



MPICH-GP : A Private-IP-enabled MPI

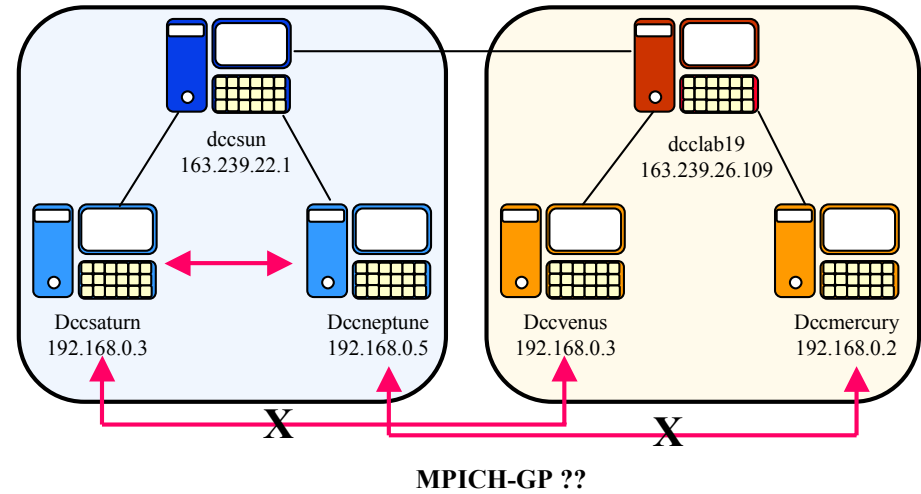
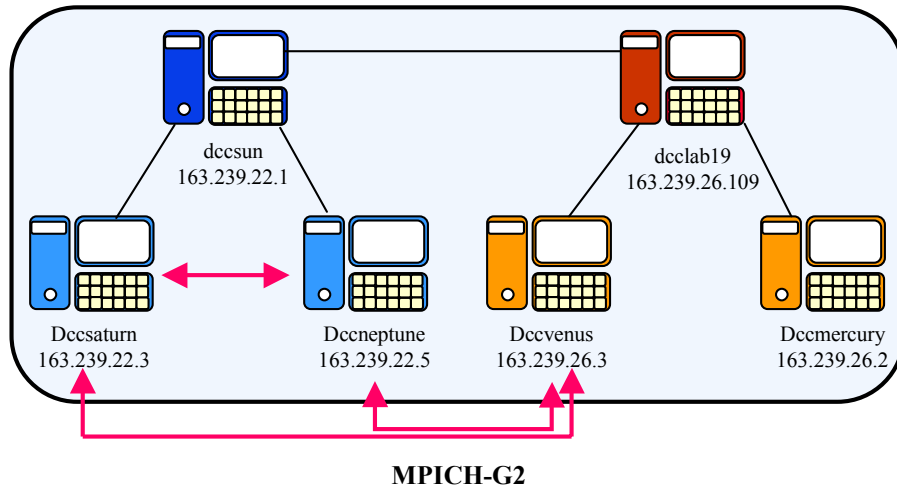
Kumrye Park ¹, Sungyoung Park ¹, Ohyoung Kwon ²

¹ Distributed Computing & Communication Lab.
Dept. of Computer Science, Sogang University
Seoul, South Korea
namul@sogang.ac.kr

² Korea University of Technology and Education
Chonan, South Korea

<http://dcclab.sogang.ac.kr>

Why is this needed?



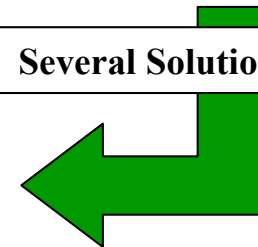
1. In MPICH-G2, all the nodes need public IPs.
2. In the private IP cluster, it can't run a program requiring bigger jobsite than the cluster size.



1. Don't need to change the existing Private IP cluster settings. (save resources and labours)
2. Can run a program both in the private IP and public IP cluster.

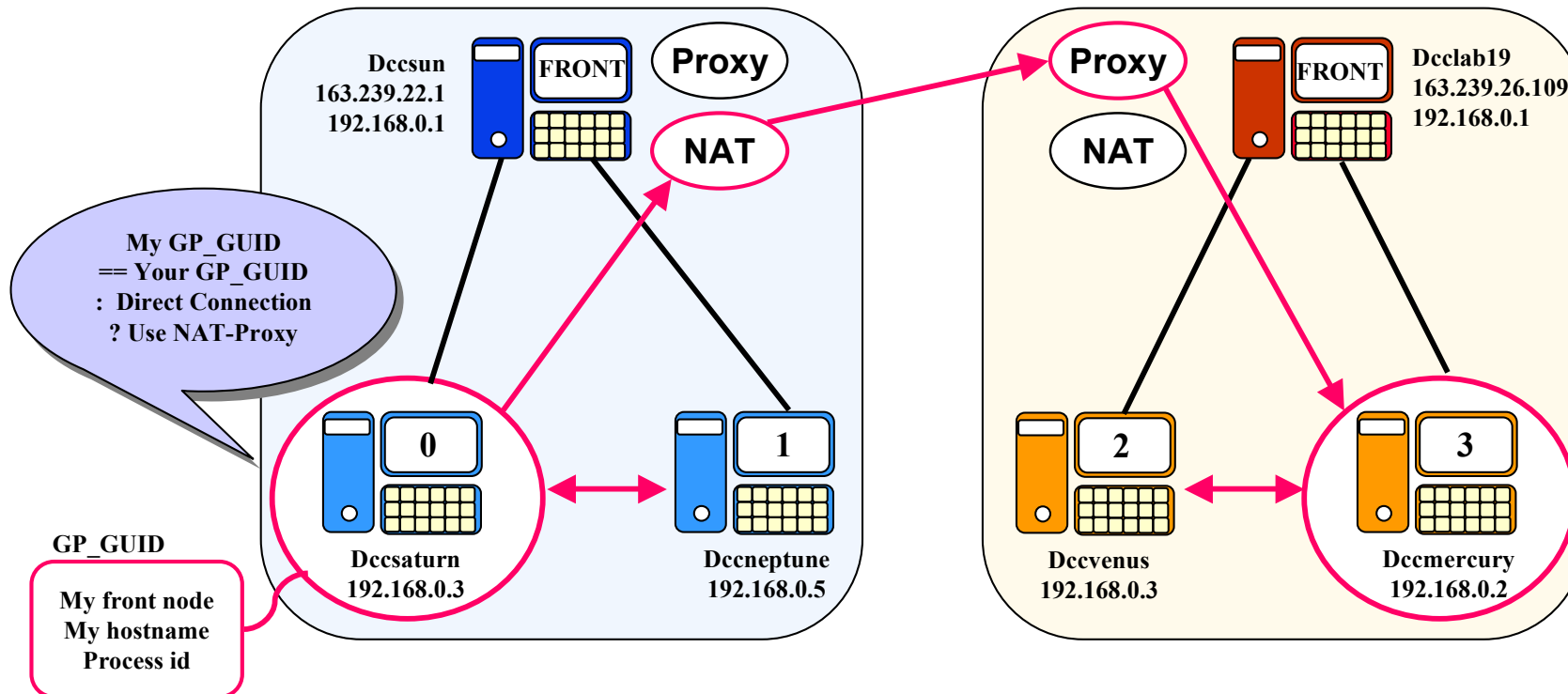
	User – Level	Kernel - Level
Advantage	Easy to implement	No User-Kernel context switching
Disadvantage	2 times of User-Kernel context switching	Lack of portability

Several Solutions can be!!



Message Relay Scheme : NAT + Proxy

- In the same cluster, communicate with each other directly (ex. rank 0 <-> rank 1)
- Outgoing streams go through the NAT service(kernel-level), and then reach the proxy daemon (ex. rank 0 <-> rank 3)
- Proxy Daemon(user-level) forwards incoming streams into the appropriate nodes



GP_GUID & Connection Decision Algorithm

- GP_GUID is an unique ID to distinguish nodes

- ❖ GP_GUID is a structure containing information as follows

front_name : FDQN of front node

compute_name : hostname

process_id : process number

- Front name

- ❖ The front name is used to distinguish internal nodes and external nodes.
- ❖ The front name can differ according to the type of the cluster to support both private IP cluster and public IP cluster.

Private IP cluster : front_name = FDQN of the front node

Public IP cluster : front_name = the hostname

- Connection Decision Algorithm

- ❖ We deploy MPICH-GP both in the private IP cluster and the public IP cluster.

Connection decision algorithm

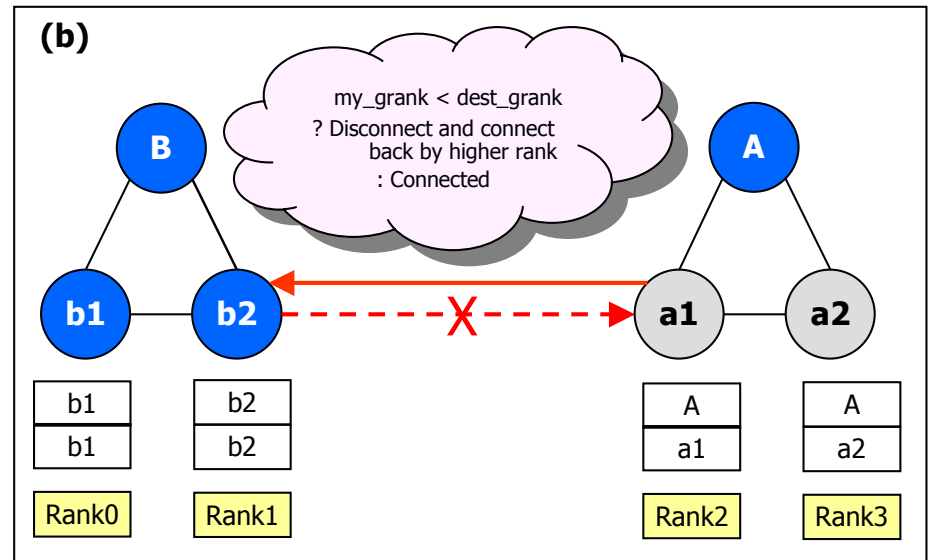
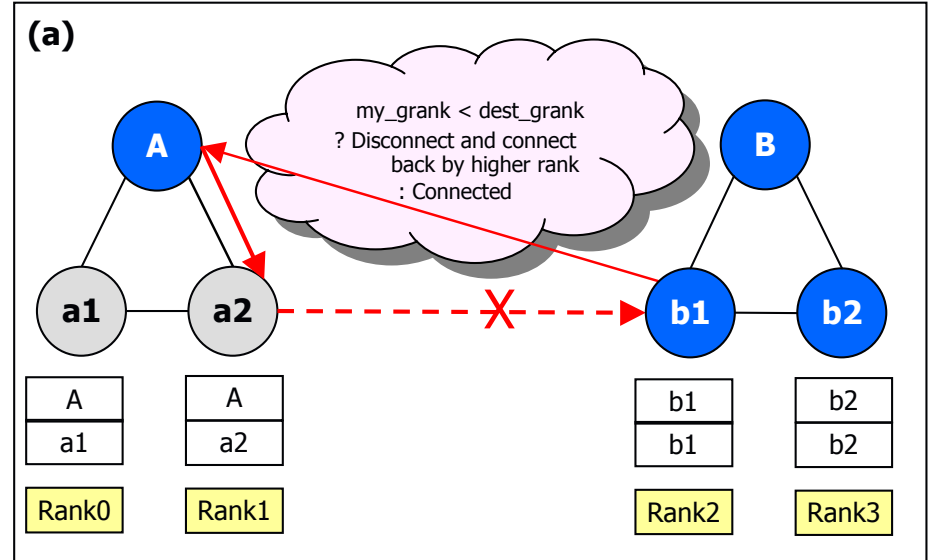
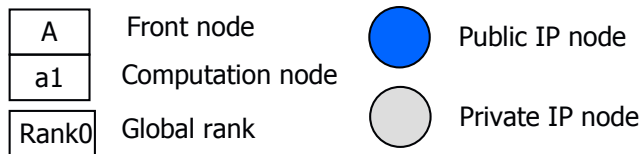
```
IF ( My front_name == Dest's front_name )
{
  The two nodes are in the same cluster.
  Make a direct connection to the destination node. }

ELSE
{
  The two nodes are in the different cluster.

  IF ( Dest's front_name == Dest's compute_name )
  {
    The destination node has a public IP.
    Make a direct connection to the destination node. }
  ELSE
  {
    The destination node is in the private IP cluster.
    Make a connection via Proxy Daemon. }
}
```

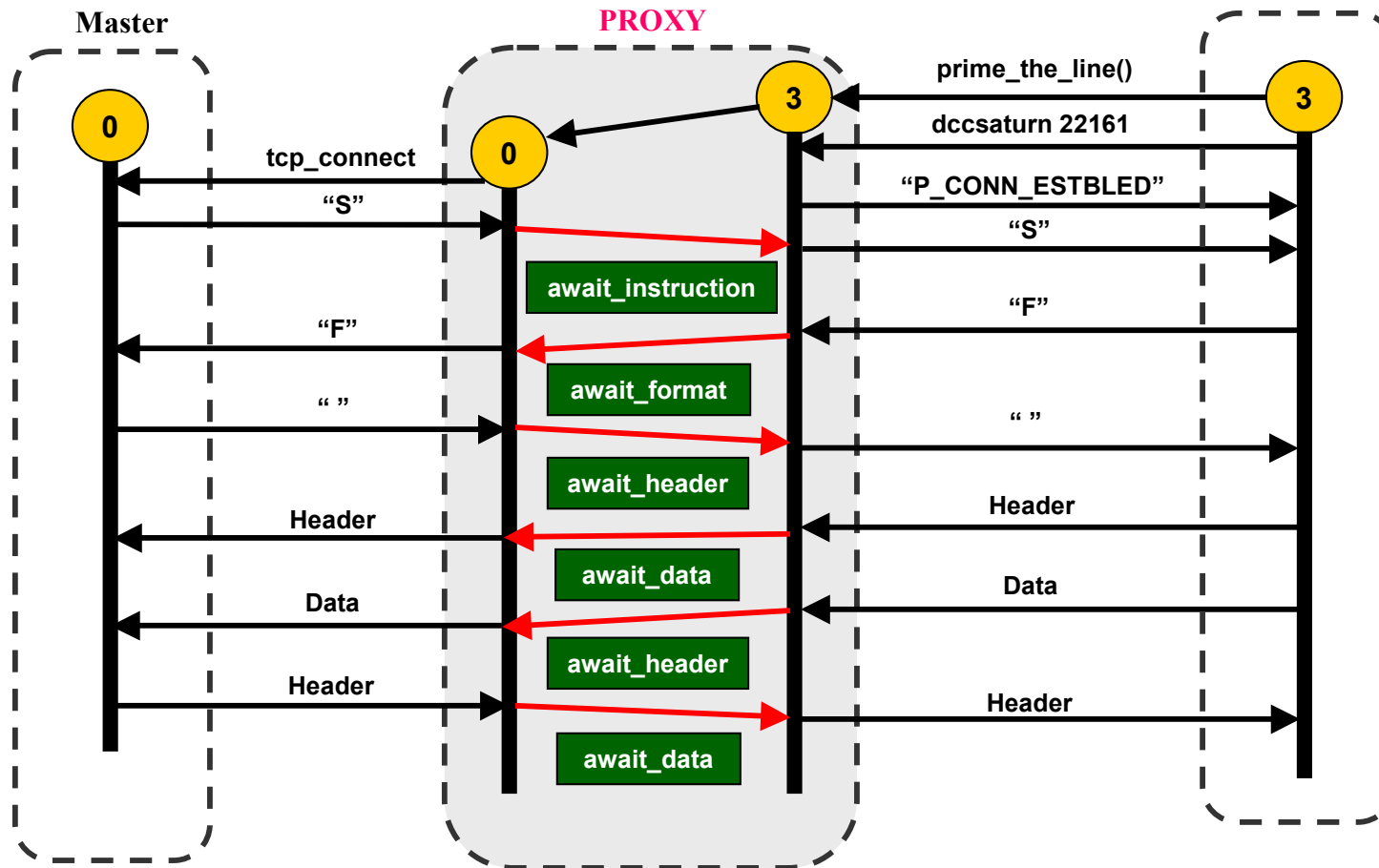
Global Rank Management

- In the MPICH-G2 protocol
 - ❖ Compare the grank of dest node with the src one -> if the dest's grank is bigger than the src -> the dest node connects back to the src.
- According to the global rank of the destination node, we can reduce the overhead of message relaying.
- It's better to give a higher global rank to a node having a public IP.



Proxy Daemon Implementation

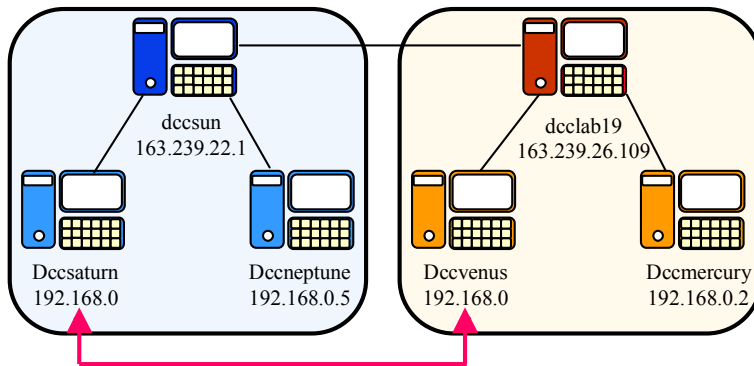
- Proxy Daemon implemented using Globus I/O.
 - ❖ MPI_Send/Recv via Proxy daemon



Performance Test(I)

■ Experimental Setting

- ❖ 800MHz CPU, 256 KB Cache, 1G Memory, 100M Ethernet NIC
- ❖ Two private IP clusters in local area network

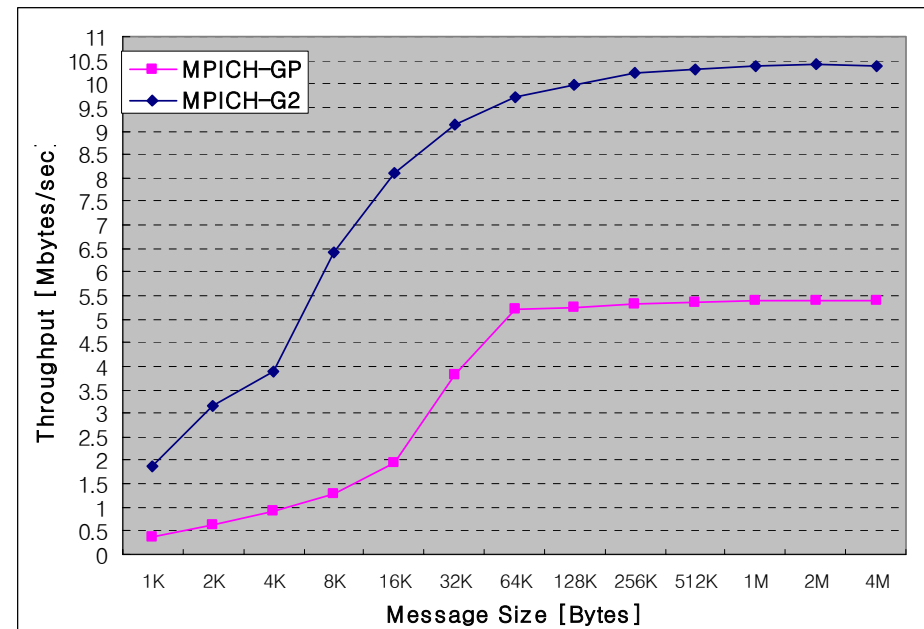


■ Benchmark Suite : PMB 2.2.1

- ❖ Ping-Pong test based on MPI_Send / MPI_Recv
- ❖ Two private IP cluster in a local area network

■ Ping-Pong test results

- ❖ Relaying Overhead (the worst case)
 - Small Msg(under 1K) : **average 16%**
 - Large Msg(over 1K) : **average 40%**
- ❖ Better performance in small message.



Performance Test(II)

■ Benchmark Suite : NPB 3.1

- ❖ Latency comparison of CG, LU, IS benchmark between MPICH-G2 and MPICH-GP
- ❖ Each of benchmarks has different communication patterns

■ CG, LU, IS benchmark results

- ❖ In CG and LU, MPICH-GP performs better than MPICH-G2
- ❖ IS Benchmark result (b) shows that PACX-MPI performs better than MPICH-G2 as the message size goes larger
- ❖ PACX-MPI
 - Use message compression
 - Use two user-level proxy (communication nodes)

- To improve the performance of MPICH-GP, we are integrating a module for selective message compression

