Tackling online game development problems with a novel network scripting language

Paul Sheppard – ITI Techmedia
George Russell – Codeplay Software
Alastair F Donaldson – Codeplay Software
Rich Rowan – Codeplay Software
Verena Achenbach – Codeplay Software

iti Techmedia

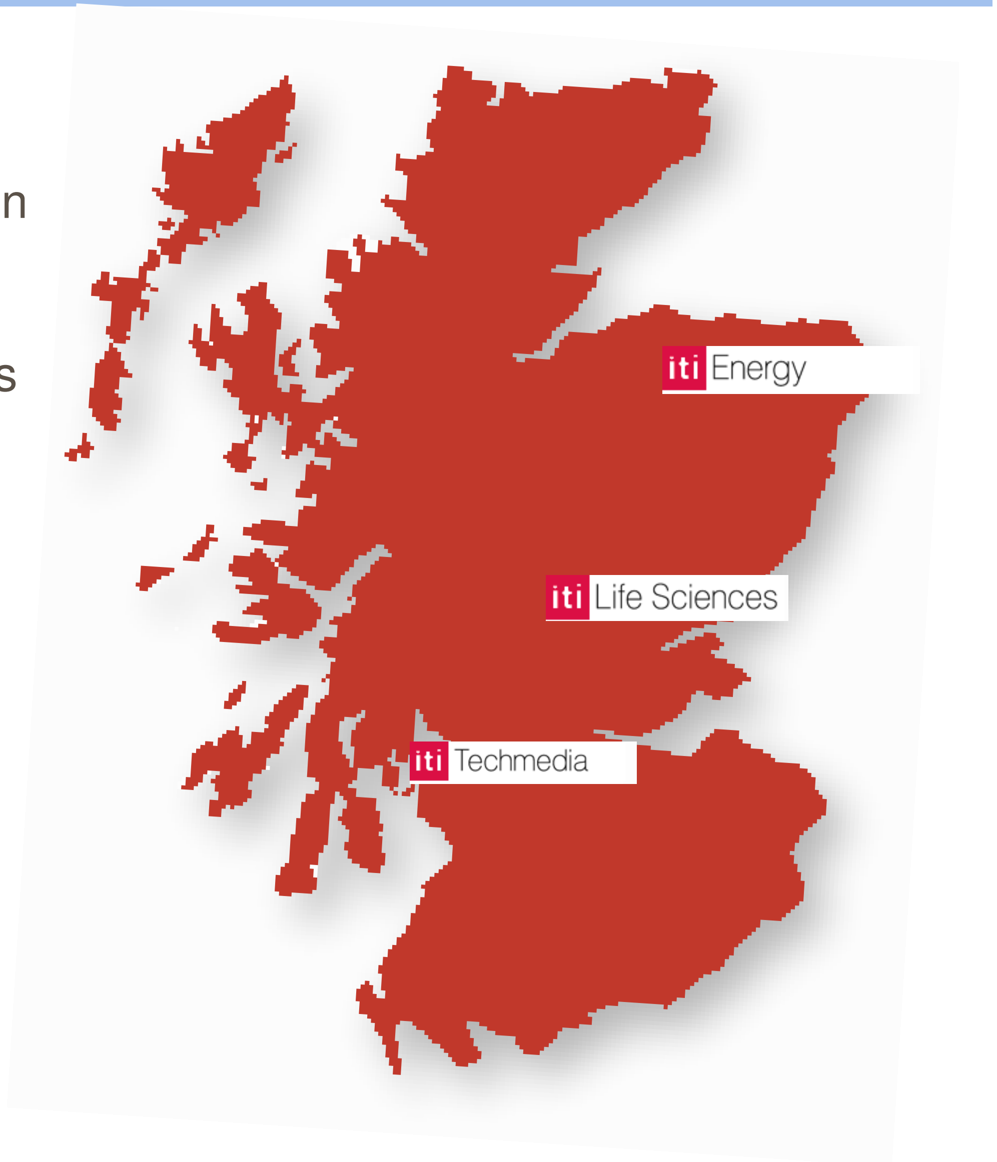
ITI Vision and Structure

ITI Scotland is a member-based commercial organisation focused on driving sustainable economic growth in Scotland through ownership of commercially targeted R&D programmes which deliver world-class intellectual assets

Established 2003

$800m funding over 10 years

$200m already committed to creating innovative, commercially-focused IP across 20 R&D programmes



R&D Programme
Online Games Development

- Online Games Programme began October 2005
- Completes in December 2008
- Budget of $10m
- 4 key strands to the programme
  - **Network Scripting Language**
  - **Games Design Toolkit**
  - **Software Productivity Tools**
  - **Procedural Content Generation**

Network Scripting Language (NSL)

iti Techmedia

DELIVERING
INNOVATION
TO INDUSTRY

Goal

To create a novel scripting language for writing bandwidth-efficient online game logic

The problem

Writing network applications is hugely complex.

- Latency
- Bandwidth Efficiency
- Concurrency
- Debugging
- Testing

Our Approach

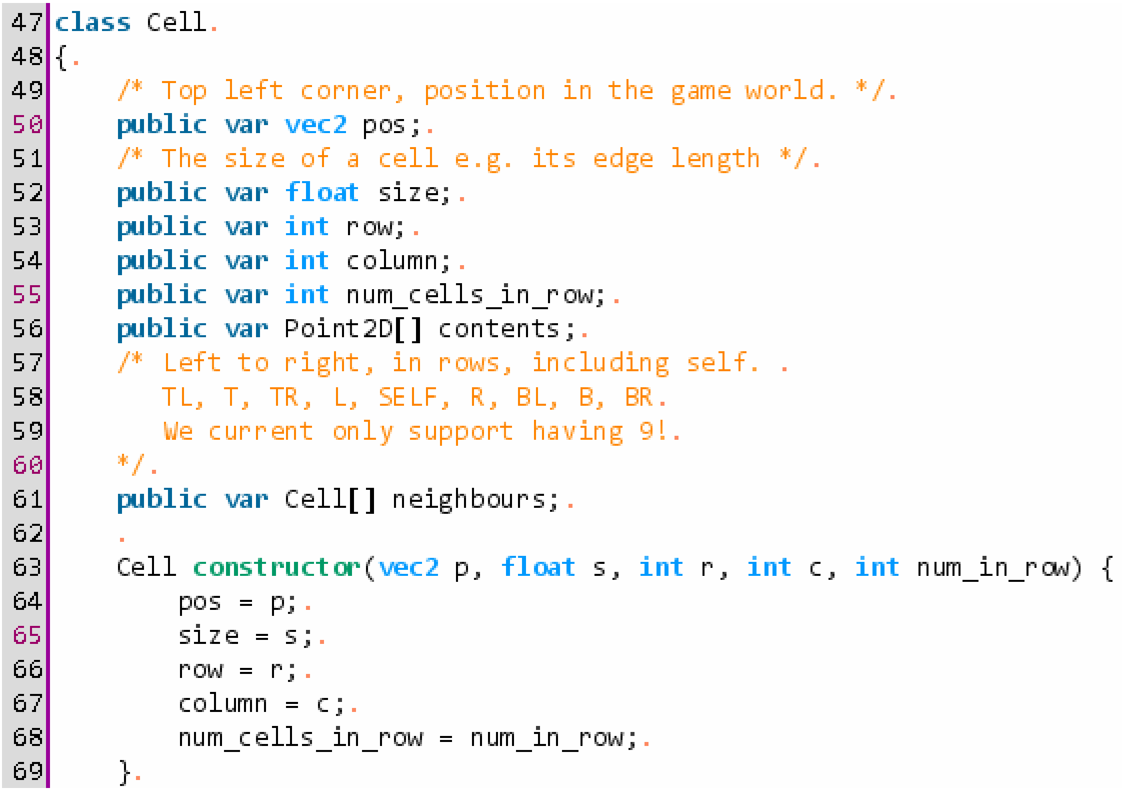
Create a simple language that handles complexity

- Easy to use
- Object oriented
- Deterministic
- Provides network transparency
- Includes debug & testing tools

Easy to use Programming

NSL Code has been designed to be easily understood by programmers

- Uses a Java-like syntax
- Game and world is composed of *objects*



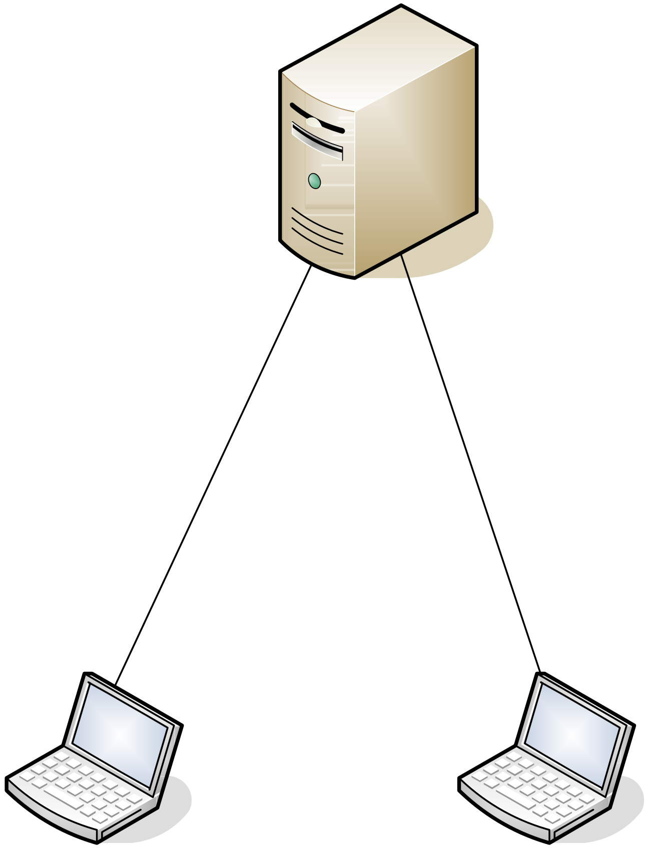
```
class Cell
{
    /* Top left corner, position in the game world. */
    public var vec2 pos;
    /* The size of a cell e.g. its edge length */
    public var float size;
    public var int row;
    public var int column;
    public var int num_cells_in_row;
    public var Point2D[] contents;
    /* Left to right, in rows, including self.
       TL, T, TR, L, SELF, R, BL, B, BR
       We current only support having 9!
    */
    public var Cell[] neighbours;

    Cell constructor(vec2 p, float s, int r, int c, int num_in_row) {
        pos = p;
        size = s;
        row = r;
        column = c;
        num_cells_in_row = num_in_row;
    }
```

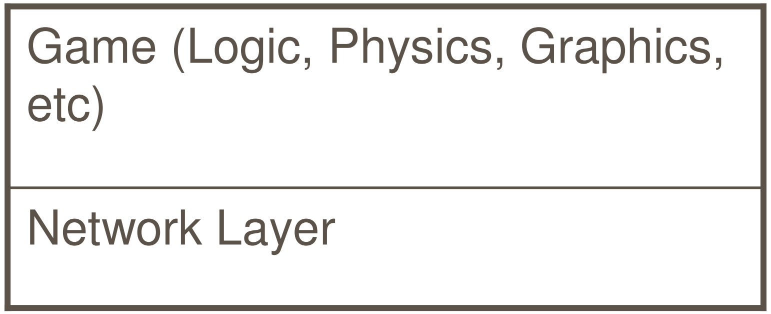
Easy to use Integration

- Easy to integrate with any game engine written in C++
- Could be implemented in other languages on other platforms
- Designed to work seamlessly on top of different networking libraries

Network Transparency



Standard Implementation

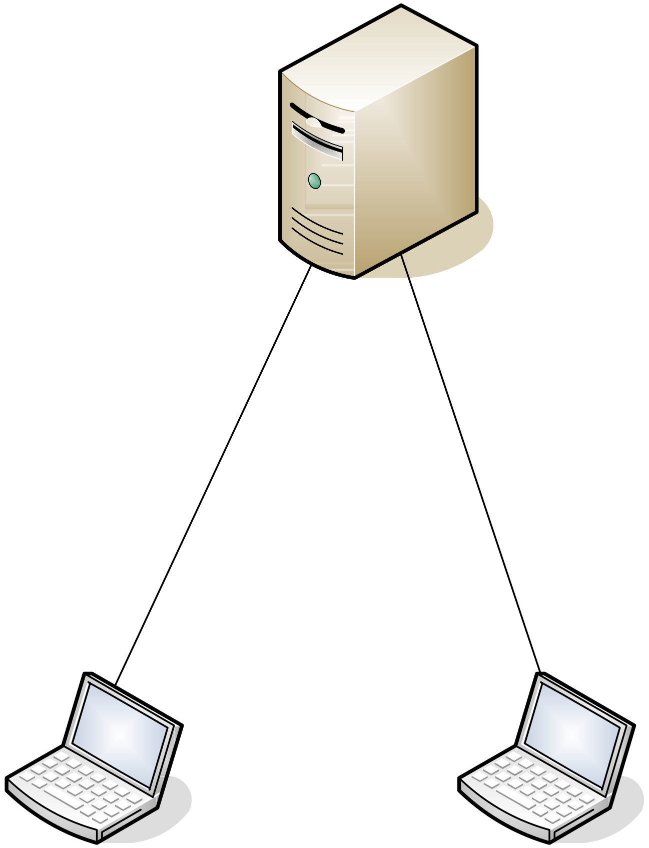


Game (Logic, Physics, Graphics, etc)

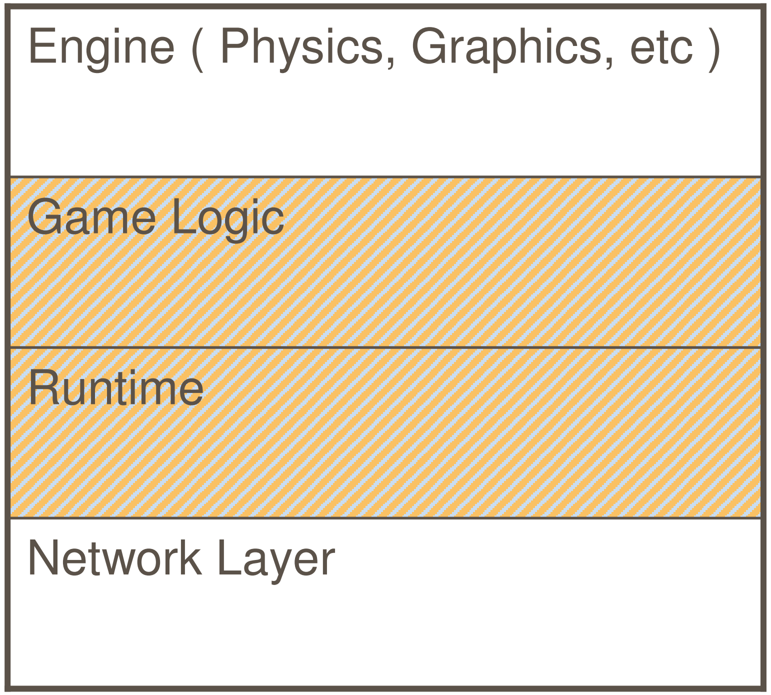
Network Layer

iti Techmedia

Network Transparency



NSL Implementation



Engine (Physics, Graphics, etc)

Game Logic

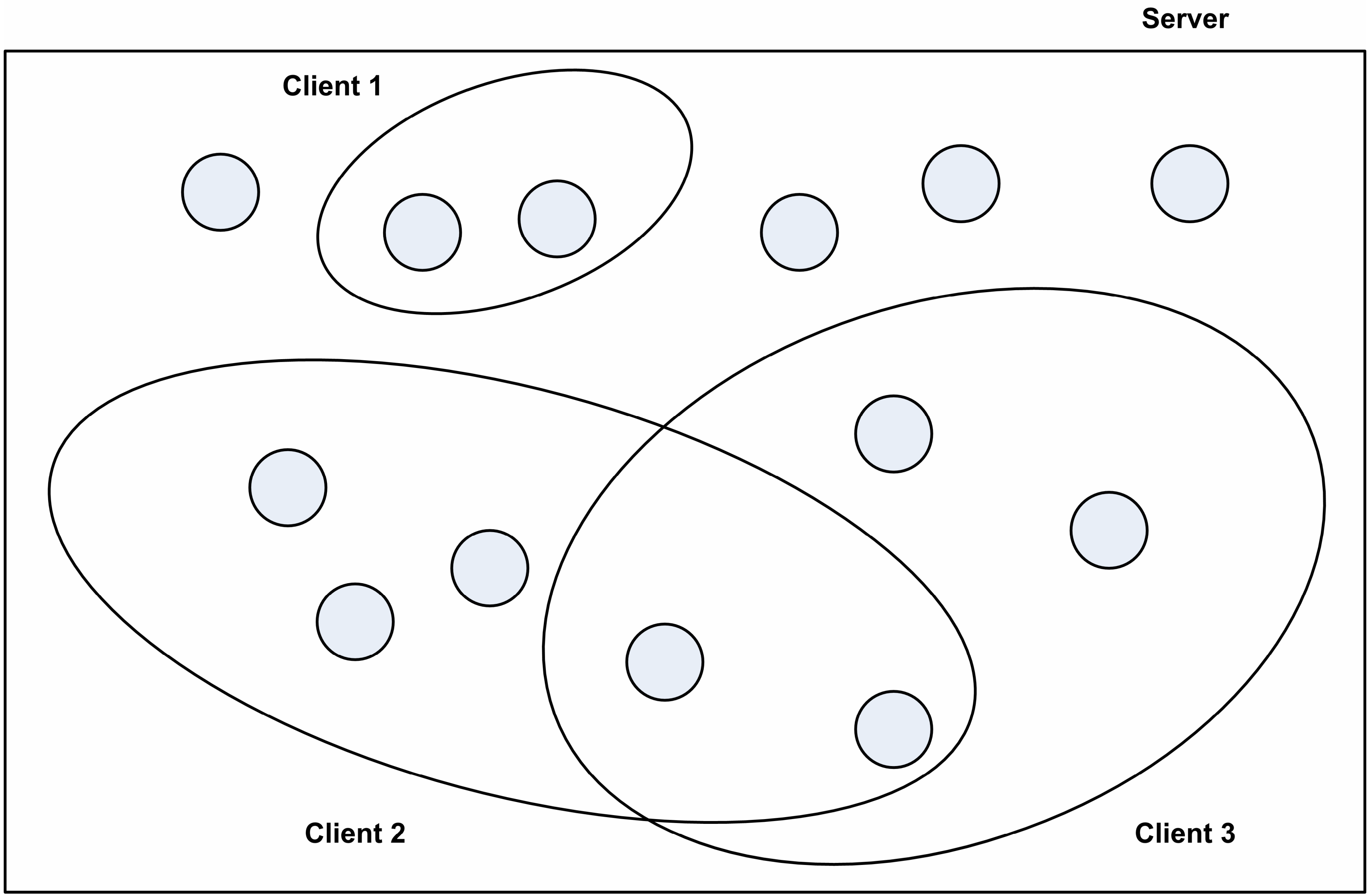
Runtime

Network Layer

Deterministic

- NSL enforces applications to be deterministic
- Server ensures clients maintain consistent state
- All clients and server will obtain the same results for the same inputs regardless of location or time of computation

Bandwidth Efficiency

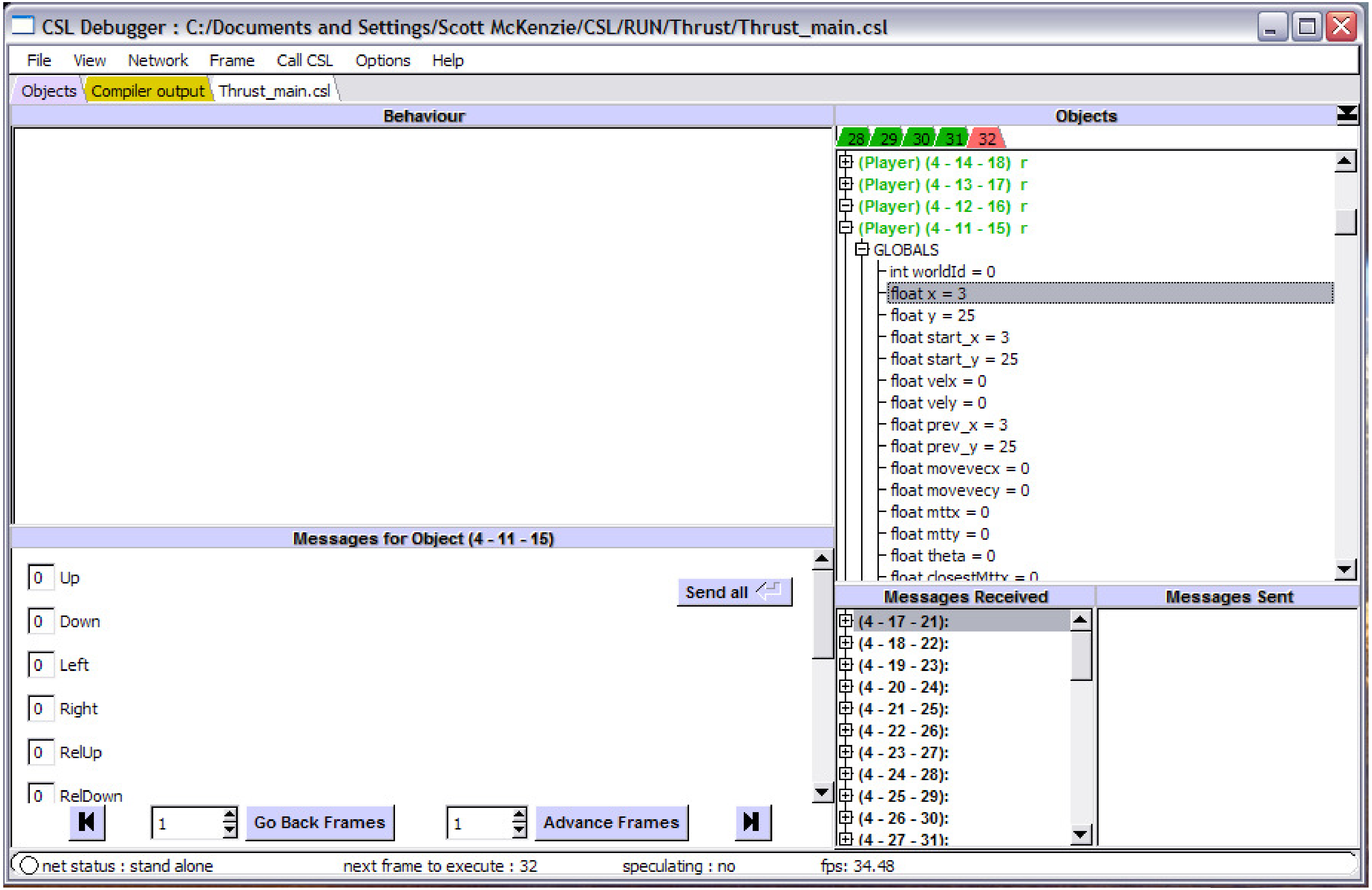


Concurrency

- Parallel by design, invisible to programmer
- Multi-threaded runtime
- Aim to achieve linear parallel performance scaling in most cases.

DELIVERING INNOVATION TO INDUSTRY

Integrated Debugger



CSL Debugger : C:/Documents and Settings/Scott McKenzie/CSL/RUN/Thrust/Thrust_main.csl

```
(Player) (4 - 14 - 18)  r
(Player) (4 - 13 - 17)  r
(Player) (4 - 12 - 16)  r
(Player) (4 - 11 - 15)  r
  GLOBALS
    int worldId = 0
    float x = 3
    float y = 25
    float start_x = 3
    float start_y = 25
    float velx = 0
    float vely = 0
    float prev_x = 3
    float prev_y = 25
    float movevecx = 0
    float movevecy = 0
    float mttx = 0
    float mtty = 0
    float theta = 0
    float closestMttx = 0
```

0 Up

0 Down

0 Left

0 Right

0 RelUp

0 RelDown

1

1

```
(4 - 17 - 21):
(4 - 18 - 22):
(4 - 19 - 23):
(4 - 20 - 24):
(4 - 21 - 25):
(4 - 22 - 26):
(4 - 23 - 27):
(4 - 24 - 28):
(4 - 25 - 29):
(4 - 26 - 30):
(4 - 27 - 31):
```

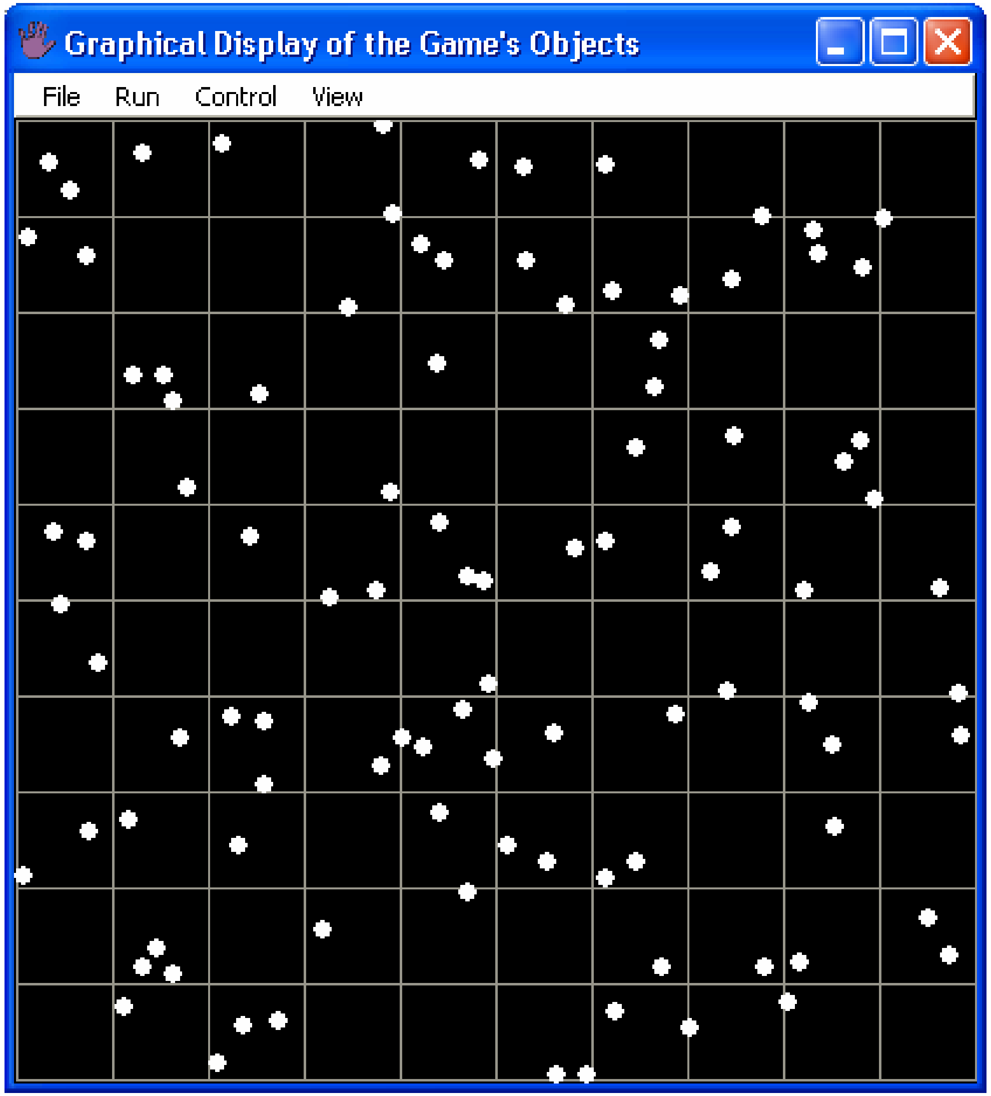
○ net status : stand alone   next frame to execute : 32   speculating : no   fps: 34.48

Testing

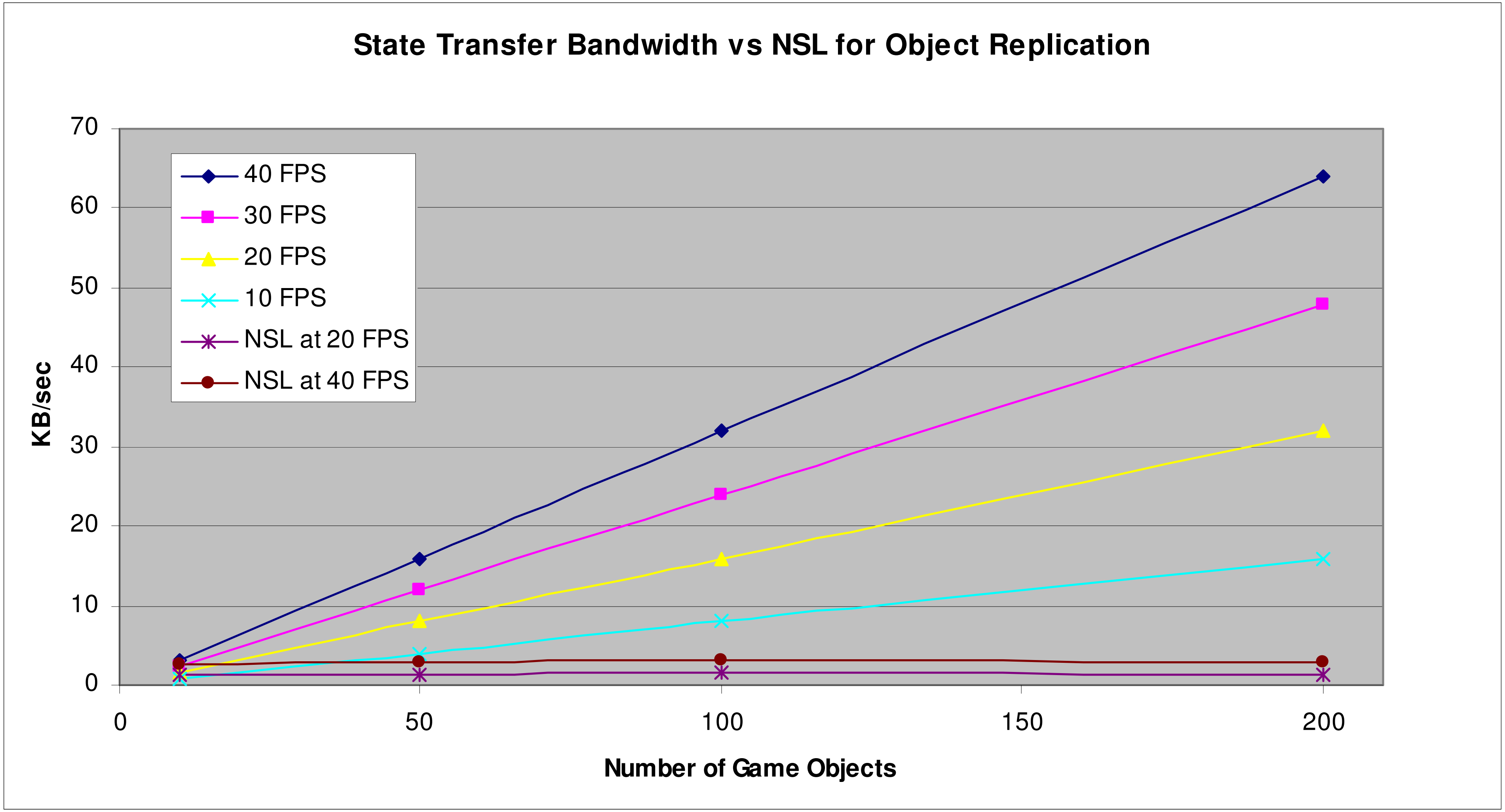
- Determinism enables us to
  - Save any frame
  - Replay then or later

- Programmer does not need to build a test harness. All handled by the default features of NSL.

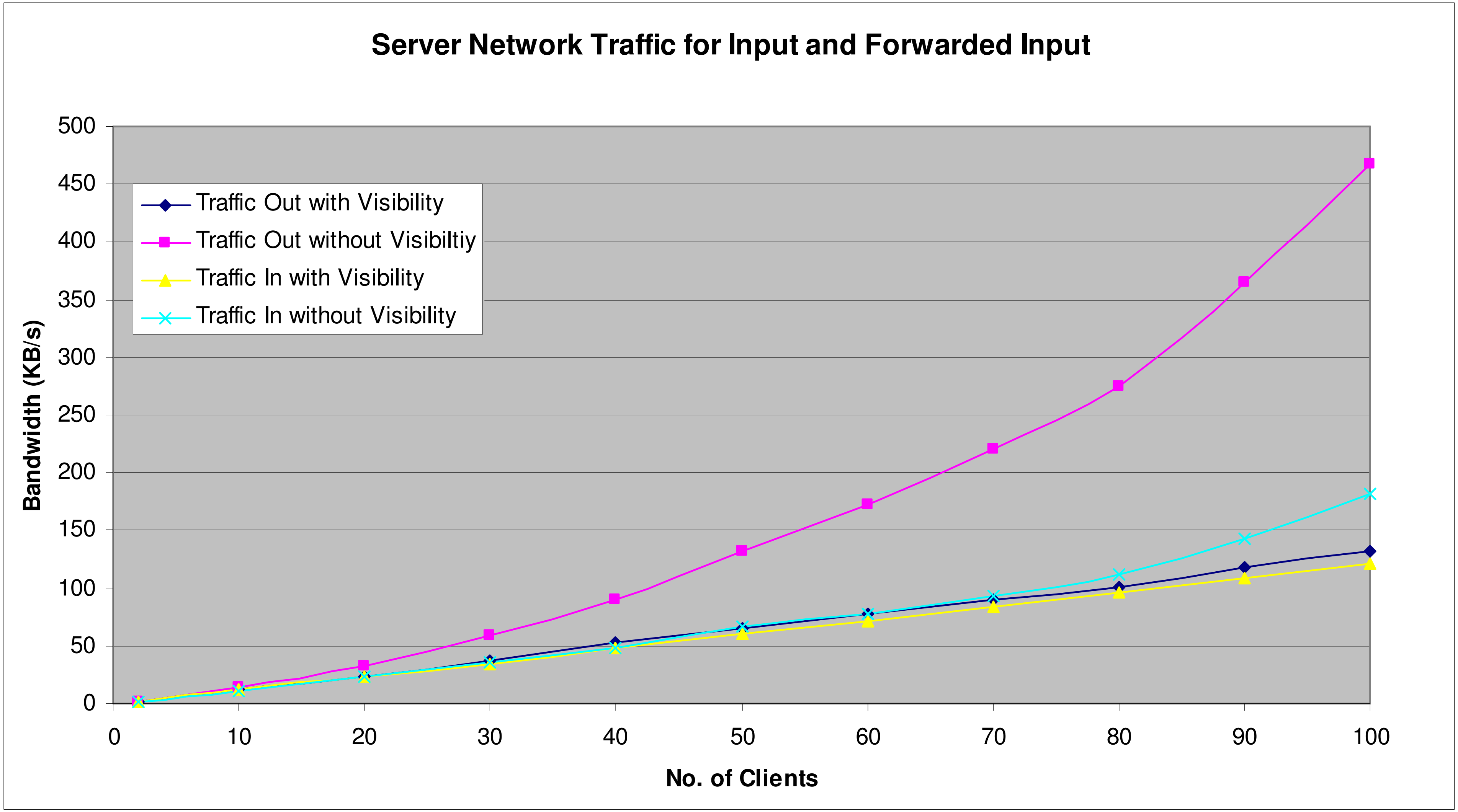
Pointworld Test



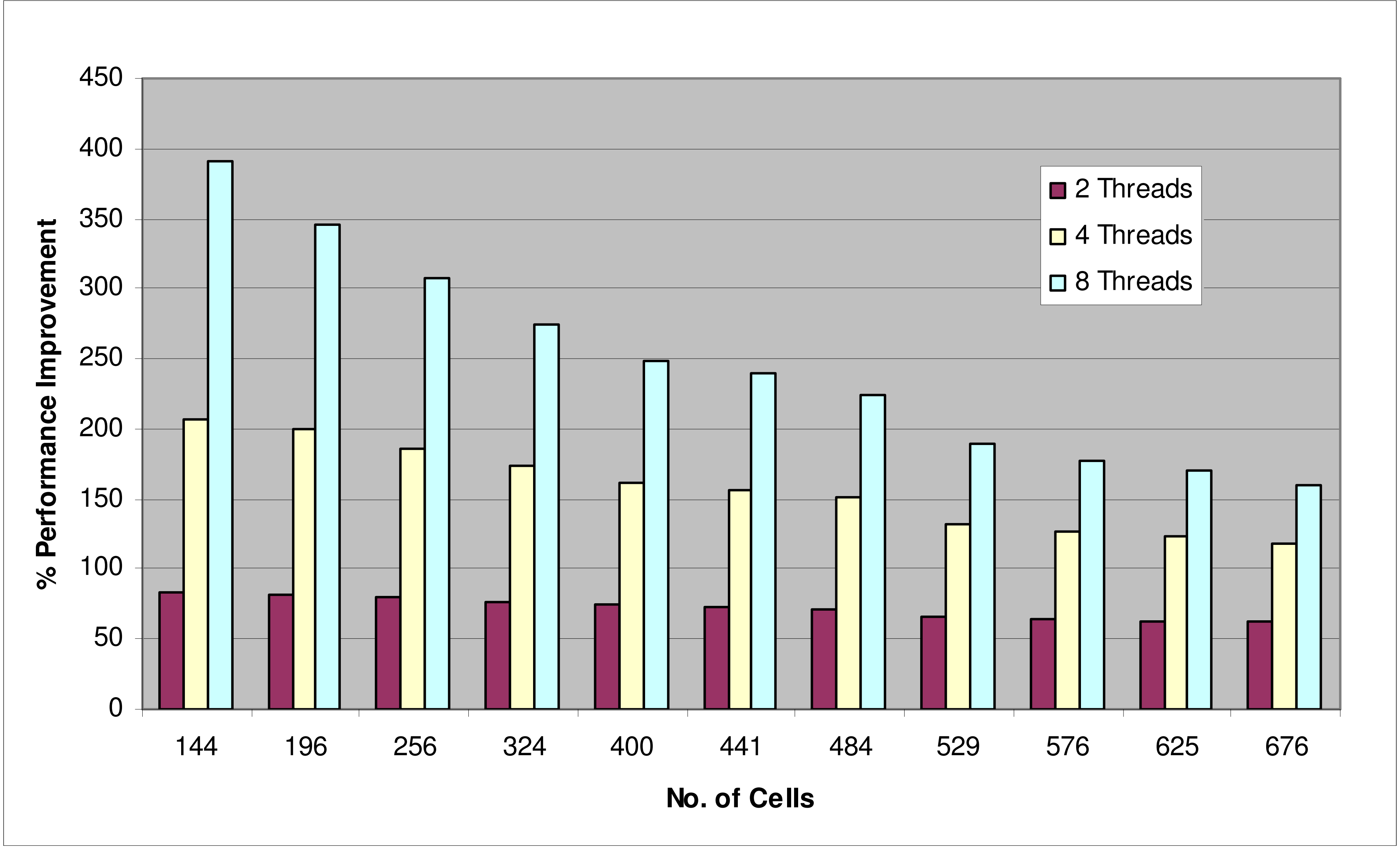
Results
Bandwidth Efficiency



Results
Server-client communication



Results
Concurrency Test



Conclusions

- Initial results show that NSL can result in efficient use of network bandwidth.
- NSL can exploit the power of multi-core processors automatically.
- Programming of game logic is simplified as the required knowledge of underlying network system is minimised.

Commercialisation Opportunities

ITI are looking for companies who are interested in licensing this research for use in commercial projects.

Please contact us if you are interested in learning more

Q & A

paul.sheppard@ititechmedia.com

richard@codeplay.com