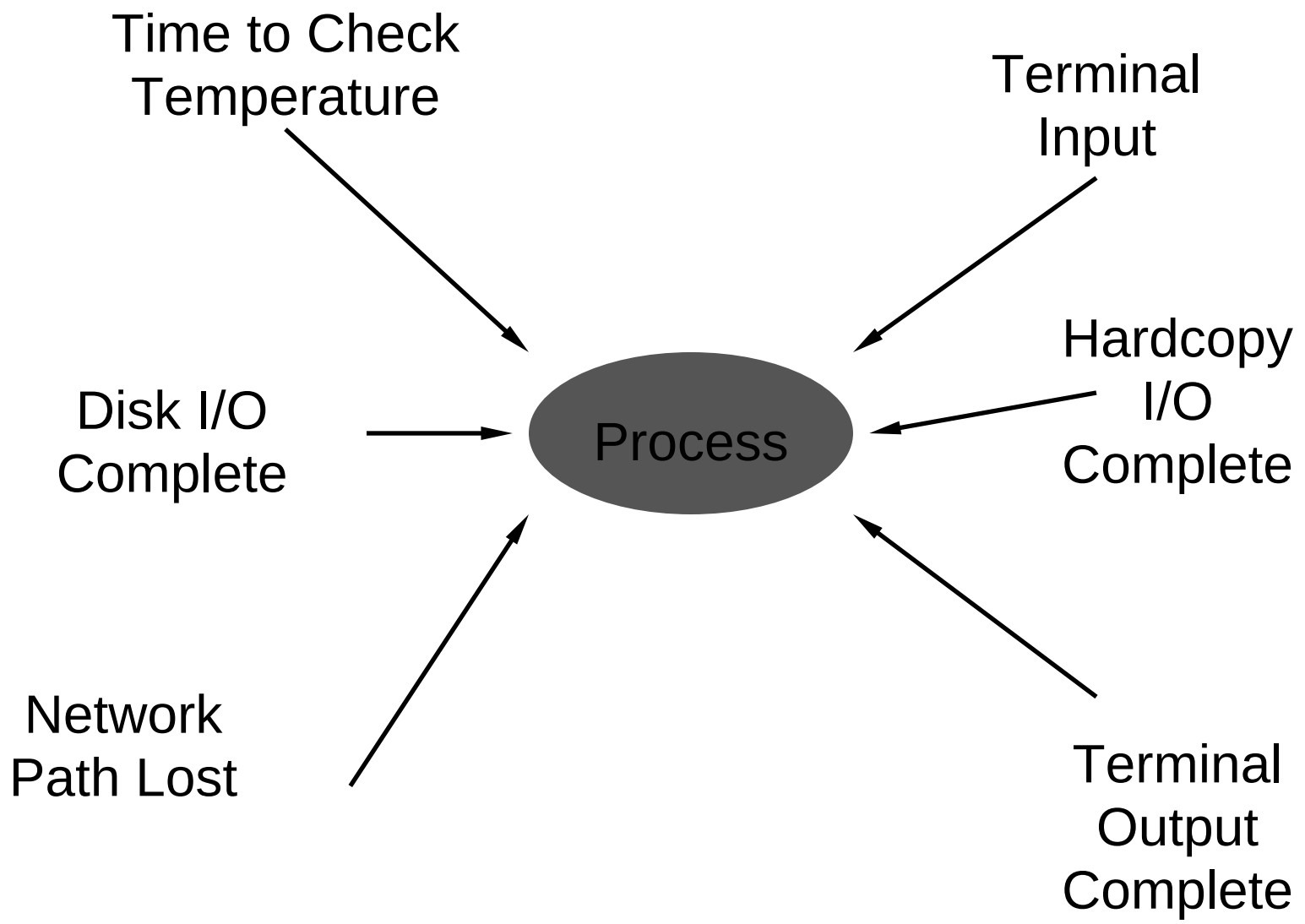
Threading Implementation Basics

- preemption model
- priority model
- debugging
- application suitability

OpenVMS ASTs – Basics

- FIFO within Access Mode
- Non-preemptable within an Access Mode
- ’Featherweight’
- AST Entry is via an asynchronous(!), simulated, CALLS instruction

Typical Event Driven Computer Application



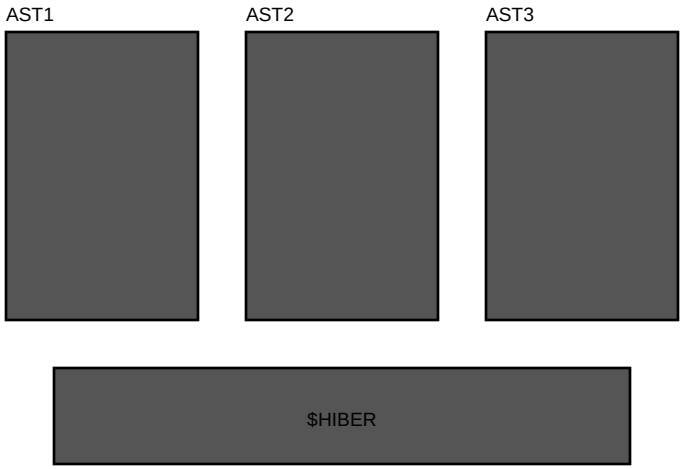
Common Root —

- External events control program
- Programs need to be efficient
- External event sequence is not under program control
- No Dispatch Routine

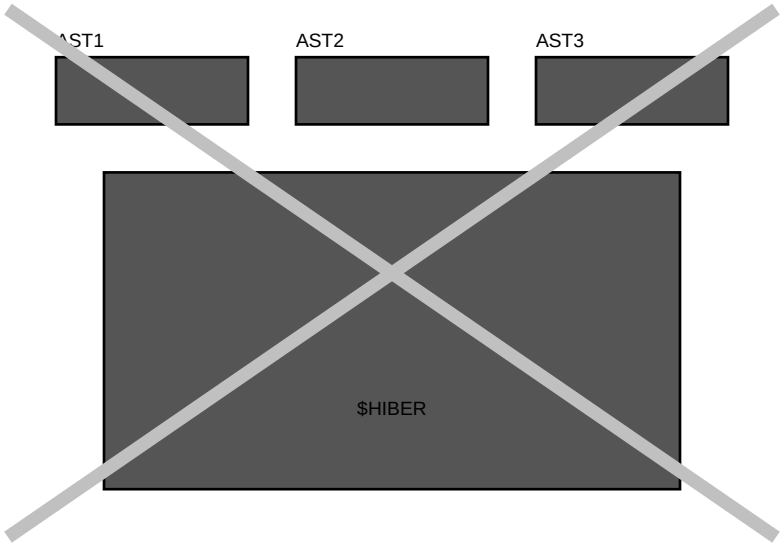
Tricks to Getting It Right

Do ALL Processing in ASTs

Avoid Performing Processing at AST level and normal Process level.



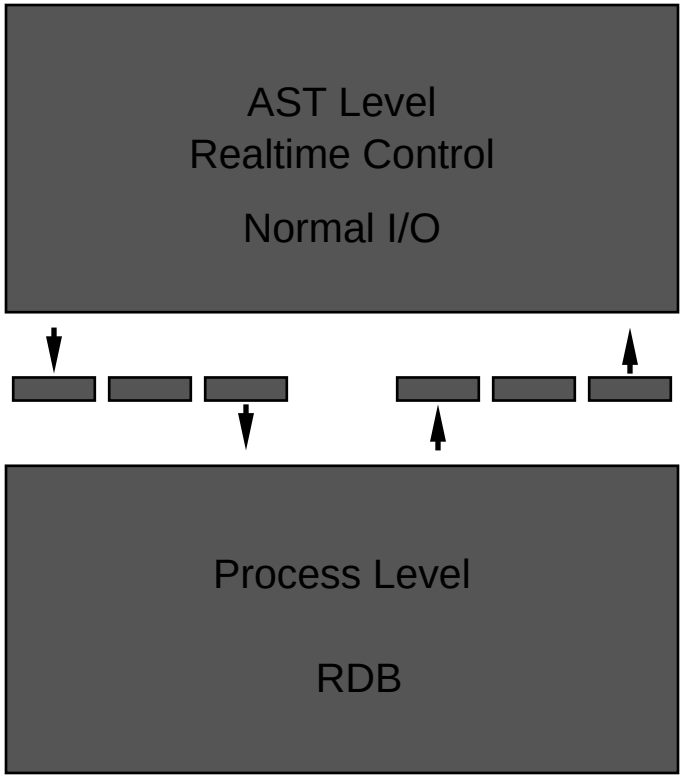
Good



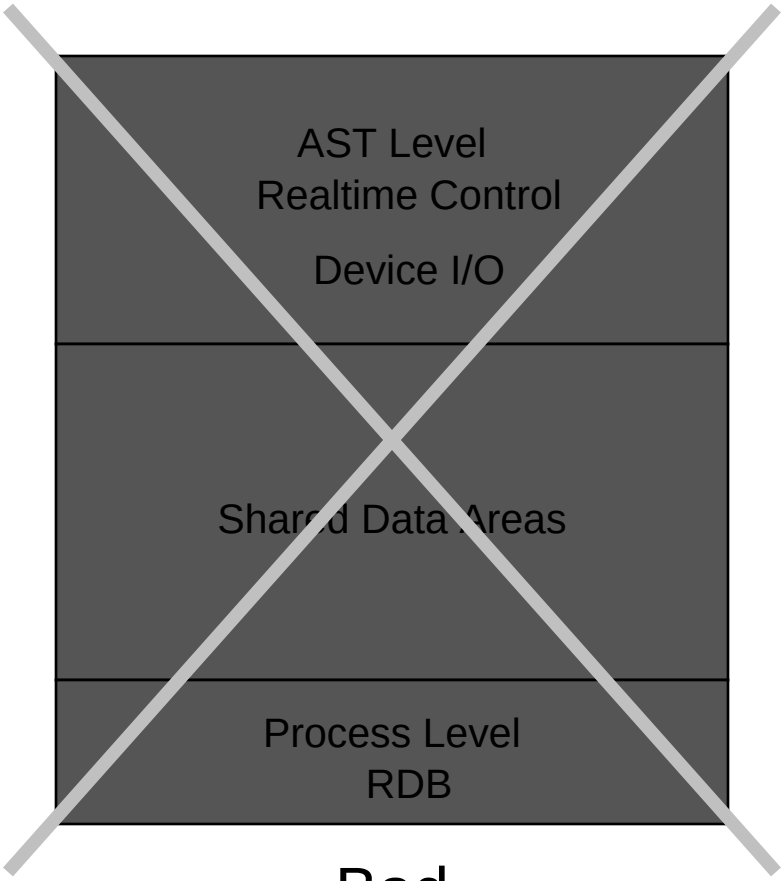
Bad

Tricks to Getting It Right

Use Work, Answer, and Free Queues to communicate.



Good



Bad

Don’t Borrow Trouble — Avoid Problems

- Kill bugs before they occur
- Increasing levels of preemption is expensive
- Analysis is cheap; debugging is expensive
- Testing is difficult, expensive, and not reassuring
- Preemption only needed to deal with latency
- DO NOT inhibit ASTs
- Use serialization where needed to simplify

Robert Gezelter
SOFTWARE CONSULTANT

Session Notes & Materials:
http://www.rlgsc.com/hpets/2002/index.html

http://www.rlgsc.com

Session 1228

Events and Threads

Questions?