

Generic RDMA Enablement in Linux

(Why do we need it, and how)

Krishna Kumar
Linux Technology Center, IBM
February 28, 2006

AGENDA

- RDMA : Definition
- Why RDMA, and how does it work
- OpenRDMA – history
- Architectural overview
- How to get there - code base
- Convergence with openIB
- Exploiters of RDMA
- References

RDMA : Definition

- *A protocol that allows applications across multiple systems to exchange data using Direct Memory Access.*
- Reduces CPU, multiple copies, and context switching overhead needed to perform the transfer.
- Preserves memory protection semantics.

RDMA : Definition (contd)

- Hence RDMA is useful in applications where high throughput, low latency networking is needed.
 - without requiring to add extra processors and memory for smaller latency and greater bandwidth.
- Current devices : Infiniband HCA's, iWARP RNIC's.

RDMA : Why is it required ?

- CPU speeds are not keeping up with networking speeds.
- Usage of CPU for network processing reduces performance of CPU intensive applications.
- Remote Direct Memory Access allows for :
 - zero-copy between the application and kernel - fewer memory copies reduce memory bandwidth and improves latency.
 - minimal demands on system resources like CPU and memory bandwidth.

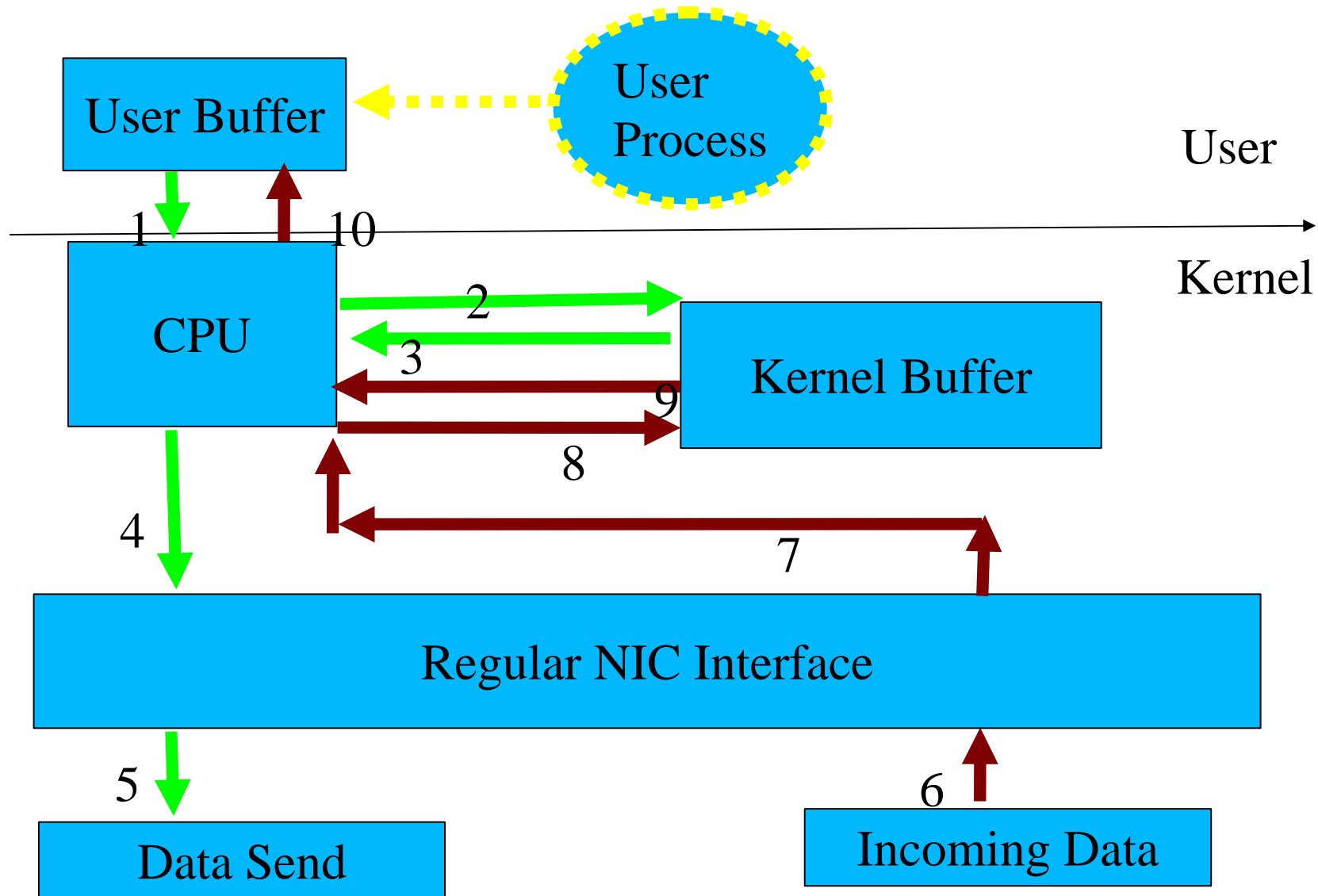
RDMA : Why is it required ?

(contd)

- Other earlier methods (like page flipping) are either not general enough or don't scale.
- CPU utilization for network intensive applications : a large amount of time spent in TCP/IP stack to copy data and manage buffers.
- Memory bandwidth requirements :
 - About 3 times link bandwidth for data receives
 - 3GB/s of host memory required to support 10GigE at full receive link bandwidth

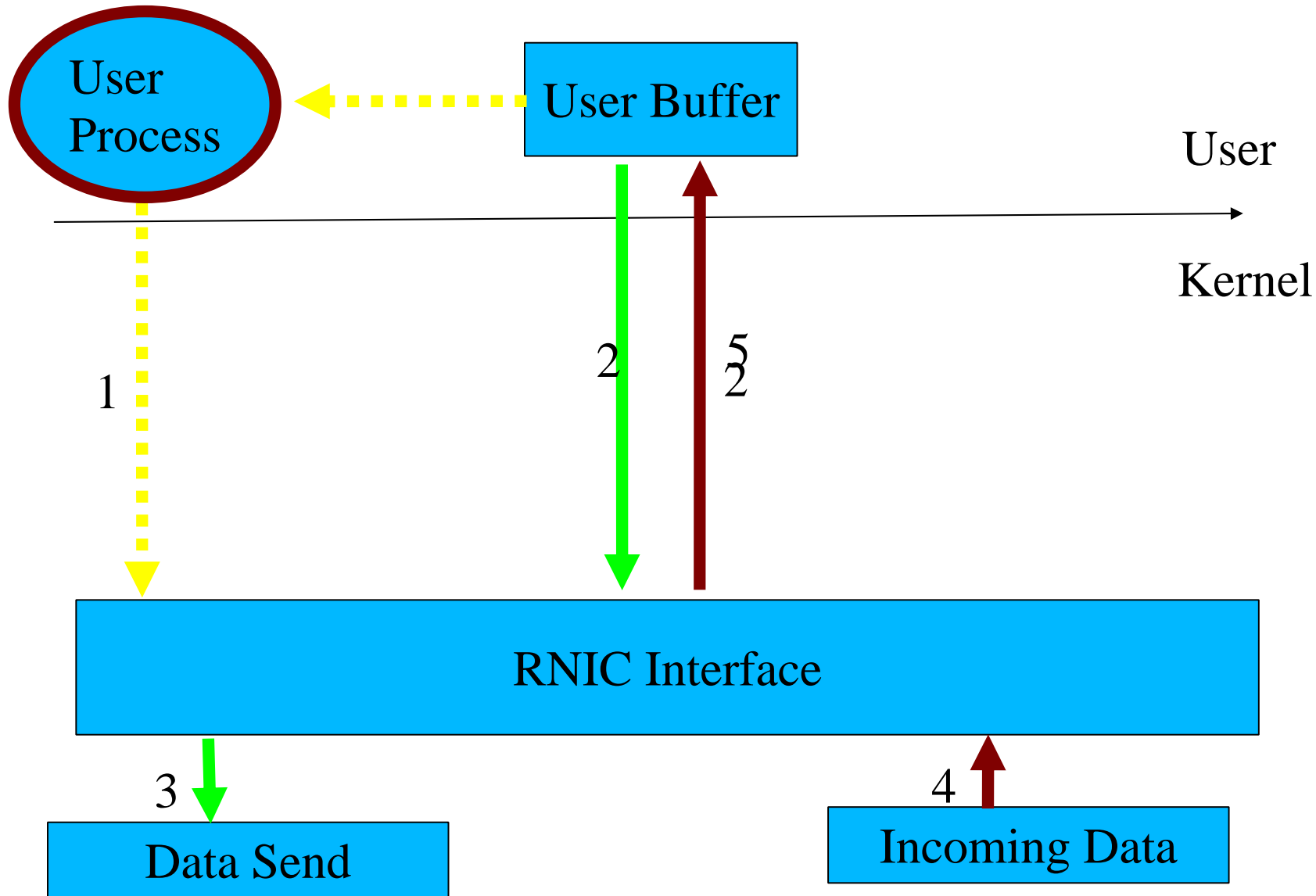
How RDMA works

- **Sending** / **Receiving** data – **old** method



How RDMA works (contd)

- **Sending** / **Receiving** data – **new method**



How RDMA works (contd)

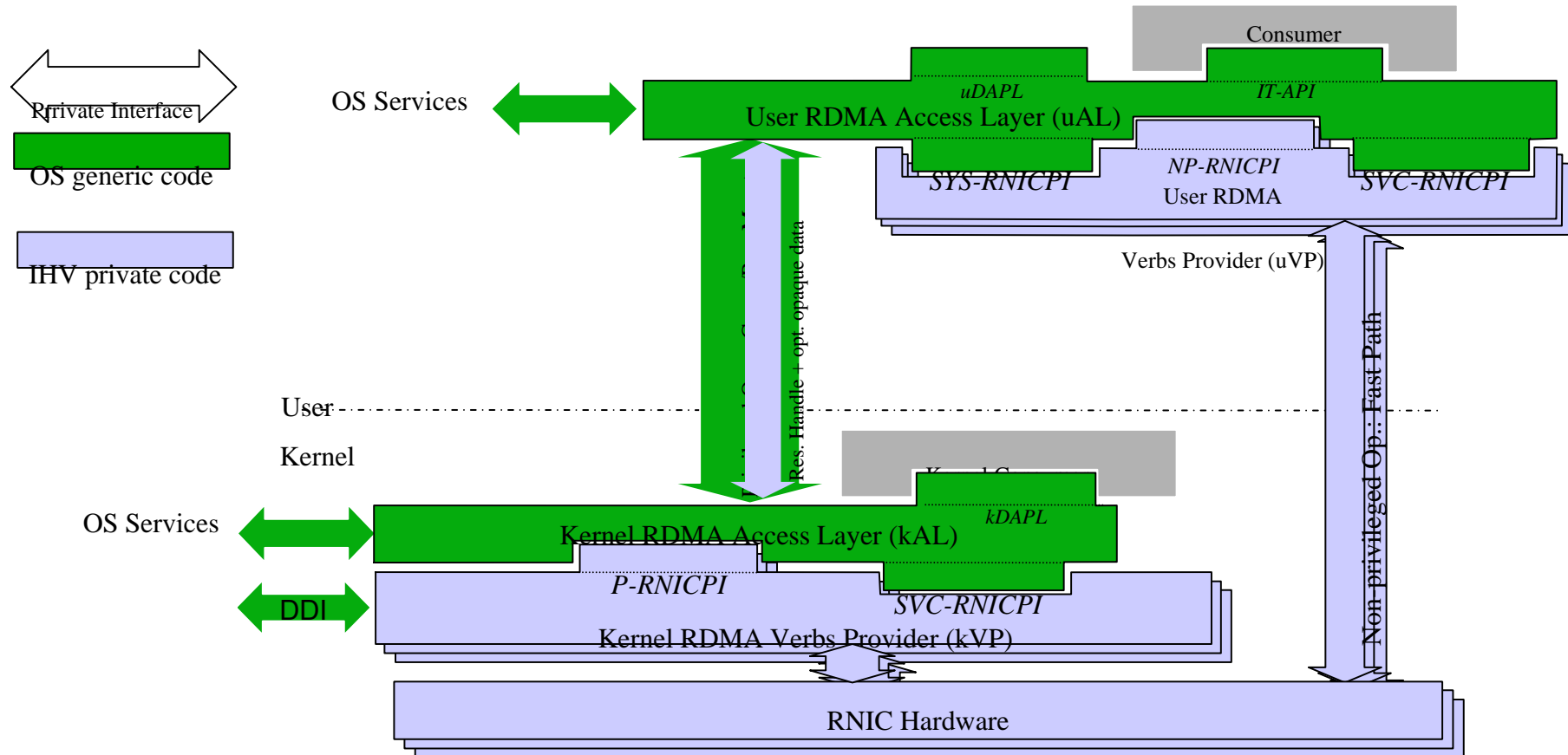
- User memory registration with adapter for direct adapter access to user memory and vice-versa
 - reduces context switches, CPU utilization
 - reduces memory bandwidth
- Some offload to card reduces CPU processing overhead for each packet.

OpenRDMA history (till Jan 2006)

- openRDMA project started in November 2004 to :
 - Enable RDMA in Linux and get acceptance in LKML
 - Common API/single stack between Infiniband and iWARP, use standard API's like DAPL and IT-API.
 - Use RNIC-PI developed by ICSC Consortium.
- An industry supported open source project
- Support hardware independent RDMA based interconnects :
 - iWARP based RNIC's
 - Infiniband HCA's.

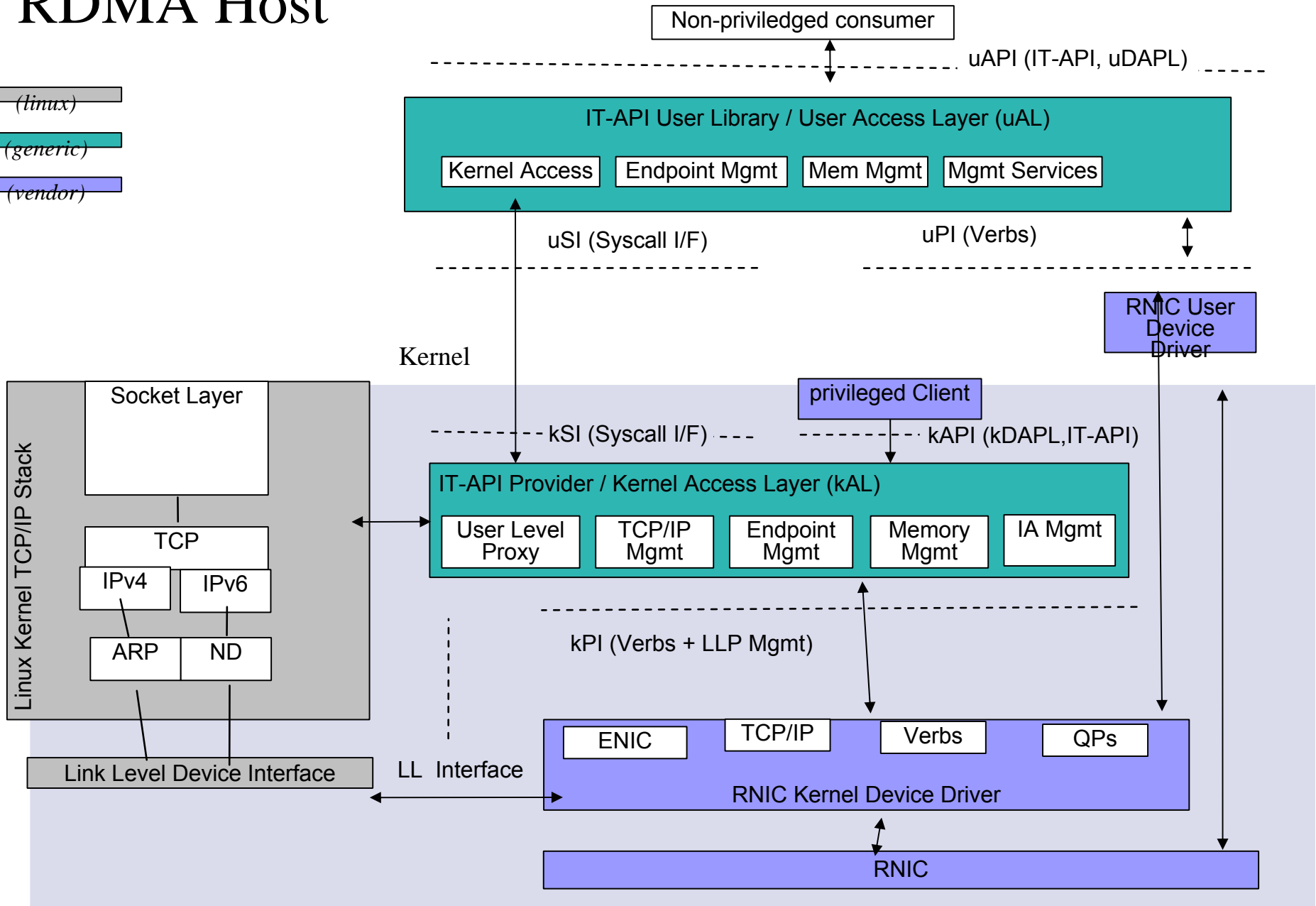
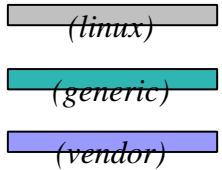
openRDMA Architecture

- OpenRDMA architecture using RNIC-PI



RNICPI Interfaces

- RDMA Host



How to get there - Code Base

- This architecture led to the creation of a base RDMA stack
 - Currently in alpha ready state (RNIC-PI 0.95)
 - being worked on by the community to add functionalities and fix issues.
- Goal is to build RDMA ecosystem with :
 - vendor independent RNIC support
 - offering multiple RDMA capable API's at both kernel and user level (DAPL, IT-API).

How to get there - Code Base (contd)

- TOE – a “no-no” for Linux kernel community.
 - TOE involves offloading TCP/IP stack processing to RNIC.
 - Hacking internals of TCP/IP stack
 - Has maintainability issues
 - Core TCP/IP stack features missing (eg netfilter)
 - Questionable performance gains

How to get there - Code Base (contd)

- Current code
 - kAL layer (rigen.o)
 - Register special device : /dev/rdma* with system, define open(), ioctl() and close() interfaces; for user access.
 - Device kVP layer will call into kAL ri_svc_attach() for each device that is “found”
 - kVP provides *rnica_ops* for implementing various verbs
 - Allocate various data structures, like QP's, CQ's, etc.
 - Call into driver specific RNIC open routine using *rnica_ops*

How to get there - Code Base (contd)

- Current code
 - kAL layer (contd)
 - ioctl() provides various handles for user access :
 - rnic handle
 - qp, cq, pd mr handles, etc
 - These handles allow various operations of the device
 - modify QP, register region, etc.
 - Separate kernel API's provided for both of the above
(common code for kernel/user)

How to get there - Code Base (contd)

- Current code
 - uAL layer (libual.so)
 - uAL init code opens special file /dev/rdma to get list of RNICS and the kVP handle identifying each RNIC.
 - User calls ri_get_rnic_handle() specifying which device to get a handle, eg device of 1 is “/dev/rdma1”.
 - Call open(“/dev/rdma1”)
 - Load uVP library provided.
 - Call ioctl(GET_RNIC_HANDLE) to trap to kernel and retrieve a handle to access the RNIC.

How to get there - Code Base (contd)

- Current code
 - uAL layer
 - User can perform operations on RNIC handle, like allocate pd, create qp/cq, register memory, allocate window, etc.
 - Other operations for RDMA : post RQ buffers, post SQ buffers.
 - `free_rnic_handle()` calls `ioctl` with `FREE_RNIC_HANDLE`
 - subsequently closes the “`/dev/rdma1`” device

OpenIB and OpenRDMA – convergence (Feb 2006)

- *Infiniband has the same concept of RDMA using slightly different API's.*
- *How to have a single API that works irrespective of the underlying fabric type ?*
 - *Convergence aspects between openIB and openRDMA is currently being discussed.*
 - *Goal is to have applications not be aware of underlying transport and to have a single implementation in linux.*

Exploiters of RDMA

- ULP's : NFSR, iSER.
- HPC applications based on MPI - typically require :
 - fast access to large amounts of data
 - high-speed interconnects between systems.
- Cluster interconnects.
- Any application with high data to control ratio
 - Wall Street financial applications

References

- An RDMA Protocol Specification
 - www.ietf.org/internet-drafts/draft-ietf-rddp-rdmap-05.txt
- Open RDMA
 - www.openrdma.org
- Interconnect Software Consortium (ICSC)
 - www.opengroup.org/icsc
- Code : Browse @ cvs.sourceforge.net/viewcvs.py/rdma/

Legal Statement

- This work represents the view of the author and does not necessarily represent the view of IBM
- Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.