Edinburgh Napier UNIVERSITY

A Comparison of MPI and CPA Networking Communication Performance

Kevin Chalmers

Centre for Information and Software Systems

Edinburgh Napier University

Edinburgh Napier UNIVERSITY

Breakdown

- Background
  - CPA Networking
- MPI and CPA Networking
- Experiments
- Future Work
  - New Network Layer
- Conclusions

Edinburgh Napier UNIVERSITY

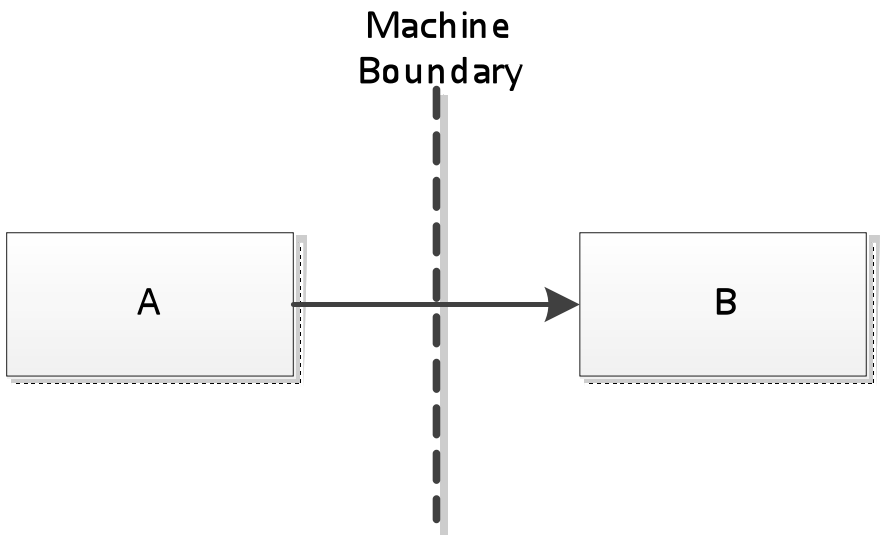
Motivation

- MPI is a standardised method of inter “process” communication in parallel computing applications
- Highly popular approach to developing parallel computing applications
- How well does CPA Networking compare to MPI for communication?
- I have been asked for a comparison for a couple of years now

Edinburgh Napier UNIVERSITY

Goal of CPA Networking

- Provide *inter-process communication* across a communication medium *in a transparent manner*
- No notion of high performance
  - Distributed channel enabling framework



Edinburgh Napier UNIVERSITY

Goal of MPI

- Provide a *high performance*, *scalable*, and *portable* inter-process communication mechanism for parallel computing applications

- Provides both point-to-point and collective communication mechanisms

- Commonly used for Single Program, Multiple Data applications

Edinburgh Napier UNIVERSITY

Comparison

- Both CPA Networking and MPI aim at *inter-process communication*
- MPI aims at HPC type applications
- CPA Networking aims at …?
  - Good question
  - Has been previous work in HPC applications
  - Essentially an enabling technology

Edinburgh Napier UNIVERSITY

CPA NETWORKING

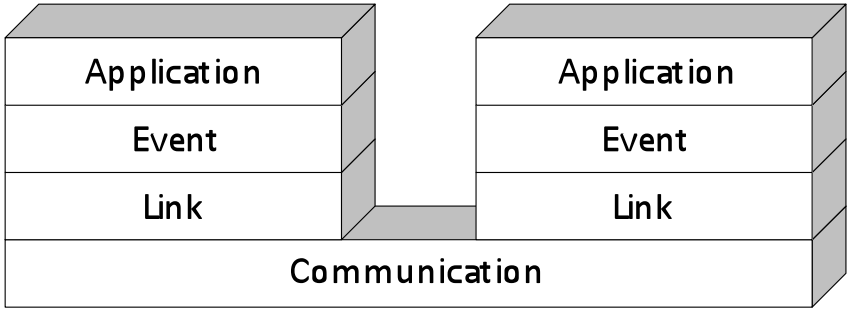
Edinburgh Napier UNIVERSITY

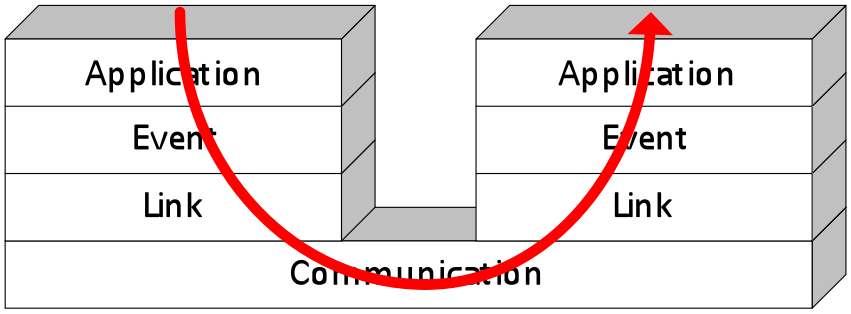
History of CPA Networking

- T9000 and Virtual Channel Processor
- JCSP.net
  - T9000 inspired
  - Highly integrated with Java and JCSP
- CPA Networking
  - Development of protocol
  - Lightly integrated with Java and JCSP
    - But still too much
  - Resource reduction

Edinburgh Napier UNIVERSITY

CPA Networking Functionality

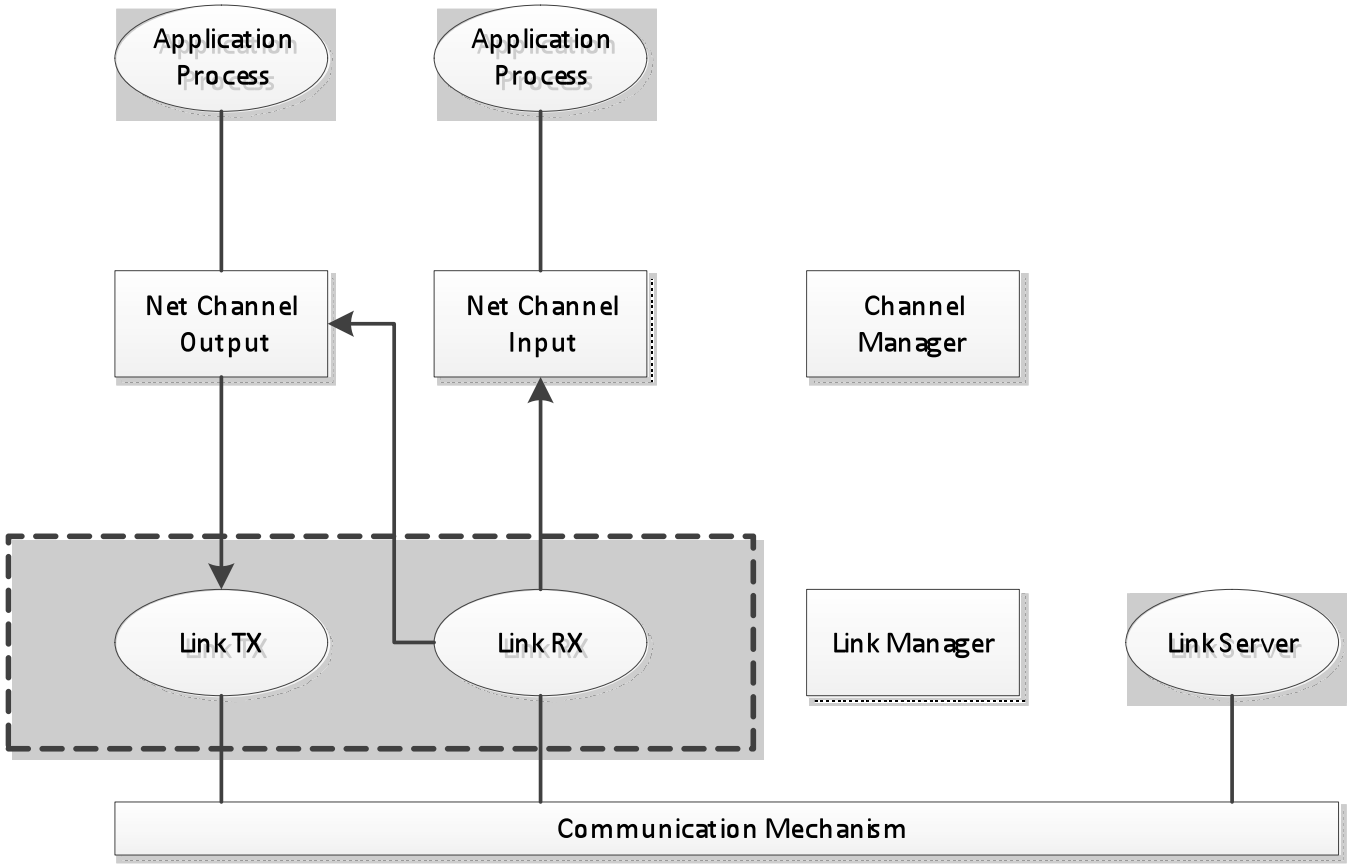




Virtual Channel

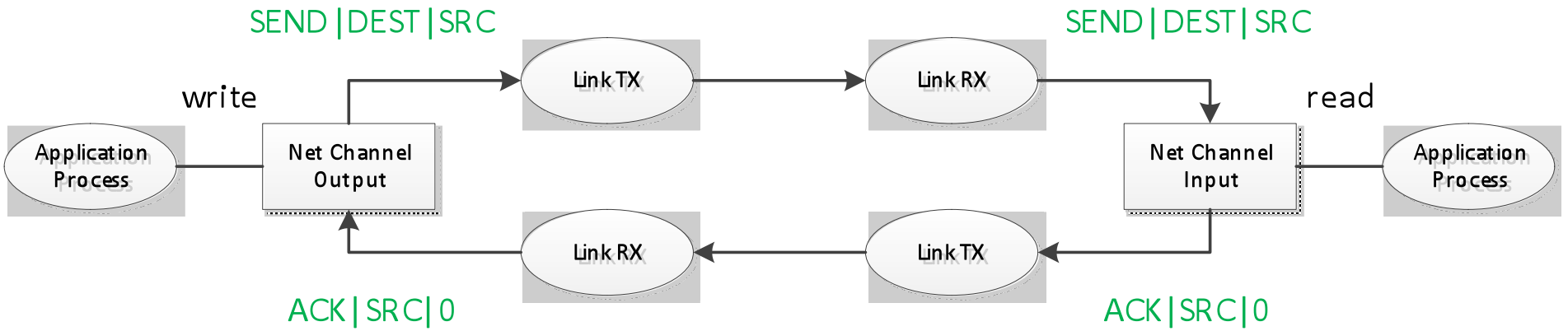
Edinburgh Napier UNIVERSITY

CPA Networking Architecture



Edinburgh Napier UNIVERSITY

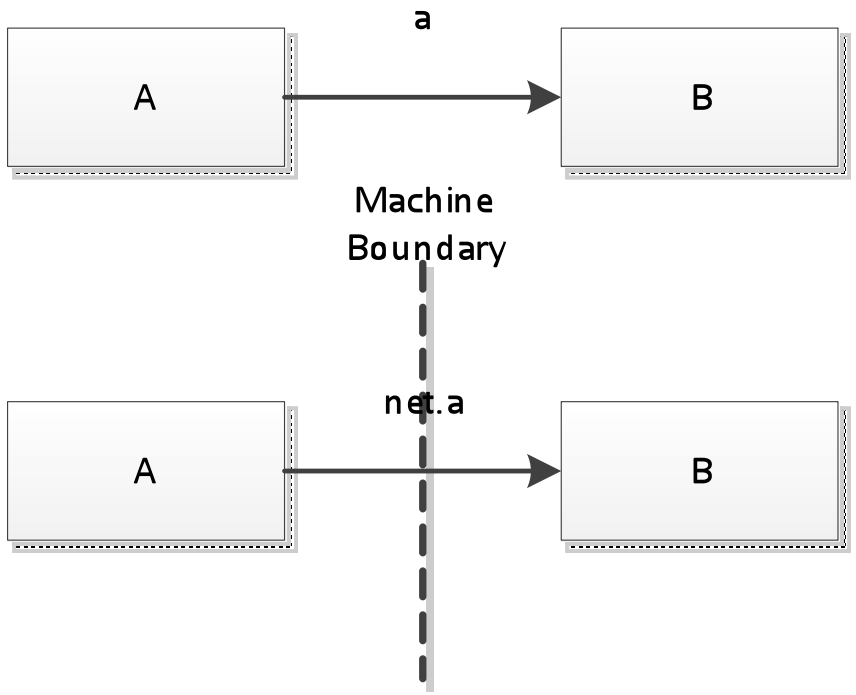
Channel Operation



- Protocol defines all messages as triples
  - TYPE | ATTR1 | ATTR2
  - Some messages have a data load
- Links process messages based on type and state of event primitive

Transparency of Distribution

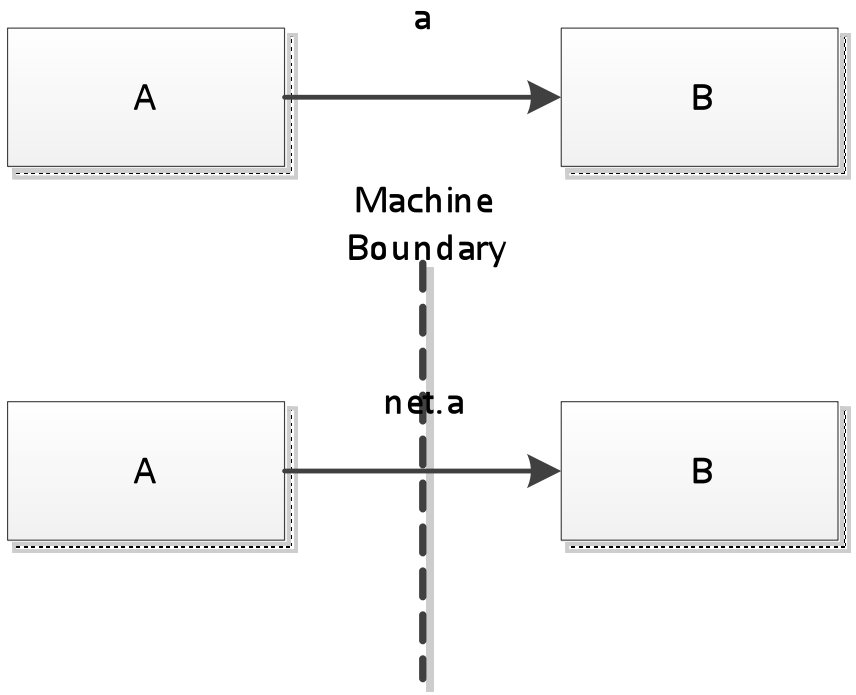
- Networked channels provide the *same interface and behaviour as standard channels*
  - A does not need to know if a is locally connected or remote connected
  - Couple of minor gotchas



Transparency of Distribution

- Powerful abstraction for distribution

- Most other approaches to distribution require you to know that you are distributed
  - For example, object-orientation aliasing is broken



Edinburgh Napier UNIVERSITY

Synchronous and Asynchronous in CPA Networking

- CPA Networking channels have asynchronous capabilities
  - Allow simpler client-server interactions
- An asynchronous communication means no ACK is sent
  - The sender completes once networked output communicates with the Link
- Networked channels are supported by infinite buffering to ensure deadlock freedom
  - Possible memory issues

Edinburgh Napier
UNIVERSITY

MPI

Edinburgh Napier UNIVERSITY

MPI Functionality

- MPI operates using a communicator mechanism
- Each process interacting with a communicator is assigned a rank
- Direct communication with a process can be achieved using the relevant rank

Edinburgh Napier UNIVERSITY

MPI Functionality

- Initially, each process belongs to the WORLD communicator
- Sub-groups of processes can create specific communicators
- Although communicators can be used to communicate with local threads, MPI is usually considered an inter-process communication mechanism
  - It is designed to cross the machine boundary

Edinburgh Napier UNIVERSITY

MPI Operations

- Some similar to CPA Networking
  - Send
  - Receive

- Some implementable in CPA Networking
  - Broadcast
  - Scatter
  - Gather

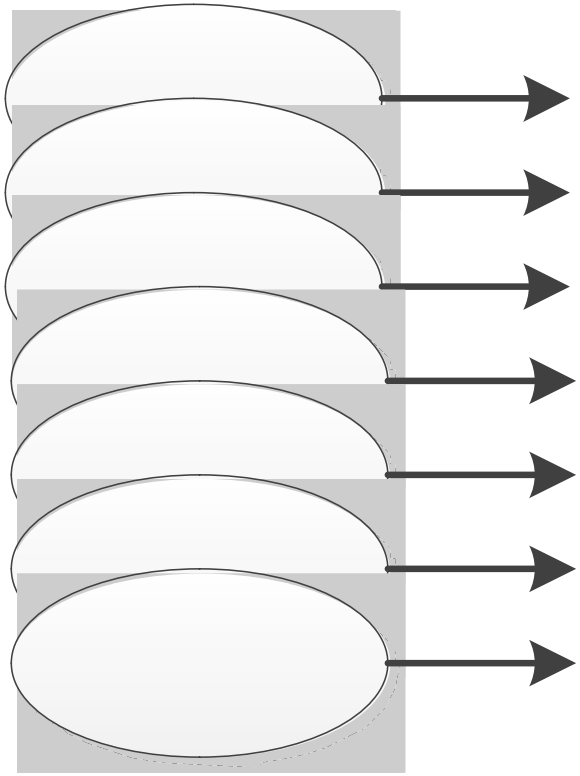
Edinburgh Napier UNIVERSITY

MPI AND CPA NETWORKING OPERATIONS

Edinburgh Napier UNIVERSITY

MPI in CPA Networking

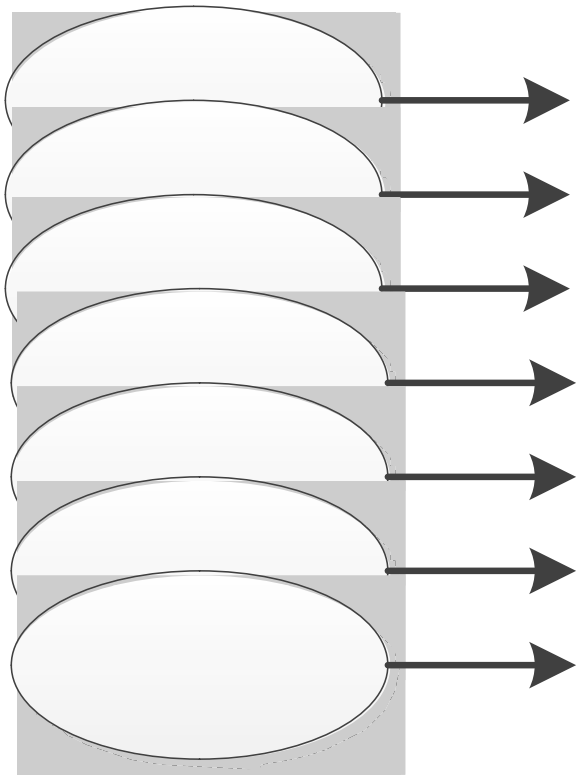
- Broadcast in MPI allows one process to send a message to all others in a communicator
- Easily simulated using a standard parallel write in CPA Networking



Edinburgh Napier UNIVERSITY

MPI in CPA Networking

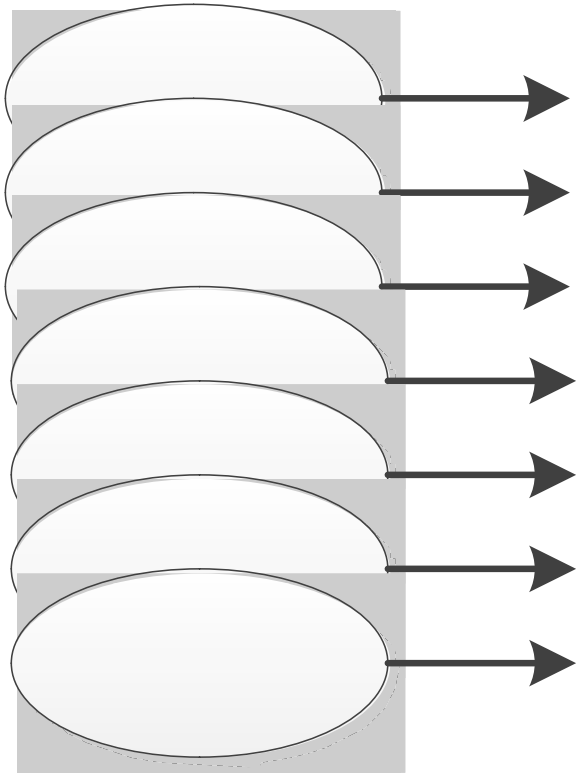
- Problem is, we create many processes to achieve this
  - In JCSP and CSP for .NET this is bad
- Would have to add a barrier communication to ensure group synchronisation



Edinburgh Napier UNIVERSITY

MPI in CPA Networking

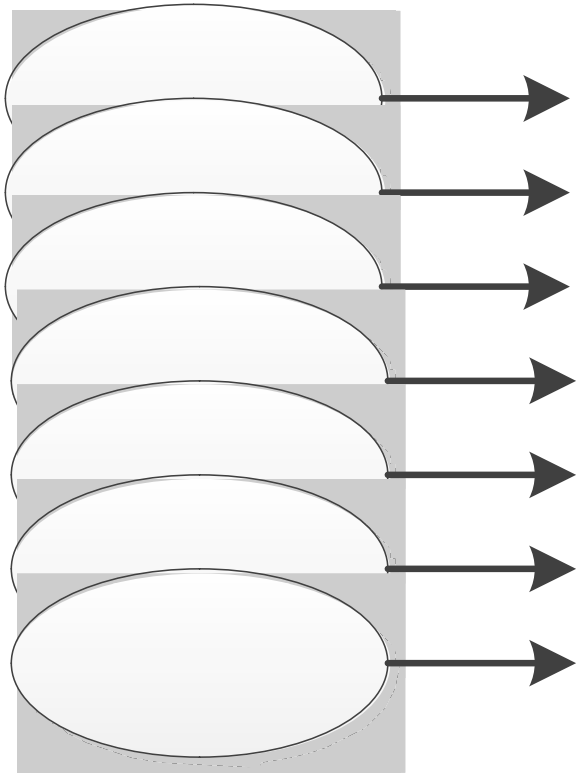
- Scatter-Gather allows a single process to send an array of messages to other processes in the group, and wait for the reply



Edinburgh Napier UNIVERSITY

MPI in CPA Networking

- Scattering can be achieved using standard parallel writes
- Gathering can be achieved using parallel reads
  - Again an extra overhead



Edinburgh Napier UNIVERSITY

Choice

- CPA Networking allows input channels to be used as guards

- They operate in the same manner as standard channel input guards

```
Alt a = new Alt(inputs);
int index = alt.Select();
data = inputs[index].Read();
```

Edinburgh Napier UNIVERSITY

Choice

- MPI does not provide the same choice mechanism
  - Cannot mix timers, input, skip, etc.

- Selection of input from a group of processes can be achieved using the probe command

```
Status status = comm.Probe(Communicator.anySource, 1);
data = comm.Receive<Data>(status.Source, 1);
```

Edinburgh Napier UNIVERSITY

CPA Networking and MPI Operations

- MPI and CPA Networking share the same general communication mechanisms
  - Send, Receive
- MPI provides collective communication mechanisms implementable in CPA Networking
  - Broadcast, Scatter-gather
- CPA Networking provides choice, and this is possible in MPI using the probe command

Edinburgh Napier UNIVERSITY

EXPERIMENTAL RESULTS

Edinburgh Napier UNIVERSITY

Approach

- Two different areas evaluated

- Base network performance
  - Latency and throughput
  - Broadcast

- Communication stress
  - Scatter-gather, request-response

Edinburgh Napier UNIVERSITY

Monte-Carlo Pi

- Monte-Carlo Pi was used as the work packet for stress
  - Allows work size to be scaled
  - Small communication size
  - Not looking for parallel speedup

```
IN: NUM_ITERATIONS
COUNT := 0
FOR i in 0 to NUM_ITERATIONS - 1
    X := random 0.0 to 1.0
    Y := random 0.0 to 1.0
    DIST := √(X * X + Y * Y)
    IF DIST <= 1.0
        COUNT := COUNT + 1
OUT: 4.0 * (COUNT / NUM_ITERATIONS)
```

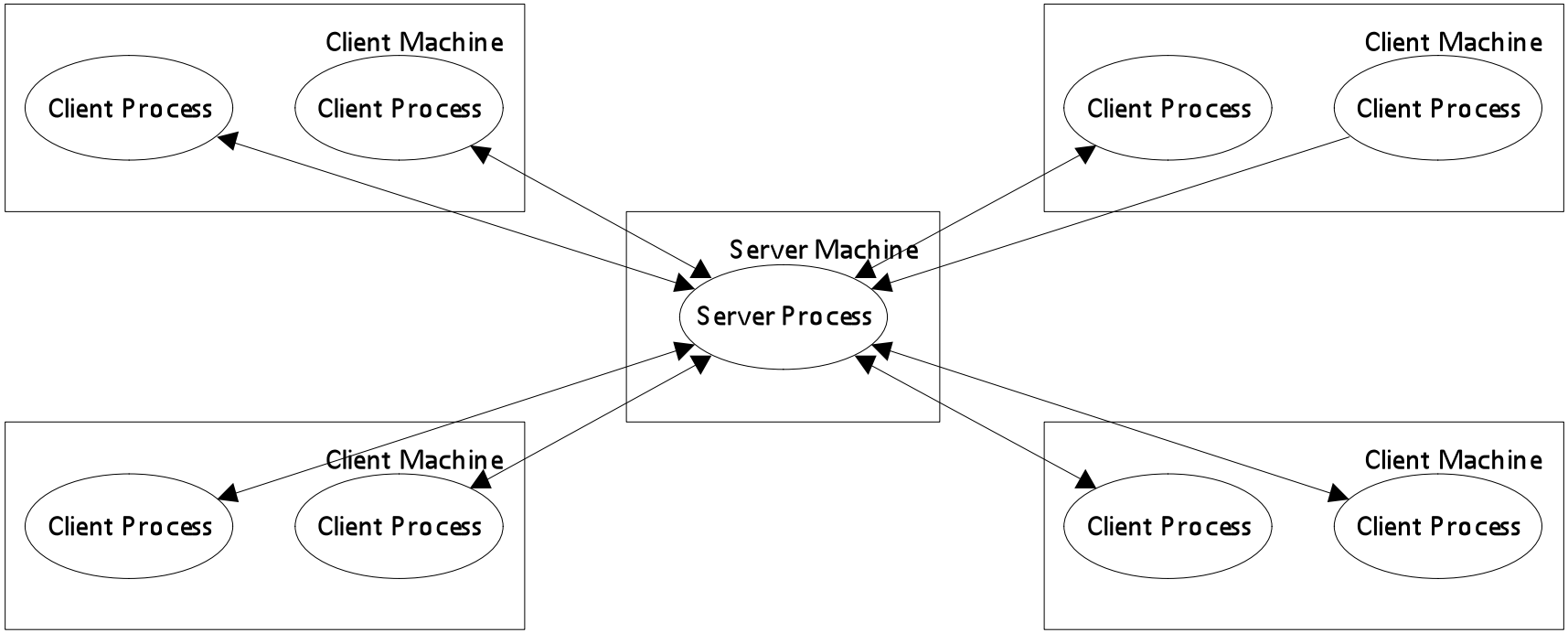
Edinburgh Napier UNIVERSITY

Platform

- Simple set up
  - Intel Core Due E8400 3.0 GHz (no HT)
  - 2 GB RAM
  - CSP for .NET versus MPI .NET
- Small Ethernet network, 100 Mbit/s
- Microsoft MPI via HPC SDK

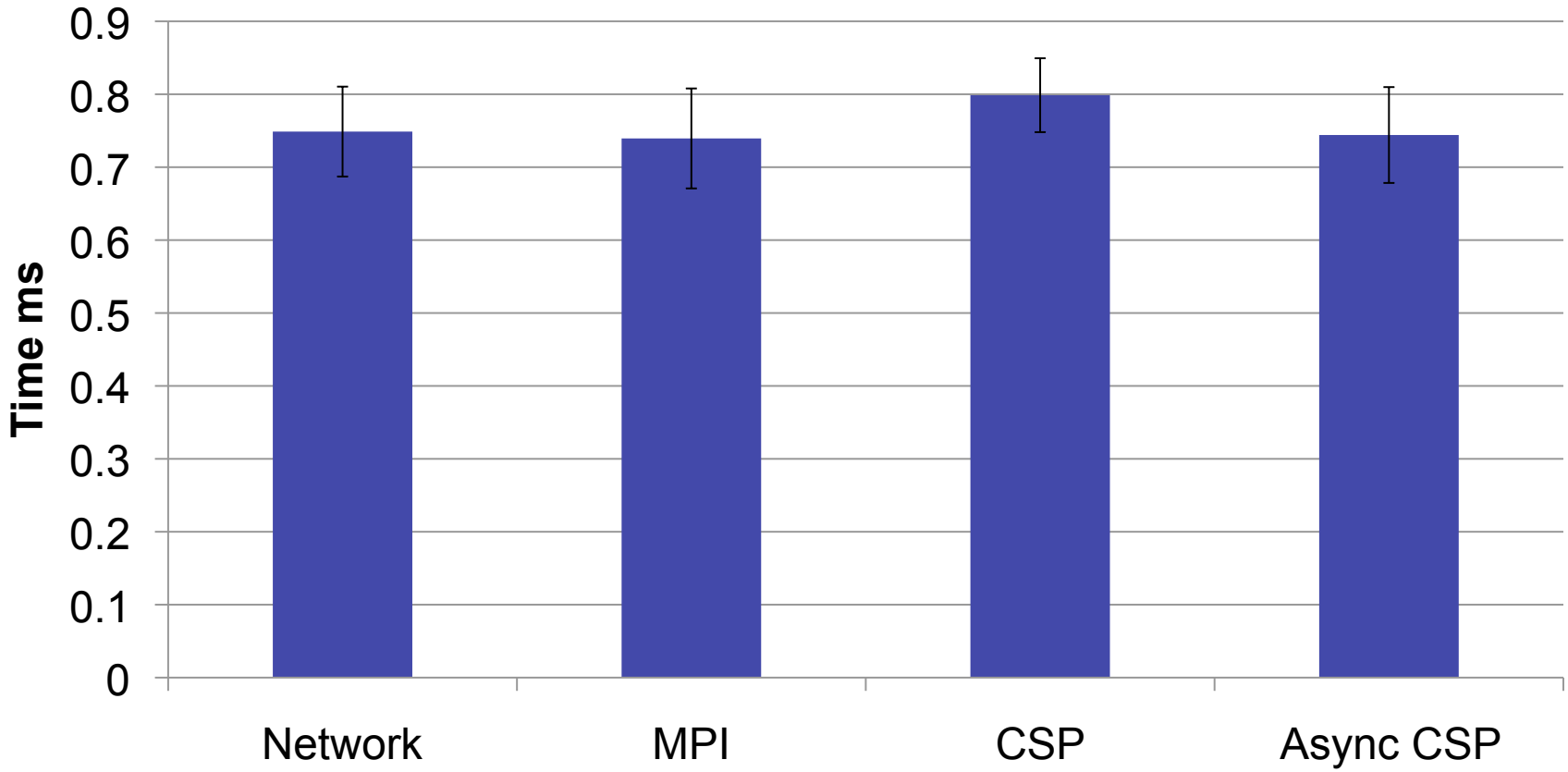
Edinburgh Napier UNIVERSITY

Machine Organisation



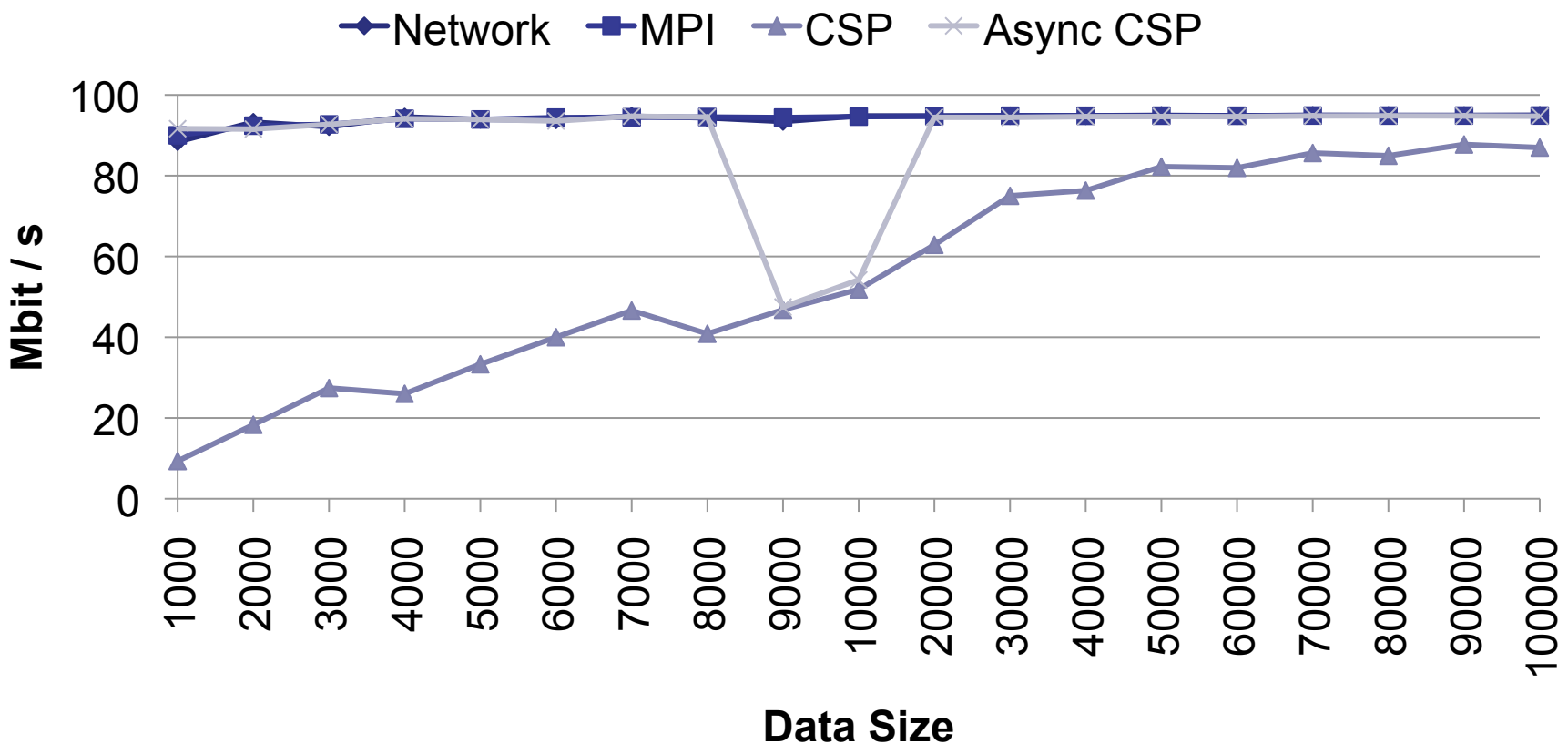
Edinburgh Napier UNIVERSITY

Ping-Pong Time



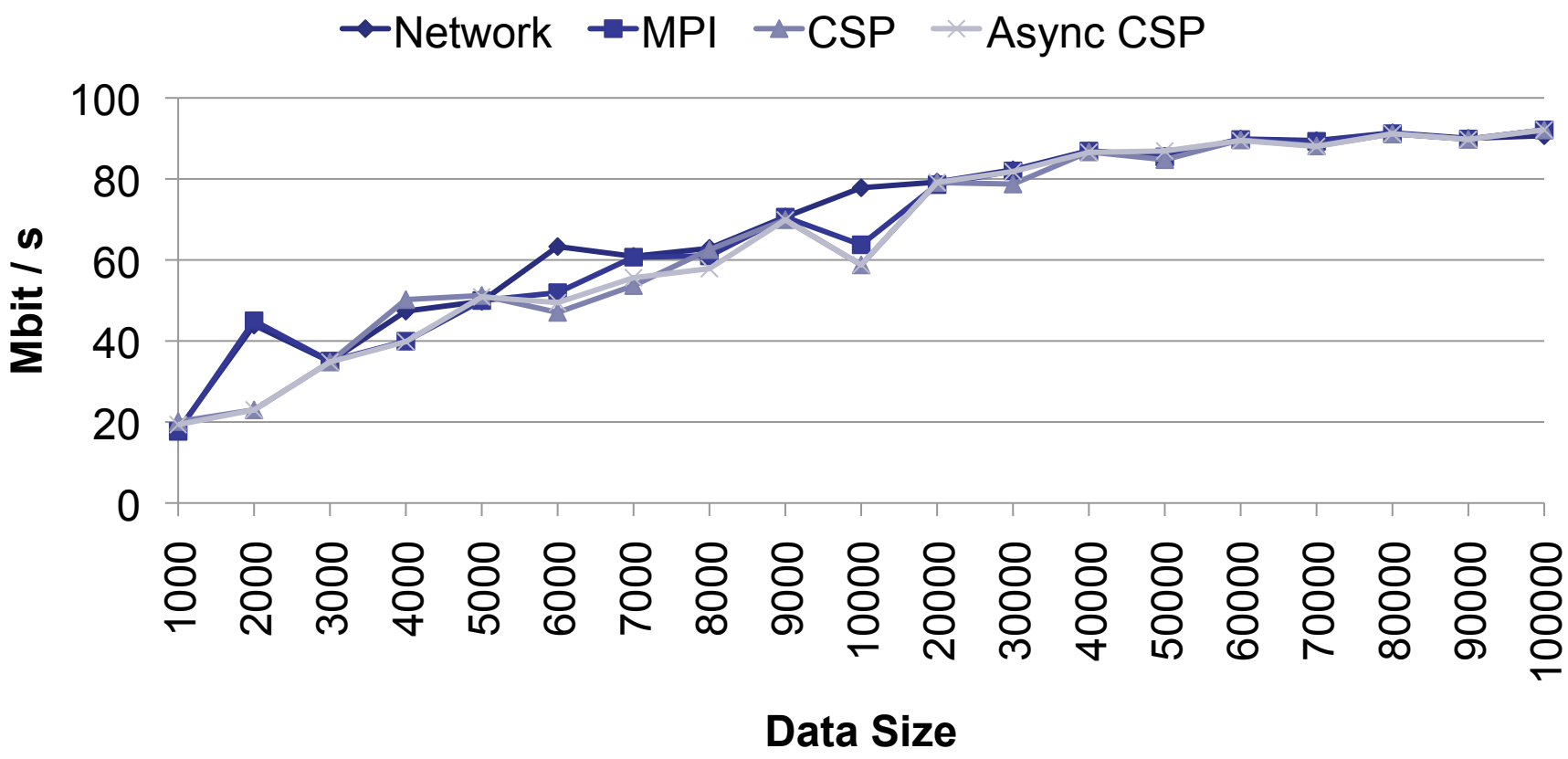
Edinburgh Napier UNIVERSITY

Throughput Point-to-Point



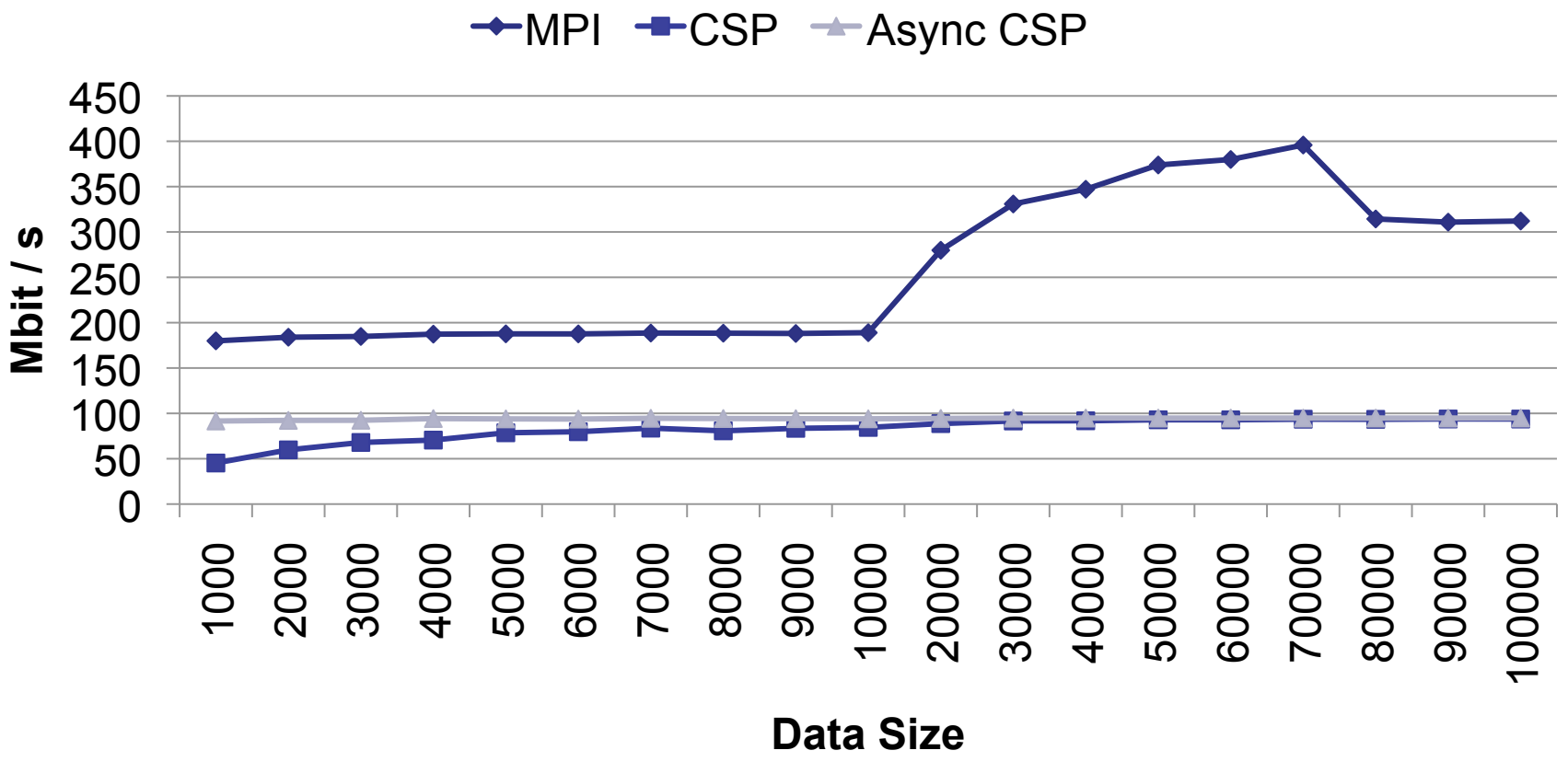
Edinburgh Napier UNIVERSITY

Throughput Ping-Pong



Edinburgh Napier UNIVERSITY

Throughput Broadcast



Edinburgh Napier UNIVERSITY

Stressed Communication

- Optimal Time

$$\frac{computation\ time + communication\ time}{number\ of\ processes}$$

- Sub-Optimal

$$\frac{computation\ time}{number\ of\ processes} + communication\ time$$

Edinburgh Napier UNIVERSITY

Stressed Communication

- Communication time
  - Each communication mechanism had an approximate 0.75ms ping-pong time

$$0.75ms \times number\ of\ packets$$

- Computation time
  - Machine can perform ~4.85 million Monte Carlo Pi iterations per second
  - Perform 1×10^9 iterations

$$computation\ time = \frac{\left(\frac{1 \times 10^9}{4.85 \times 10^6}s\right)}{8} = 25773ms$$

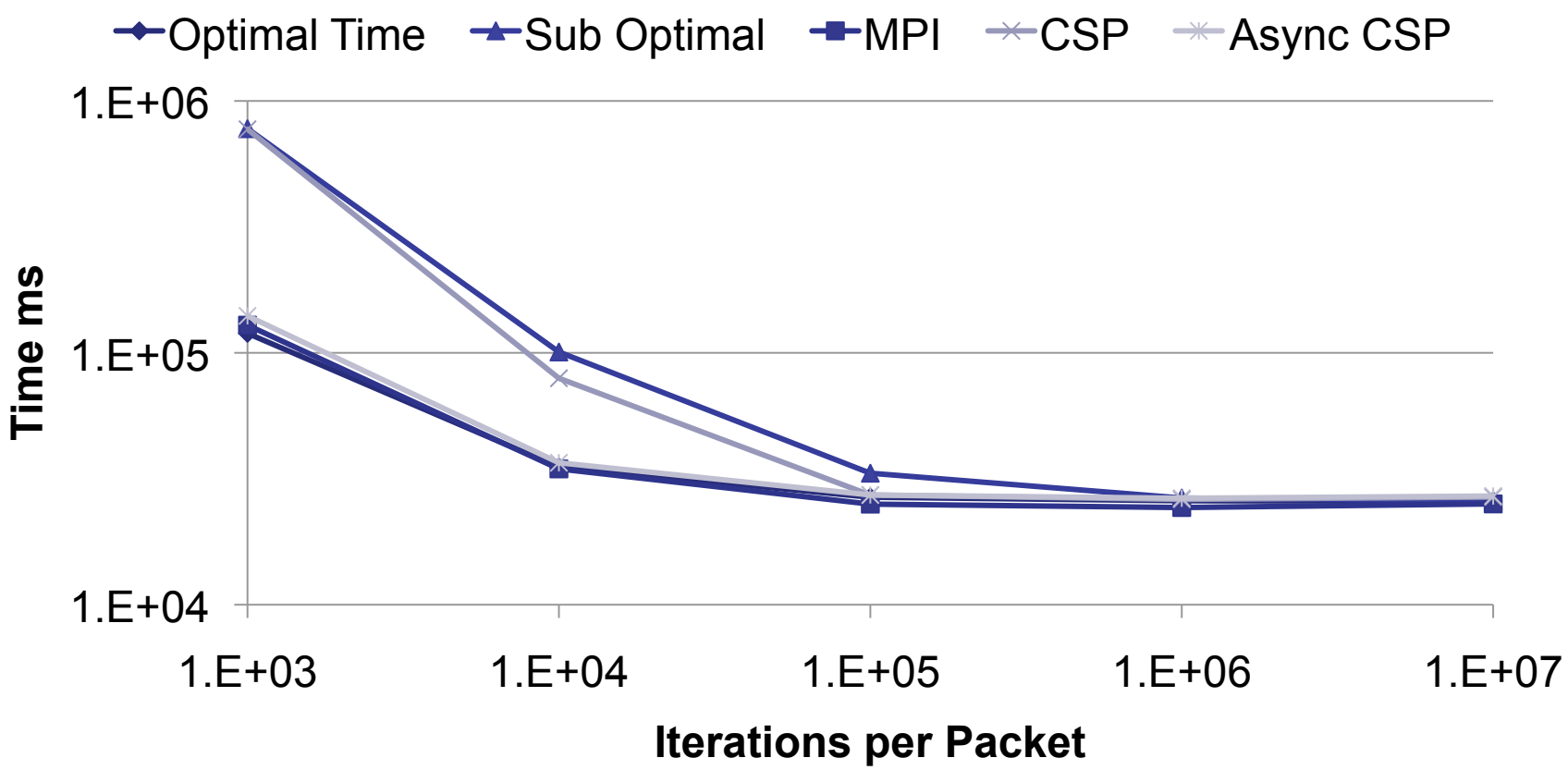
Edinburgh Napier UNIVERSITY

Approximate Optimal and Sub-Optimal Times

Iterations Per Packet	Num Packets	Comm Time	Comp Time	Optimal	Sub-Optimal
1×10^3	1×10^6	750000	25773	119523	775773
1×10^4	1×10^5	75000	25773	35148	100773
1×10^5	1×10^4	7500	25773	26711	33273
1×10^6	1×10^3	750	25773	25867	26523
1×10^7	1×10^2	75	25773	25782	25848

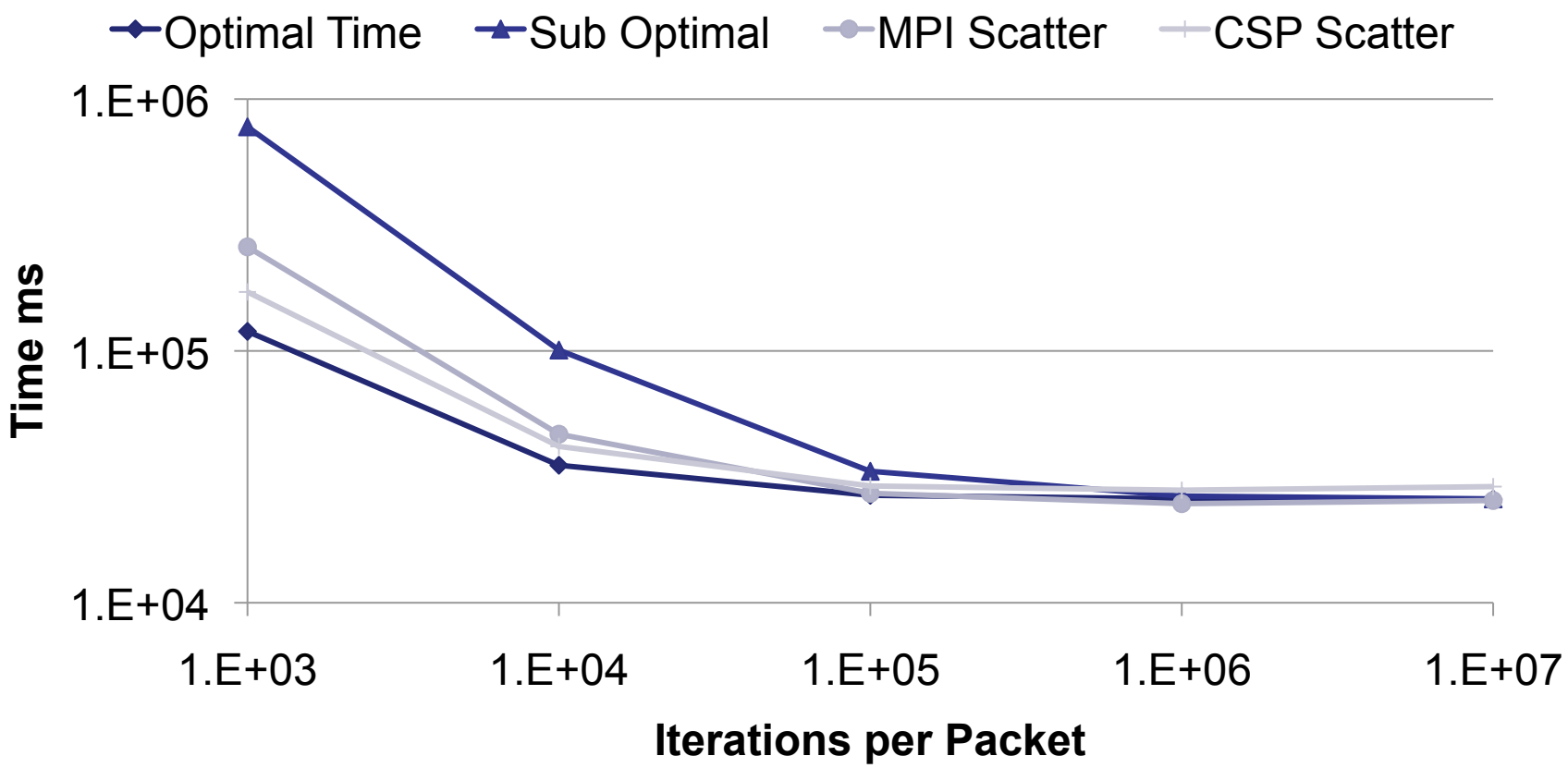
Edinburgh Napier UNIVERSITY

Monte-Carlo Pi Request-Respond



Edinburgh Napier UNIVERSITY

Monte-Carlo Pi Scatter-Gather



Edinburgh Napier UNIVERSITY

CONCLUSIONS AND FUTURE WORK

Edinburgh Napier UNIVERSITY

Quick Summary

- So MPI and CPA Networking provide no great difference in communication performance
  - You could probably optimise to a particular scenario
  - Different scenarios might favour one over the other

- So why do we have CPA Networking? Why don't we just use MPI and be done with it?
  - This had me thinking a bit

Edinburgh Napier UNIVERSITY

Advantages of CPA Networking

- It provides distributed channel semantics, transparently to the application programmer
  - And hopefully in a cross-platform manner
- …
- We have mobility?
  - But I could never work out a good purpose, or a reasonable approach to achieve channel mobility

Edinburgh Napier UNIVERSITY

Limitations of CPA Networking

- It is still limited in the platforms it supports
  - Actually only JCSP and CSP for .NET
  - Tried others – will discuss next
- It has a protocol definition that was developed to support JCSP style concurrency
- It is still closely coupled with the network layer
  - Expects stream connections internally

Edinburgh Napier UNIVERSITY

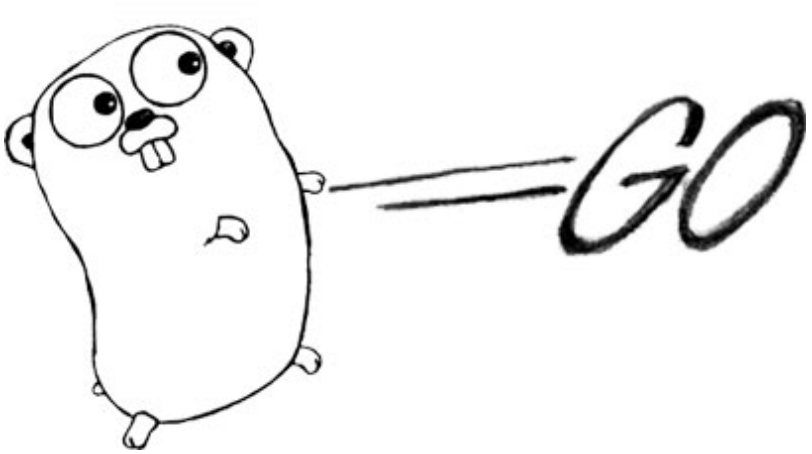
Integration of CPA Networking into a Library

- CPA Networking is still tightly coupled within a library
  - JCSP, CSP for .NET
- It relies on extending functionality of an existing framework
- This has led to problems in implementation on other platforms / frameworks
  - Tried occam-π
  - Tried C++CSP

Edinburgh Napier UNIVERSITY

Integration of CPA Networking into a Library

- What about other languages that support a CPA style?
  - Google Go
  - Erlang
  - etc.



ERLANG

Edinburgh Napier UNIVERSITY

New Network Layer

- We need a new network layer
- We need a better network layer
- We need a network layer that is decoupled from the library / language that wishes to use it
- We have a protocol and existing verified architecture, we just need to adapt it for general purpose

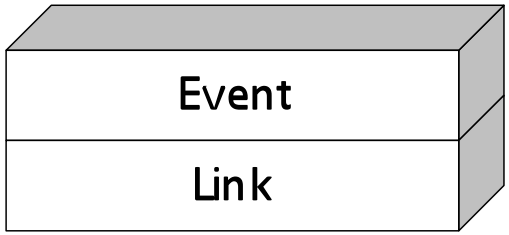
Edinburgh Napier UNIVERSITY

New Network Layer

No concept of sharing

Management

Simple commands



Management

Simple commands

Write your own
communication wrapper

Edinburgh Napier UNIVERSITY

New Network Layer

- Write it in something low level
- Don't rely on channels internally?
  - All we really have is unbounded queues – there is no requirement of choice in the architecture
- Can hook in existing communication layers
  - TCP/IP
  - MPI

Edinburgh Napier UNIVERSITY

Networked Mobile Channels

- Considering using MPI as a base layer has made me decide on a model to support channel mobility

- Use mailboxes to store messages, the receiver requests the next message when it is ready
  - It can only be ready when it is not mobile

- All communicating applications will belong to the same group, thus allowing simple access to the mailbox

Edinburgh Napier UNIVERSITY

Mobile Processes

- We have been able to write distributed mobile processes for a long time in JCSP
  - About 2005

- Still the only framework that can do this (as far as I know)
  - Code mobility system
  - I know a bit too much about Java class loading than is probably healthy

Component Model for Mobility

Edinburgh Napier UNIVERSITY

Traditional Model

- Code
  - Code that describes the mobile component
- State
  - Active state – program counter, etc.
  - Passive state – data attributes of the component

CPA Model

- Type
  - Name of the type
  - Code (if required for strong mobility)
- State
  - Connection state (required for strong mobility)
  - Data – attributes of the component
  - Behaviour – active state of the component (required for strong mobility)

Edinburgh Napier UNIVERSITY

Transparency of Mobility

- Really, we want to have transparency of mobility
  - Send a channel across a network or local channel
  - Send a process across a network or local channel

- We do have most of the requirements met in the current version of JCSP
  - Local to distributed channel mobility is the hard part
  - Protocol driven on the network layer

Edinburgh Napier UNIVERSITY

Mobile Agent Framework in CPA Networking and MPI

- We actually have the technology to develop a robust mobile agent framework that can
  - Either allow known components to migrate between frameworks, maintaining connection state
  - Or strong mobility with dynamic code loading on a single framework
- Using MPI as a base communication layer would make this fairly trivial to use, once the pieces are in place
- The question is, does anyone want such a system?

Edinburgh Napier UNIVERSITY

Conclusions

- MPI and CPA Networking, although aimed at different audiences, provide similar performance for communication
- We can simulate many of the different operations in either approach
  - Although performance may be an issue
- A revaluation of CPA Networking is probably required to allow more general usage