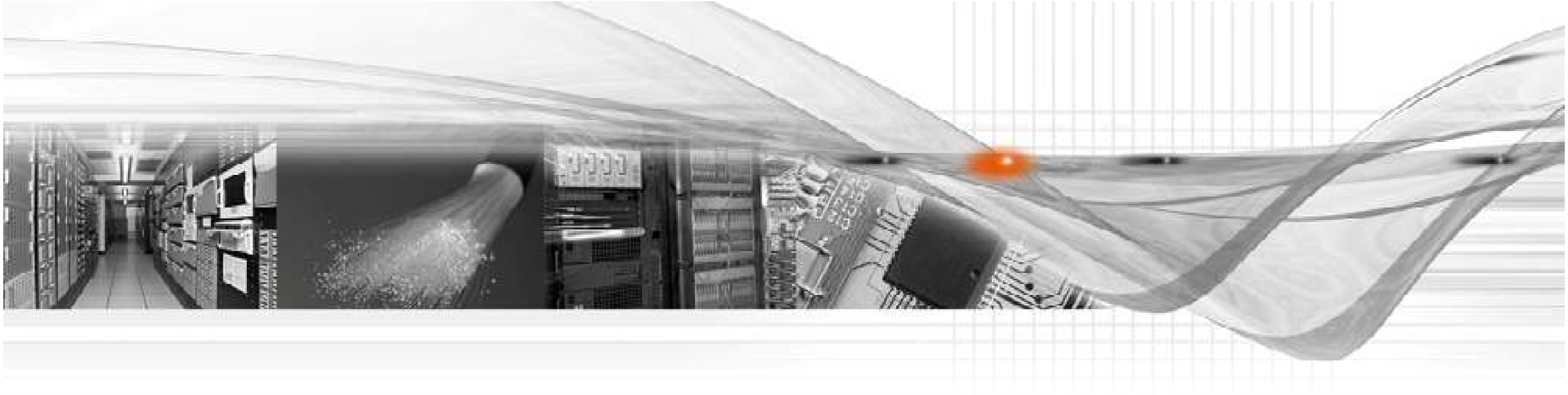


OPENFABRICS ALLIANCE

OpenFabrics Alliance Training Program

Author: Rupert Dance
Date: 11/14/2010

OFA Launches New Training Initiative

- The OFA has created two training courses
  - Introduction to OpenFabrics Software Mini-Course
  - Programming with OpenFabrics Software
- Overall Goals
  - Provide an introduction to, as well as hands-on experience with OpenFabrics Software and related software tools
  - Training is aimed at mainstream application developers, system architects and IT managers in EDC, HPC and Financial Services

Intro to OpenFabrics Software - Goals

- Create a course with appeal to a wide audience, from programmers, system architects to IT Managers.
- Introduction to OpenFabrics Software and Remote Direct Memory Access (RDMA) concepts
- Give developers, system integrators and managers an overview of the benefits of OpenFabrics technologies
- Designed to help attendees understand the value proposition of OpenFabrics Software:
  - Low latency
  - High bandwidth
  - Kernel bypass
- De-mystify OpenFabrics Software and RDMA
- Spark interest in deploying or learning more about OpenFabrics Software and RDMA programming.

Intro to OpenFabrics Software - Syllabus

- Technologies: IB, iWARP, RoCE
- The OpenFabrics concept
  - Reliable and Unreliable Transport Services
  - Message Semantics and RDMA Semantics
  - Fault Tolerance and reliability
  - Multicast communication
- The OFA Stack and ULPs
  - IPoIB
  - iSER
  - NFS-RDMA
  - RDS
  - SDP
  - SRP

- OpenFabrics Utilities
  - OpenSM (OSM): InfiniBand Subnet Manager
  - Diagnostic tools
  - Performance tests
- Value proposition examples
  - Latency – with Financial Services example
  - Bandwidth – with Enterprise datacenter example
  - Kernel Offload – with Cloud/Virtualization example
  - Applications using OFA Software
    - MPI
    - RDS
    - uDAPL

Programming with OpenFabrics Software

- Provide developers with the knowledge and experience they need for writing application programs using RDMA
- Focus on the OFED API, RDMA concepts and common design patterns
- Provide detailed classroom instruction in writing an application to the verbs API
- Provide hands-on experience writing, compiling and executing an application program using the OFA stack
- Opportunity to develop applications on a full-fledged OFA cluster in a working HPC environment
  - The OFA cluster at UNH-IOL includes equipment from all the major InfiniBand and iWARP companies

Programming Course Requirements

- Requirements
  - Knowledge of “C” programming including concepts such as structures, memory management, pointers, threads and asynchronous programming
  - Knowledge of Linux since this course will not include Windows programming
- Helpful
  - Knowledge of Event Handlers
  - Knowledge of sockets or network programming
  - Familiarity with storage protocols

Programming Course Format

- Led by Dr. Robert D. Russell, professor in the CS Department at the University of New Hampshire
- Comprehensive 2 day training program
- 8 hours of presentation and 8 hours of lab work
- Alternating 4 hour slots of training and hands on work
- Includes access to the OFA Cluster located at UNH-IOL
- Primary location will be the University of New Hampshire Interoperability Lab (UNH-IOL)
- The course can be delivered at conventions and company sites
  - Additional fees apply for travel, instructors and equipment required to support the training materials and exercises.

UNH Interoperability Lab

Programming Course Syllabus

- Overview
  - Introduction to OFA architecture
  - A description of the ‘verbs’ versus the verbs API
  - Network perspective
  - Routing/forwarding
    - VLs
    - Congestion control
  - Host perspective
  - Asynchronous processing
  - Channel vs. RDMA semantics
- Data-path basics
  - QP
  - CQ
  - Memory registration
  - LIDs, partitions and protection domains
- Object management basics
  - PD
  - QP
  - CQ
  - SRQ
  - MKEY

- Connection management basics
  - Establishing connections using RDMACM
    - Common practices for managing connections
  - CM protocol state machine
  - RDMA CM state machine
  - RDMACM API
  - RoCE
- Basic RDMA programming
  - Memory registration
  - Object creation
  - Posting requests
  - Polling
  - Waiting for completions using events
  - Common practices for implementing blocking wait
- RDMA/Send Receive/Kernel Bypass design patterns
  - Send-receive
  - RDMA cyclic buffers
  - Rendezvous
  - Buffers:
    - Copy in / copy-out
    - ZCopy using RDMA
    - Pin-down cache
    - Asynchronous pipelining

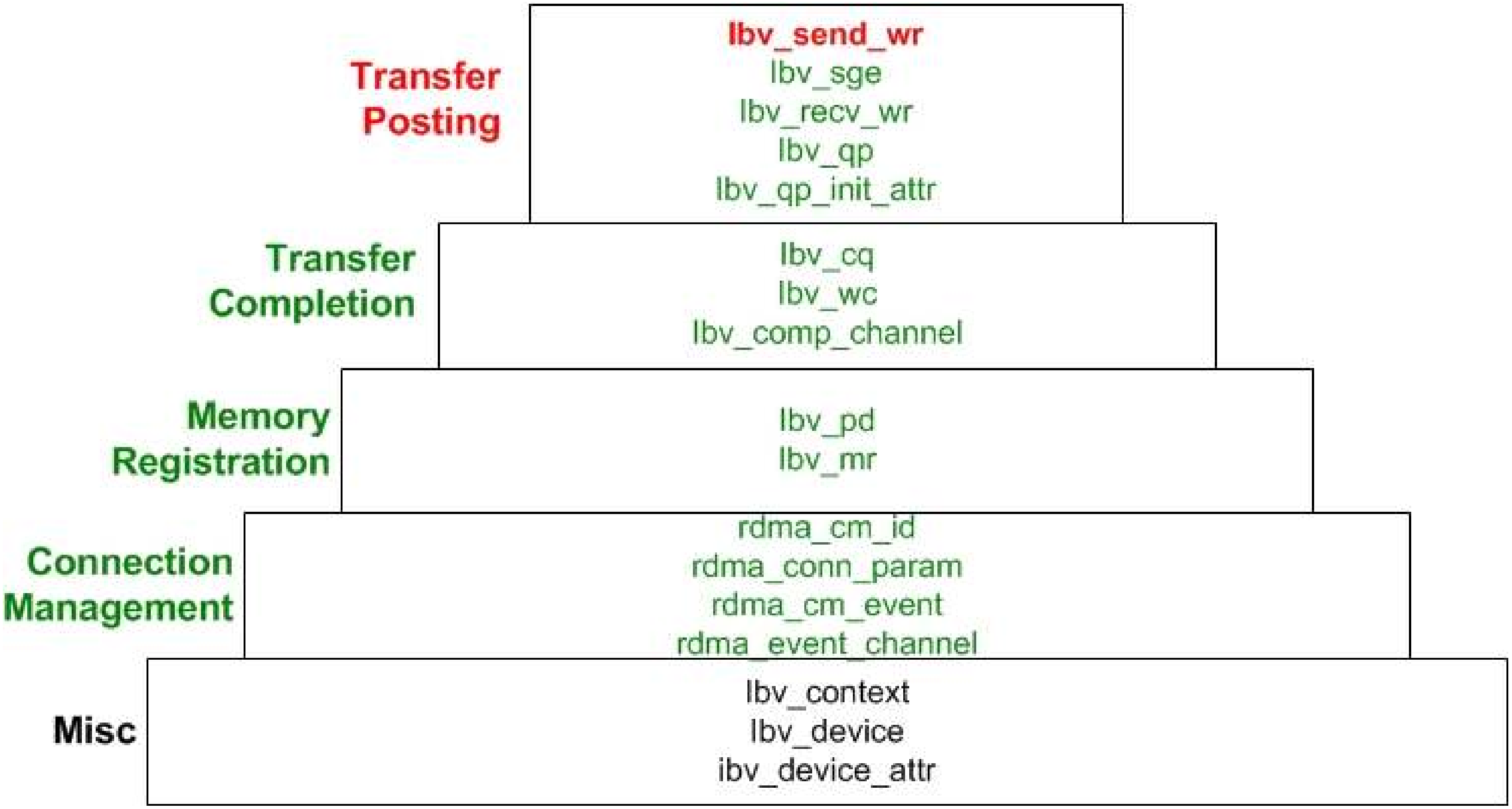
Programming Course Sample

- Description of the verbs
- Description of the data structures
- Preparation for posting a send operation
- Create the work request
- Gathering data from memory
- Putting gathered elements on the wire
- Code samples
- The Big Picture

Programming Course - OFED Verbs

	Setup	Use	Break-Down
Transfer Posting	rdma_create_qp	**Ibv_post_send** Ibv_post_recv	rdma_destroy_qp
Transfer Completion	ibv_create_cq ibv_create_comp_channel	Ibv_poll_cq Ibv_wc_status_str Ibv_req_notify_cq Ibv_get_cq_event Ibv_ack_cq_events	ibv_destroy_cp ibv_destroy_comp_channel
Memory Registration	Ibv_alloc_pd Ibv_reg_mr		Ibv_dealloc_pd Ibv_dereg_mr
Connection Management	rdma_create_id rdma_create_event_channel	rdma_resolve_addr rdma_resolve_route rdma_connect rdma_disconnect rdma_bind_addr rdma_listen rdma_get_cm_event rdma_ack_cm_event rdma_event_str rdma_accept rdma_reject rdma_migrate_id rdma_get_local_addr rdma_get_peer_addr	rdma_destroy_id rdma_destroy_event_channel
Misc		rdma_get_devices rdma_free_devices rdma_query_devices	

Programming Course – Data Structures



Transfer Posting

Ibv_send_wr
Ibv_sge
Ibv_recv_wr
Ibv_qp
Ibv_qp_init_attr

Transfer Completion

Ibv_cq
Ibv_wc
Ibv_comp_channel

Memory Registration

Ibv_pd
Ibv_mr

Connection Management

rdma_cm_id
rdma_conn_param
rdma_cm_event
rdma_event_channel

Misc

Ibv_context
Ibv_device
ibv_device_attr

Posting to send data

- Verb: **ibv_post_send()**
- Parameters:
  - Queue Pair - QP
  - Pointer to linked list of Send Work Requests – SWR
  - Pointer to bad SWR in list in case of error
- Return value:

  == 0 all SWRs successfully added to send queue (SQ)

  != 0 error code

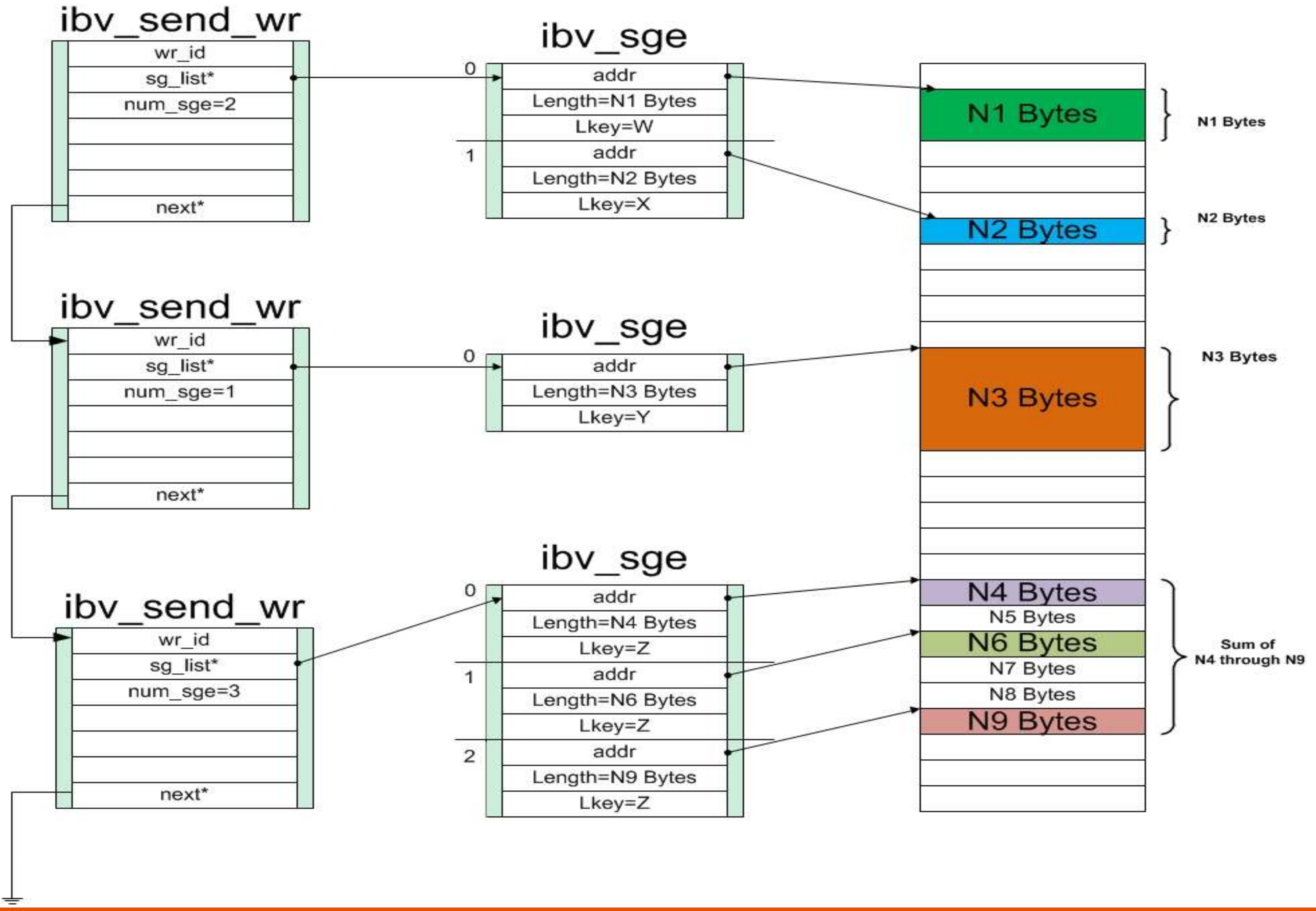
Send Work Request (SWR)

- Purpose: tell network adaptor what data to send
- Data structure: **struct ibv_send_wr**
- Fields visible to programmer:

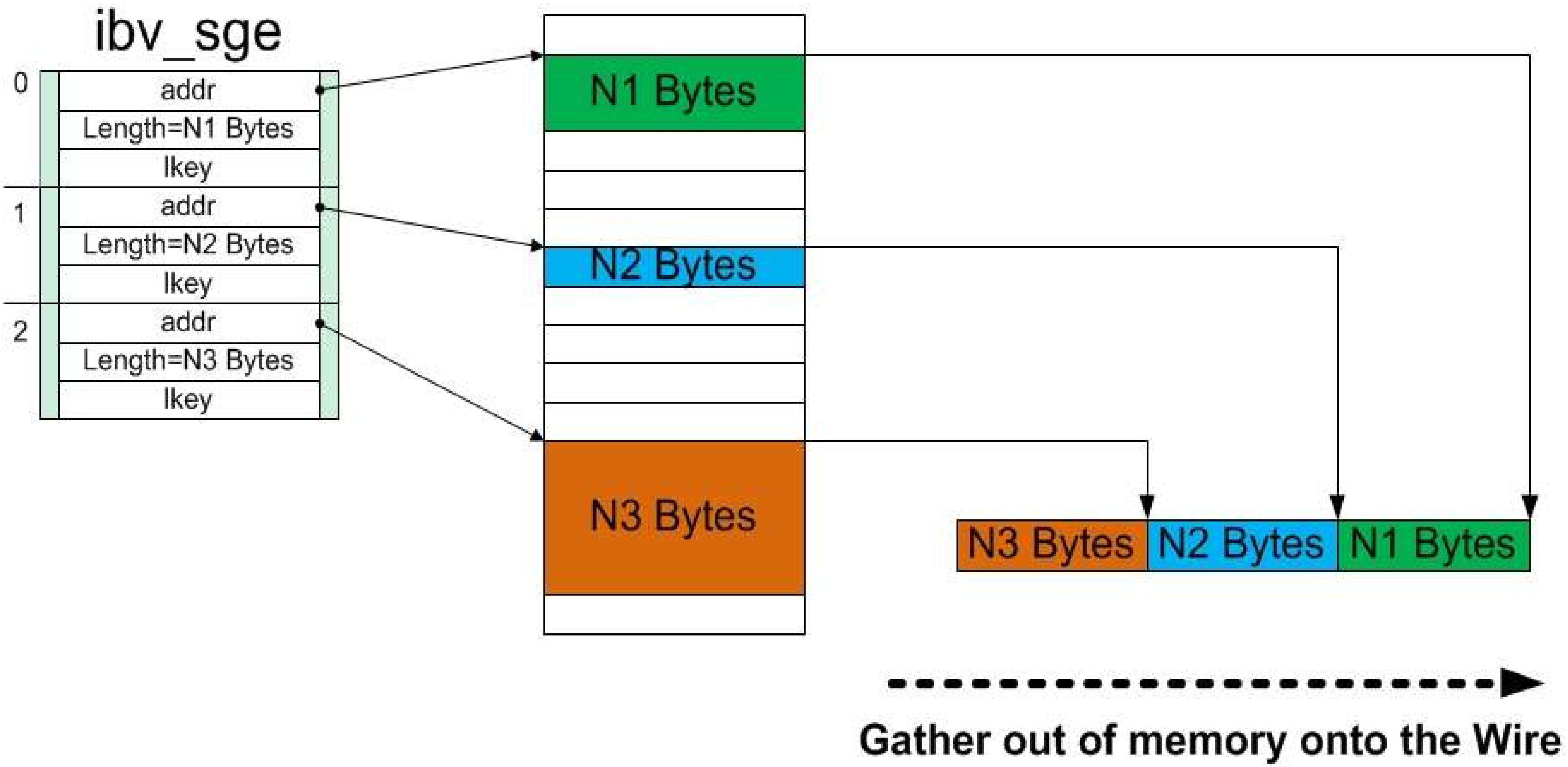
next	pointer to next SWR in linked list
wr_id	user-defined identification of this SWR
sg_list	array of scatter-gather elements (SGE)
opcode	**IBV_WR_SEND**
num_sge	number of elements in **sg_list** array
send_flags	**IBV_SEND_SIGNALED**

- Programmer must fill in these fields before calling **ibv_post_send()**

Creating Scatter Gather Elements



Gather during ibv_post_send()



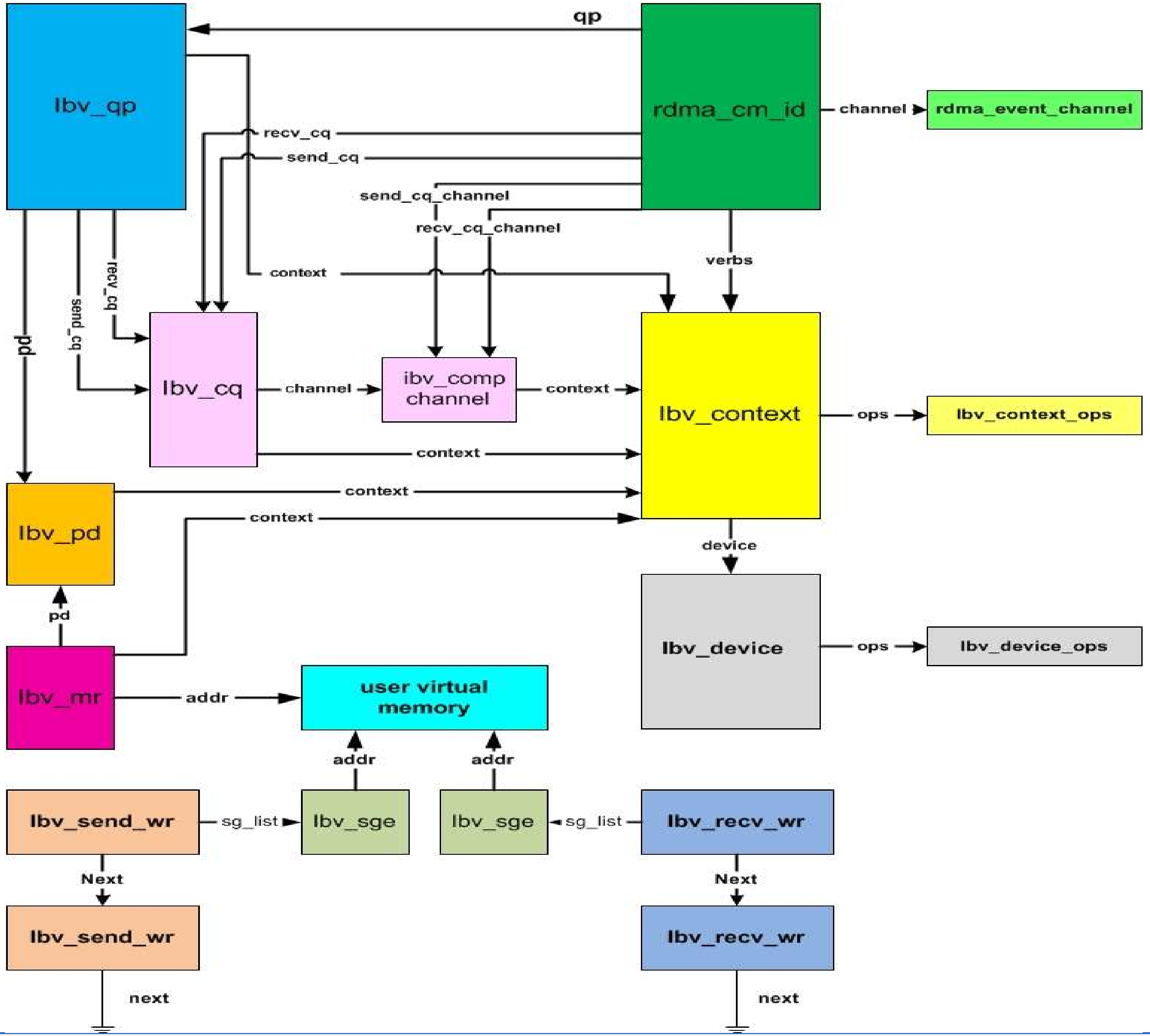
ibv_post_send() code snippet

```
int
our_post_send(struct our_control  *conn, struct ibv_send_wr  *send_work_request,
                struct our_options  *options)
{
struct ibv_send_wr      *bad_wr;
int   ret;

errno = 0;
ret = ibv_post_send(conn->queue_pair,  send_work_request,  &bad_wr);
If (ret != 0) {
      if (our_report_wc_status(ret,  "ibv_post_send",  options) != 0)  {
            our_report_error(ret,  "ibv_post_send",  options);
      }
}
return ret;
}      /* our_post_send */
```

our_setup_send_wr() code snippet

```
static void
our_setup_send_wr(struct our_control  *conn,  struct ibv_sge  *sg_list,
                    enum ibv_wr_opcode  opcode,  int  n_sges,
                    struct ibv_send_wr  *send_work_request)
{
        /* set the user's identification to be pointer to itself */
        send_work_request->wr_id = (uint64_t)send_work_request;

        /* not chaining this work request to other work requests */
        send_work_request->next = NULL;

        /* point at array of scatter-gather elements for this send */
        send_work_request->sg_list = sg_list;

        /* number of scatter-gather elements in array actually being used */
        send_work_request->num_sge = n_sges;

        /* the type of send */
        send_work_request->opcode = opcode;

        /* set SIGNALED flag so every send generates a completion */
        send_work_request->send_flags = IBV_SEND_SIGNALED;

        /* not sending any immediate data */
        send_work_request->imm_data = 0;
}       /* our_setup_send_wr */
```

Programming Course – The Big Picture



Future OFA Software Training Course

- System Administration
  - System configuration
  - Cluster optimization
- Advanced Programming topics
  - Kernel level programming
- ULP Training
  - MPI
  - RDS
  - SRP

OFA Training Course Availability

- **Introduction to OpenFabrics Software Mini-Course** – available upon request at your location or the University of New Hampshire InterOperability Lab
- **Programming with OpenFabrics Software** – January 19-20, 2011 from 8-5 at the University of New Hampshire InterOperability Lab
  - Provide an introduction to RDMA concepts and theory
  - Provide detailed classroom instruction in writing an application to the verbs API.
  - Provide hands-on experience writing, compiling and executing an application program using the OFA stack and software tools.
- Registration available at www.openfabrics.org/training
- For more information contact: rsdance@soft-forge.com