The PTP Runtime Architecture

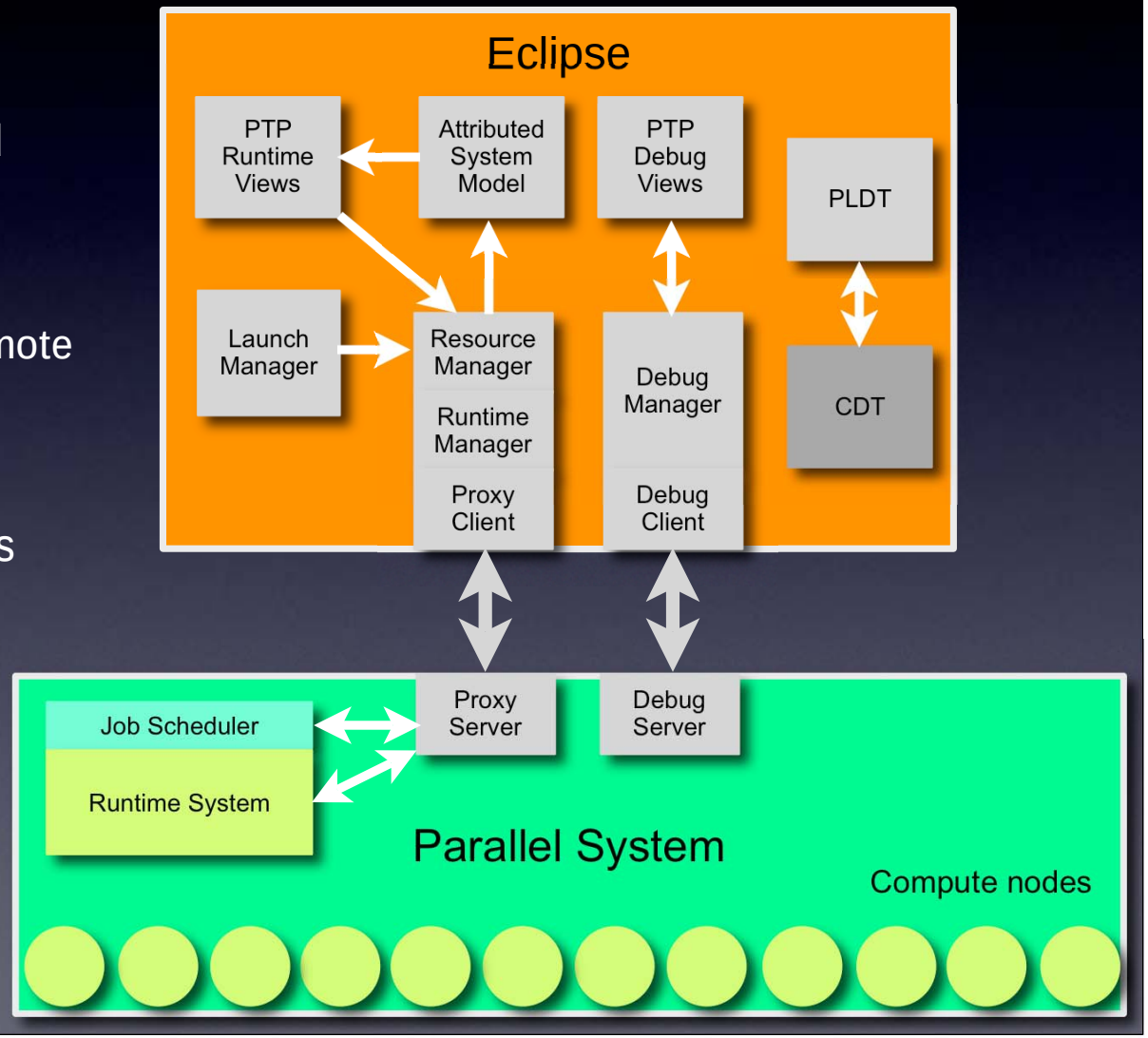
Randy M. Roberts
Los Alamos National Laboratory
rsqrd@lanl.gov

Overview

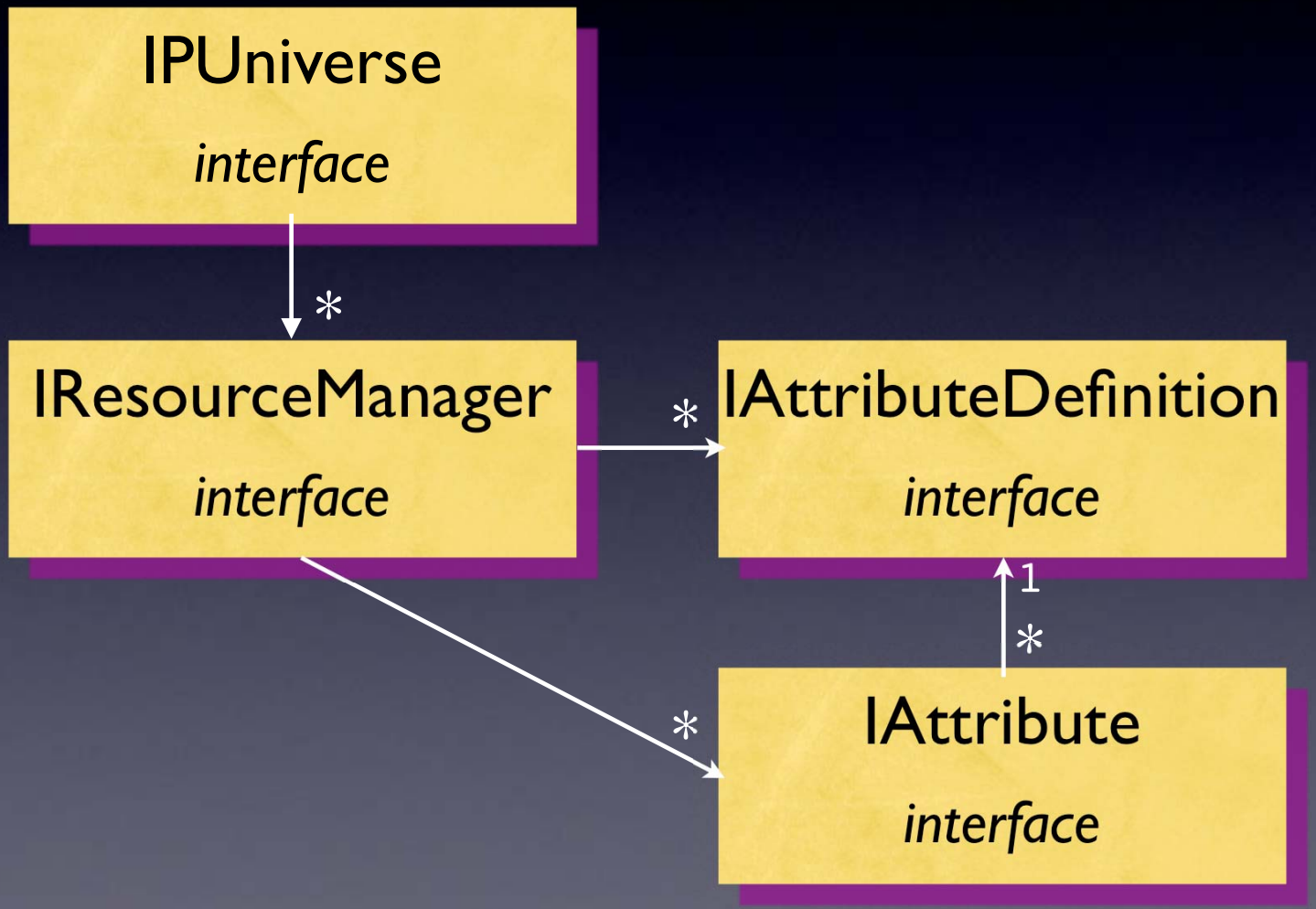
- Connection between PTP and remote system
- Model mirrors remote system's physical structure
- Manages parallel job submission
- Manages debug job launch
- Views to display model elements

PTP Architecture Details

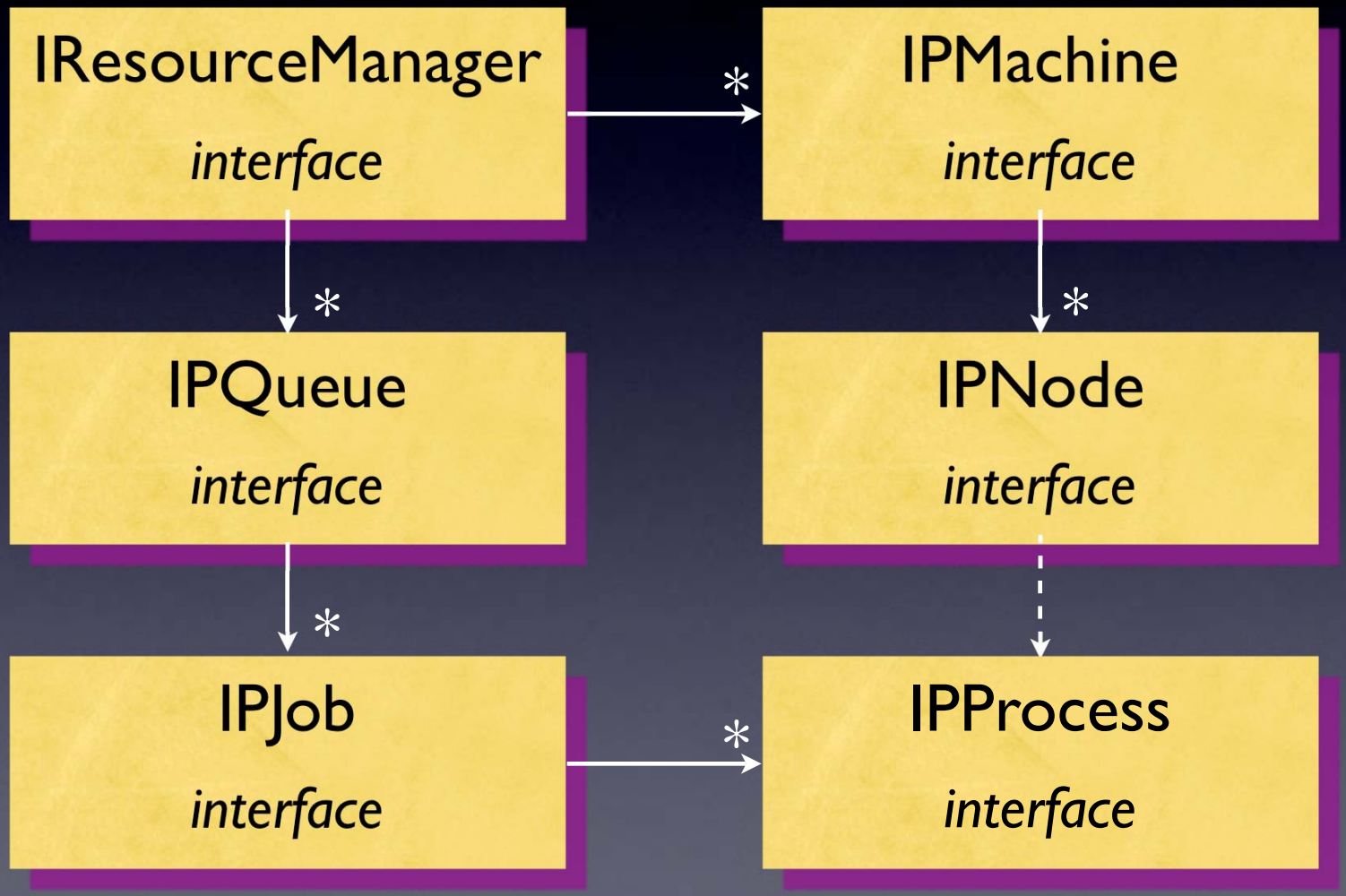
- The PTP runtime architecture is separated into two components
  - Java-side UI and parallel model
  - Native-side manages remote system
  - Communication via TCP sockets
- The PTP debug system is also separated into two components
  - not detailed here



Runtime Model



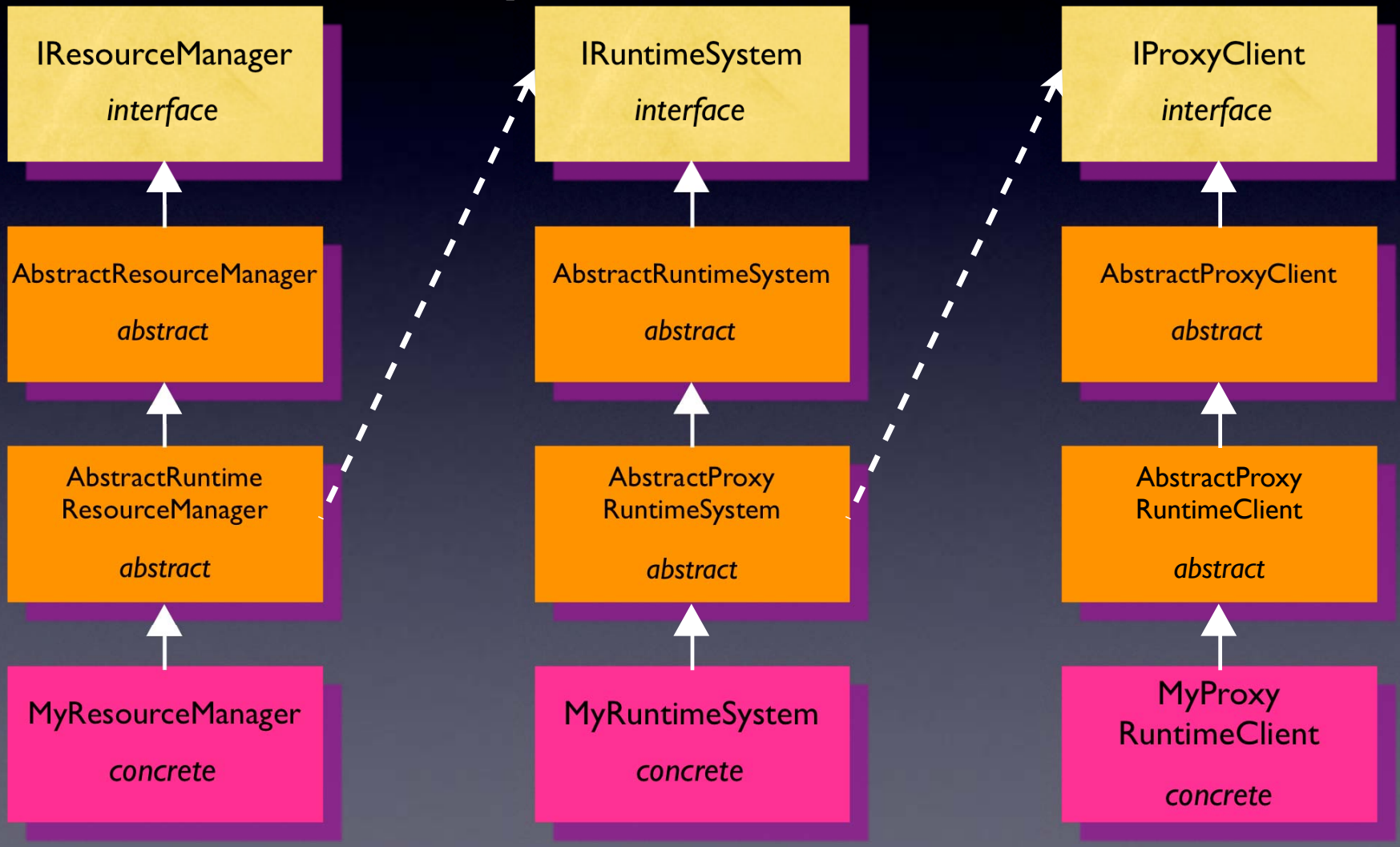
Runtime Model



Resource Manager Plug-Ins

- Resource Managers are contributed via Eclipse plug-Ins
- Plug-Ins supply:
  - resource manager
  - preference pages
  - creation wizard pages
  - job launch dynamic tab
- Multiple resource manager instances for each contributed resource manager

Resource Manager Implementation



Concrete Runtime Resource Manager

- AbstractRuntimeResourceManager receives asynchronous events from the concrete *IRuntimeSystem*, implements *IRuntimeEventListener*
- Most work provided by concrete class' protected *doCreateRuntimeSystem*() method
- Generic information passing via attributes

Concrete Runtime System

- Decouples resource manager from proxy client
- Almost everything handled by the base class, *AbstractProxyRuntimeSystem*
- In most cases only passes the *IProxyClient* to *super()*

Concrete Runtime Proxy

- Runtime proxy client communicates with remote system via TCP connection
- Wire protocol is encapsulated by the proxy runtime hierarchy
- Implements client side of TCP proxy protocol
- Most work handled by abstract base classes
- Java-side communication handled by abstract base class

Proxy Server

- Directly controls and monitors remote system through native interfaces
- Implements server side of TCP proxy protocol
- Need not be written in Java
- Runs as a user process, not as a daemon
- Needs no special privileges