PERFORMANCE OF THE DISTRIBUTED CPA PROTOCOL AND ARCHITECTURE ON TRADITIONAL NETWORKS

Kevin Chalmers
Institute of Informatics and Digital Innovation
Edinburgh Napier University

Breakdown

- Background
  - And why we haven’t got occam-π networking working yet
- Network performance
  - Latency
  - Throughput
- Mandelbrot performance
- Conclusion and future work

What I hoped to be talking about today…

- occam-π talking to JCSP talking to PyCSP

- This is possible
  - occam-π version very unstable
  - occam-π version very inefficient

- Something interesting using this setup on an HPC
  - JCSP is good for user interfaces
  - PyCSP good for scripting
  - occam-π good for heavy lifting

Background

- On-going work on a unified protocol and architecture for CPA based distributed computing
  - Once I have this, I can move back to getting mobility built into the protocol
- JCSP Net 2.0 package has been around for a few years now
  - 2008
- Previously we have only looked at mobile device communication using JCSP Net 2.0
- Upgrade to CSP for .NET 2.0

Problem with occam

- Networking architecture relies on a number of dynamically sizing lookup tables internally
  - Channel lookup table
  - Barrier lookup table
  - Link lookup table
- Channels and barriers are created with an indexing value in the range 0 to 2^{32}-1
  - This can be defined by the application programmer
- occam currently doesn't allow complex data structures easily
  - Going into native code an option

Tests Performed

- We are looking at general network performance using the CPA architecture
  - Network latency
  - Network throughput (unidirectional and bidirectional)
- Baseline network, CSP Sync and CSP Async gathered

- We are also going to do a naïve (non-optimised) distributed Mandelbrot

- Results gathered using both JCSP and CSP for .NET 2.0

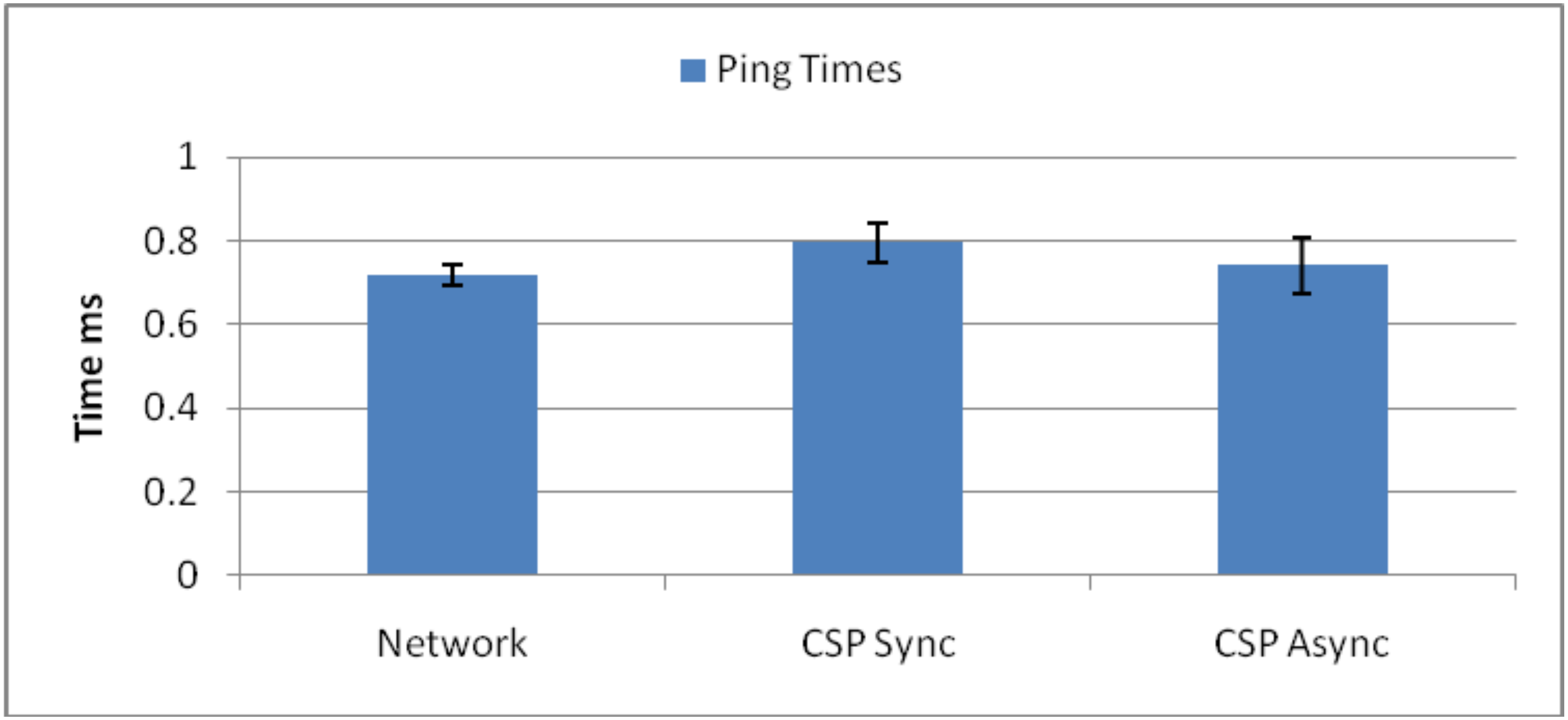
Experimental Framework

- Experiments were performed in a standard computing lab

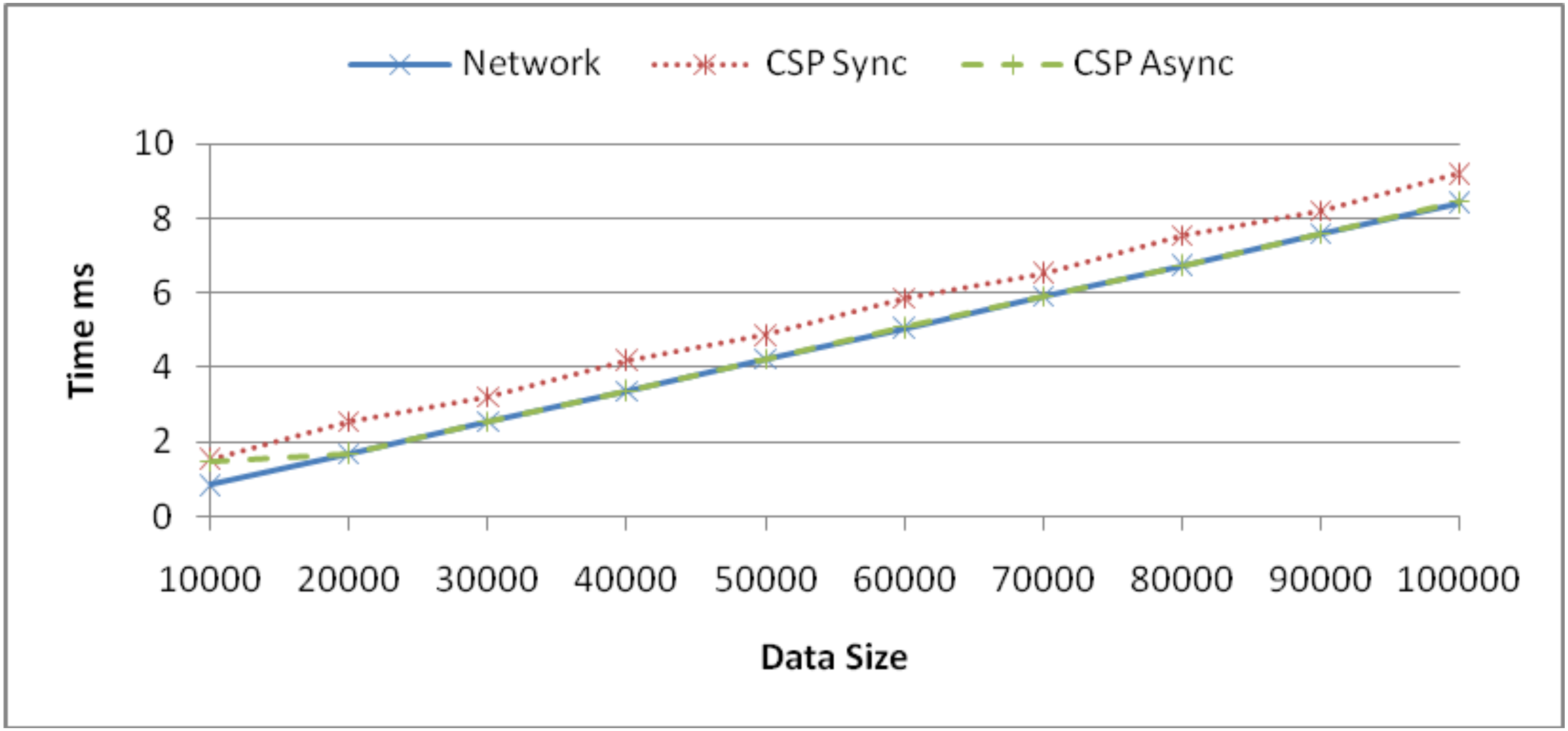
- Machines specs
  - Intel Core Duo E8400 3.0 GHz (no hyper-threading)
  - 2 GB RAM
  - Windows 7 32-bit
  - .NET 3.5, Java 6

- Network
  - 100 Mbps switched Ethernet

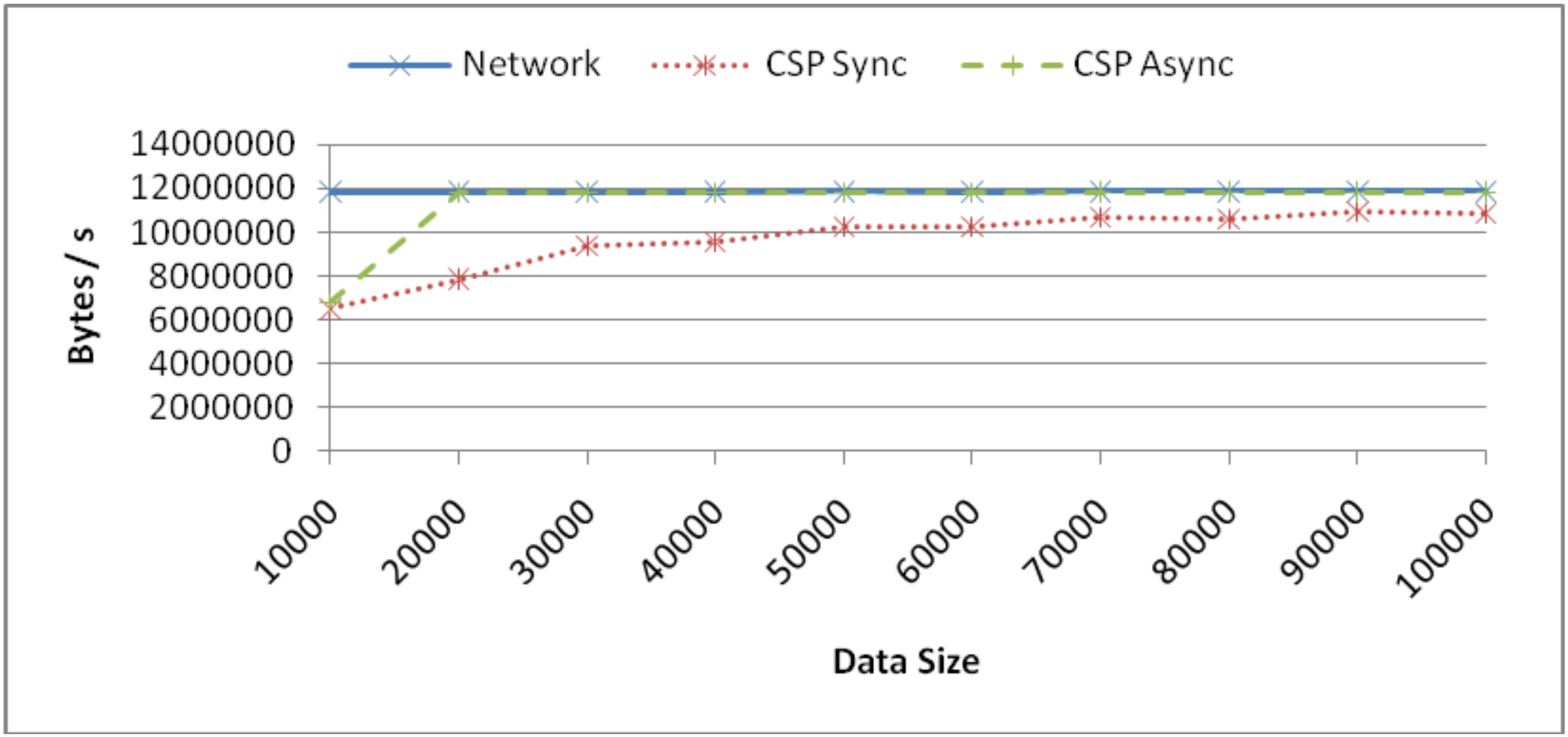
Ping Times



Sending Times

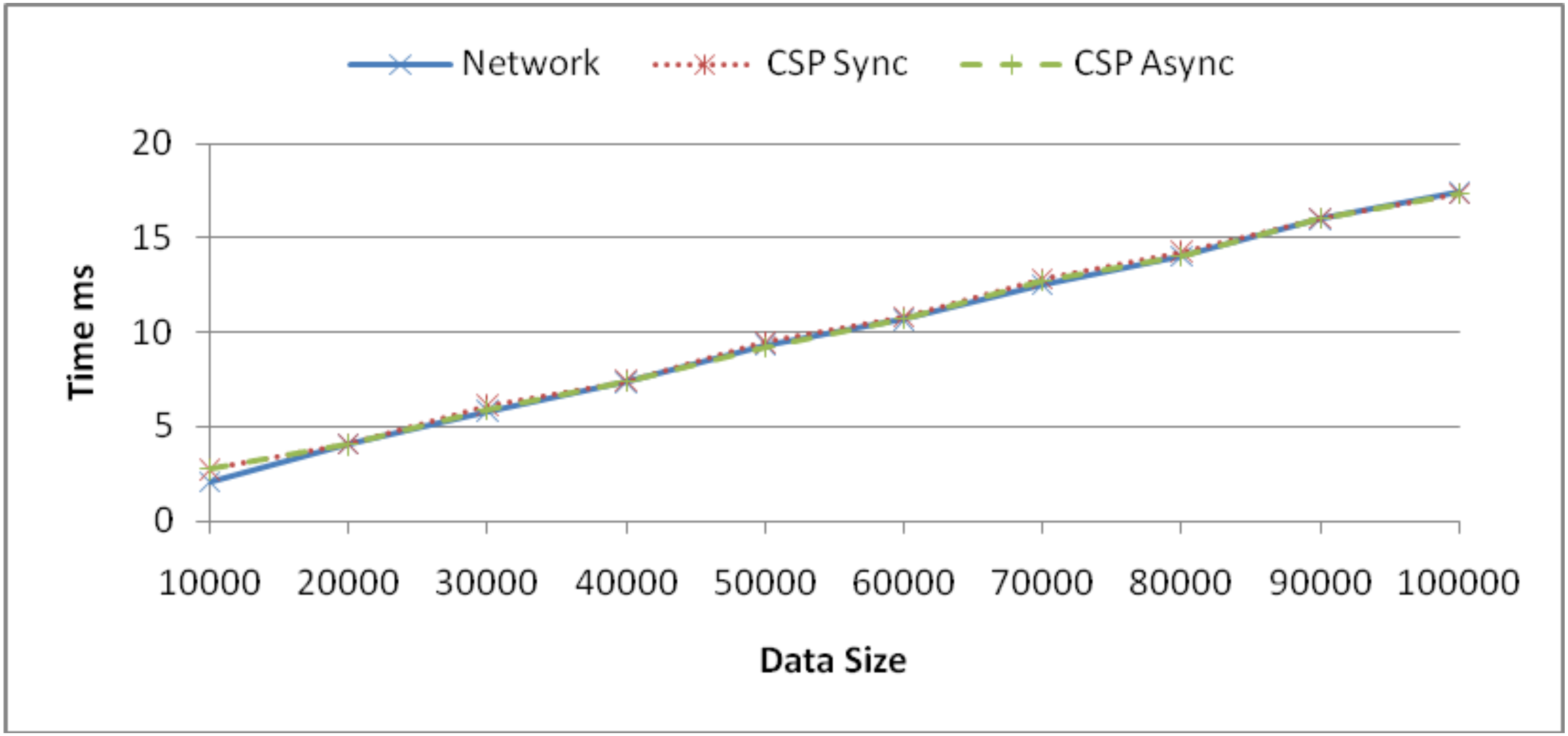


Throughput

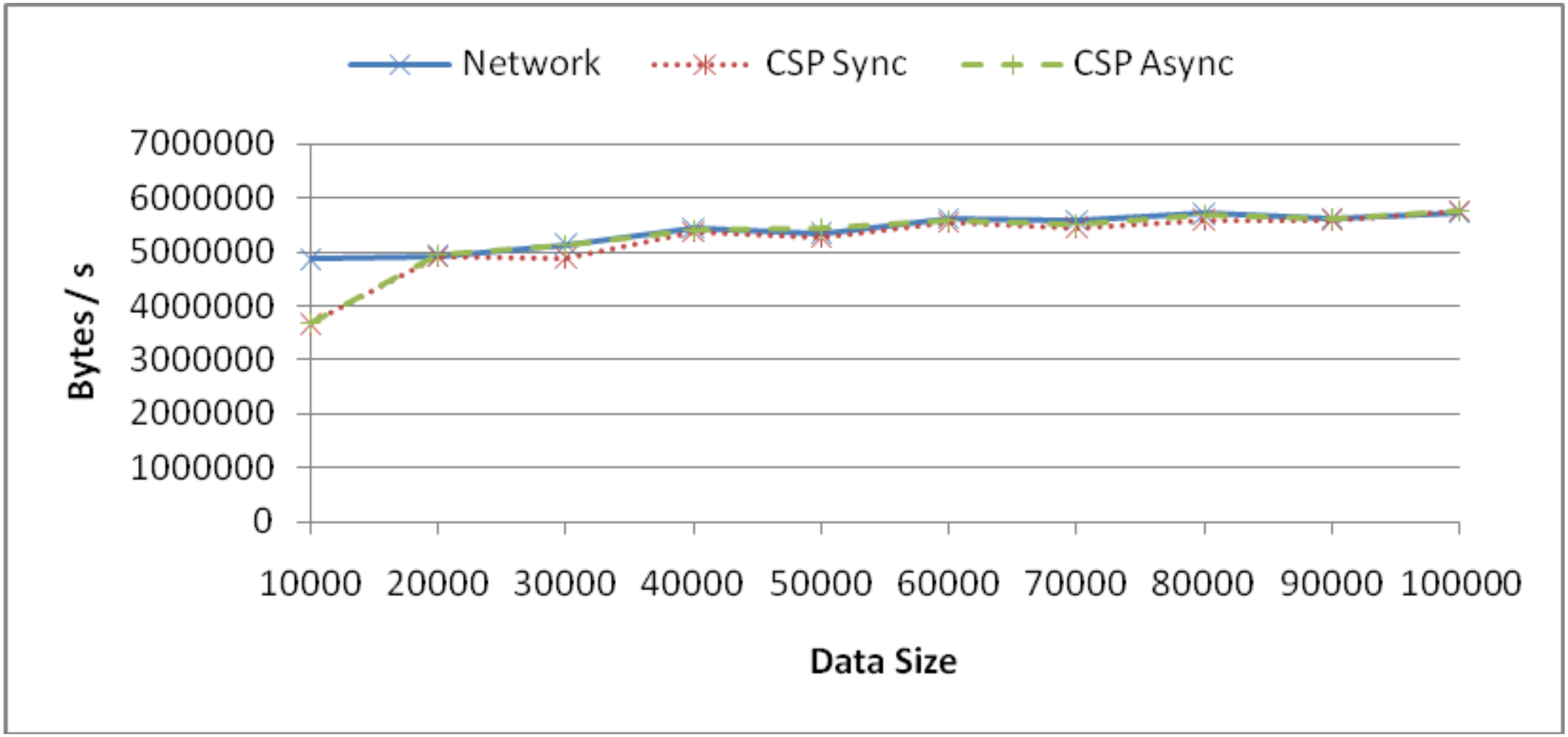


Network
CSP Sync
CSP Async
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
Bytes / s
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
Data Size

Send-Receive Times



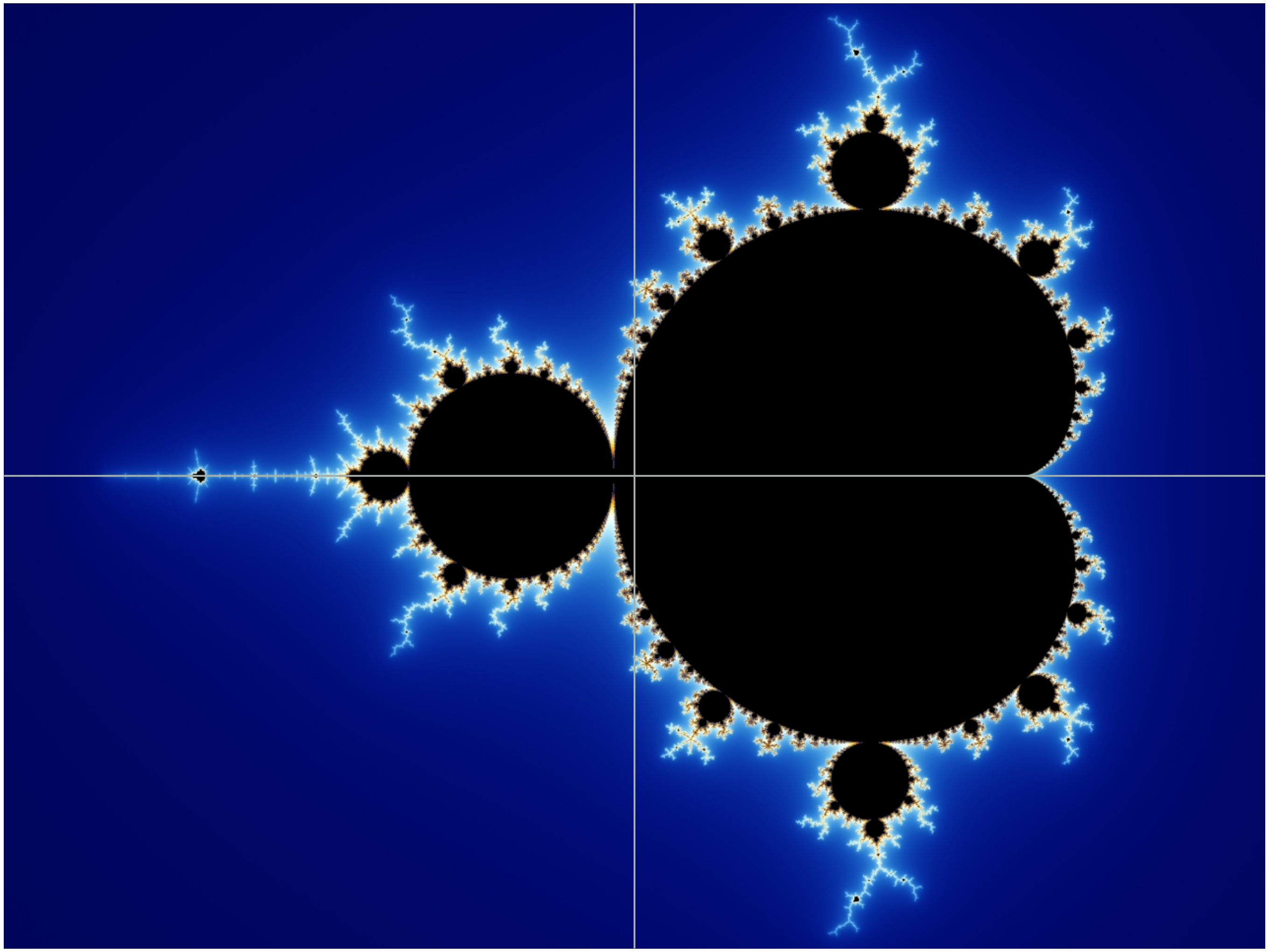
Send-Receive Throughput



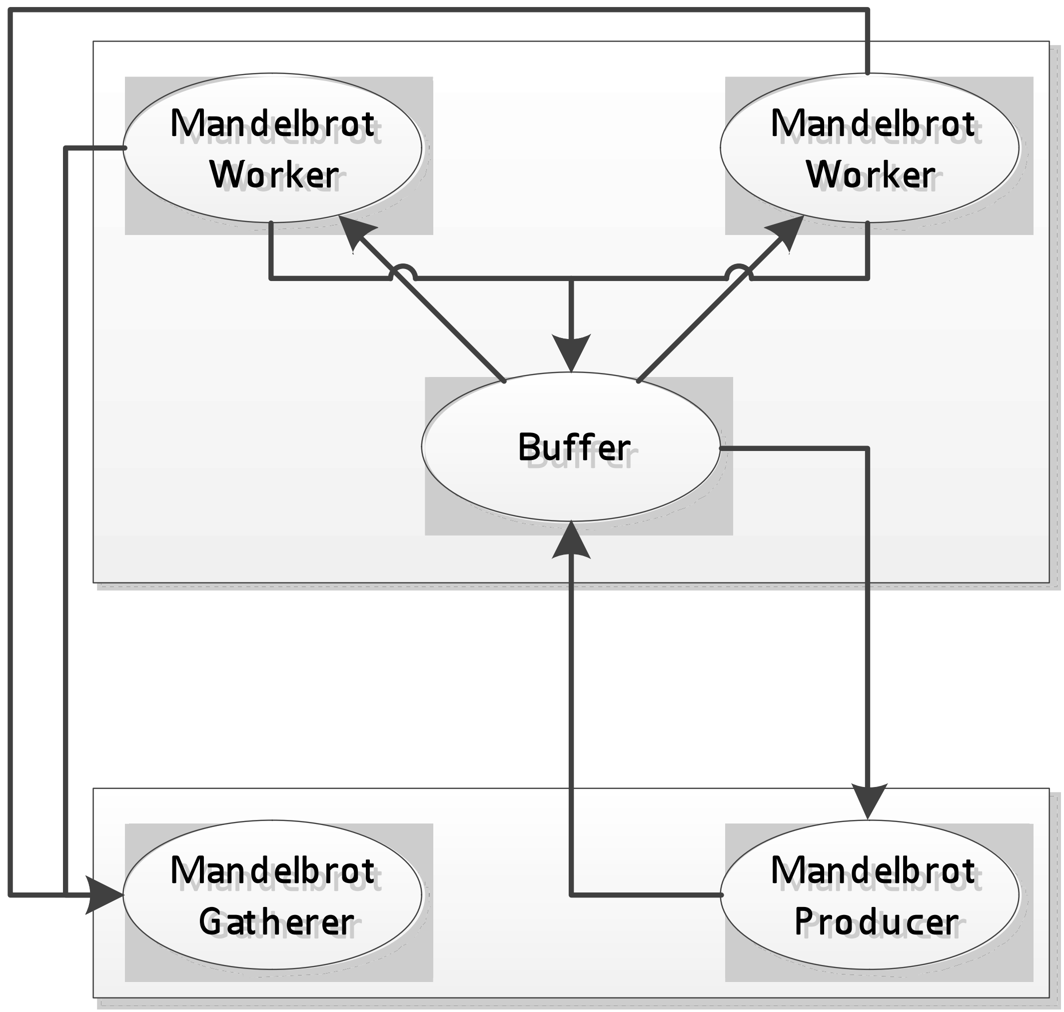
Mandelbrot

- Producing 3500 x 2000 pixel bitmaps representing parts of the Mandelbrot set
- Split a single image into multiple parts
- Scaling the set to produce multiple bitmaps
  - 2 x scale = 4 parts (7000 x 4000 total image size)
  - 3 x scale = 9 parts (10500 x 6000 total image size)
  - etc.
- Using the escape time algorithm

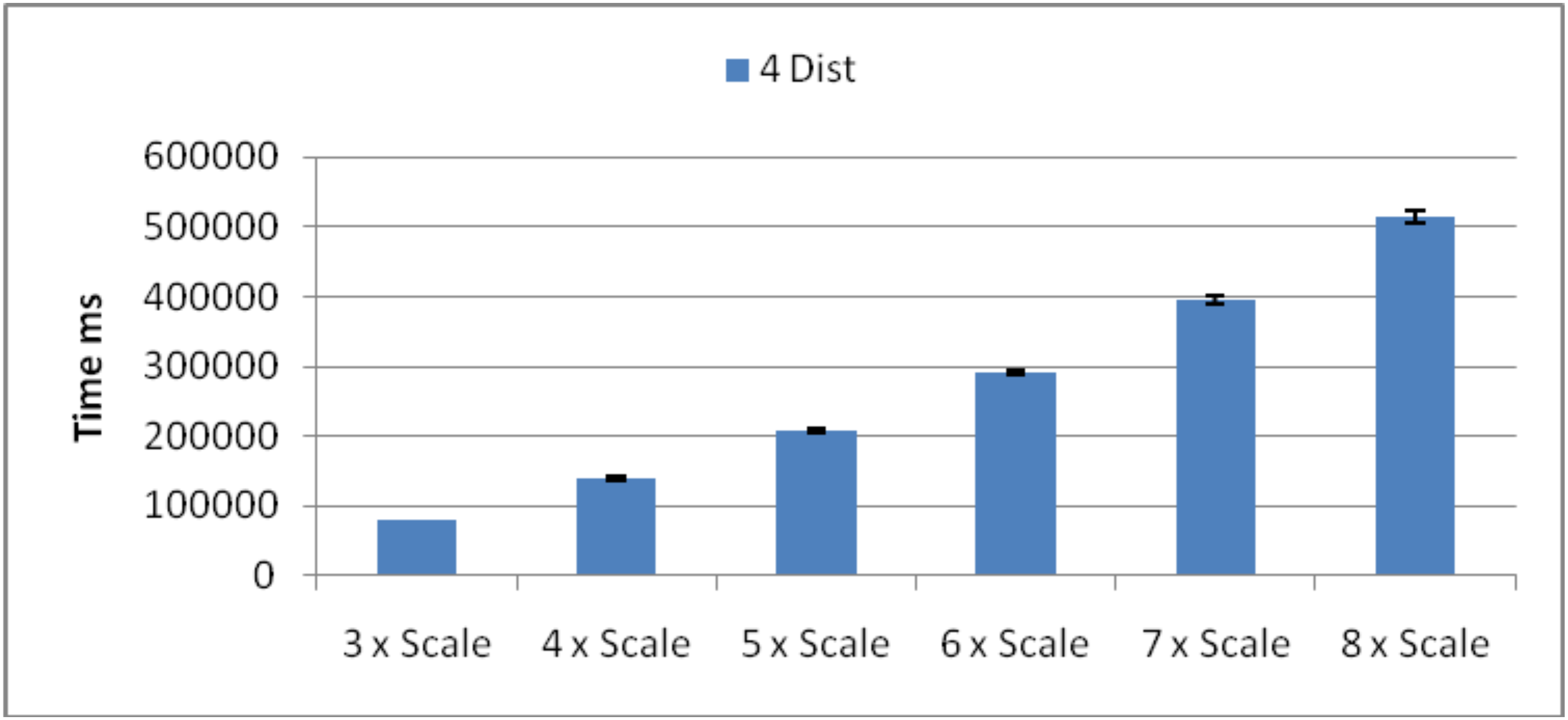
Mandelbrot Tiling



Mandelbrot Architecture



Mandelbrot Results



Throughput

Scale	Data Points	Bytes	DP / s	Bytes / s
1	7×10^6	2.8×10^7	3.25×10^5	1.3×10^5
2	2.8×10^7	1.12×10^8	6.67×10^5	2.66×10^6
3	6.3×10^7	2.52×10^8	8.04×10^5	3.21×10^6
4	1.12×10^7	4.48×10^8	8.04×10^5	3.22×10^6
5	1.75×10^8	7×10^8	8.46×10^5	3.38×10^6
6	2.52×10^8	1.01×10^9	8.65×10^5	3.46×10^6
7	3.43×10^8	1.37×10^9	8.69×10^5	3.47×10^6
8	4.48×10^8	1.79×10^9	8.71×10^5	3.48×10^6

Future Work

- Currently working on a C++CSP version of the network architecture
  - All CSP based libraries can plug-in and use
  - Hopefully finished towards the end of summer
  - Will not be in an optimised state

- Tackle some good problems with this on an HPC

- Comparison work against MPI, Erlang, etc.

- Mobility built into the protocol
  - Still no “ideal” solution

Conclusion

- We have inter-framework communication
  - Granted only between JCSP and CSP for .NET
- occam-π has a few problems when implementing the architecture we want
  - C++CSP networking should solve this
- Distributed CPA protocol and architecture gives performance comparable to the baseline network
  - Particularly at large data sizes and back and forth communication
- Some speedup when performing Mandelbrot – but not much
  - Naïve Mandelbrot implementation

QUESTIONS?

Thanks to Julien Mateos for his work on CSP for .NET 2.0, and his current work on implementing networking for C++CSP