A Critique of JCSP Networking

Kevin Chalmers, Jon Kerridge and Imed Romdhani

School of Computing

Napier University

NAPIER UNIVERSITY
EDINBURGH

Breakdown

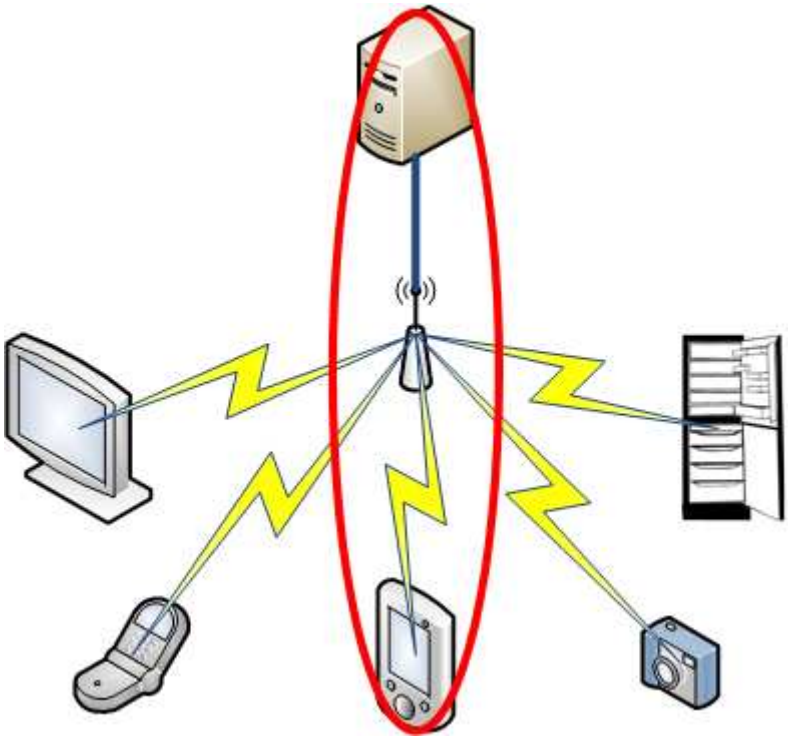
- Introduction
- Object Serialization in Java
- Current JCSP Implementation
- Problems
- Towards a Better Solution
- Conclusion

Introduction

- Numerous network aware CSP inspired frameworks
  - JCSP, pony, CSP.NET, C++CSP, PyCSP
- Majority based on T9000 virtual channel model
  - Links and channels multiplexing over one another
- None interact
  - No reason why not
- Performance usually judged on task parallelisation
  - I'm interested in the communication abstraction

Introduction

- Mobile devices
- Mobile processes (agents)
- Ubiquitous computing
  - Environment of autonomous interacting devices
  - Complex



Object Serialization in Java

- Functionality
- Example – `Integer` object

Object Serialization in Java

- Java object ⇜⇝ bytes
- Requirements
  - `Serializable` or `Externalizable` interface
  - `ObjectInputStream` and `ObjectOutputStream`
- Class description and object data sent
  - Class description includes inheritance information
- Control signals
- Use of references in the stream
  - Aliasing is a problem

Object Serialization in Java

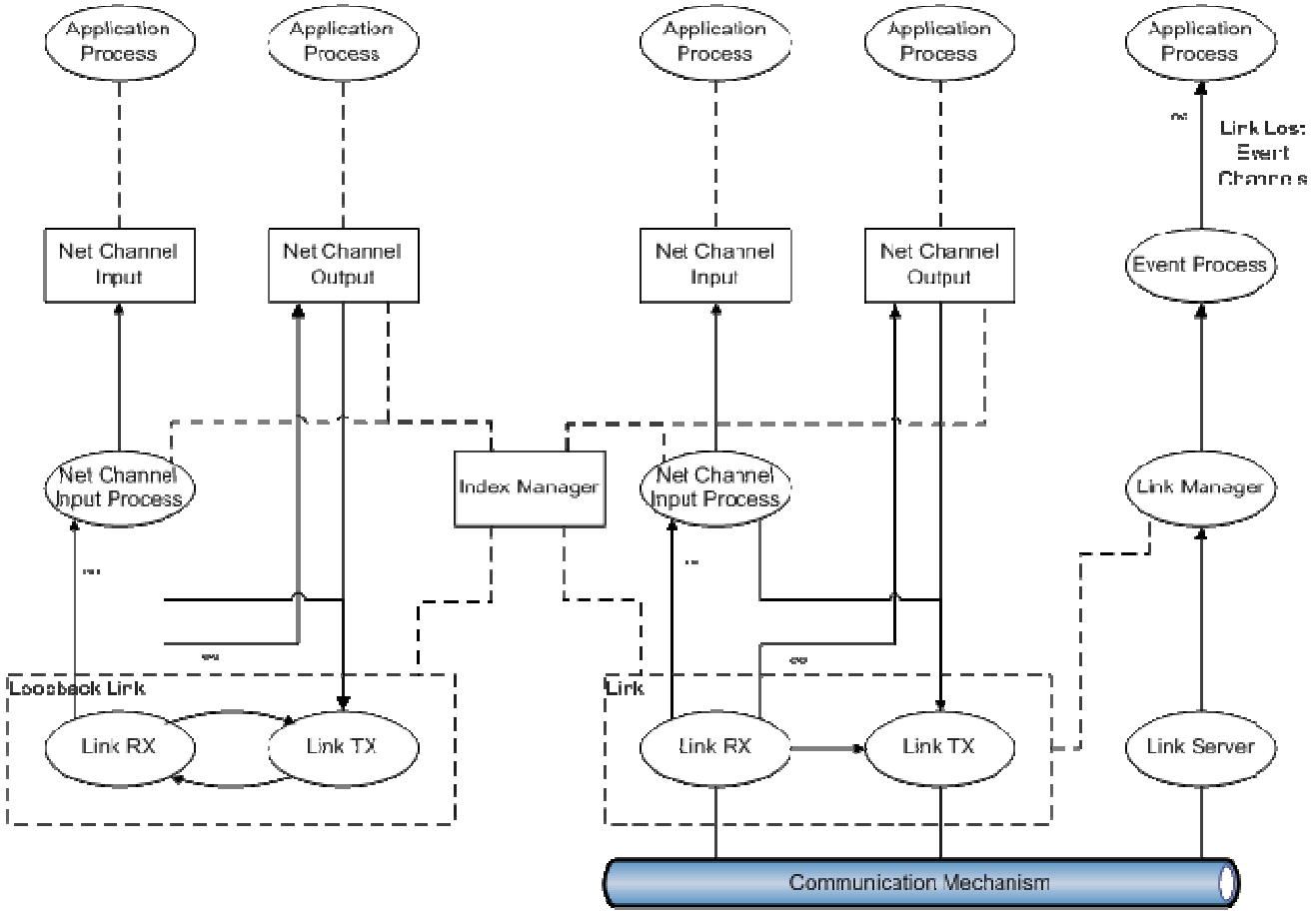
- `Integer` object
  - Extends `Number`
  - Wraps 32-bit value in an object

	0	1	2	3	4	5	6	7	8	9
0	TC_OBJECT	TC_CLASSDESC	Name length (17)		j	a	v	a	.	l
10	a	n	g	.	l	n	t	e	g	e
20	r	Class Serialization Identifier (1360826667806853064)								Flags
30	Variable count (1)		I(nteger)	Name length (5)		v	a	l	u	e
40	TC_ENDBLOCKDATA	TC_CLASSDESC	Name length (16)		j	a	v	a	.	l
50	a	n	g	.	N	u	m	b	e	r
60	Class Serialization Identifier (-8742448824652078987)								Flags	Variable count (0)
70		TC_ENDBLOCKDATA	TC_BASE	value						

Current JCSP Implementation

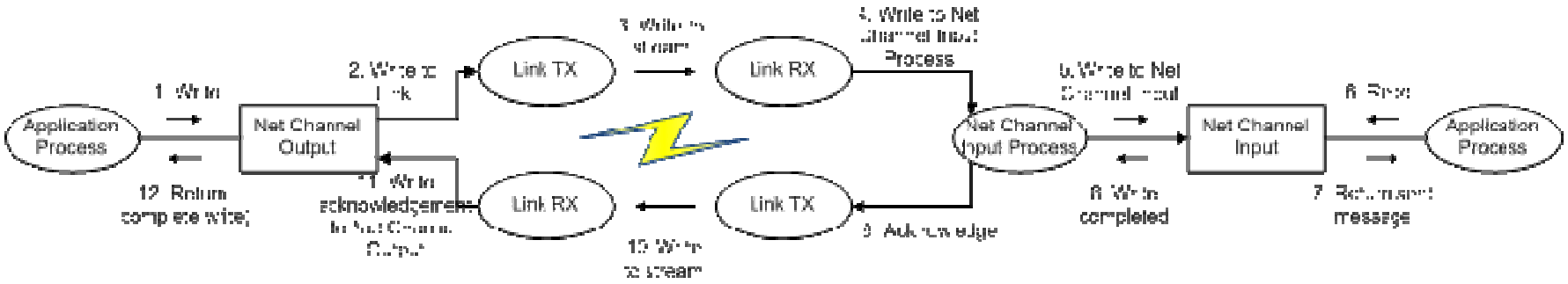
- High level architecture
- Virtual channel
- Message hierarchy

Current JCSP Implementation

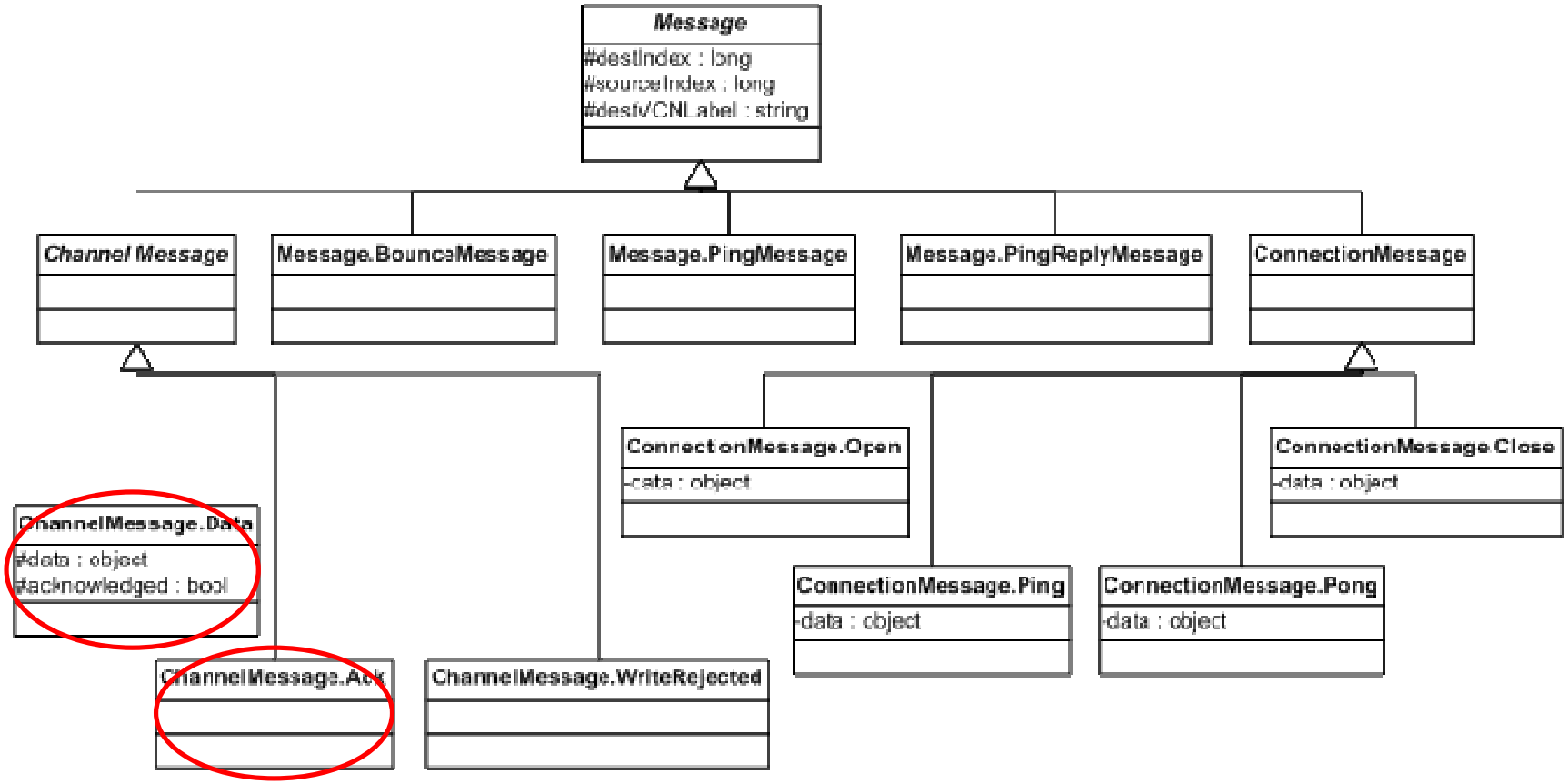


Current JCSP Implementation

- Virtual channel
  - `NetChannelOutput` to `NetChannelInput` (via the `LinkTx` and `LinkRx`)
  - At least five processes required



Current JCSP Implementation



Problems

- Resource usage
- Complexity
- Message cost
- (Java) objects only
- Performance
- High priority `Link` processes
- Exception handling
- Lack of protocol

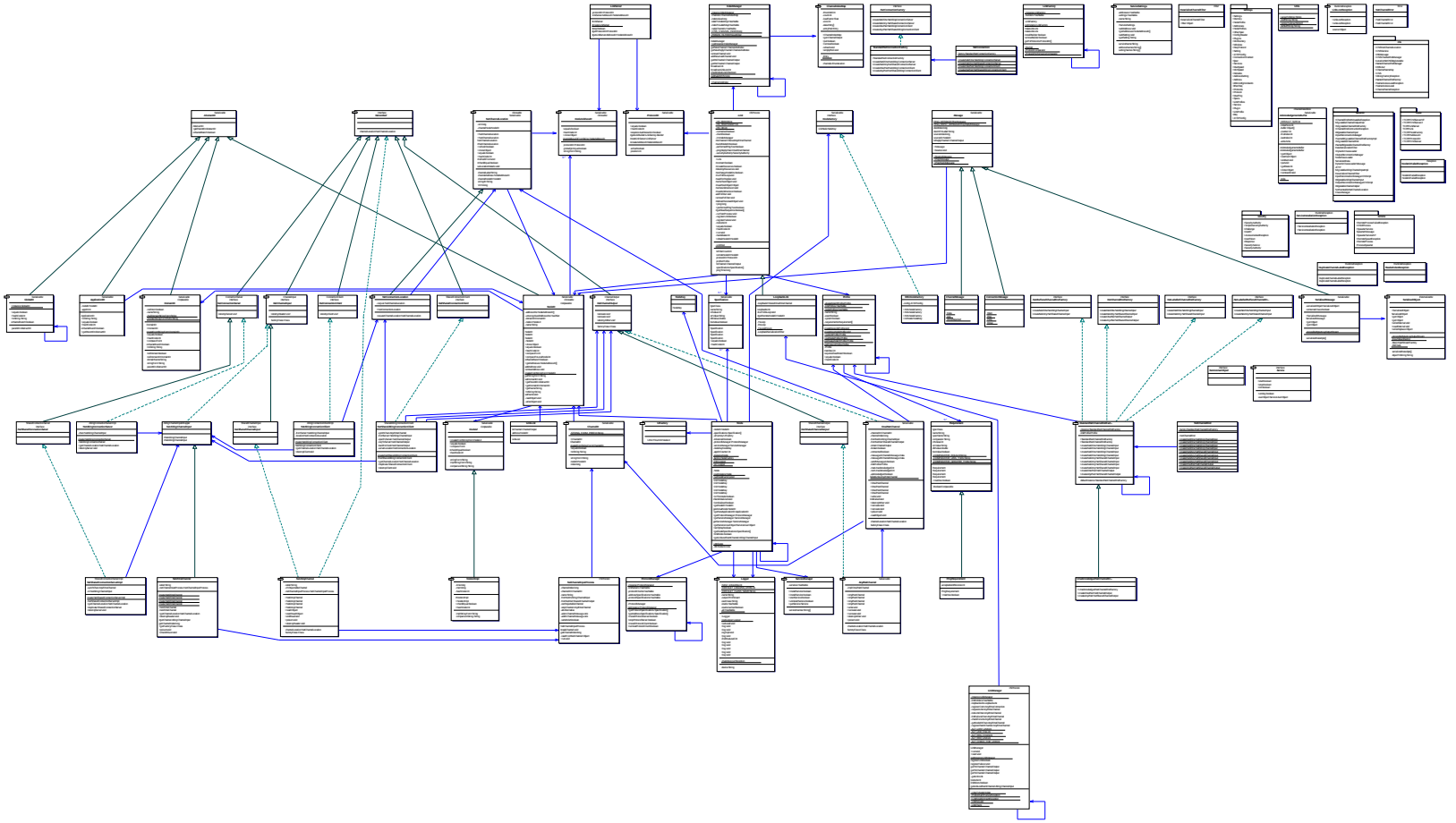
Problem – Resource Usage

- Numerous processes in operation
- Start up
  - `LoopbackLink` (2), `LinkServer`, `LinkManager`, `EventProcess`
- Extra set up
  - First `NetChannelOutput` creates handler (CNS requires one)
  - `CNSService` process
- Five processes created and destroyed during `Link` creation
- Subsequent operations
  - Each `Link` to a `Node` requires two processes (CNS Link)
  - Each `NetChannelInput` requires one process (one to CNS)
- PDA limited to 400 threads
  - Standard initialised Node uses 11 (including main)

Problem - Complexity

- Subjective
- Difficult to extend functionality
- Difficult to extract functionality
- Difficult to modify functionality
- Premise is simple
  - Crossbar between Links and channel ends
- Implementation complicated

Problem - Complexity



Problem – Message Cost

- Object messages are expensive
  - Serialization of message + serialization of sent object
- Object streams reset after each send
  - Internal pool of messages
  - Aliasing on stream
  - Class information sent each time
- Deal of sent information
  - Type, destination, source and possibly data

Problem – Message Cost

	0	1	2	3	4	5	6	7	8	9
0	TC_OBJECT	TC_CLASSDESC	Name length (32)		o	r	g	.	j	c
10	s	p	.	n	e	t	.	C	h	a
20	n	n	e	l	M	e	s	s	a	g
30	e	$	D	a	t	a	Class Serialization Identifier			
40					Flags	Variable count (2)		Z (boolean)	Name length (12)	
50	a	c	k	n	o	w	l	e	d	g
60	e	d	L (Object)	Name length (4)		d	a	t	a	TC_STRING
70	Name length (18)		L	j	a	v	a	/	l	a
80	n	g	/	O	b	j	e	c	t	;
90	TC_ENDBLOCKDATA	TC_CLASSDESC	Name length (27)		o	r	g	.	j	c
100	s	p	.	n	e	t	.	C	h	a
110	n	n	e	l	M	e	s	s	a	g
120	e	Class Serialization Identifier								Flags
130	Variable count (0)		TC_ENDBLOCKDATA	TC_CLASSDESC	Name length (20)		o	r	g	.
140	j	c	s	p	.	n	e	t	.	M
150	e	s	s	a	g	e	Class Serialization Identifier			
160					Flags	Variable count (3)		J (long)	Name length (9)	
170	d	e	s	t	l	n	d	e	x	J (long)
180	Name length (11)		s	o	u	r	c	e	l	n
190	d	e	x	L (Object)	Name length (12)		d	e	s	t
200	V	C	N	L	a	b	e	l	TC_STRING	Name length (18)
210		L	j	a	v	a	/	l	a	n
220	g	/	S	t	r	i	n	g	;	TC_ENDBLOCKDATA
230	TC_BASE	destIndex								sourceIndex
240								destVCNLabel	acknowledged	data

Problem – Message Cost

	0	1	2	3	4	5	6	7	8	9
0	TC_OBJECT	TC_CLASSDESC	Name length (31)		o	r	g	.	j	c
10	s	p	.	n	e	t	.	C	h	a
20	n	n	e	l	M	e	s	s	a	g
30	e	$	A	c	k	Class Serialization Identifier				
40				Flags	Variable count (0)		TC_ENDBLOCKDATA	TC_CLASSDE	Name length (27)	
50	o	r	g	.	j	c	s	p	.	n
60	e	t	.	C	h	a	n	n	e	l
70	M	e	s	s	a	g	e	Class Serialization Identifier		
80						Flags	Variable count (0)		TC_ENDBLOCKDATA	TC_CLASSDESC
90	Name length (20)		o	r	g	.	j	c	s	p
100	.	n	e	t	.	M	e	s	s	a
110	g	e	Class Serialization Identifier							
120	Flags	Variable Count (3)		J (Long)	Name length (9)		d	e	s	t
130	l	n	d	e	x	J (Long)	Name length (11)		s	o
140	u	r	c	e	l	n	d	e	x	L (Object)
150	Name length (12)		d	e	s	t	V	C	N	L
160	a	b	e	l	TC_STRING	Name length (18)		L	j	a
170	v	a	/	l	a	n	g	/	S	t
180	r	i	n	g	;	TC_ENDBLOCKDATA	TC_BASE	destIndex		
190						sourceIndex				
200				destVCNLabel						

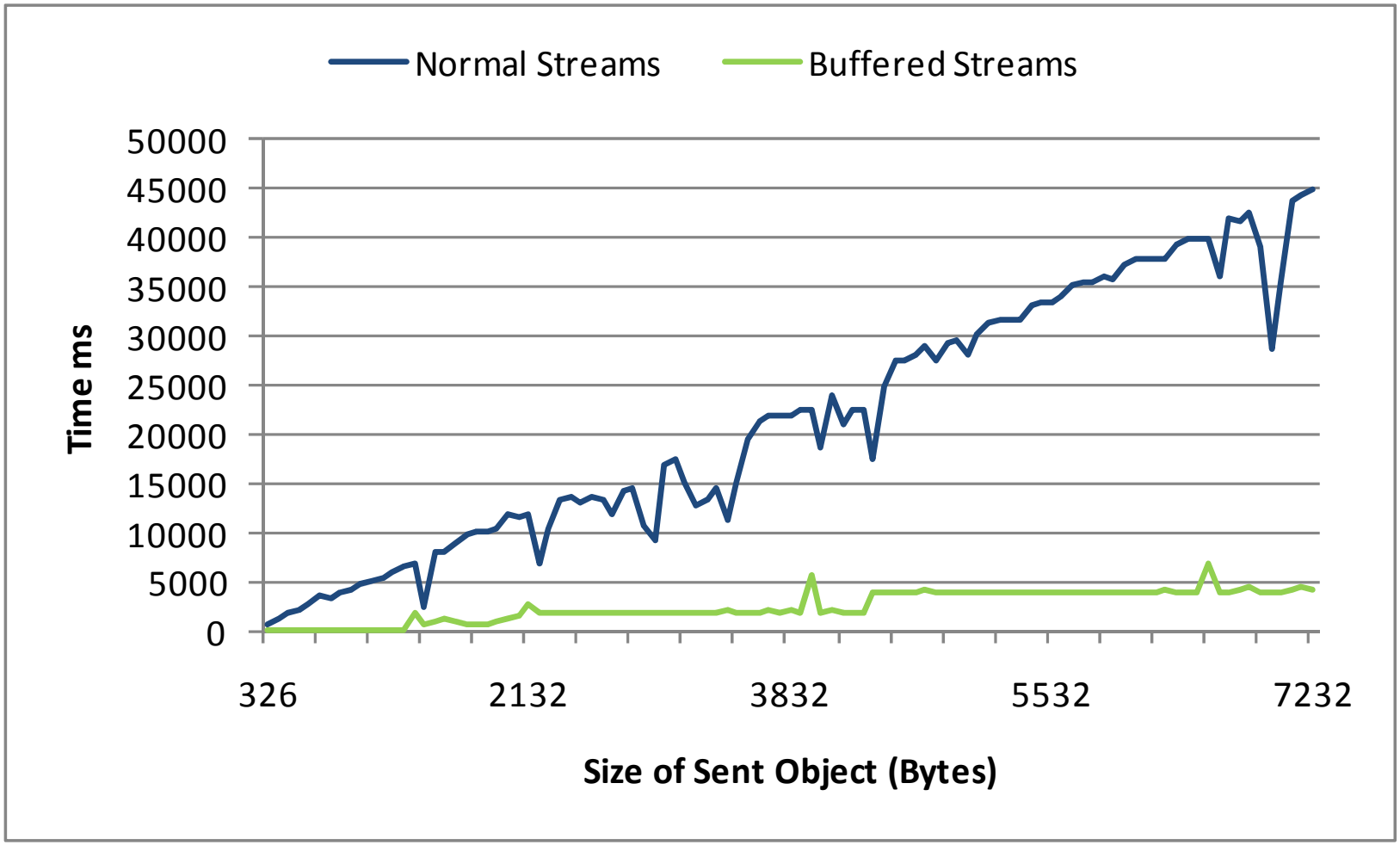
Problem – (Java) Objects Only

- Channel will only take objects
  - `ChannelMessage.Data` only takes object data
- Workarounds
  - Auto boxing, convert into byte array first
- Still a Java object
  - Serialization overhead
  - No inter-framework communication

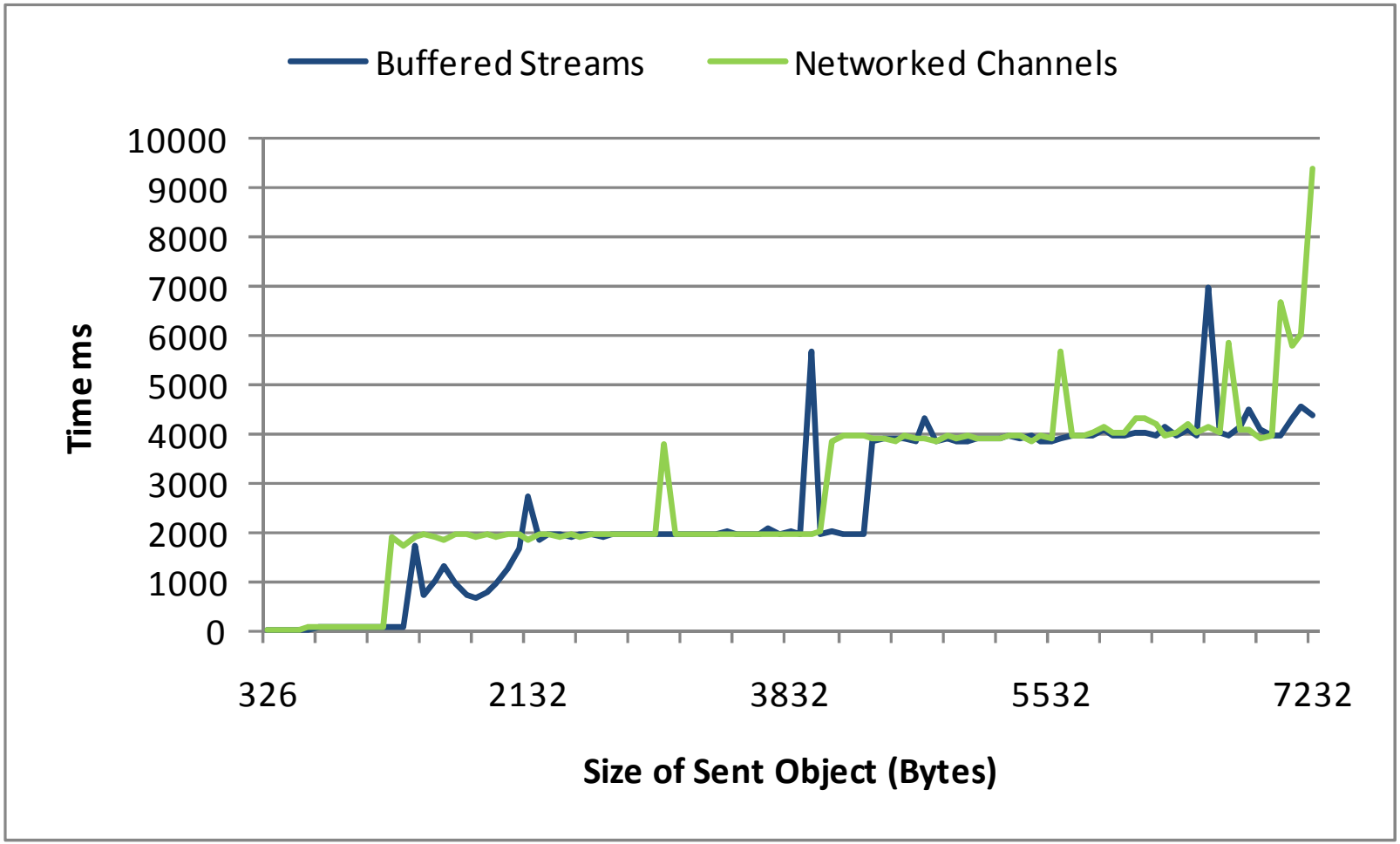
Problem - Performance

- Different test classes developed
  - Varying in complexity (unique objects to object references)
  - Internal object arrays (`Integer` and `Double`)
  - Varying in size *n* (0 to 100)
- `TestObject4`
  - Large and fairly complex
- Size
  - (n = 0) ⟶ 326 bytes
  - (n = 1) ⟶ 500 bytes
  - (n > 1) ⟶ 500 + 68(n – 1) bytes
- Complexity
  - Object references: 10 + 8n
  - Unique objects: 10 + 4n
- Roundtrip time – PDA to PC

Problem - Performance



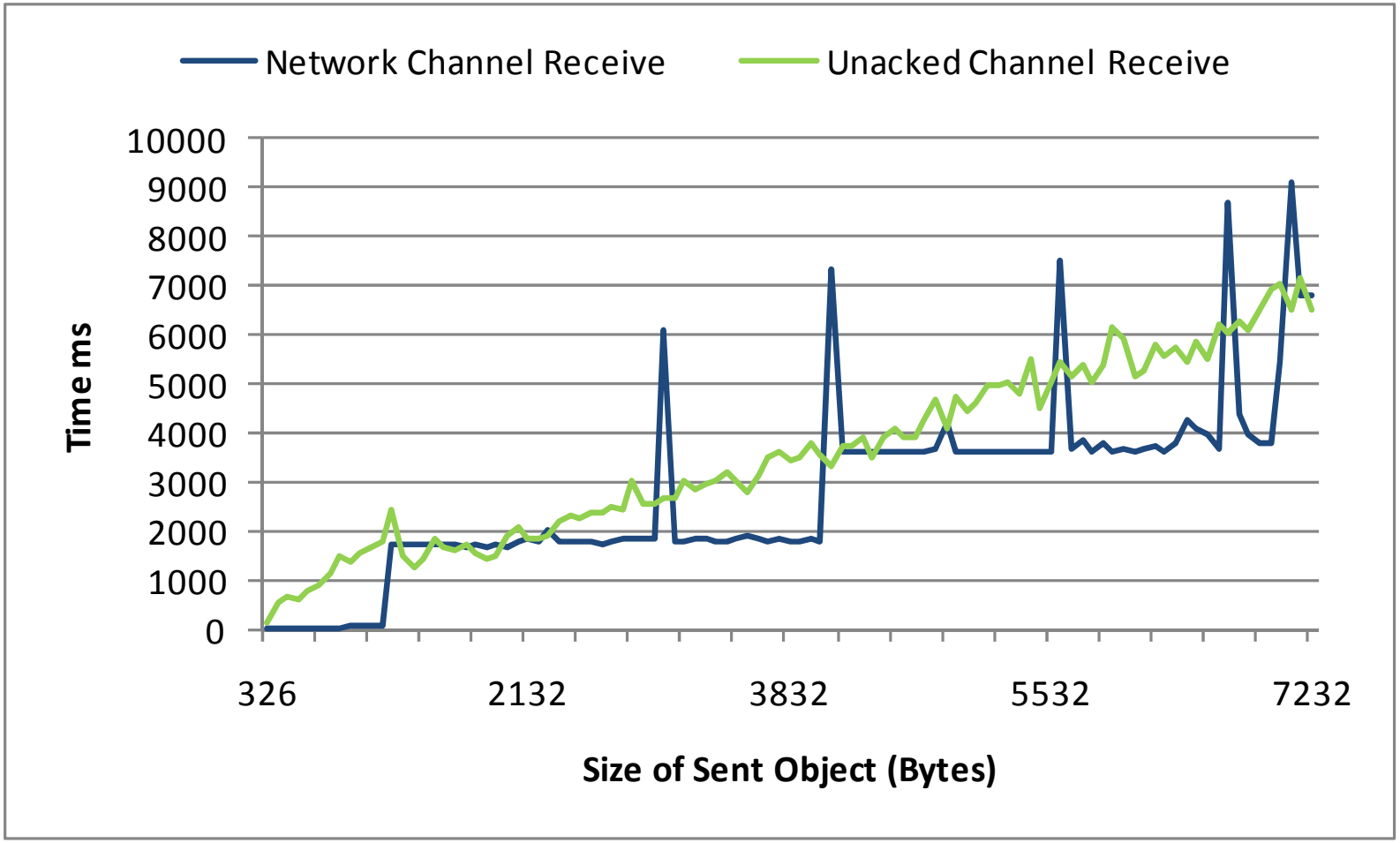
Problem - Performance



Problem – High Priority Links

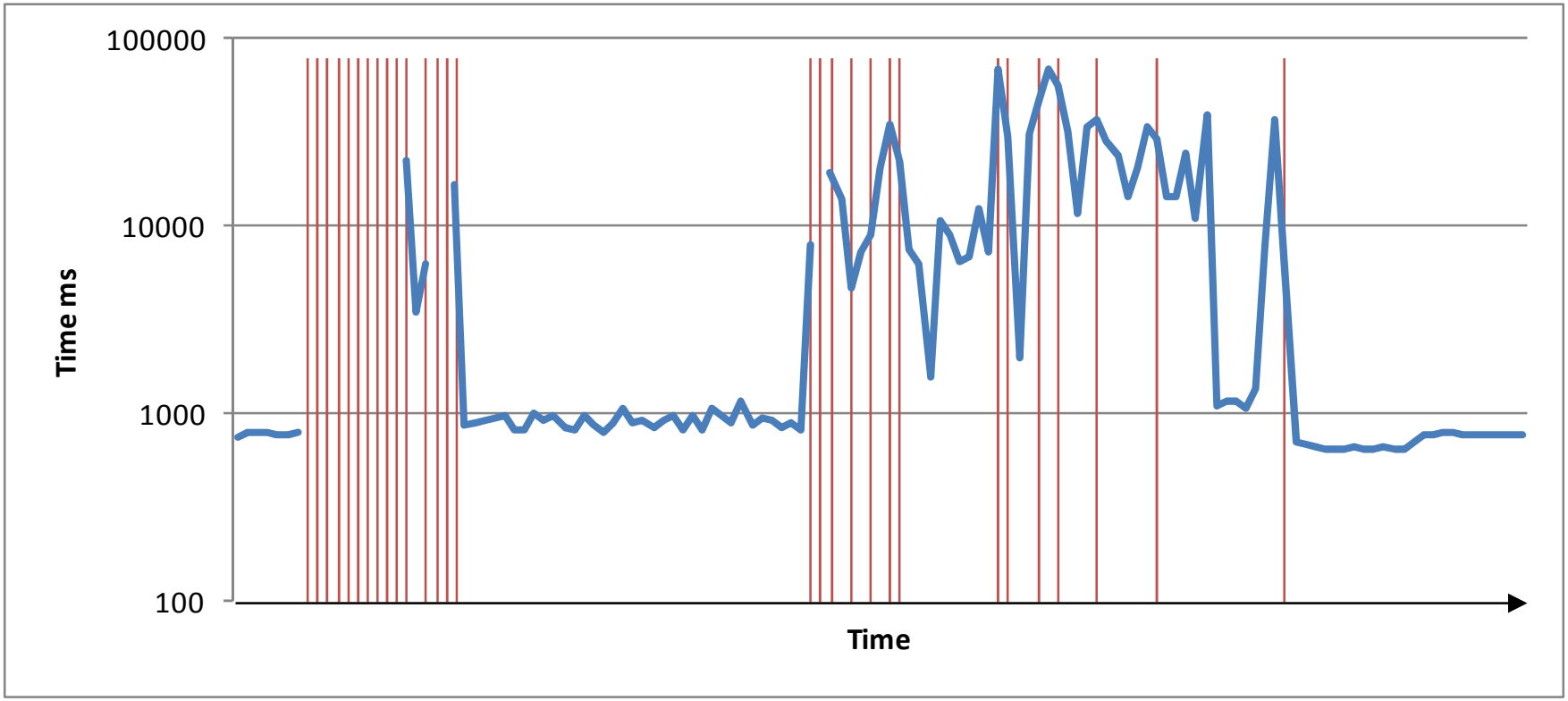
- Link processes given maximum priority in the JVM
  - Depends on JVM implementation
- Computation must wait for I/O to complete
  - Within obvious I/O performance and multi-core properties
- Adding Links and channels increases I/O requirements
  - And thus reduces computation resources
- JCSP networking aimed at high computation to low communication ratio systems

Problem – High Priority Links



Problem – High Priority Links

- CommsTime on PDA



Problem – Exception Handling

- Poor exception handling
  - `write()` may block if `Link` to input end fails
  - Exceptions thrown internally but not propagated
  - If lucky might get a `LinkLostException`
  - If luckier might get a `ChannelDataRejectedException`
    - Rejectable channels now deprecated
  - Use `LinkLostEventChannel` to detect `Link` failure
    - `NetChannelOutput` not guarded

Problem – Lack of Protocol

- No interaction between frameworks
  - Even between different versions of Java
- Difficult to add new communication primitives
  - Can use underlying network channels (extra resources)
  - Numerous places to add functionality
- Data should be extracted from communication message
  - Serialization (again)

Ongoing Work and Conclusions

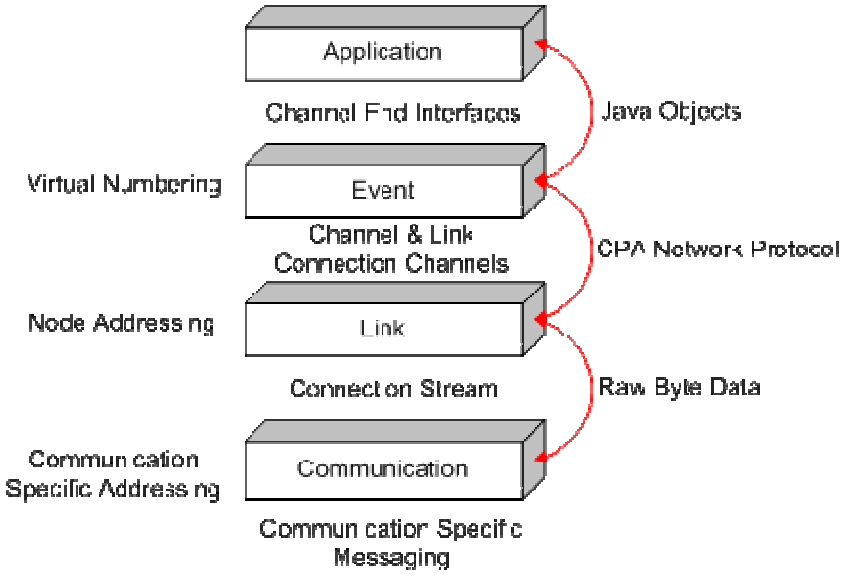
- JCSP Networking 2.0
  - Features
- Conclusions

Towards a Better Solution

- JCSP Networking 2.0
  - See Fringe session
- Features
  - Lower resource usage
    - Input channels are now lightweight
  - Reduced message size
    - 9 bytes standard, 13 for data messages
  - Better performance and scalability
  - No reliance on serialization
  - Data conversion extracted to channel layer

Towards a Better Solution

- Features (continued)
  - Simpler (layered) model
    - Layers only understand certain message types
  - Easier to extend
  - Networked barrier
  - Properties exposed
    - Link priority, buffer size
  - Error handling improved
  - Well defined protocol
  - Verified model
    - Using Spin (mobile channels)
    - FDR? – mobile channels in CSP



Conclusions

- JCSP Networking had some problems
  - Resource usage
  - Performance – serialization
  - Interoperability – serialization
  - Configuration and extension
  - Exception handling
  - All reflect badly for JCSP outside parallel computing usage
- These can be overcome
  - New JCSP Networking implementation
  - New protocol
- Future work – framework interaction?
  - JCSP, pony, C++CSP, PyCSP, CSP.NET

Questions?