

UDI Architecture In-Depth

<http://www.sco.com/forum1999/conference/developfast/f8>

Robert Lipe
UDI Development Team Lead
E-mail: robertl@sco.com



Agenda

- **UDI Portability and Extensibility**
- UDI Execution Model
- UDI Data Model
- Intro to UDI Specifications
- Q & A

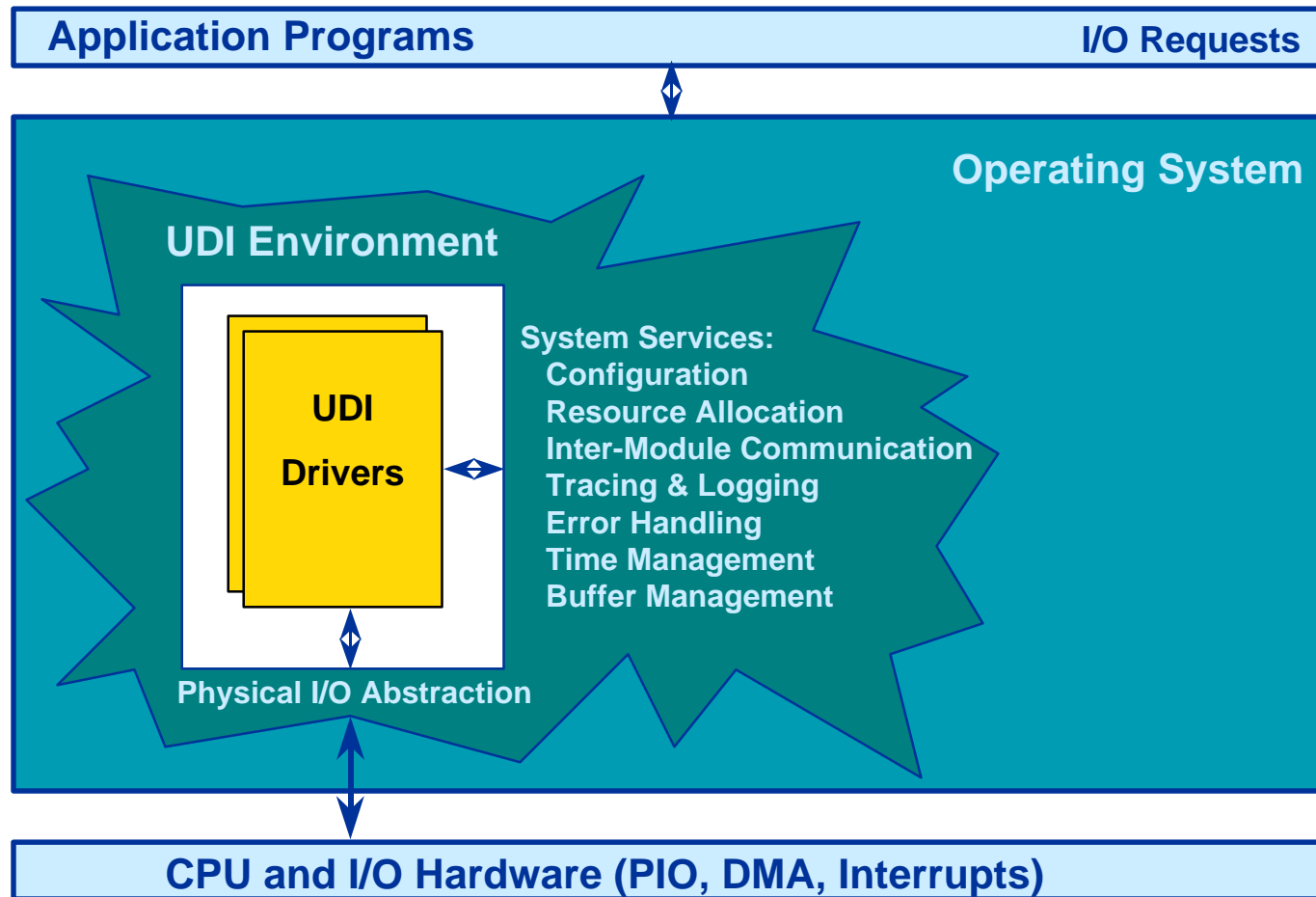


Driver Portability with UDI

- **100% driver source portability**
 - Binary portability for IA32 and IA64 and ...
- **Complete driver encapsulation**
 - All APIs defined
- **OS-neutral and platform-neutral**
 - OS policy removed from driver
 - Transparent endianness conversions

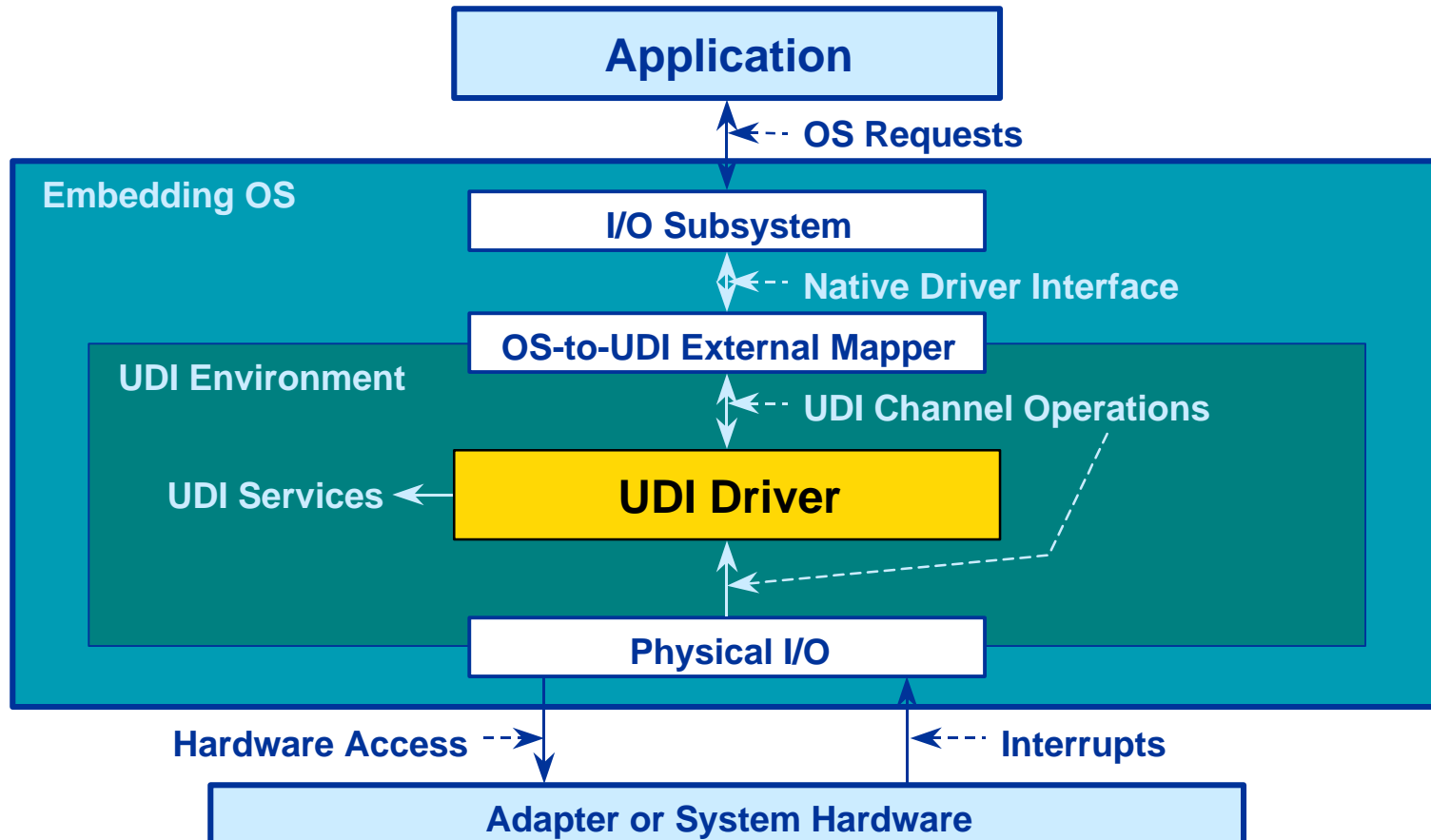


UDI Fully Encapsulates Drivers



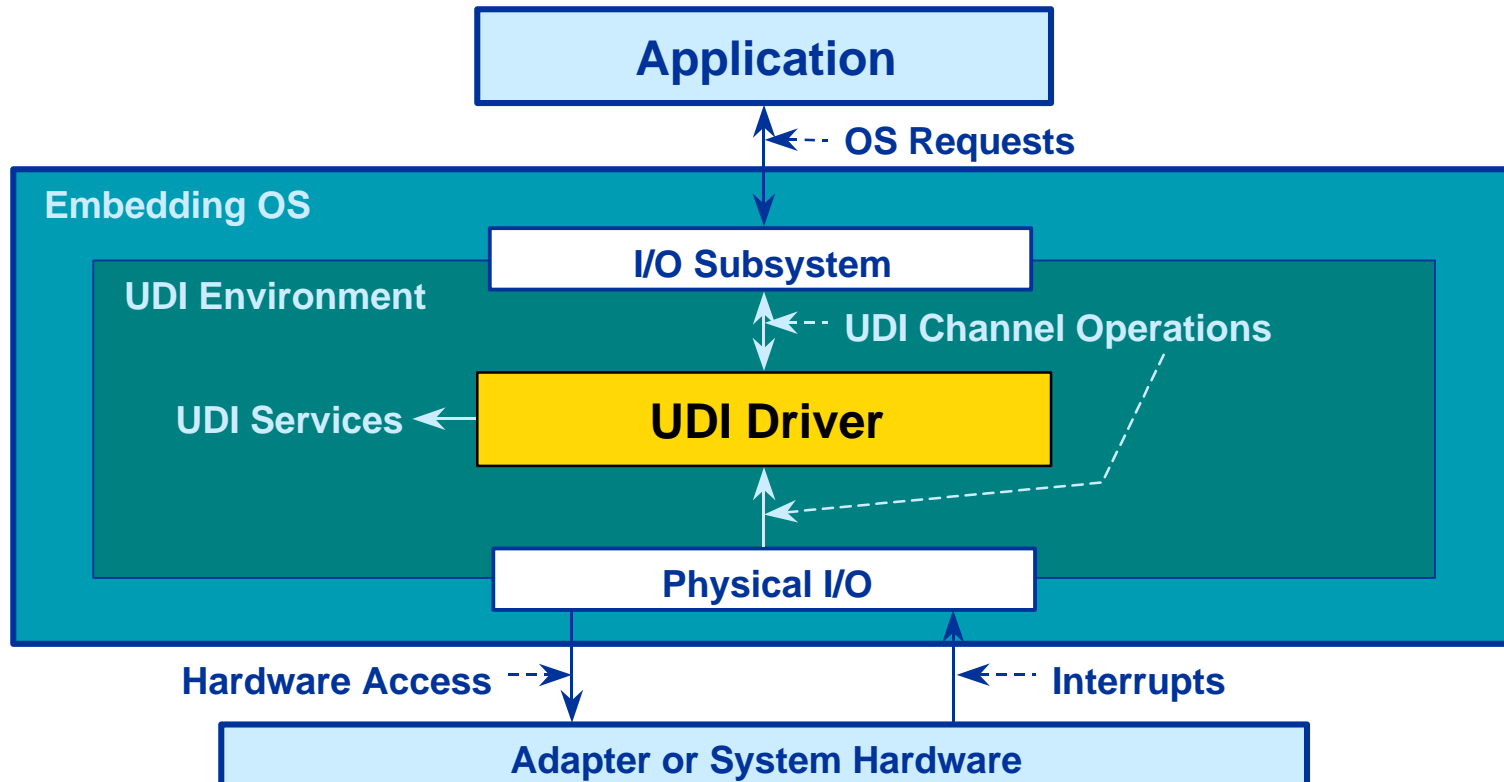
Path From Application to Driver

Layered Implementation



Path From Application to Driver

Integrated Implementation



Uniformity Across Device Types

- **Basic model common for all drivers**
 - Execution and Data Models
 - » Common buffer model
 - Configuration Model
 - Inter-Module Communication
 - » Between drivers and/or environment modules
 - System Services and Utility Functions



UDI Metalanguages

- **Device-type specific communication**
- **Defines communication paradigm between cooperating modules**
 - Operations and sequences to implement technology-specific functionality
- **Analogous to SCSI CAM, DLPI, etc.**



Agenda

- UDI Portability and Extensibility
- **UDI Execution Model**
- UDI Data Model
- Intro to UDI Specifications
- Q & A

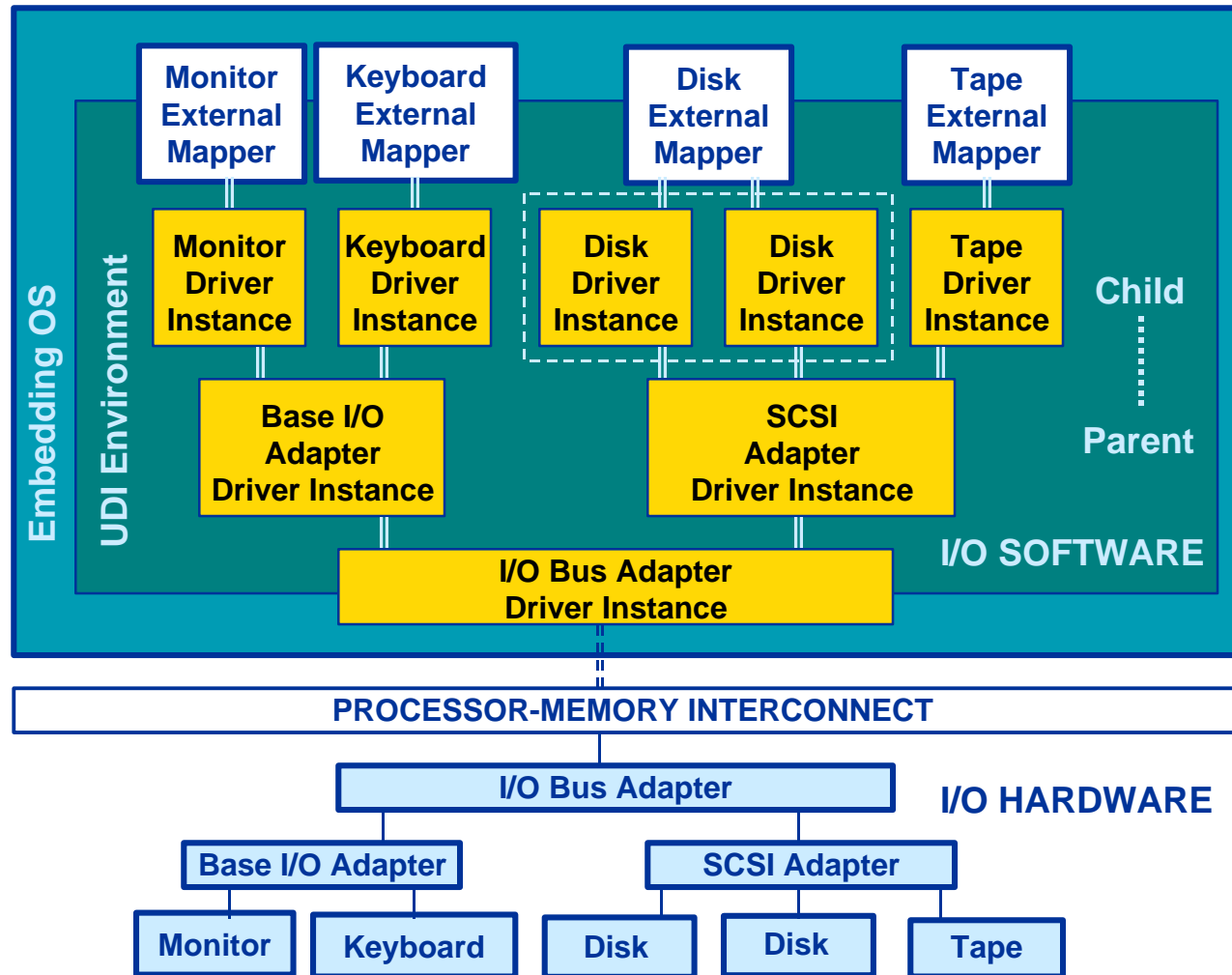


UDI Execution Model

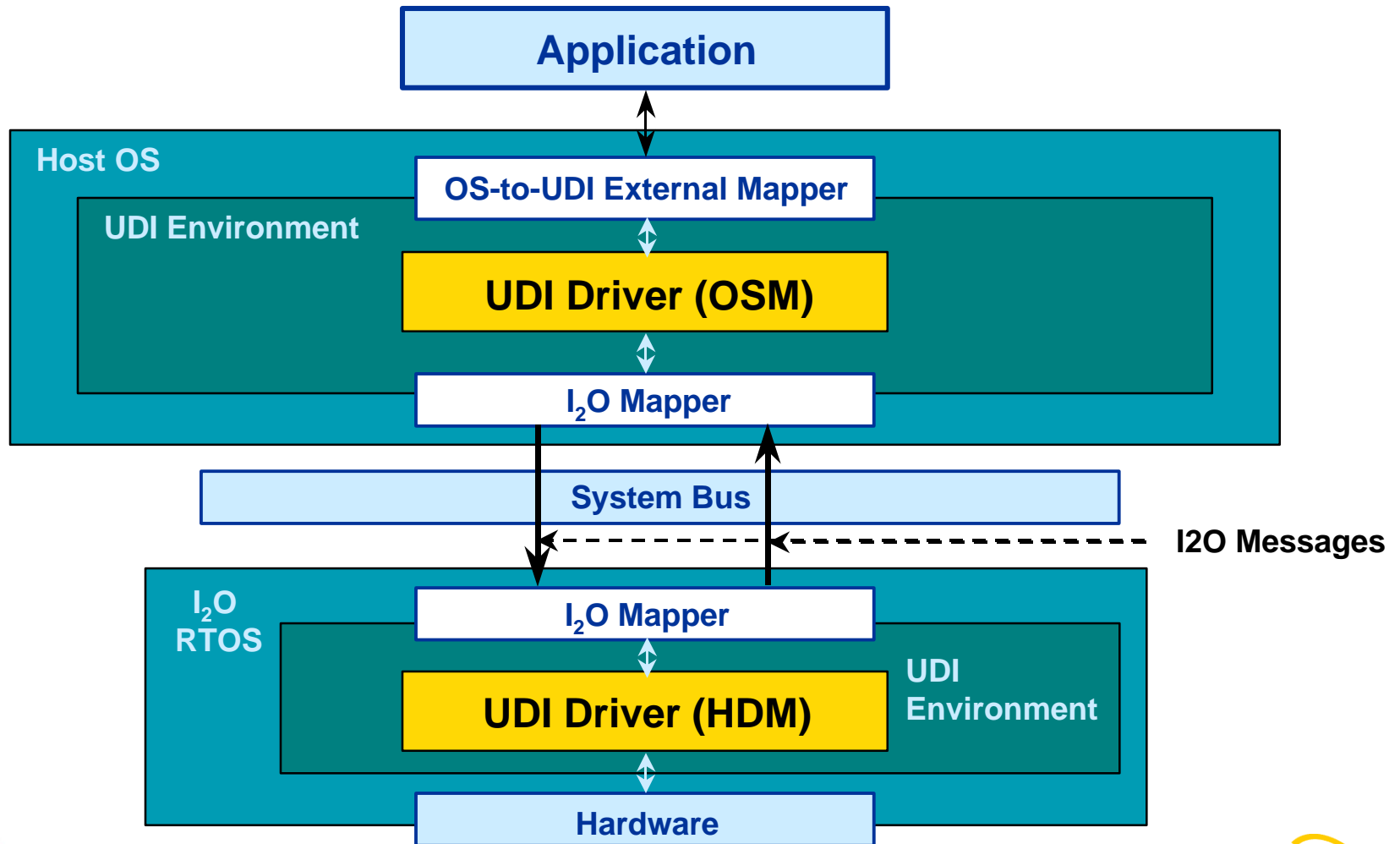
- **No global entry points**
- **Driver's udi_init_info structure contains entry-point pointers, size requirements...**
- **All driver code executed in the context of a *region***
 - Regions are associated with driver instances
 - » One for each adapter/device controlled



Example Driver Hierarchy



Example UDI & I₂O Combination



UDI Regions

- **Basic unit for execution and scheduling**
 - Each call into the driver region is serialized
- **No direct data sharing between regions**
 - *You have to go through channels*
- **Region attributes (e.g. priority hints) specified at build time**



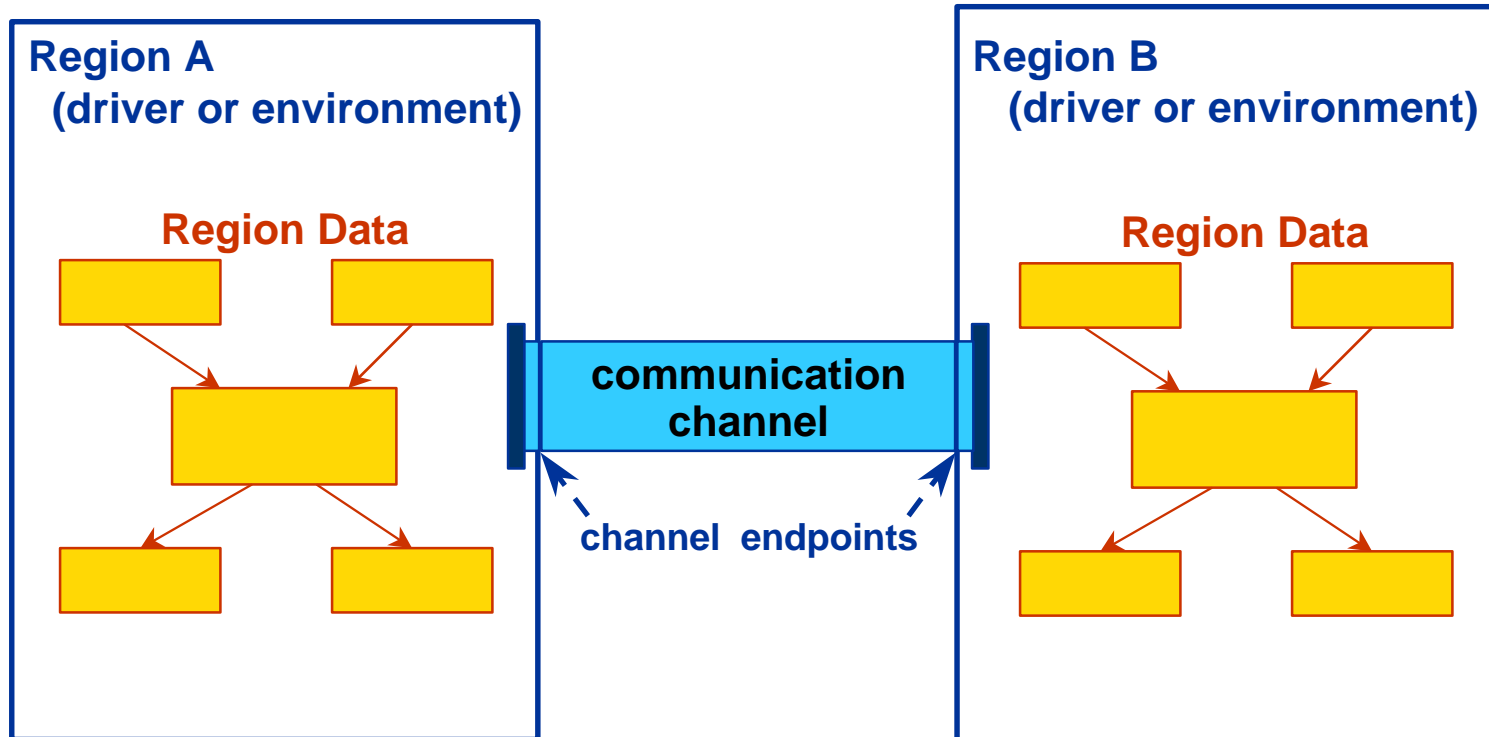
UDI Regions

(continued)

- **One driver instance per device instance**
- **One or more regions per driver instance**
 - Multi-region drivers may have higher parallelism
- **Enables *instance-independence***
 - Driver state separate for each device instance
- **Enables *location-independence***
 - Each region may operate in a different domain
 - » e.g. address space, NUMA or network node



Regions and Channels



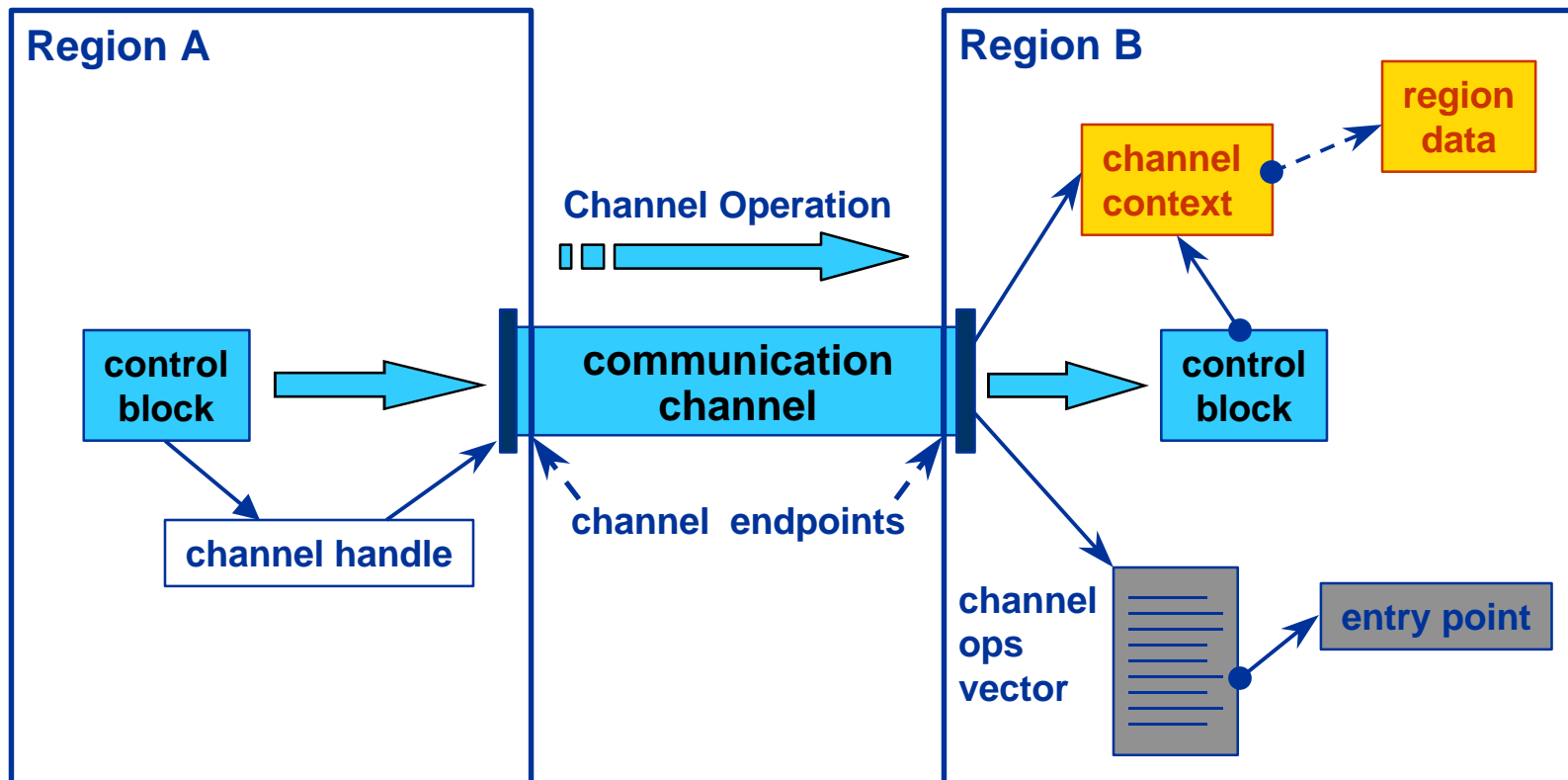
UDI Channels

Basis for Inter-Module Communication (IMC)

- **Bi-directional channels connect regions**
- **Communication via *channel operations***
 - Strongly typed function-call interface
 - Asynchronous one-way operations
 - » Each request has a corresponding response
 - » Context managed via *control blocks*



UDI Channel Communications



Metalinguages and Channels

- **Metalinguages define:**
 - Number and types of channels
 - Channel operation types on each channel
 - » Control block plus meta-specific parameters
 - Control block types for each operation
 - » Struct includes meta-specific fields
 - » Generic control block header common to all



UDI System Services

- **System interface & resource management**
 - Implemented for all UDI environments
 - Abstract OS services
- **Calls from driver to environment services are called *service calls***



UDI Service Calls

Two Styles

- **Synchronous service calls**
 - Complete without blocking
 - Results returned “immediately”
- **Asynchronous service calls**
 - Return without blocking
 - Delayed completion
 - Results returned via callback function



Non-Blocking Execution Model

- **All service calls and channel operations return without blocking**
- **Drivers usually return after making one service call or channel operation call**
- **Gives environment complete control over thread usage and driver scheduling**
- **“Pseudo-threads” interleaved between callbacks**



Agenda

- UDI Portability and Extensibility
- UDI Execution Model
- **UDI Data Model**
- Intro to UDI Specifications
- Q & A



UDI Data Model

- **Context managed via *control blocks***
 - Used with channel ops & async service calls
 - Environment uses CB to hold service call state
 - Driver uses context pointer in CB to find its data
- **No memory shared between regions**
 - Memory allocated in region private to that region
 - Regions share data with channel operations



UDI Control Blocks

- **CB contains *scratch* and *context* pointers (preserved across service calls, not ops)**
 - » Scratch space in CB holds per-request state
 - » Context pointer lets driver find the context of a channel op or callback
 - Initially set to channel context
 - Channel context struct points to global data
- **All CBs begin with generic `udi_cb_t`**



Implicit Synchronization

- **No locking primitives required in UDI**
 - All data accesses implicitly synchronized
 - » Region data accessible only from that region
 - » Only one thread per region active at a time
 - Other calls deferred until active call returns
 - Typically by adding CB to a region queue
 - Driver controls its parallelism by picking number and type of regions



Region Kill

- **Different environments have different levels of trust in drivers**
- **UDI environments can:**
 - detect misbehaved drivers (e.g. bad pointers)
 - track resource ownership and transfers
 - abruptly terminate (“region-kill”) driver instances
 - » Frees all resources and shuts down device



Agenda

- UDI Portability and Extensibility
- UDI Execution Model
- UDI Data Model
- **Intro to UDI Specifications**
- Q & A



UDI Specifications

<http://www.sco.com/UDI/specs.html>

- **UDI Core Specification**

- UDI Architecture
 - » Execution, Data and Configuration Models
- Fundamental Data Types
- Core Services and Utility Functions
- Core Metalanguages
- Packaging & Distribution



UDI Specifications

(continued)

- **UDI Physical I/O Specification**
 - DMA, Programmed I/O (PIO), Interrupts
 - Bus Bridge Metalanguage
- **UDI PCI Bus Binding Specification**
 - PCI enumeration attributes, etc.



UDI Specifications

(continued)

- **UDI Network Driver Specification**
 - Network Interface Card Metalanguage
 - See session F13: UDI Network Drivers
- **UDI SCSI Driver Specification**
 - SCSI Metalanguage
 - See session F12: UDI SCSI Drivers



Fundamental Data Types

- **Specific-length types**
 - udi_ubit8_t, udi_sbit8_t, udi_ubit16_t, udi_sbit16_t, udi_ubit32_t, udi_sbit32_t
 - udi_boolean_t (udi_ubit8_t)
- **Abstract types**
 - udi_size_t, udi_index_t



Fundamental Data Types

(continued)

- **Opaque types**

- Contain environment-private fields and structure
- Must be allocated using UDI service calls
- Opaque handles
 - » `udi_channel_t`, `udi_constraints_t`
- Semi-opaque types
 - » `udi_cb_t *`, `udi_buf_t *`



Core Services

- **Inter-Module Communication (IMC)**
- **Memory Management**
- **Buffer Management**
- **Time Management**
- **Tracing and Logging**



Core Utility Functions

- **String/Memory Utilities**
 - udi_strcpy, udi_strlen, udi_memcmp et al
 - udi_snprintf, udi_strtou32
- **Queue Management Utilities**
- **Endianness Management Utilities**



Core Metalanguages

- **Management Metalanguage**
 - Environment-initiated control operations
- **Generic I/O Metalanguage**
 - Generic read/write plus custom ops
 - Useful for prototyping and “one-off” extensions
 - Used to access driver diagnostics



Agenda

- UDI Portability and Extensibility
- UDI Execution Model
- UDI Data Model
- Intro to UDI Specifications
- **Q & A**



UDI Information

Web page

<http://www.sco.com/UDI>

Project UDI contacts

Chair: Kevin Quick, +1 214 654 5173, kquick@iphase.com

Vice Chair: Mark Evenson, +1 408 447 5601, mevenson@cup.hp.com

Secretary: John Lee, +1 650 786 5323, john.lee@eng.sun.com

Editor: Kurt Gollhardt, +1 908 790 2277, kdg@sco.com

Advisor: Mark Bradley, +1 303 684 4753, markb@btc.adaptec.com

