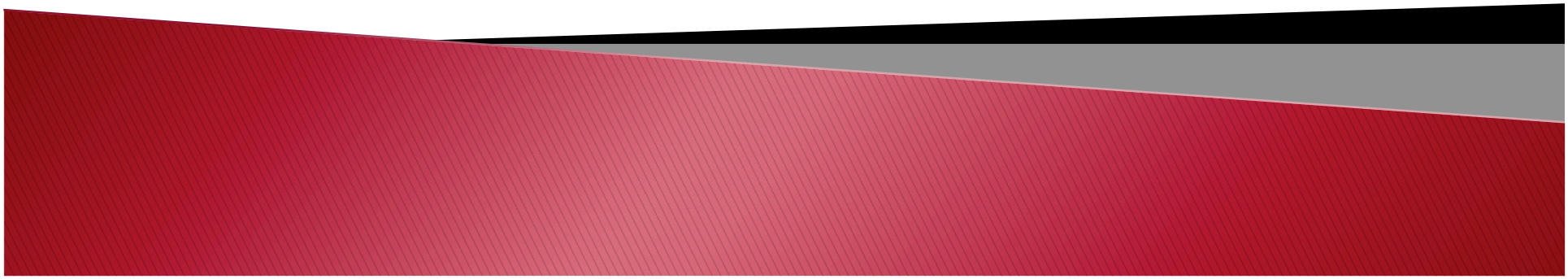
Improving Tool Startup and Runtime Performance

Dorian Arnold

University of New Mexico

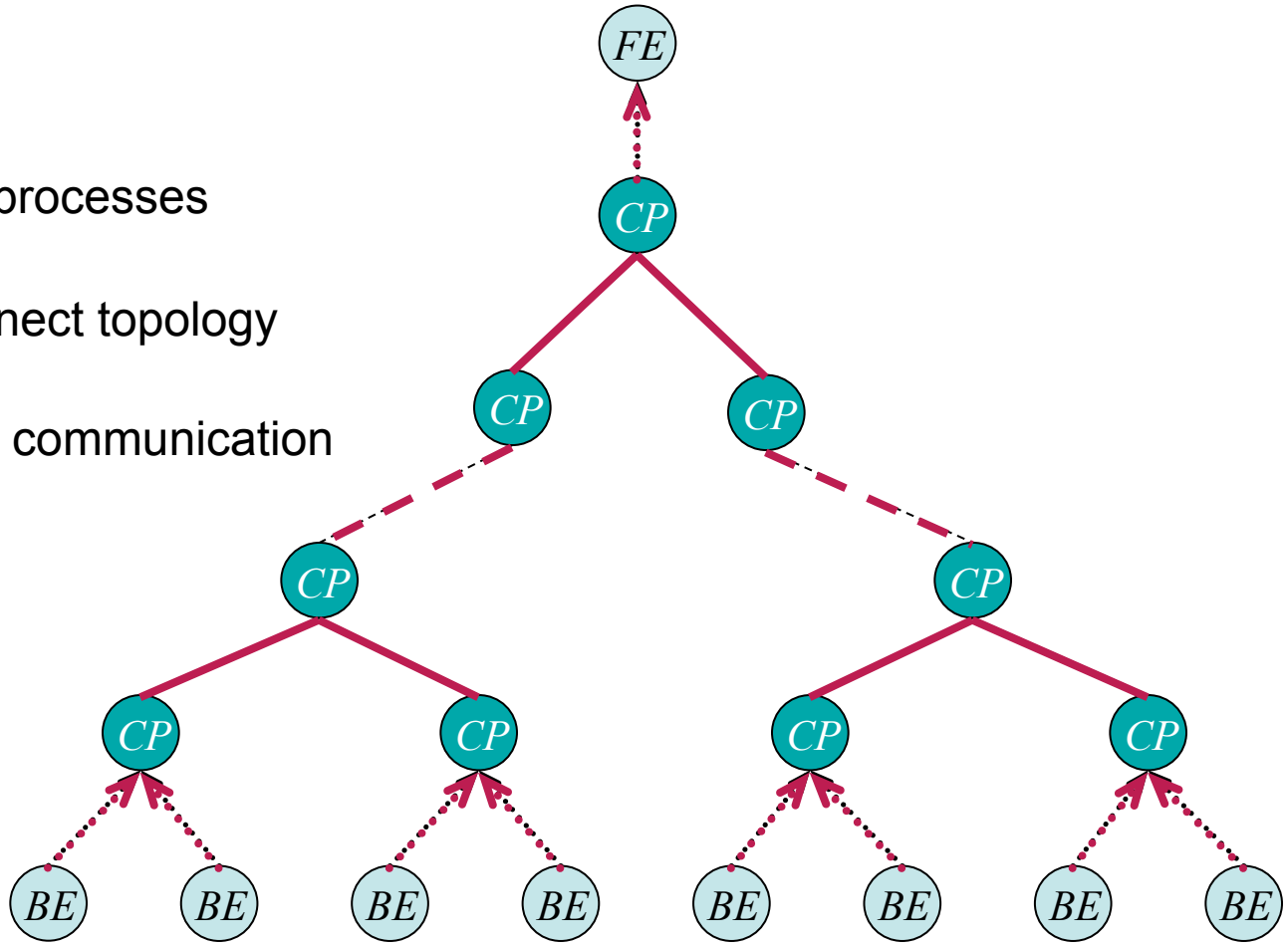


MRNet architecture

(1) Launch processes

(2) Interconnect topology

(3) Runtime communication

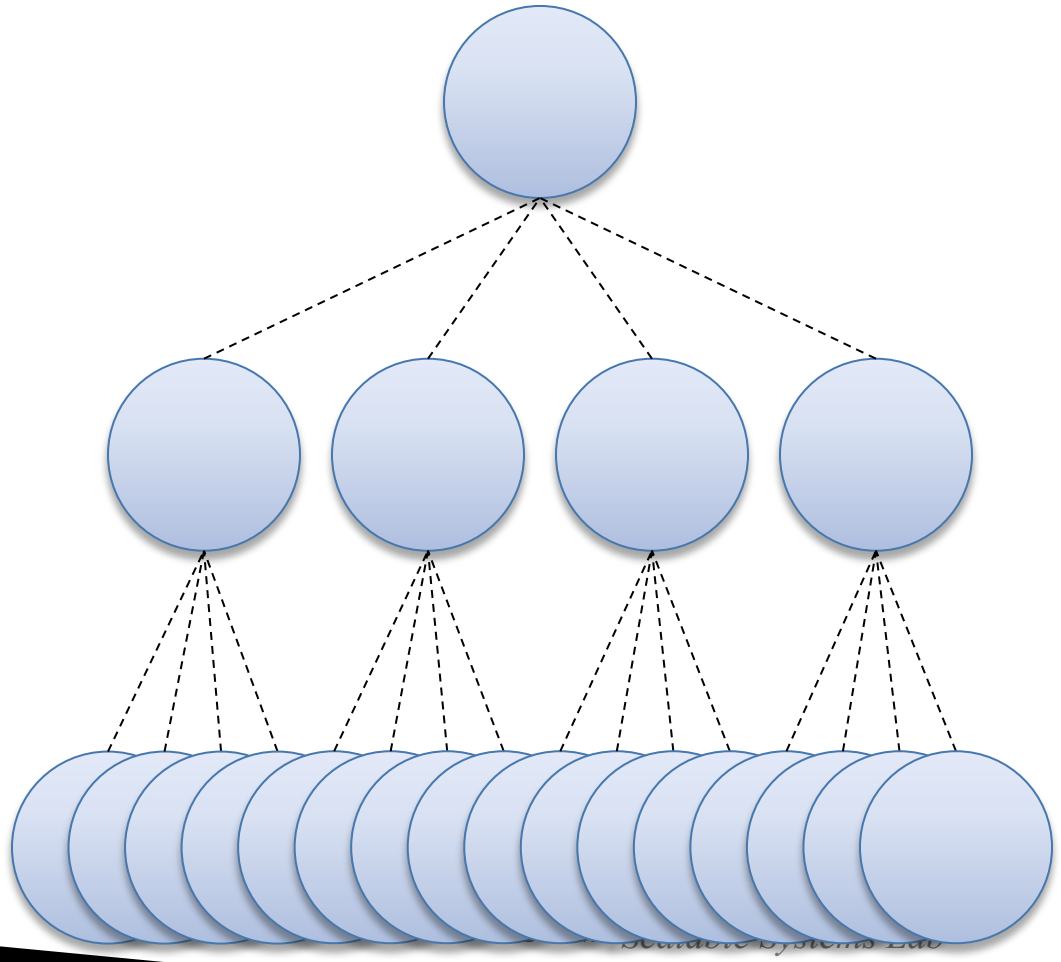


This talk

- How do we plan to leverage native (high-speed) services for improving MRNet?
  - Job launchers/resource managers
  - Communication services and fabrics

Current (sequential) process launch

- Parent creates children
  - Local ➔ `fork()`/`exec()`
  - Remote ➔ `rsh`-based mechanism
- MRNet's "standard"

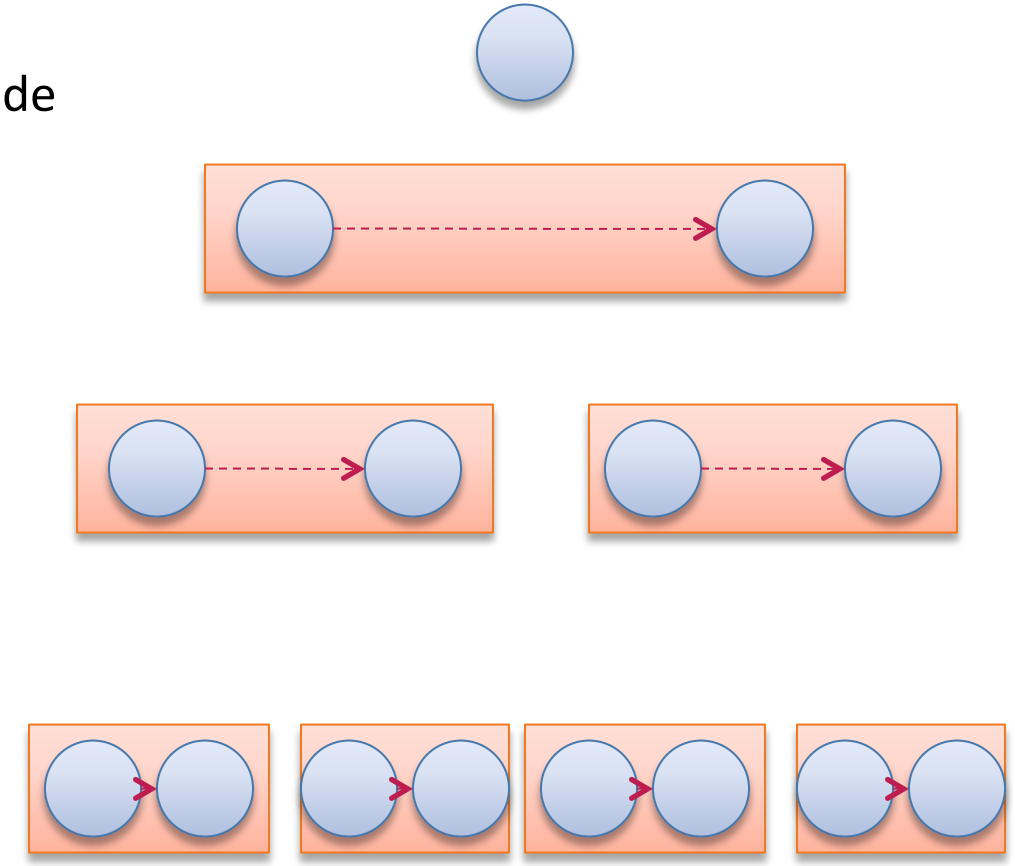


Problems with sequential launch

- Serialized process
  - Often much slower than data collection and analysis
- Resource contention
  - File system (e.g. for program binary)
  - Network

Current XT process launch

- Bulk-launch 1 process per node
- Process launches collocated processes



Current MRNet Topology Dissemination

- Hierarchical, sequential dissemination
  - For both sequential and XT process launch mechanisms

1. Front-end passes to some processes

2. Processes iteratively propagate to other processes

Current MRNet IPC

- TCP/IP for inter-process communication
- Broadcast & point-to-point primitives
- Doesn't necessarily use high-performance networks
- Point-to-point messages transit multiple hops
- No scatter operation

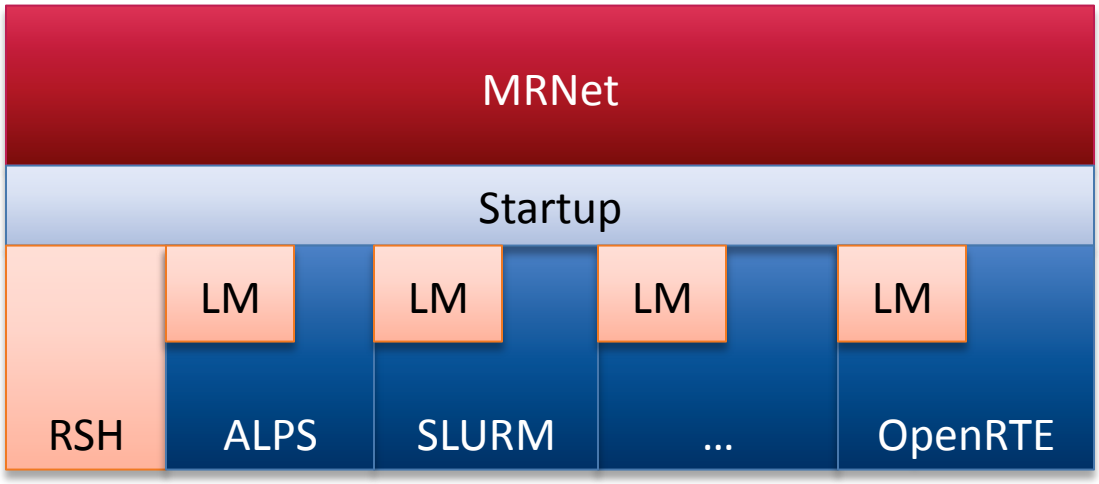
Generic Goals

- Use high-performance services (when available)
- Use reasonable defaults otherwise
  - I.e., current default mechanisms
- Use uniform abstractions and protocols independent of underlying mechanisms
- Increase MRNet portability to new systems

More specifically, we want to …

- Develop a single set of abstractions and protocols for job launch, information dissemination and IPC
- Use native resource managers/job launchers for process creation
- Use scalable services for information dissemination
- Use high-performance runtime IPC

Refactoring MRNet process launch



LaunchMON

- Facilitate creating, porting and maintaining individual tools to large scale HPC systems
- Abstract common operations into a single API with plug-ins for platform specific implementations
- Basic (relevant) services
  - Launch or attach to a job (priming it for tracing)
  - Co-locate tool processes with running application processes

Have you heard the one about …

We need a TBON to scalably* bootstrap our TBON

"That's just so crazy, it just might work!"

* I've applied for U.S. citizenship ☺

A lightweight framework for MRNet bootstrapping

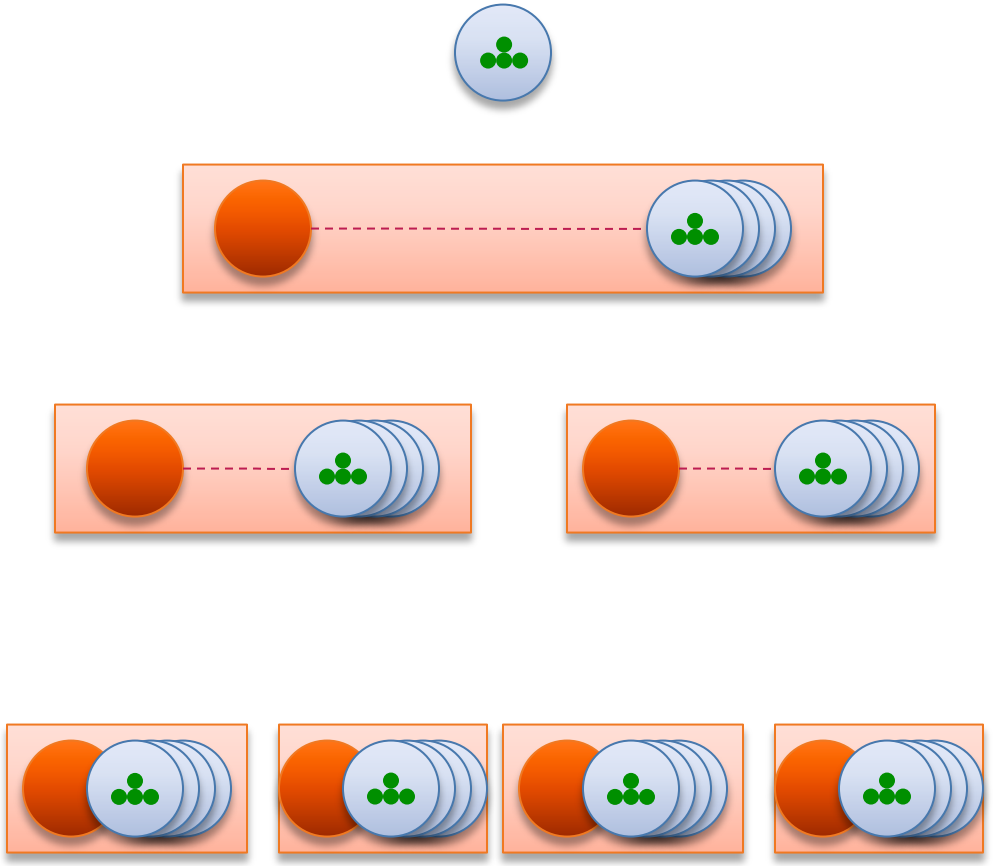
- LIBI: Lightweight infrastructure bootstrapping infrastructure
  - Name is a work in progress ☺
  - Generic service for scalable system instantiation and initialization
  - Used for MRNet startup and torn down afterwards

LIBI Services

- Process launch
- Scalable, low-level collectives

Using LIBI to initialize MRNet

1. Front-end launches LIBI
2. Use LIBI to launch MRNet processes
3. Use LIBI to scatter topology information
   - Parent info
4. MRNet finalizes initialization



Advantages of MRNet of LIBI

- Complete separation of process launch from topology information dissemination
- Consistent, platform-independent framework for process deployment and interconnection
- Refactors platform-dependent mechanisms into single, isolated component

Proposed LIBI implementation

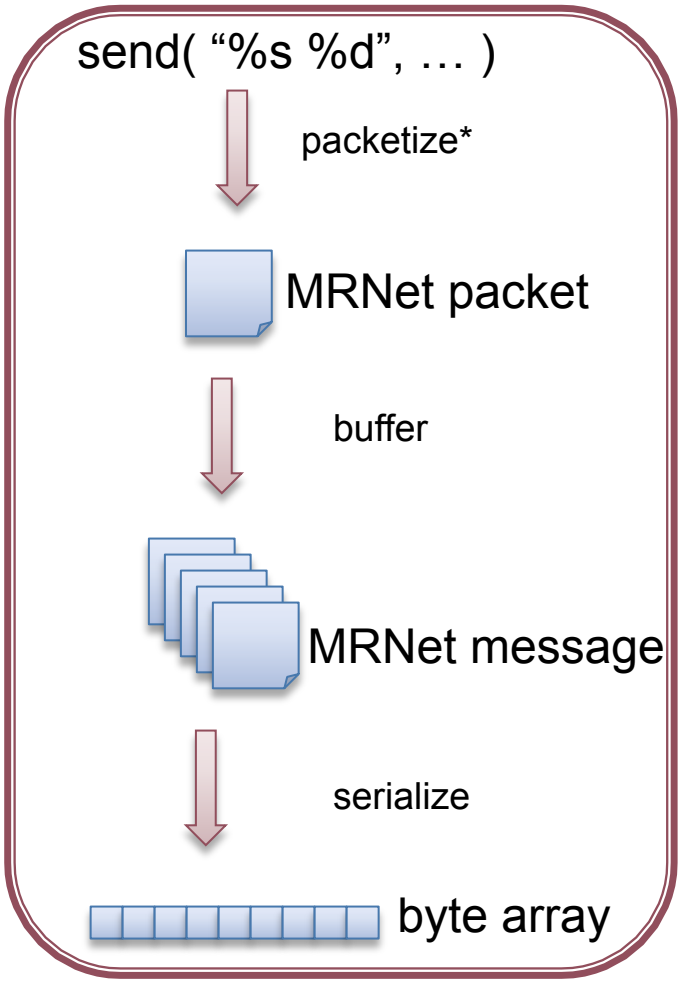
- On initialization, launch LIBI processes
  - 1 LIBI process per relevant node
  - Bulk-launch service when available
  - Rsh-based mechanism when bulk-launch not available
- Organize LIBI processes into tree
- LIBI launch service
  - LIBI front-end retrieves and distributes binaries via LIBI tree to limit file system and network contention
    - Similar to our "scalable binary relocation service"
- LIBI communication service
  - Rudimentary data transfer
  - PMGR-based with COBO as reference implementation

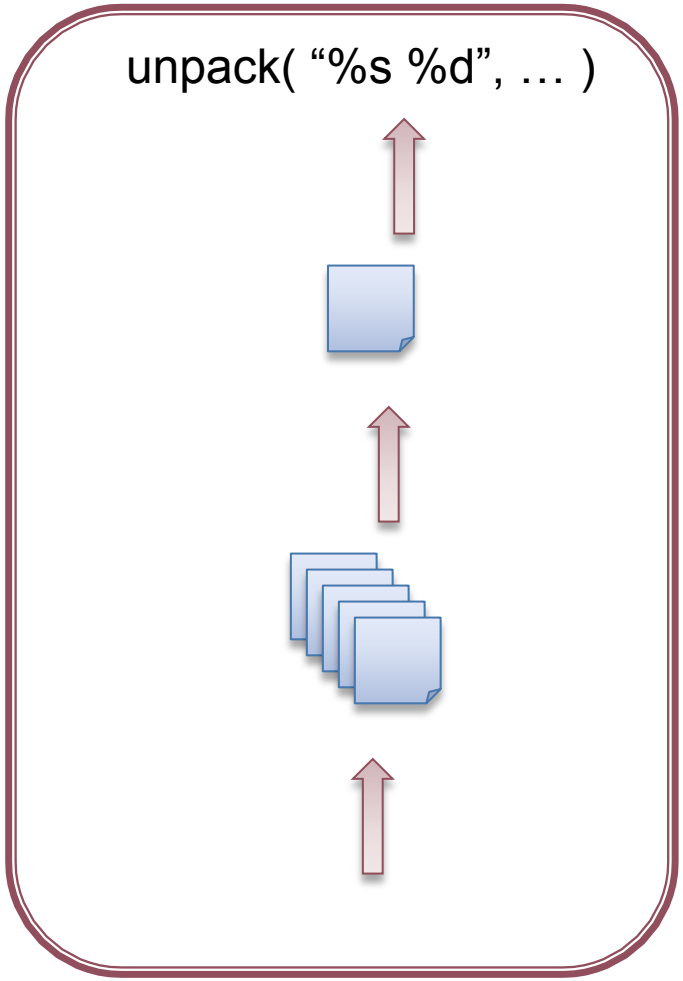
Related work

- SLURM: Simple Linux utility for resource management
  - Persistent daemons
  - Dynamic trees when SLURM command is invoked
  - LIBI would leverage SLURM when available
  - SLURM offers no communication services

- ScELA: Scalable and extensible launching architecture
  - MVAPICH MPI
  - Launches nodes serially
  - No mechanisms for easing file system load
  - Unclear whether ScELA is readily extractable from MVAPICH
    - Did the get the “componentization” memo?

MRNet IPC



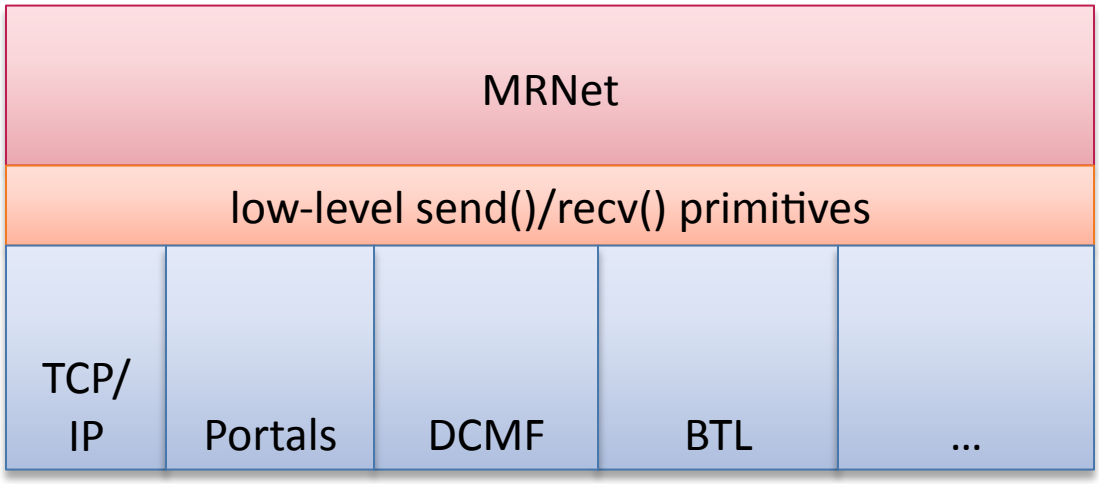


* Also in the GWBED, right after “misunderestimate”

MRNet IPC

- TCP/IP connections
  - Multicast over unicast approach
- Abstract communication layer
  - Point-to-point, group, scatter operations
  - Allow flexible implementation replacement
    - True multicast
    - TCP, Custom-networks, MPI, IP multicast, …
    - Bypass tree for direct point-to-point
    - Shared memory
    - One-sided communication, RDMA, …

Refactoring MRNet IPC



Basic primitives

- session establishment
  - single end-point
  - group of end-points
  - bi-directional
  - Should back-ends be allowed to establish sessions
- send data
  - unicast, broadcast (implied by session establishment parameters)
  - scatter
- receive data

Abrupt Transition Ahead

Place abrupt transition here …

- Tools and failure/recovery models
- As systems scale up and failures increase, how does tools/tool infrastructure need to evolve?
- Failure models:
  - crash stop, byzantine, silent errors, hardware vs. software errors, …
- Fault-tolerance models:
  - Ignore and continue, restart, save/restore (process/ communication) state, …