Supercomputer Design Through Simulation

Cray User Group (CUG) Meeting
Lugano, Switzerland

Rolf Riesen
rolf@cs.sandia.gov

Sandia National Laboratories

May 9, 2006

Talk Overview

Section Outline

Goal

- Simulate a supercomputer; e.g., Red Storm, using federated discrete event simulators
- With enough fidelity to make future purchase and design decisions concerning things like:
  - CPU choice
  - Memory size and speed
  - Network interface
  - Topology
  - Application behavior
  - Research directions
  - etc.
- Created initial prototype with promising attributes
- This talk describes simulator
- Collective results on Thursday

Section Outline

Node (Application)

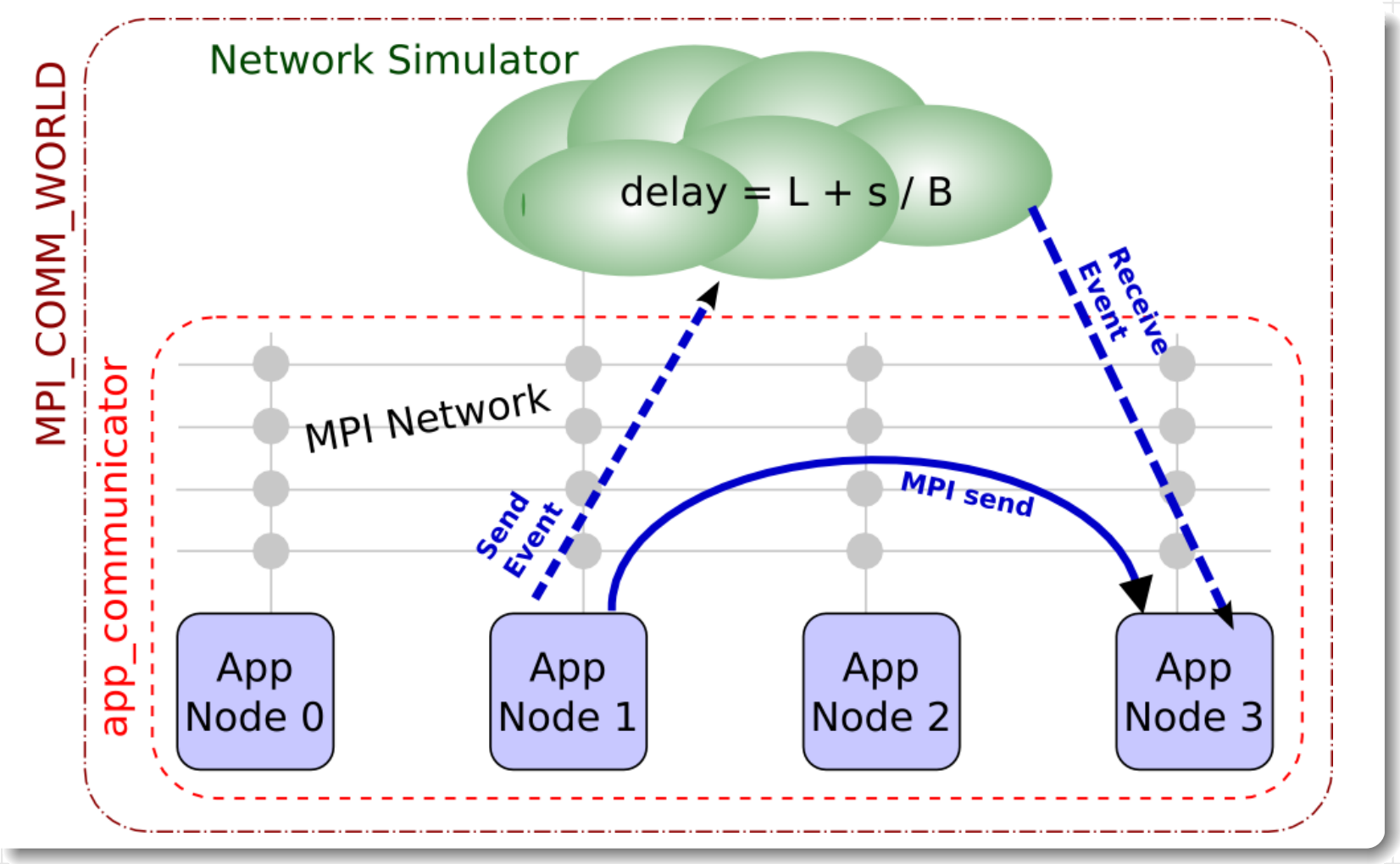
Hybrid simulator:

- App runs regularly and uses MPI to exchange data
- Each MPI send and receive generates an event to the network simulator
- Sim generates rcv events that are matched by clients
- Algorithm determines when and how to update virtual time on each node
- Use MPI wrappers and profiling interface
- Current network simulator uses simple model:

$$\Delta = \frac{s}{B} + L$$

Δ	network delay	B	network bandwidth
s	message size	L	network latency

MPI Wrapper Library



MPI Wrapper Library

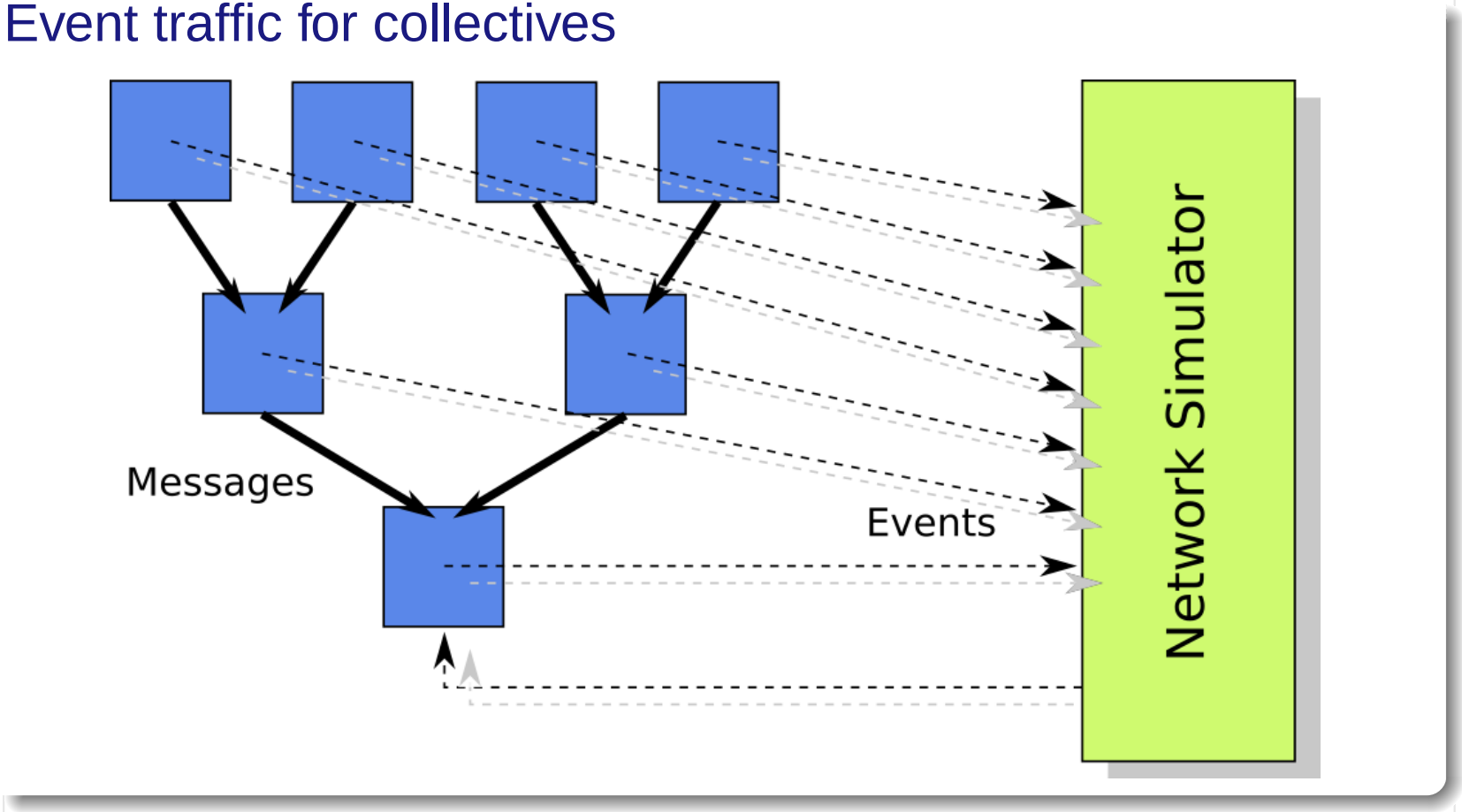
```
int  MPI_Send(void *data,
              int  len,
              MPI_Datatype dt,
              int  dest,
              int  tag,
              MPI_Comm comm)
{
    t_x = get_vtime();

    //  Send the MPI message
    rc= PMPI_Send(data, len, dt, dest, tag,  comm);

    //  Send event to simulator
    event_send(t_x , len, dt, dest,  tag);

    return rc;
}
```

MPI Wrapper Library

```
int MPI_Recv(void *data, int len, MPI_Datatype dt, int src,
             int tag, MPI_Comm comm, MPI_Status *stat)
{
    t1 = get_vtime();

    // Receive the MPI message
    rc = PMPI_Recv(data, len, dt, src, tag, comm, stat);

    // Wait for the matching event
    event_wait(&tx, &Δ, stat->MPI_TAG, stat->MPI_SOURCE);

    if (tx + Δ > t1)
        t3 = tx + Δ;
    else
        t3 = t1;
    set_vtime(t3); // Adjust virtual time
    return rc;
}
```

MPI Wrapper Library

Event traffic for collectives



MPI Communicators

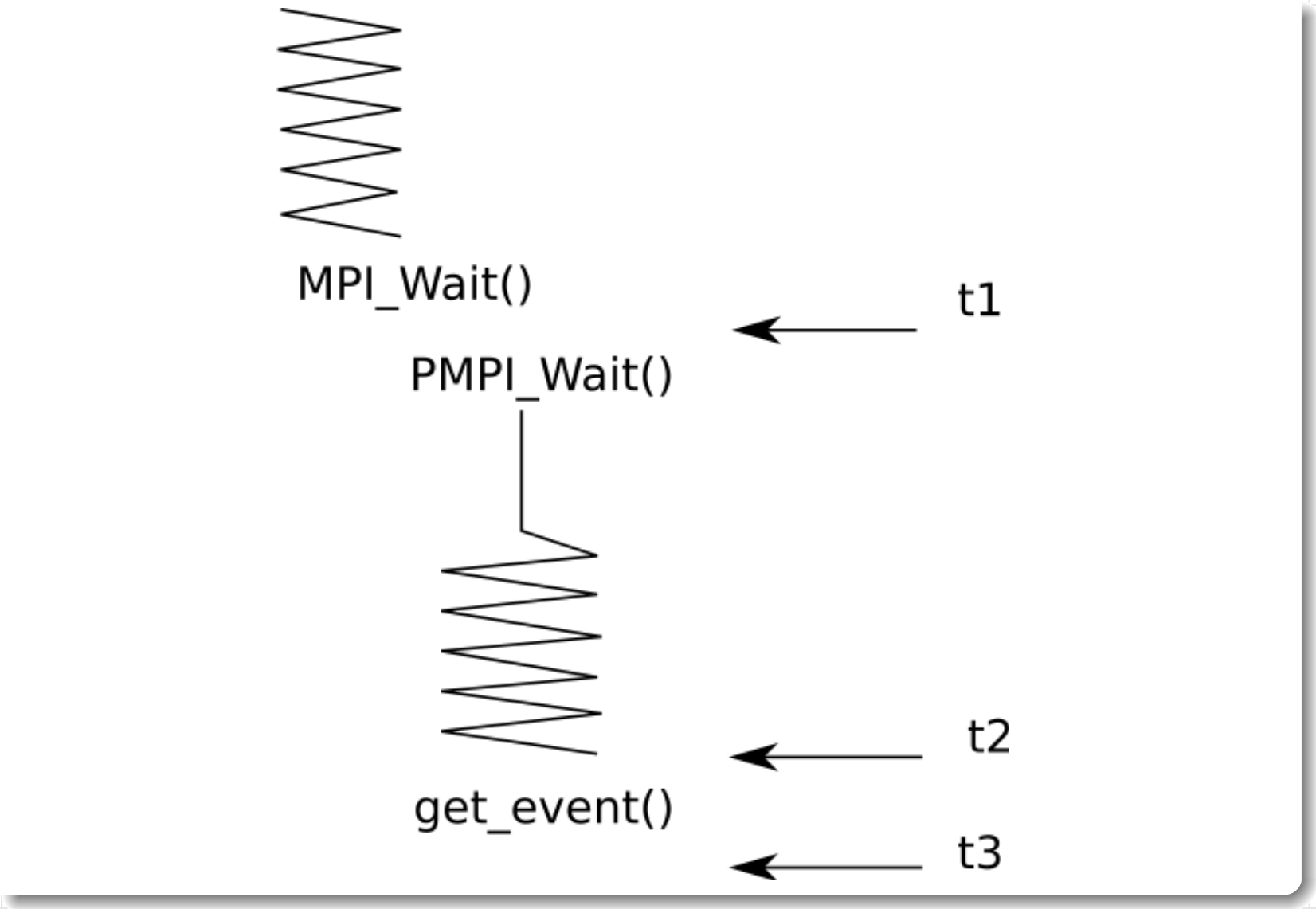
- Simulator framework sets up communicator for application nodes only
- **`MPI_COMM_WORLD`** covers application and simulator
- Wrappers swap **`MPI_COMM_WORLD`** with internal communicator when application calls MPI
- Application never sees real **`MPI_COMM_WORLD`**

Virtual Time

```
if  (t_x + Δ > t_1)
      t_3 = t_x + Δ;
else
      t_3 = t_1;
set_vtime(t_3);
```

- If message was sent earlier than we started looking for it, we have to assume it was already here
  - Just “erase” the time we spent actually receiving it
- If message arrived after we started waiting for it, use the virtual send time + Δ to set local virtual clock

Virtual Time



Linking

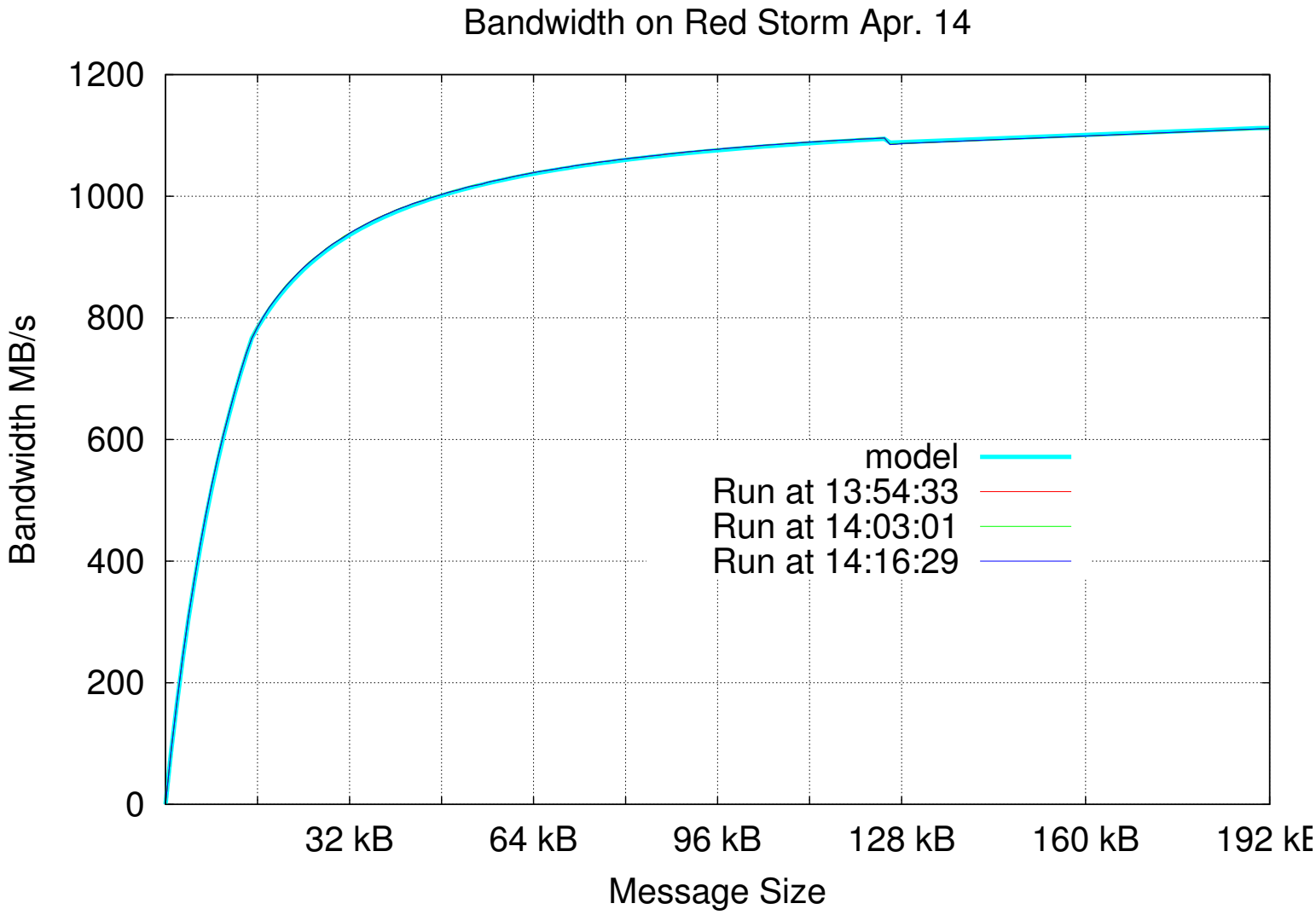
- Currently need to rename **main()** to **main_node()**
- Should not be necessary when we use **MPI_Init()**
- In Fortran programs **`program`** has to be changed to **`subroutine main_node`**
- No Changes to application are necessary!

Section Outline

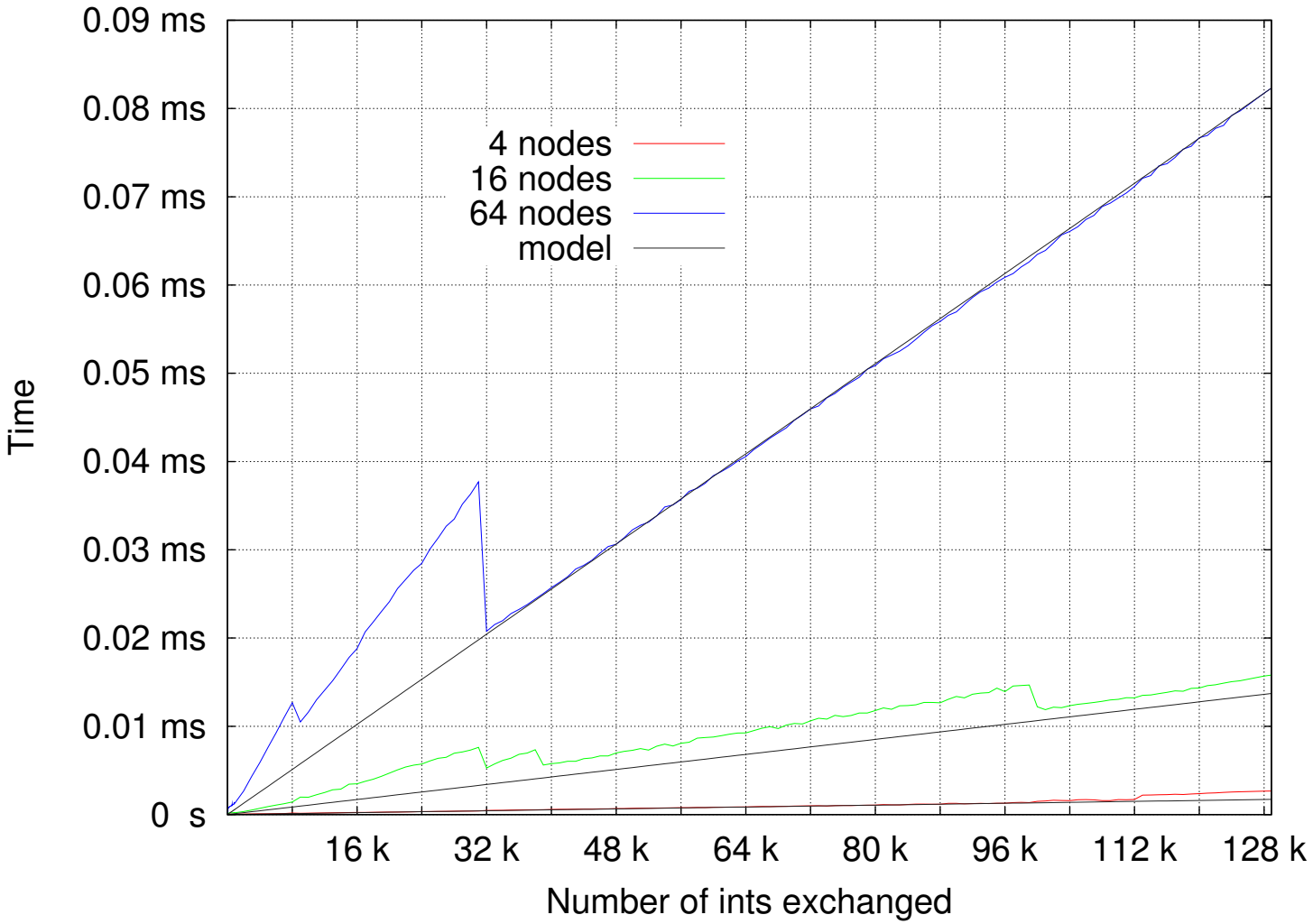
Usage

- Two steps:
  - Create point-to-point model
  - Create collective model
- Measure two-node latency curve and write function to model it
- Measure all-to-all performance and write model

Usage

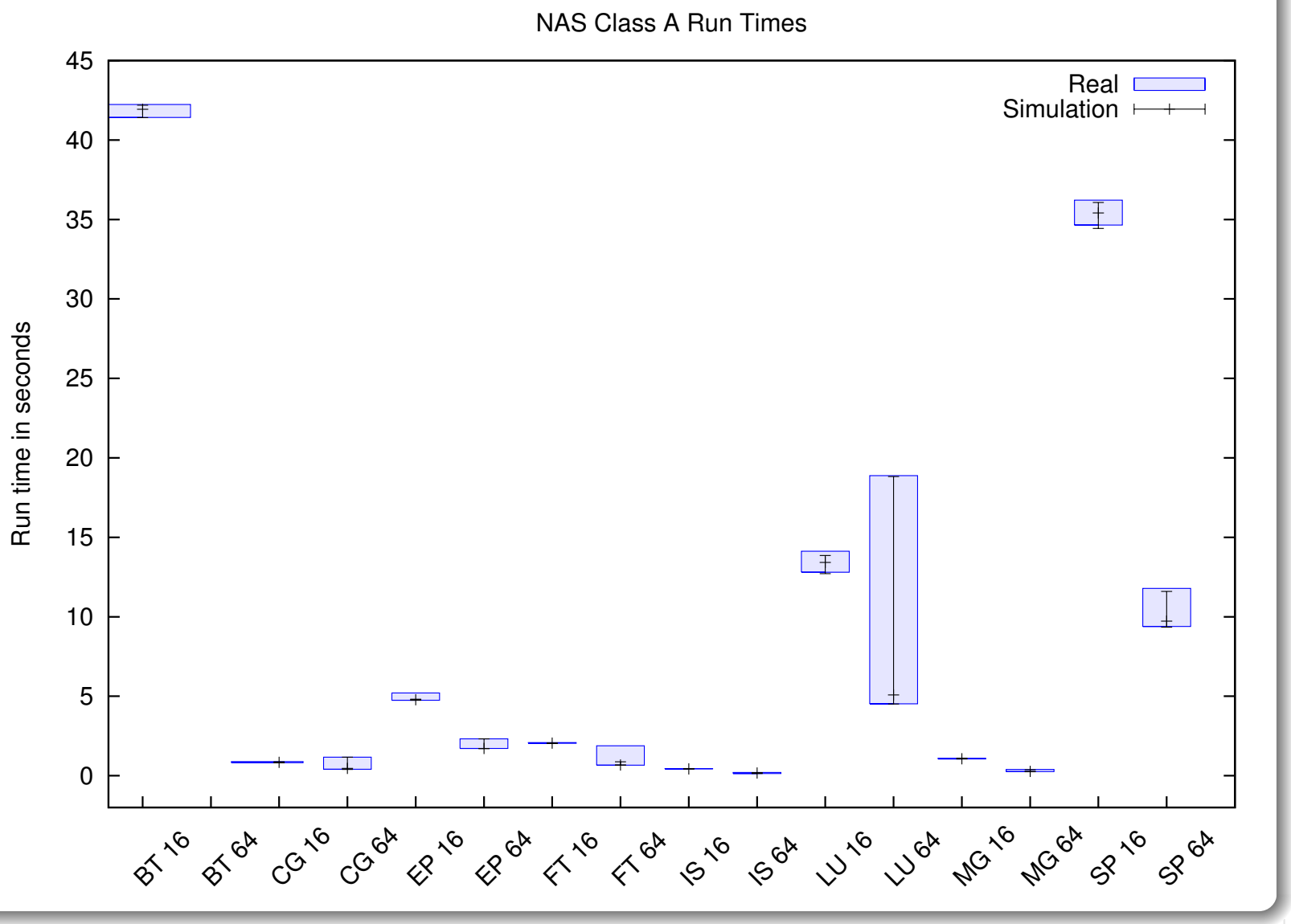


Usage



Section Outline

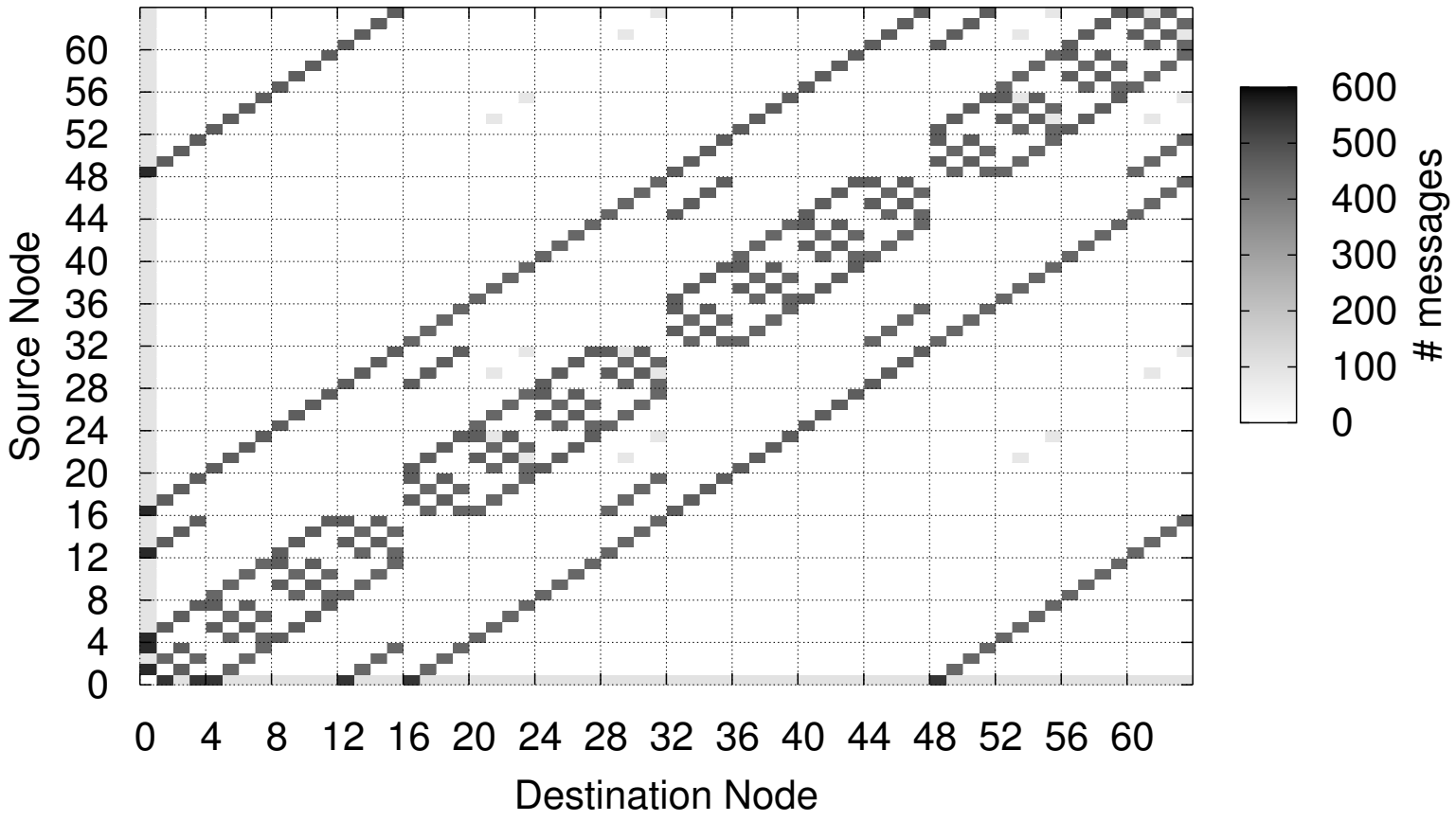
Validation



Section Outline

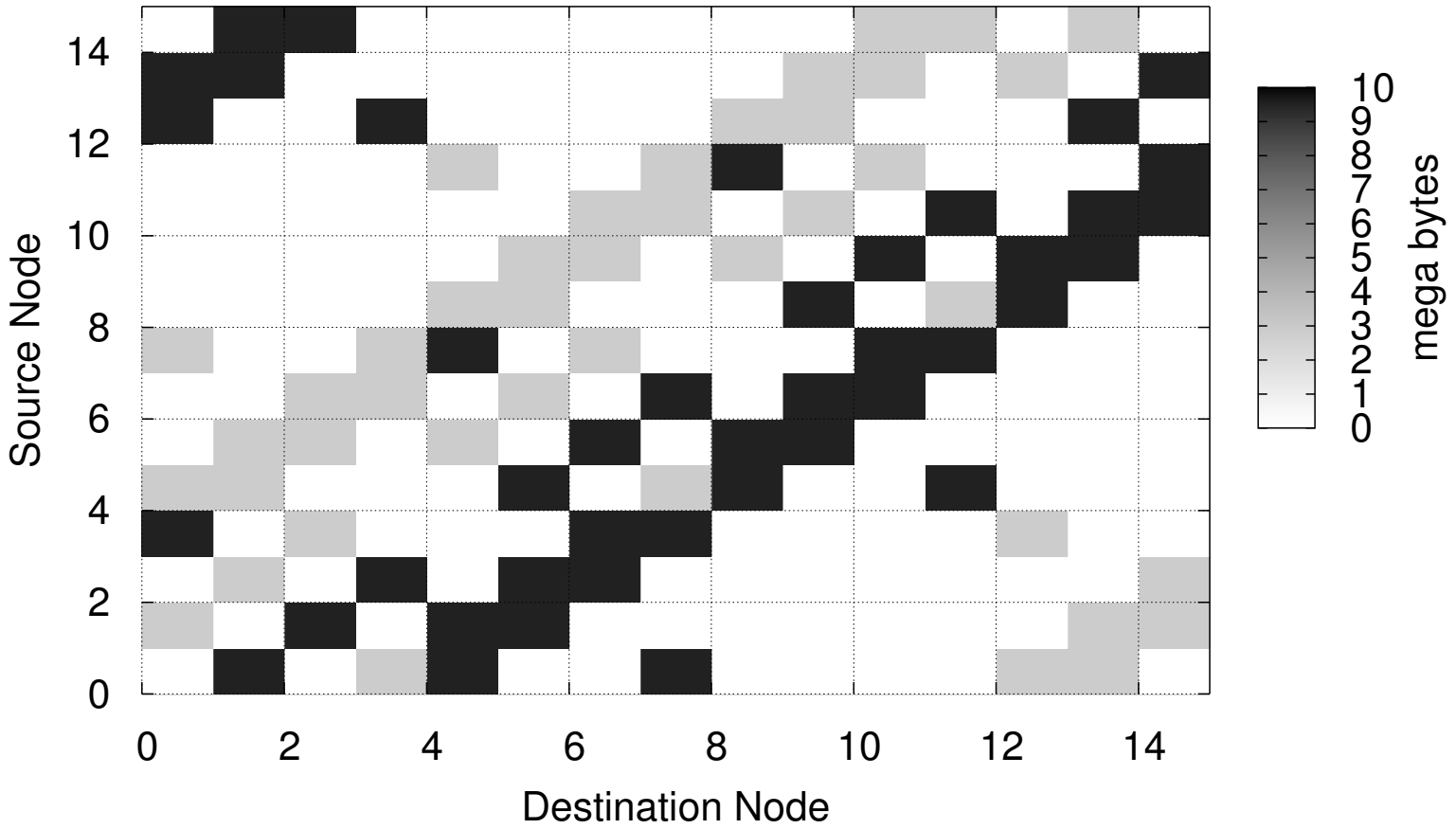
Communication Patterns

MG (class B) message density distribution



Communication Patterns

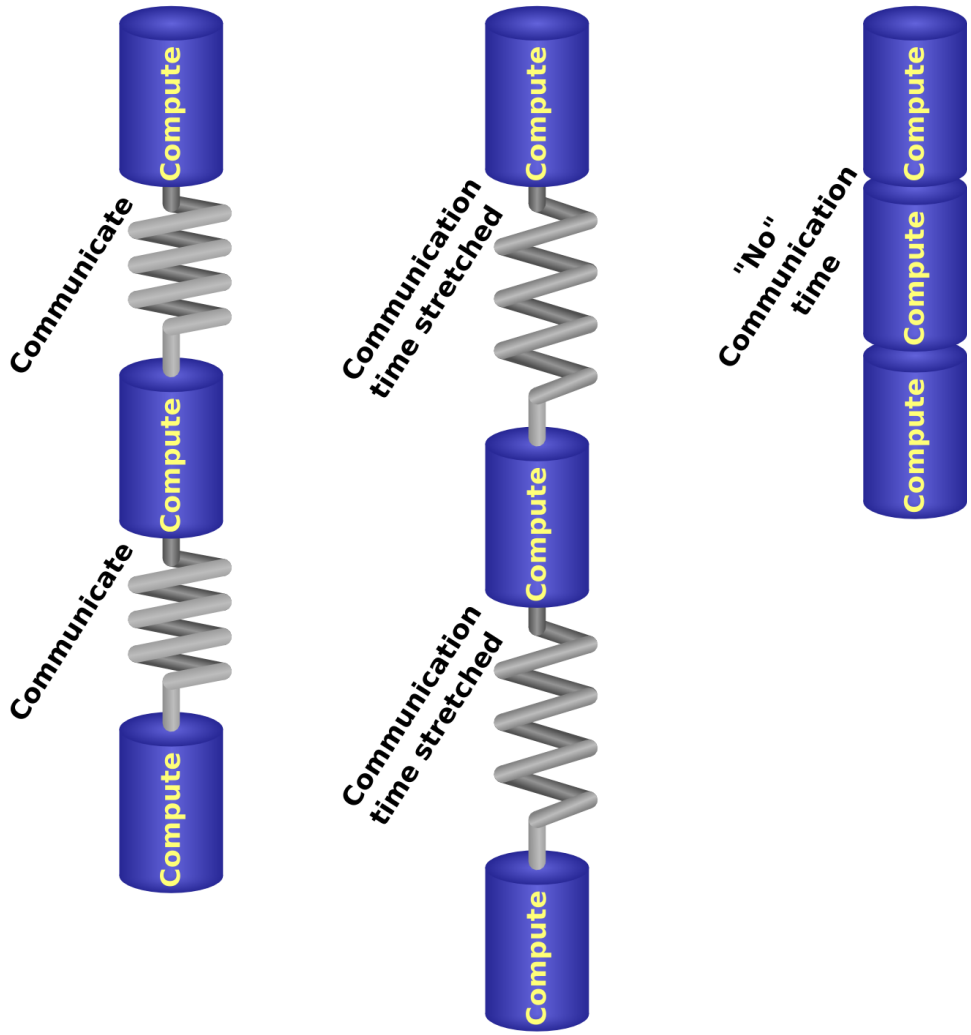
BT (class A) data density distribution



Varying Bandwidth and Latency

- Simulator can change bandwidth and latency independently
- This can be used to evaluate application performance under varying network characteristics
- $\rightarrow$ predict impact of new network

Varying Bandwidth and Latency



Zero-Cost Collectives

- Putting collectives into NIC, building specialized NIC, or optimizing them is interesting
- How much application performance can be gained is not clear
- Simulator can assign $\Delta = 0$ to collectives and leave point-to-point alone
- See talk on Thursday

Intrusion-Free MPI Traces

- So far gathered only limited amounts of data
- Simulator can gather, and save to disk, large amount of data
  - Without changing application virtual time

Section Outline

Comparison to Other Work

- This approach seems to be new
- Combines low-intrusion measurement research with discrete event simulation
- Needs more validation, but seems to be very accurate
- Opens up many different and simple ways of evaluating applications and research directions

Comparison to Other Work

- No instrumentation code inserted into app
  - Rename **`main()`** (**`program`**) only change to app
- No disturbance of (virtual) runtime of app
  - Independent of amount of data collected.
- No extra memory needed on compute nodes to store trace data
- Language independent (Fortran, Fortran 90 with MPI-2, and C)

Section Outline

Future Work

Continuing Work

- Need to incorporate more accurate network model
- This will allow simulation of congestion, and evaluation of topology choices, node allocation, etc.
- Move below MPI into NIC for more fine-grained simulation
- Incorporate non-network simulators; CPU and NIC sims

Section Outline

Summary

- Novel tool to collect MPI data
- Language independent
- Only linking with application needed
- Virtual runtime of application is not changed
- Lots of future possibilities

End

Questions?