To Boldly Go …

(an occam-π mission on engineering emergence)

Peter Welch[1], Kurt Wallnau[2], Adam Sampson[1], Mark Klein[2]
[1] School of Computing, University of Kent
[2] Software Engineering Institute, Carnegie-Mellon University

Emergenet, York, 20th. April, 2010

To Boldly Go:
an occam-π Mission

A thesis, *boids* and a demo …

Process architecture and *boids* …

Observations of emergence…

Summary and Conclusions …

Engineering Emergence

THESIS

Some future systems will be too complex to design and implement *explicitly*.

Instead, we will have to learn to engineer the desired behaviours *implicitly*.

We will do this through the discovery and programming of *simple* rules of behaviour, applied to a mass of *dynamically configured and interacting components*, from which desired *complex* behaviours *emerge* …

Engineering Emergence

THESIS

Some future systems will be too complex to design and implement *explicitly*.

Instead, we will have to learn to engineer the desired behaviours *implicitly*.

The components *individually* will be *simple*, showing not a hint of the *complex* behaviours that can emerge when *a lot of them* get together …

Engineering Emergence

Examples?

- Mechanisms design (game theory, micro-economics)
  - Rational actors have local, private information
  - Emergent: optimal allocation of scarce resources
  - Optimal decisions rely on truth revelation

SEI Computational Mechanism Design

Engineering Emergence

Examples?

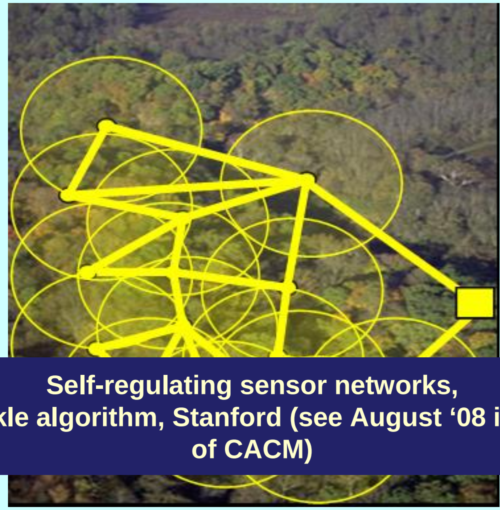
- Swarming behaviour (flocks, wasp colony behavior)
  - Autonomous (non-rational) actors, local interactions only
  - Emergent: “swarm” behavior
  - UAV swarms and autonomous robots

UAV SWARM HEALTH MANAGEMENT
Aerospace Controls Laboratory, MIT
(see http://vertol.mit.edu/)

Engineering Emergence

Examples?

- Social communication (gossip, epidemic algorithms)
  - Large, ad hoc, dynamic networks
  - Emergent: minimum power to achieve eventual consistency
  - Low power, low reliability sensors and data propagation



**Self-regulating sensor networks,
Trickle algorithm, Stanford (see August '08 issue of CACM)**

Engineering Emergence

Case study

Boids: avoid collisions, match vector with those of birds is sight, head for the centre of mass of birds in sight, take fright if a ***hoik*** is spotted, be attracted by ***foid***, …

Emergent behaviours: flocking, squabbling, migration waves, panic scattering, orbiting points of attraction *(if only a small group)*, feeding frenzy *(if a large enough flock)*, turbulence, maze solving, …

demo … occoids … cylons …

To Boldly Go:
an **occam-π** Mission

A thesis, *boids* and a demo …

Process architecture and *boids* …

Observations of emergence…

Summary and Conclusions …

Lightweight Communicating Processes

- **Fine-grained**
- **Massively parallel (zillions)**
- **Process-oriented**

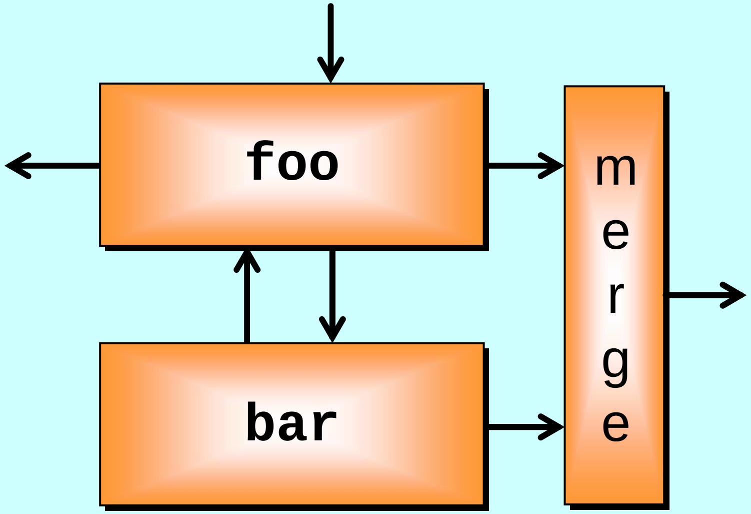
This is the way of the world ...

- **Processes, networks, networks-within-networks**
  - Channel (reader-writer) synchronisation
  - Barrier (multiway synchronisation)

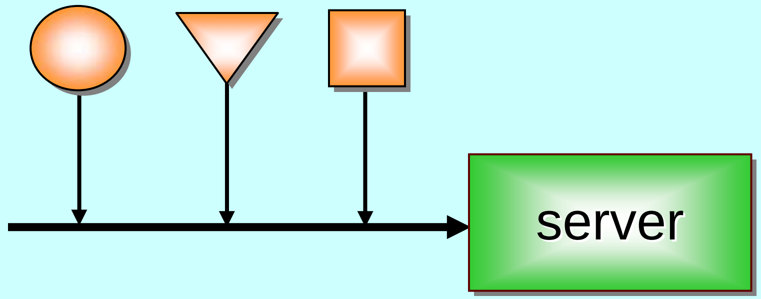
CSP / occam-π

- **Ever-changing network topologies**
  - Dynamic birth, re-connections, death
  - Mobile channels and processes
  - Mobile process location and neighbour awareness

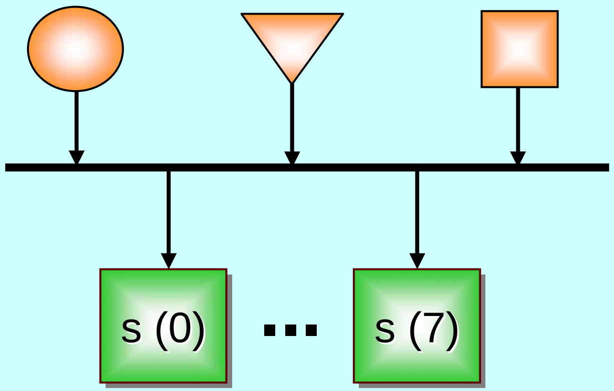
π-calc / occam-π



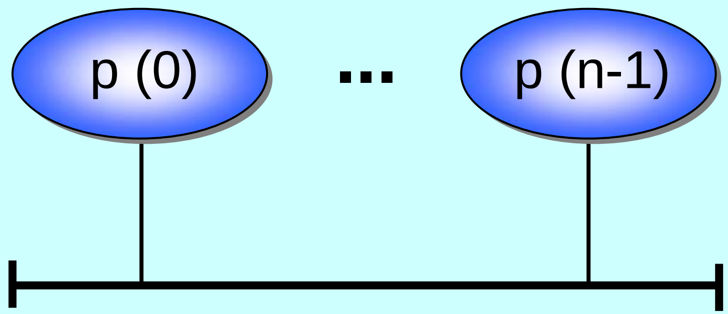
(a) a network of three processes, connected by four internal (hidden) and three external channels.



(b) three processes sharing the client end of a channel bundle to a server process.

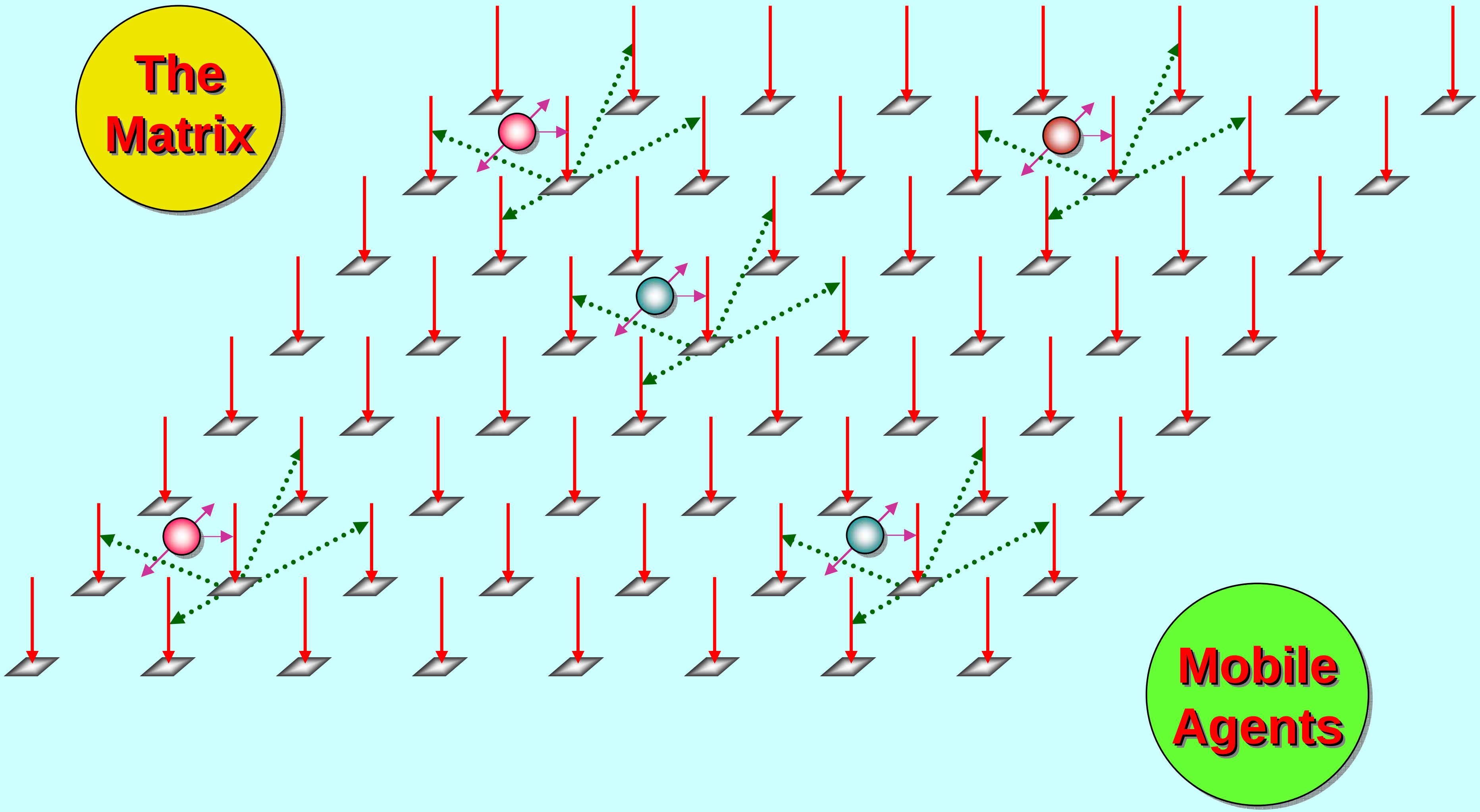


(c) three processes sharing the client end of a channel bundle to a bank of servers sharing the other end.

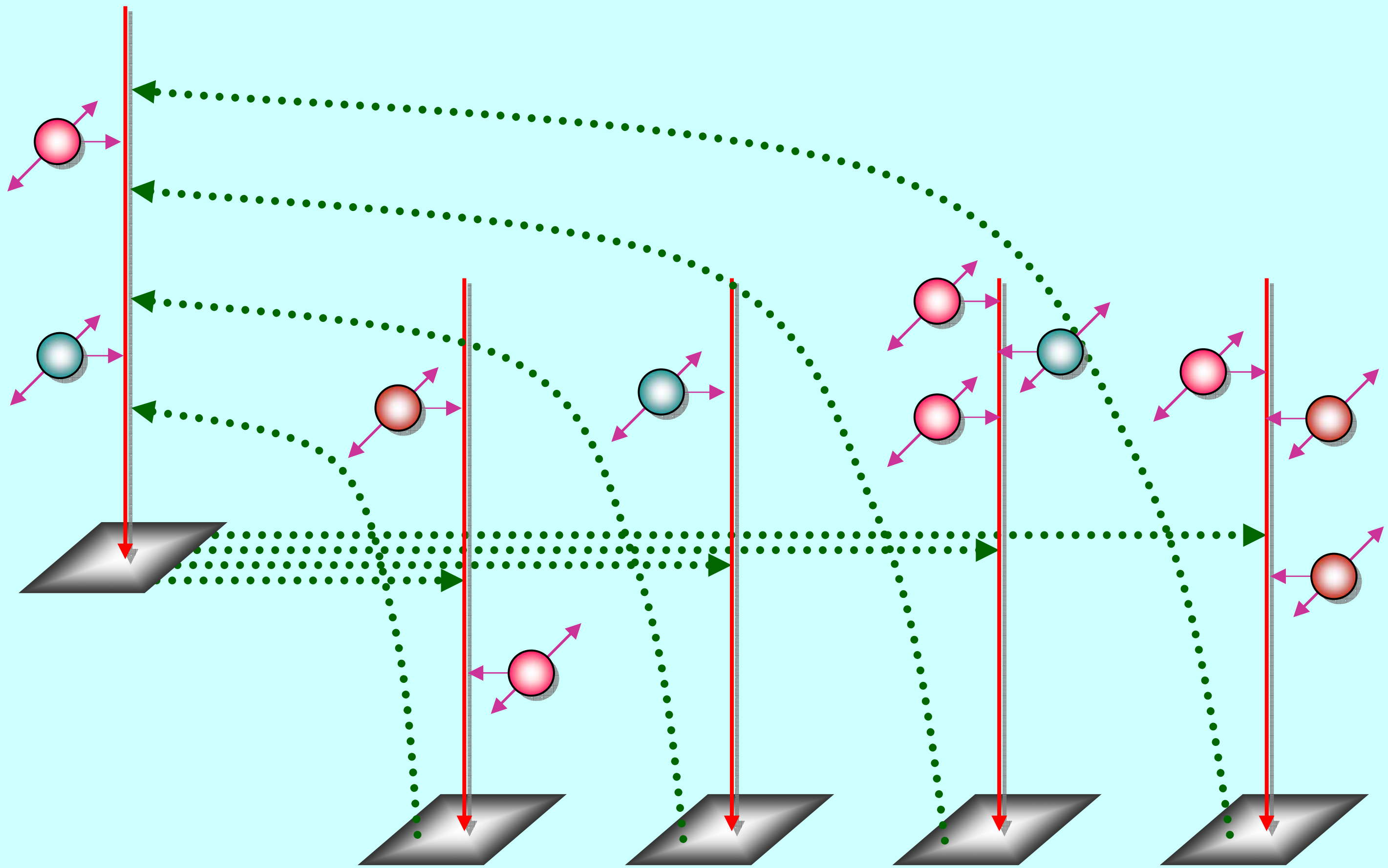


(d) n processes enrolled on a shared barrier (any process synchronising must wait for all to synchronise).

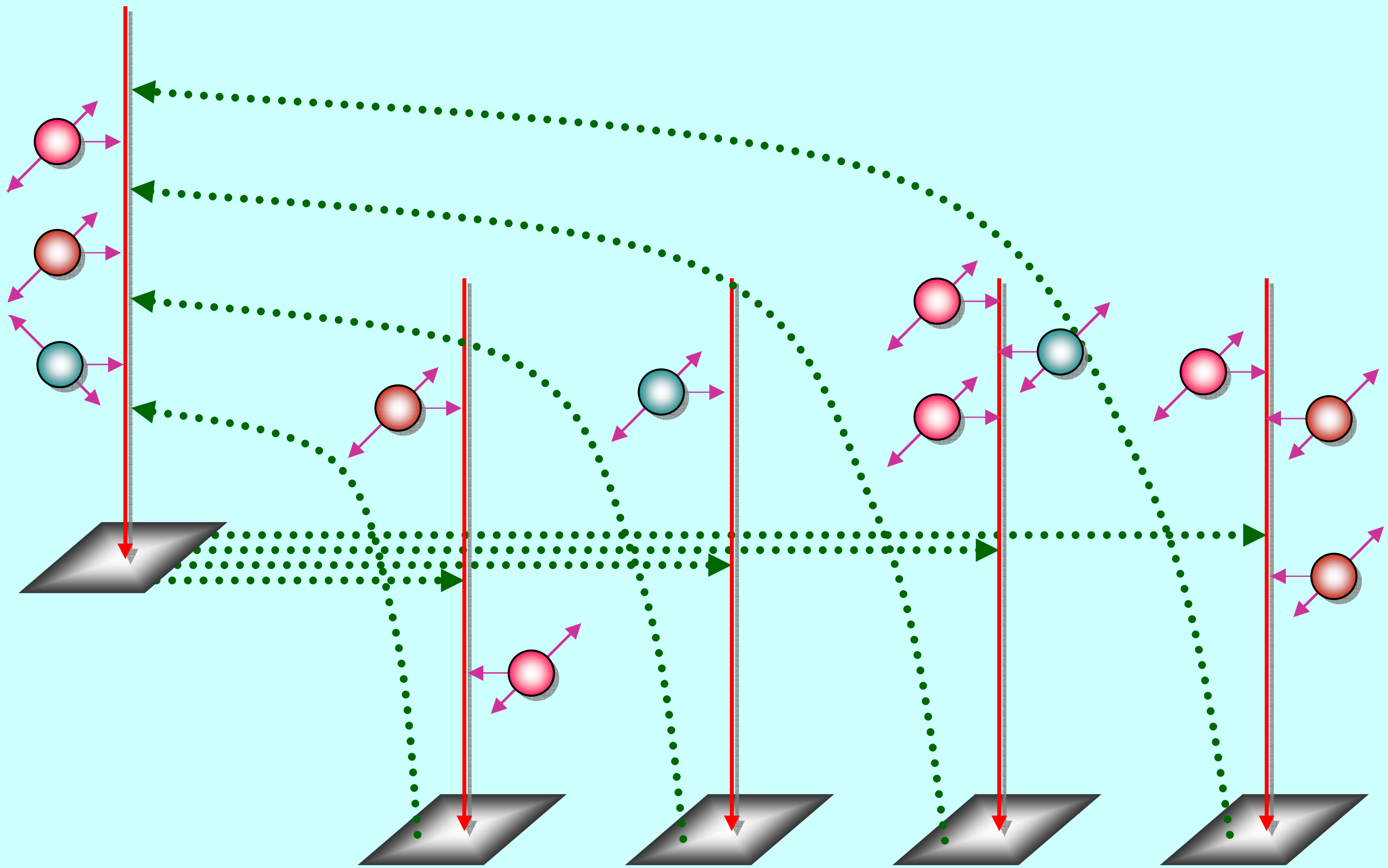
Location (Neighbourhood) Awareness



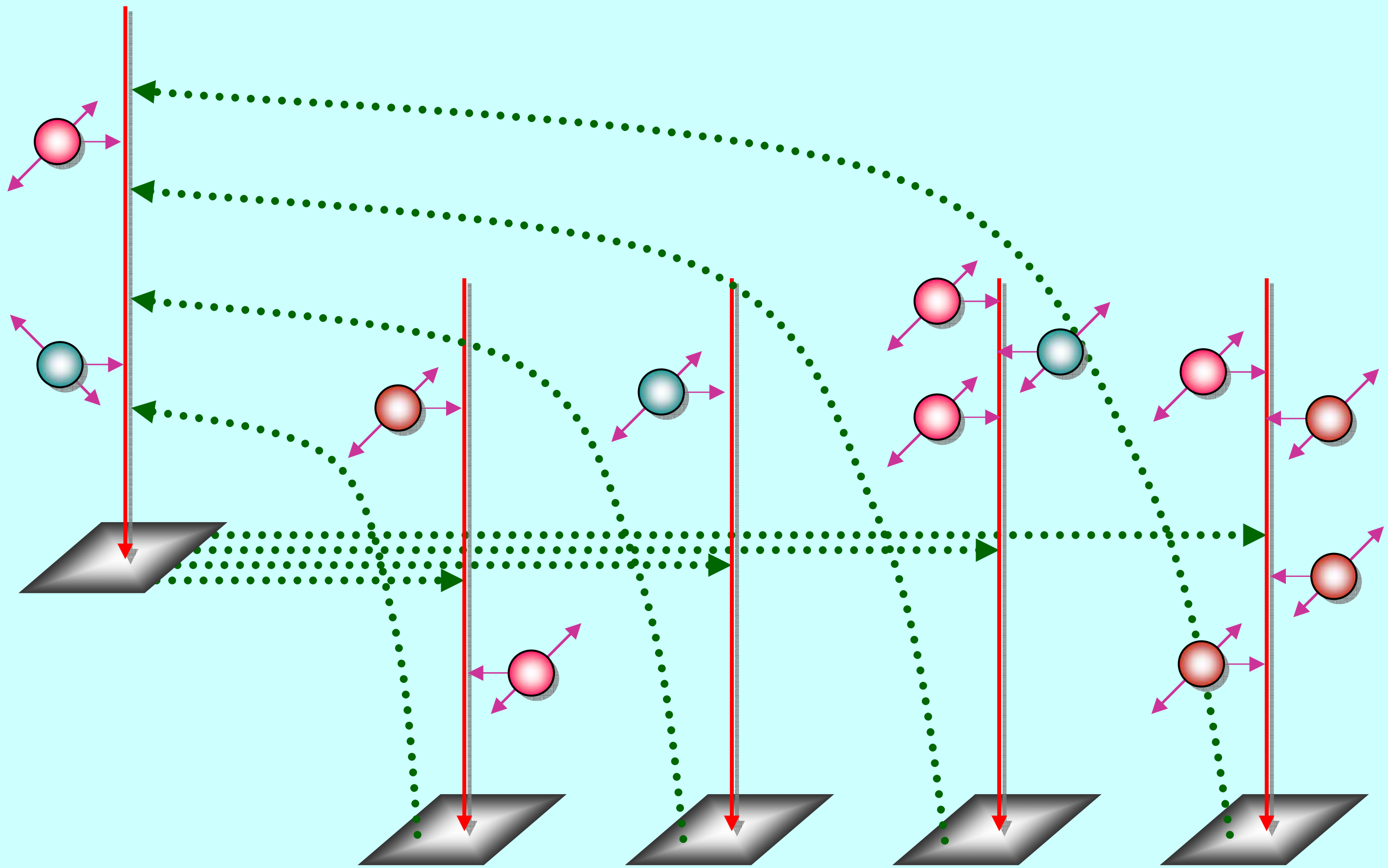
Location (Neighbourhood) Awareness



Location (Neighbourhood) Awareness



Location (Neighbourhood) Awareness



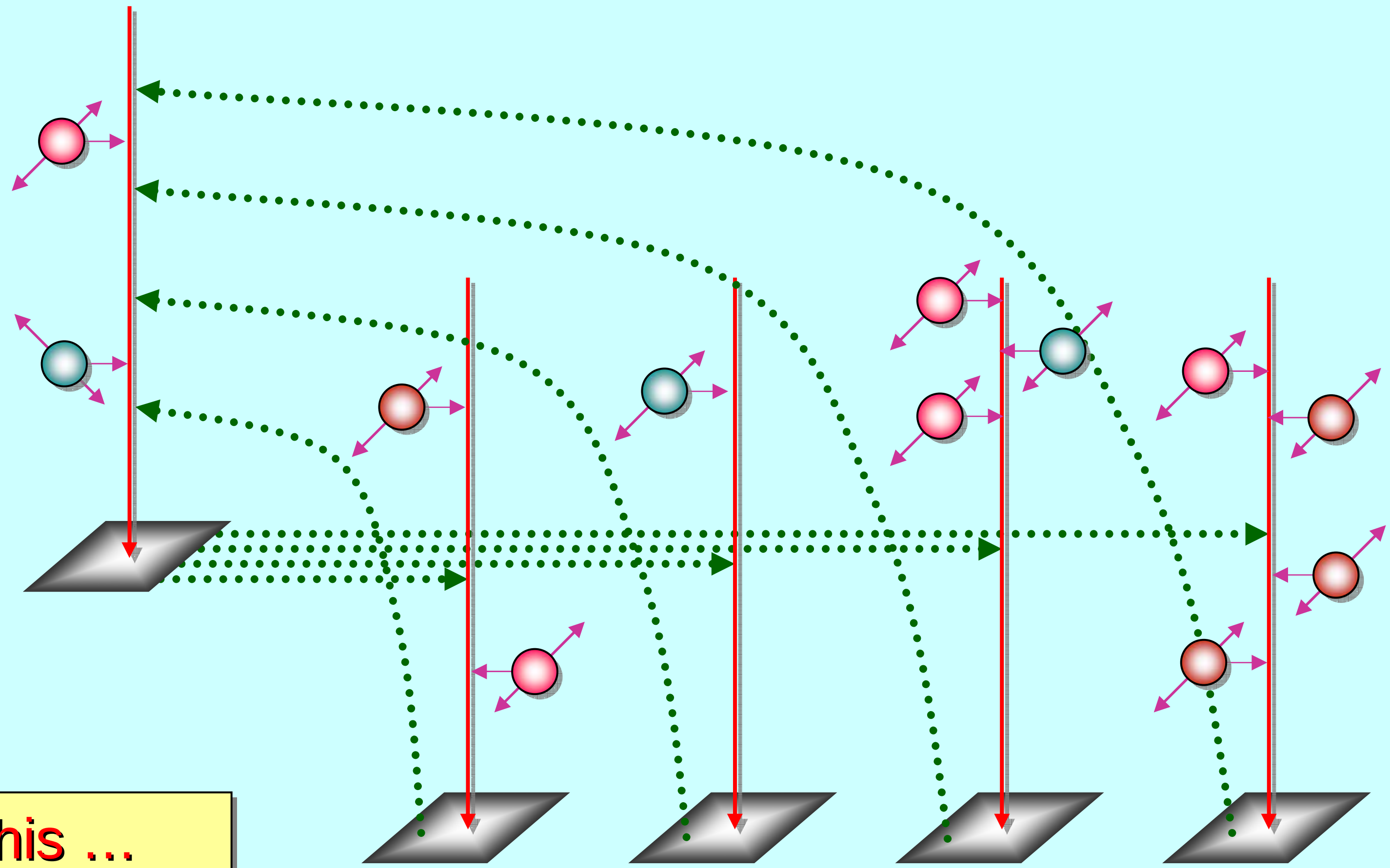
occam-π Boids Model

- Each **server** is responsible for its own region of space …

- A region may hold many **birds** … or none …

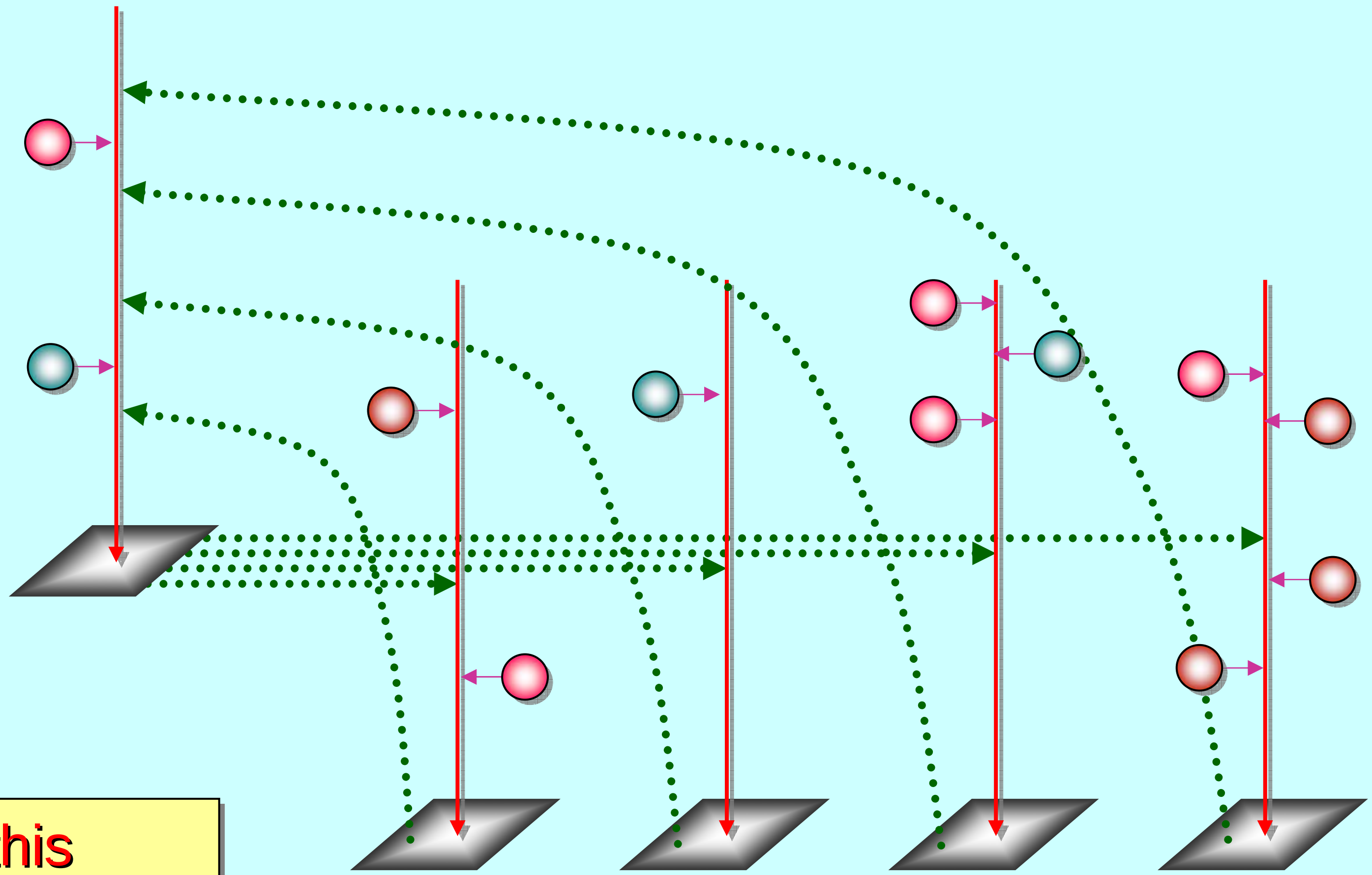
- Each **bird** is in only one region at a time … but can consult with its immediately neighbouring regions …

Location (Neighbourhood) Awareness

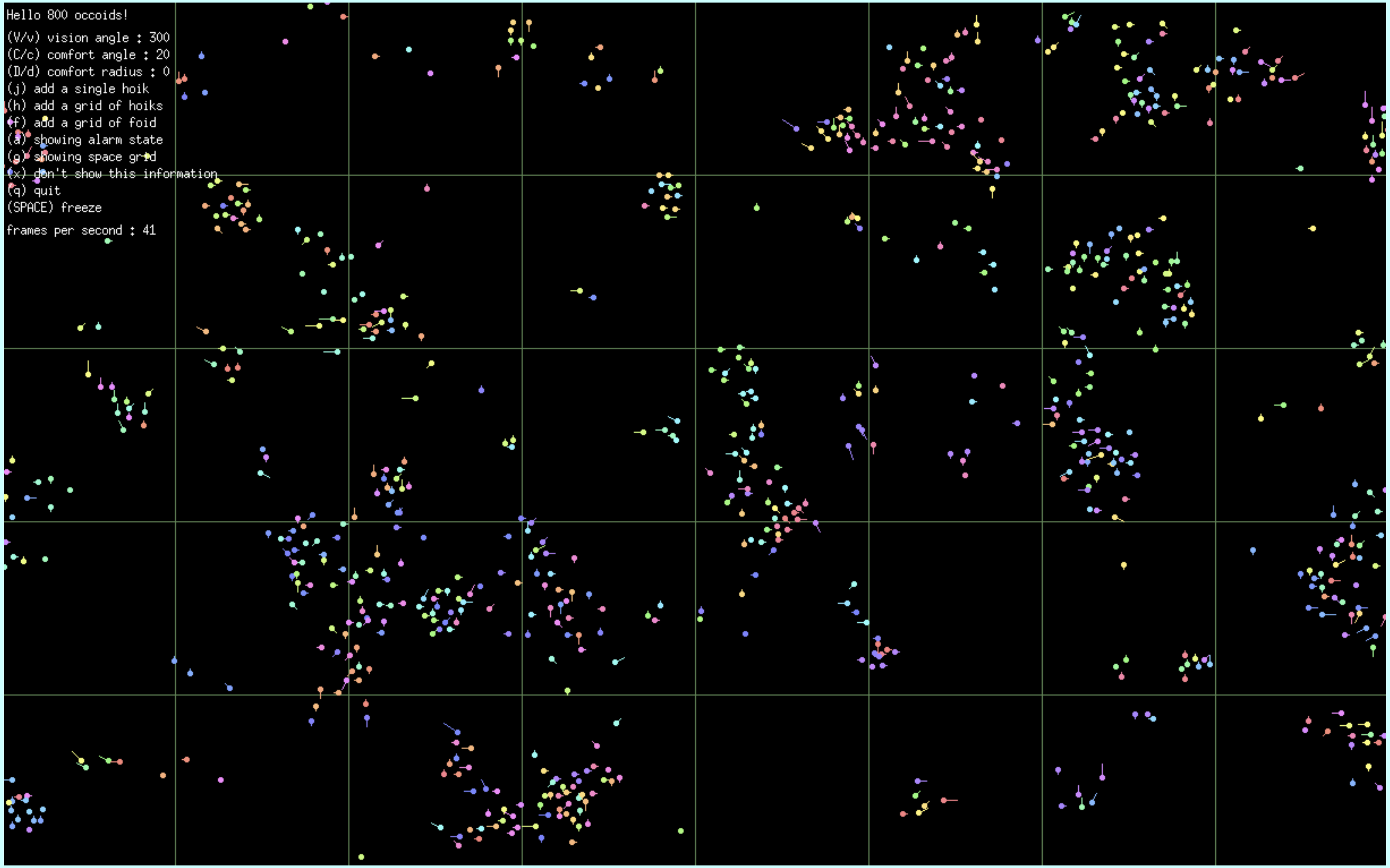


So, not this …

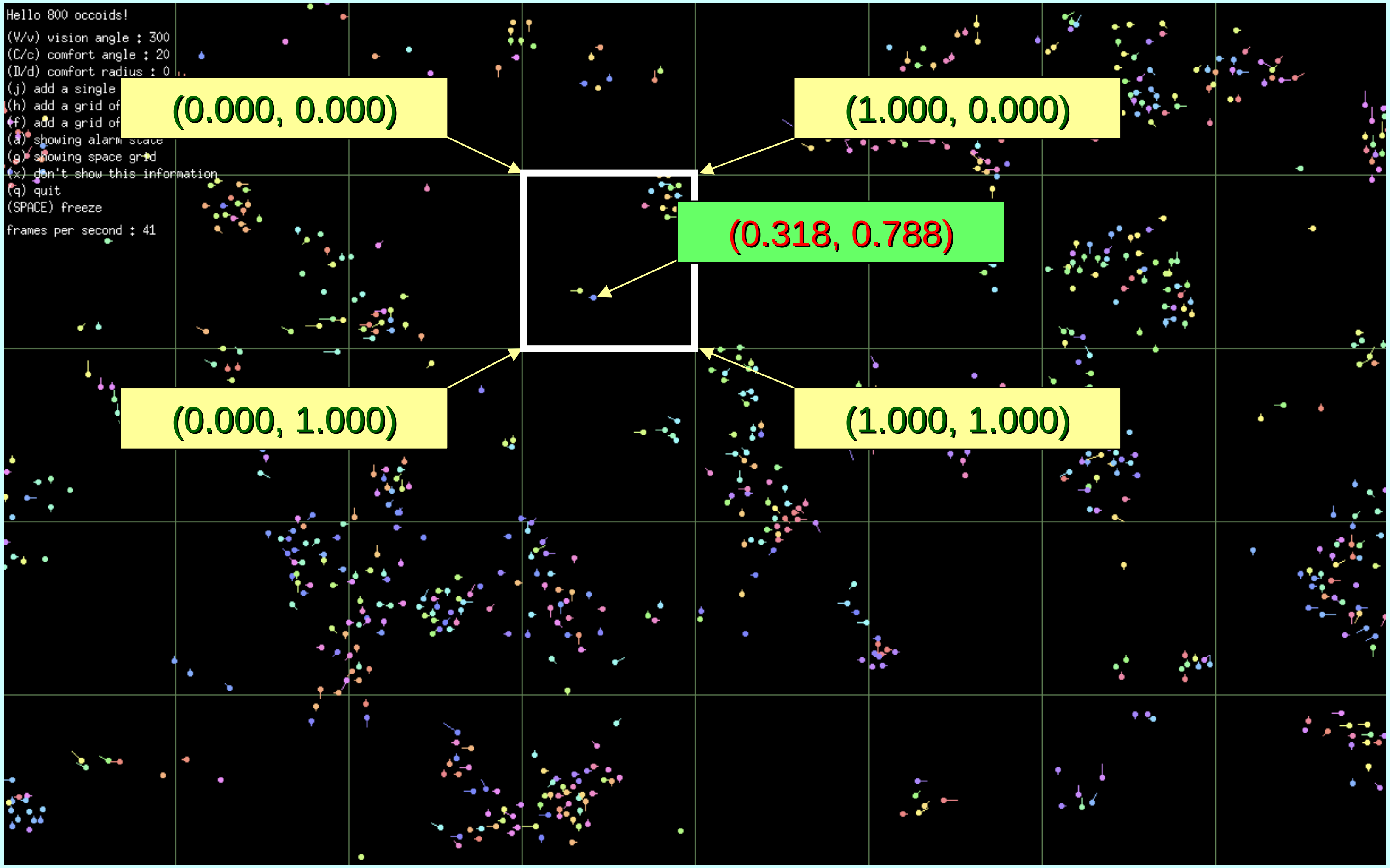
Location (Neighbourhood) Awareness



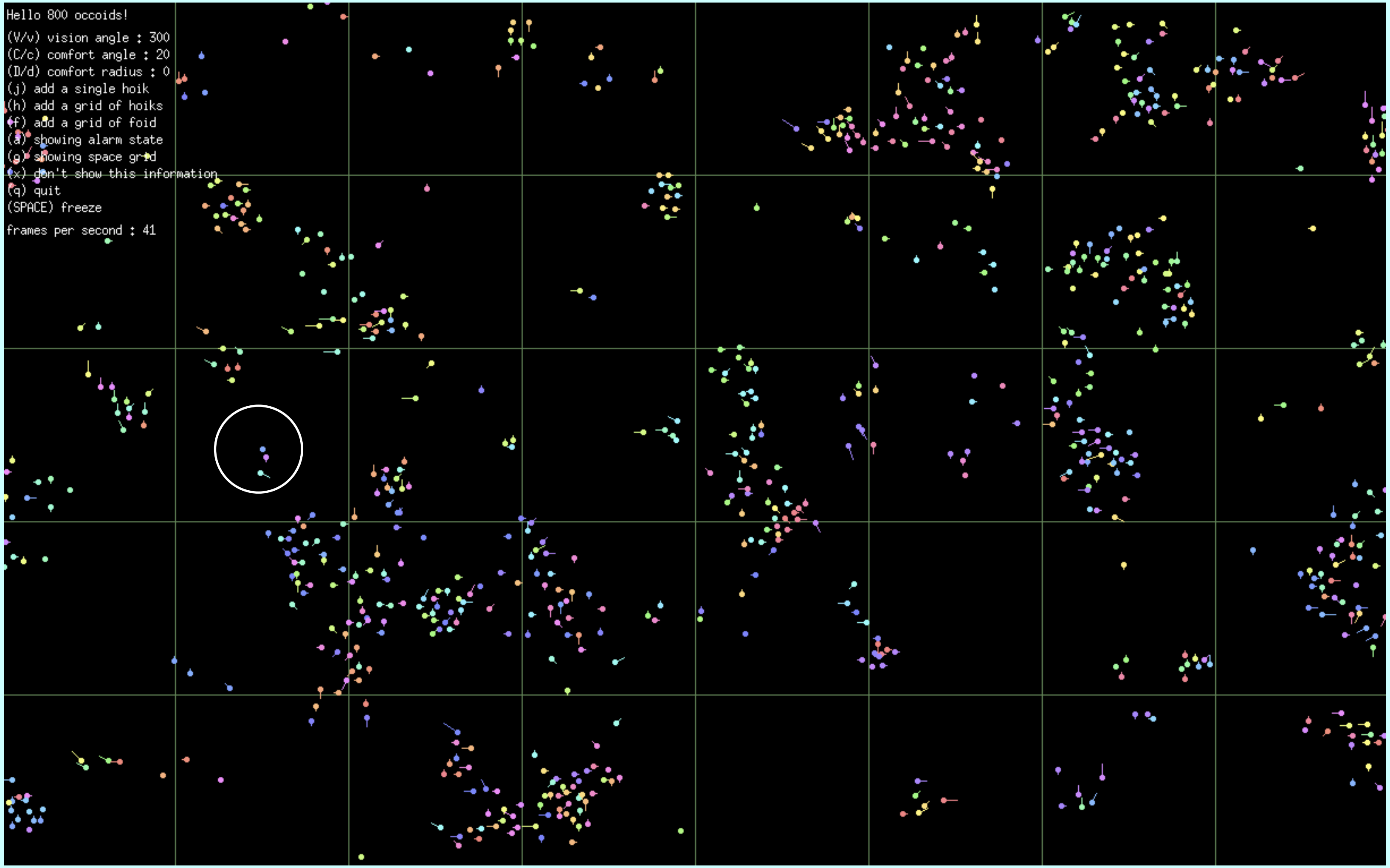
… but this



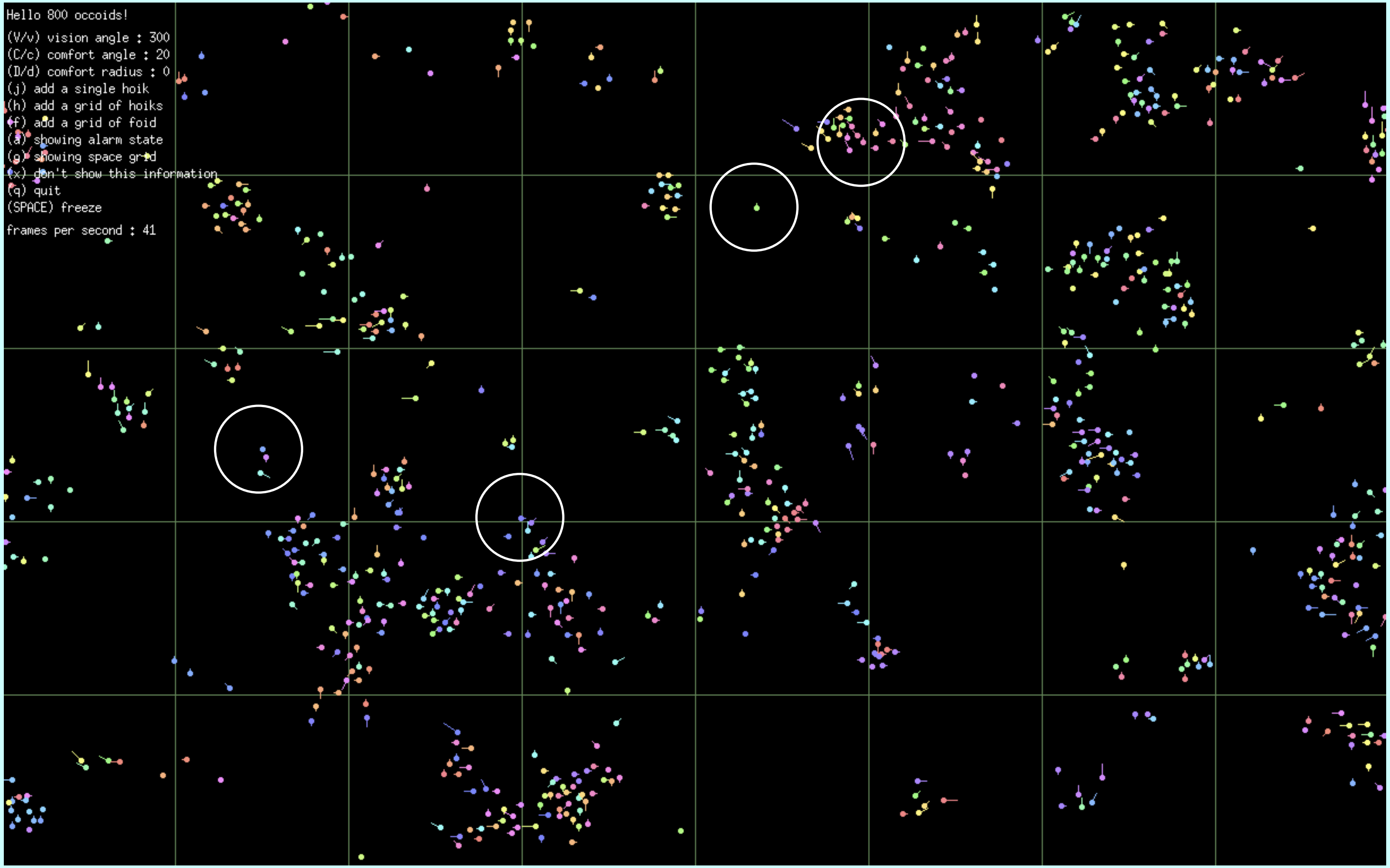
- **Each bird registers its state (position, vector, alarm state, colour, etc.) to the server for its region …**



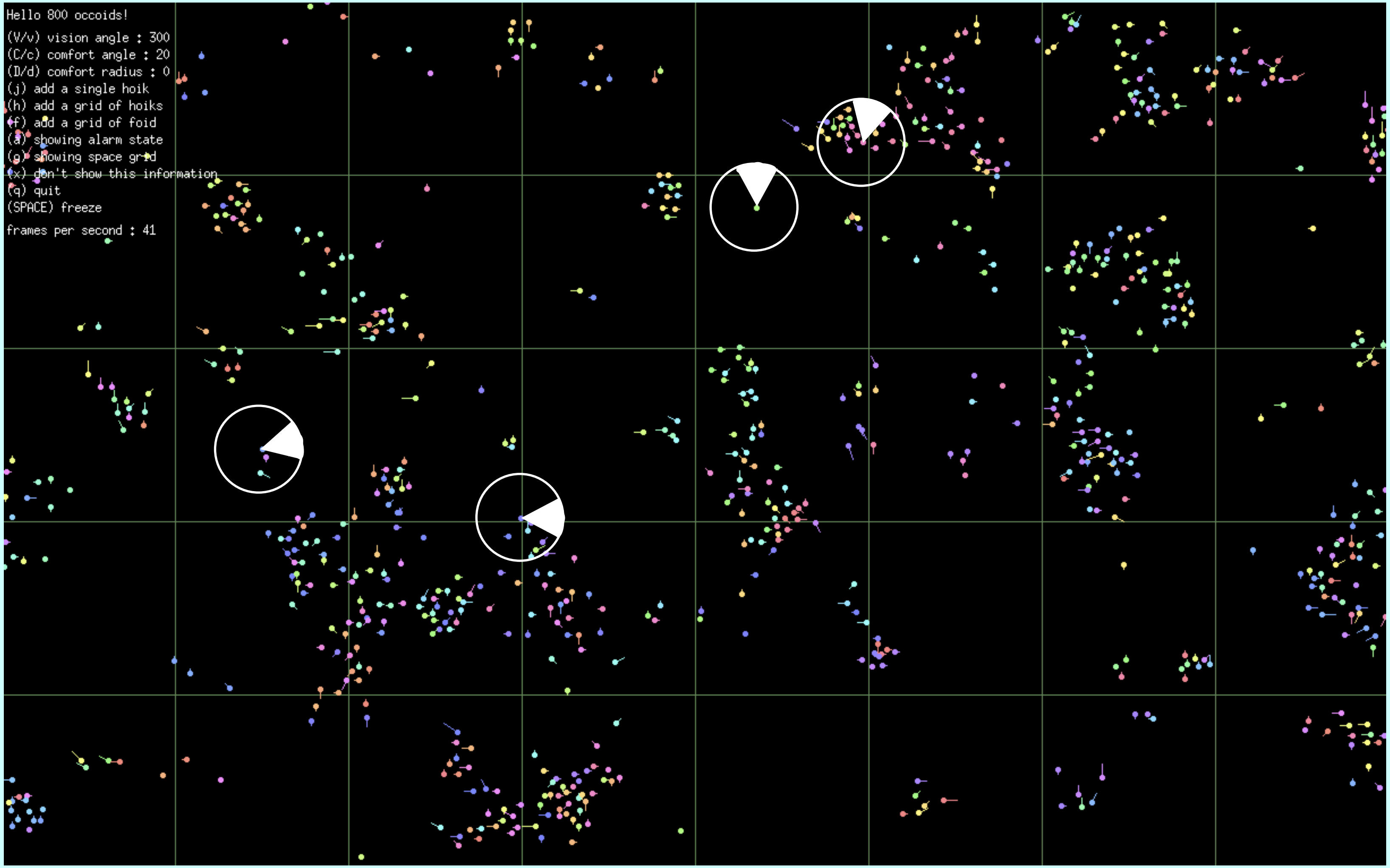
- **Each bird knows its position relative to its current region of space – it doesn't know which region that is …**



- **Birds have a maximum range of vision (up to a radius of 1) …**



- **Birds have a maximum range of vision (up to a radius of 1) … so may need to consult up to 4 servers …**



- **Birds also have a restricted angle of vision … in this case to 300° (i.e. missing 60° rear view) …**

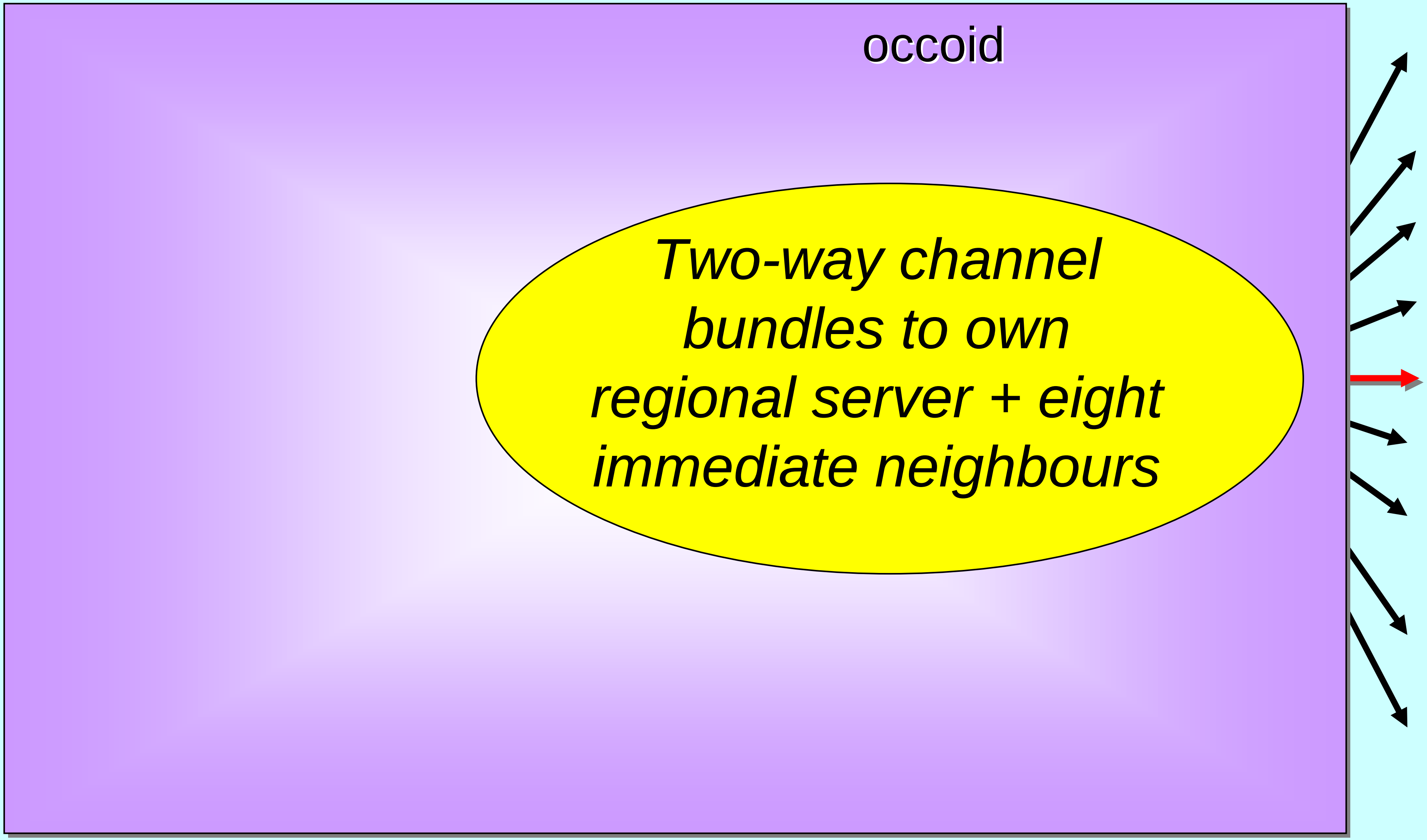
occam-π Boids Model

- A **bird** process follows a general pattern for mobile agents …

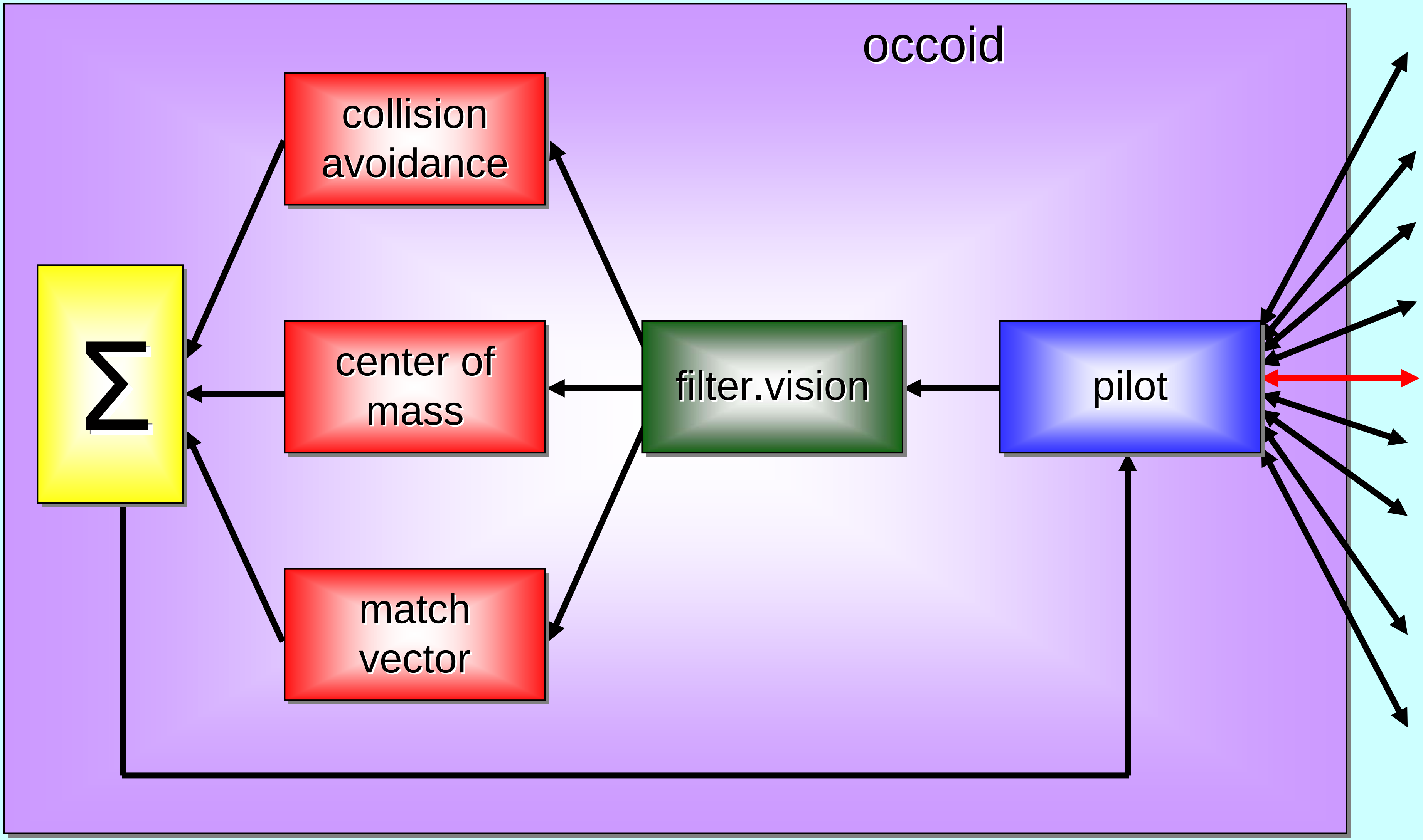
- It has a **pilot** sub-process, responsible for dealing with the **servers** in its immediate neighbourhood and, when necessary, moving between them. The **pilot** is the eyes and wings of the **bird** …

- It has **brain** sub-processes, receiving vision information from the **pilot** and computing wing muscle forces back to the **pilot** …

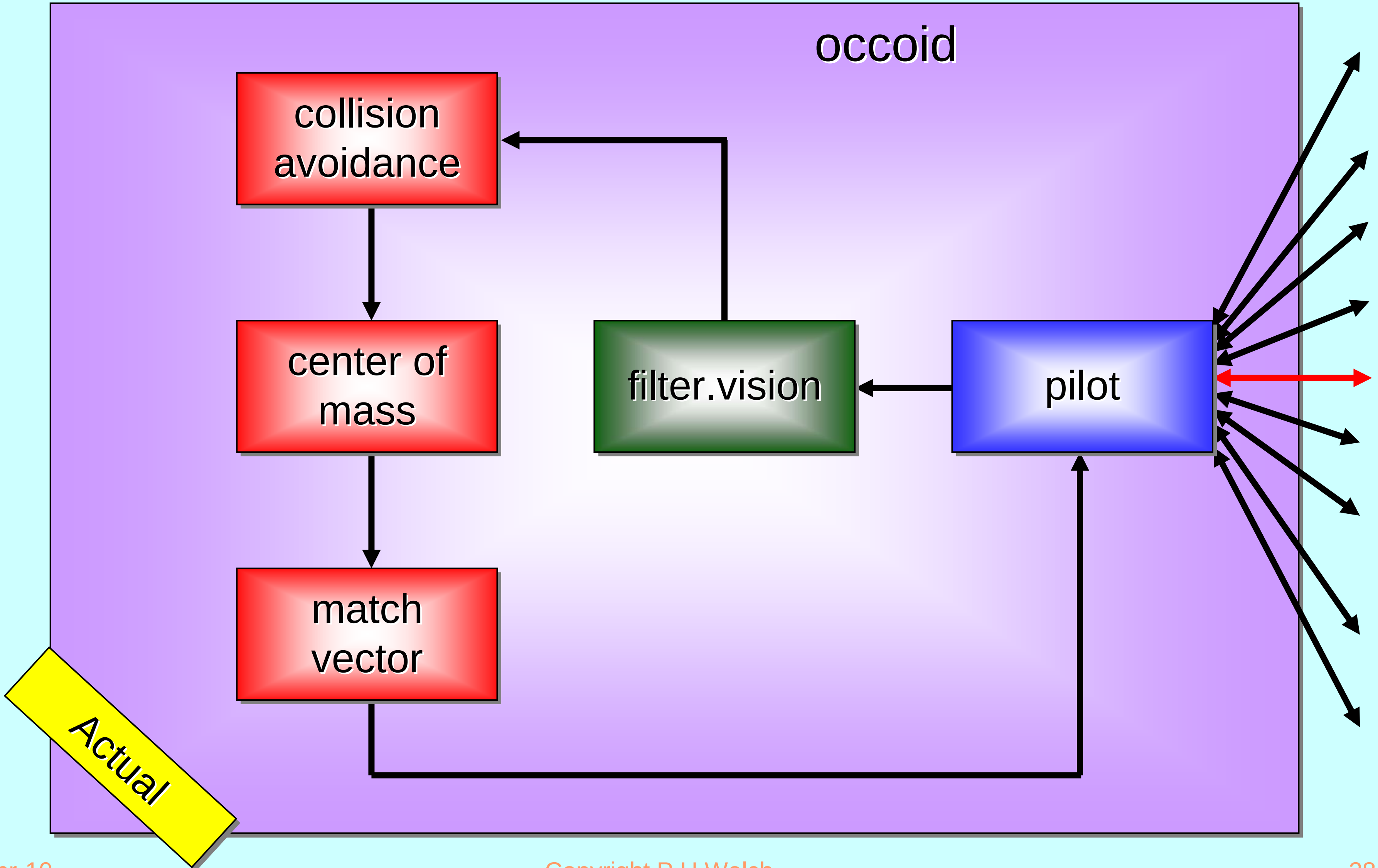
occam-π Boids Model



occam-π Boids Model



occam-π Boids Model



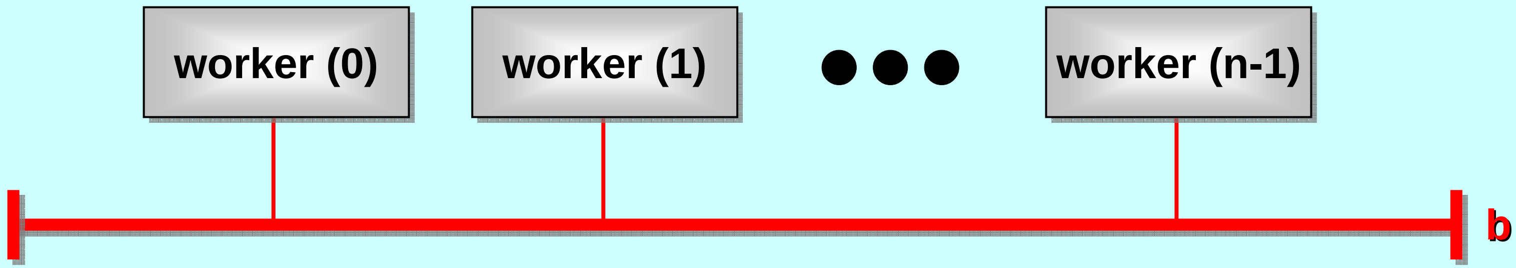
occam-π Boids Model

- A bird process follows a general pattern for mobile agents …

- The birds are kept in step with each other (and with a visual renderer process) by *barrier syncs* … which also provides a model of time. The pilot process does this …

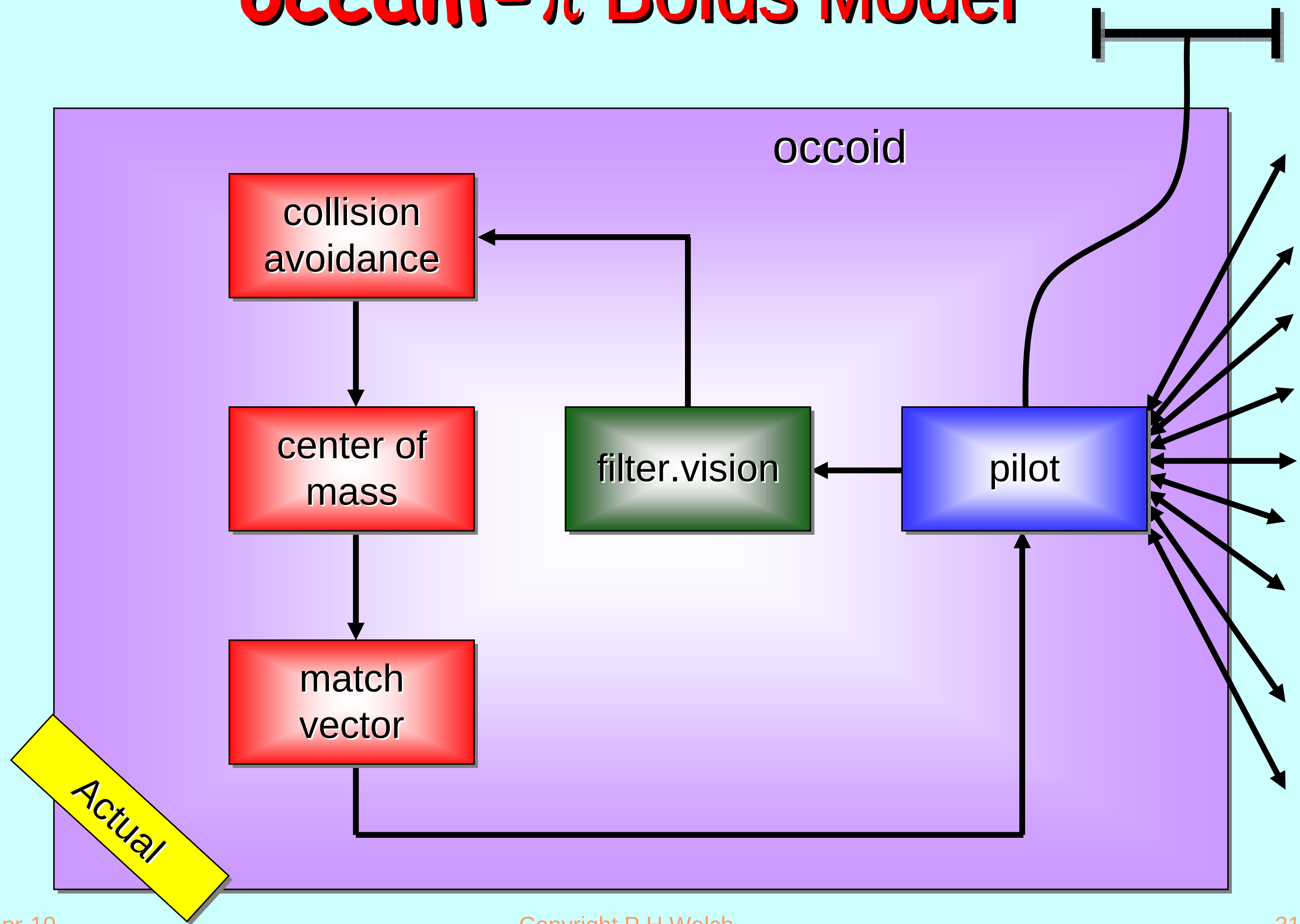
Barrier Synchronisation

The occam-π **BARRIER** type corresponds to a multiway CSP ***event***, though some higher level design patterns (such as *resignation*) have been built in.



Basic CSP semantics apply. When a process *synchronises* on a barrier, it blocks until *all* other processes *enrolled* on the barrier have also *synchronised*. Once the barrier has completed (i.e. all *enrolled* processes have *synchronised*), all blocked processes are rescheduled for execution.

occam-π Boids Model



occam-π Boids Model

- A **bird** process follows a general pattern for mobile agents …

- The **birds** are kept in step with each other (and with a **visual renderer** process) by *barrier syncs* … which also provides a model of time. The **pilot** process does this …

```
WHILE alive
  SEQ
    SYNC tick
    ...  observe local neighbourhood
    SYNC tick
    ...  change local neighbourhood
```

all see a consistent view

and, possibly, move

occam-π Boids Model

- A regional server process holds a dynamic array of all visiting birds …

- It supplies this information to all observers: the birds, the process doing the rendering … and, in future, live hawks, food, etc.

- These server processes do not sync on the barrier … they have no need keep note of time … or keep in step with the birds.

To Boldly Go: an **occam-π** Mission

A thesis, *boids* and a demo …

Process architecture and *boids* …

Observations of emergence…

Summary and Conclusions …

Engineering Emergence

Case study – reminder

Boids: avoid collisions, match vector with those of birds is sight, head for the centre of mass of birds in sight, take fright if a ***hoik*** is spotted, be attracted by ***foid***, …

Emergent behaviours: flocking, squabbling, migration waves, panic scattering, orbiting points of attraction *(if only a small group)*, feeding frenzy *(if a large enough flock)*, turbulence, maze solving, …

Engineering Emergence

Case study – reminder

Almost all processes have been described – **(5x800)** bird processes, **(8x5)** regional servers. There are only **4** others (for visual rendering and keyboard input).

Emergent behaviours: flocking, squabbling, migration waves, panic scattering, orbiting points of attraction *(if only a small group)*, feeding frenzy *(if a large enough flock)*, turbulence, maze solving, …

Engineering Emergence

Case study – reminder

There is ***nothing*** in the design or programming dealing with ***flocking***, ***scattering***, ***orbiting***, ***feeding frenzies***, ***migration waves***, ***turbulent flow*** or ***solving mazes!***

Emergent behaviours: flocking, squabbling, migration waves, panic scattering, orbiting points of attraction *(if only a small group)*, feeding frenzy *(if a large enough flock)*, turbulence, maze solving, …

Engineering Emergence

Case study – reminder

We don't like the ***scattering*** … we would prefer the flock to ***maintain cohesion*** when danger is spotted and ***turn-as-one away*** from it … but what are the rules for engineering this behaviour?

There is no concept of flock (for example) in the design … so there is nothing to program directly.

The panic signal propagates fast across a flock … but the birds don't have the right rules for the right response to emerge. Any ideas? ☺ ☺ ☺

Engineering Emergence

Scheduling dyamics – reminder

The network topology changes all the time as the birds move …

The computational loading on each bird and each server varies dynamically and cannot be predicted in advance …

Nevertheless, the occam-pi kernel (CCSP) does a good job of very lightweight load balancing across all the cores (that we have right now!) …

To Boldly Go: an **occam-π** Mission

A thesis, *boids* and a demo …

Process architecture and *boids* …

Observations of emergence…

Summary and Conclusions …

To Boldly Go – Summary

We have described an architecture for the *intentional emergence* of complex systems behaviour.

Processes (*mobile*, *communicating* and *lightweight*) are good candidates for supporting such an architecture. **occam-π** provides this computaional model and scales well across both shared and distributed memory.

Engineering the desired behaviour is *indirect*. We need to discover simple *low-level rules* for pieces that we can program and, then, *run masses* of them. For complex systems, there will be *no high-level components* that directly work the behaviour we want.

To Boldly Go – Summary

It's programming, Jim, but not as we know it …

Scientific Instruments

Scientist / Engineer   Instrument   Complex System

Simulation as a Scientific Instrument *

Scientist / Engineer

Simulation

Complex System

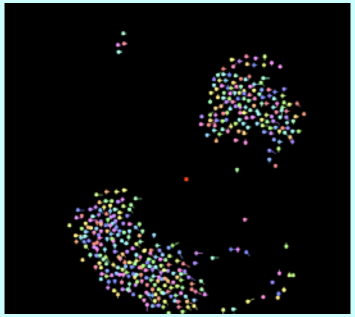
Modelling – Simulation – Validation cycle

Research and discovery of low-level processes from which observed complex behaviours emerge.

* "Simulation as an experimental design process for emergent systems", Andrews-Stepney-Winfield

Discovering Unexpected Relations between Phenomena

Scientist / Engineer



Simulation

Starlings

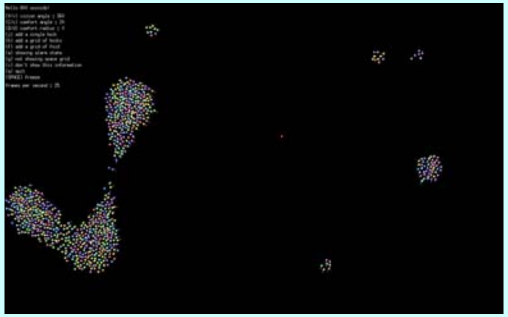
Modelling – Simulation – Validation cycle

Research and discovery of low-level processes from which observed complex behaviours emerge.

Discovering Unexpected Relations between Phenomena

Starlings

Scientist / Engineer

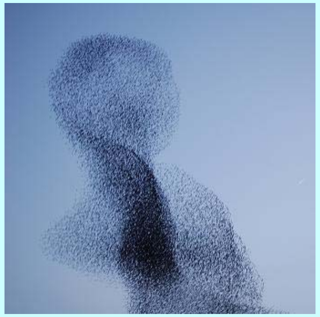


Simulation

Modelling – Simulation – Validation cycle

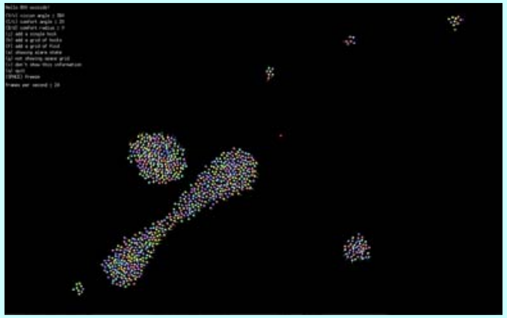
Research and discovery of low-level processes from which observed complex behaviours emerge.

Discovering Unexpected Relations between Phenomena



Starlings

Scientist / Engineer



Simulation

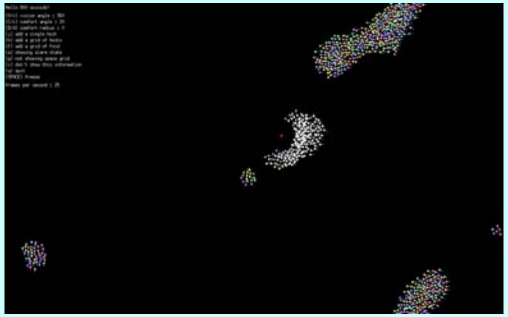
Modelling – Simulation – Validation cycle

Research and discovery of low-level processes from which observed complex behaviours emerge.

Discovering Unexpected Relations between Phenomena

Starlings and Hawk

Scientist / Engineer



Simulation

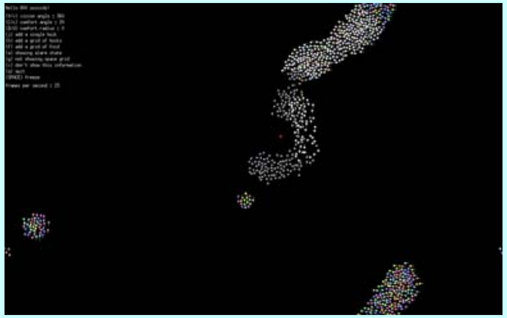
Modelling – Simulation – Validation cycle

Research and discovery of low-level processes from which observed complex behaviours emerge.

Discovering Unexpected Relations between Phenomena

Starlings and Hawk

Scientist / Engineer



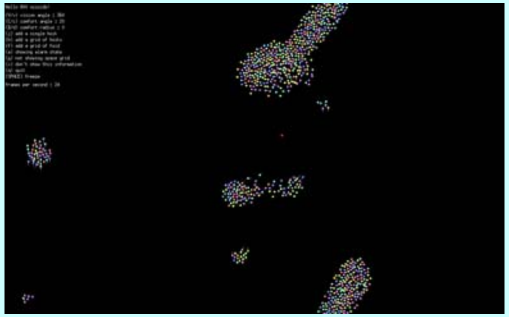
Simulation

Modelling – Simulation – Validation cycle

Research and discovery of low-level processes from which observed complex behaviours emerge.

Discovering Unexpected Relations between Phenomena

Starlings and Hawk



Simulation

Scientist / Engineer

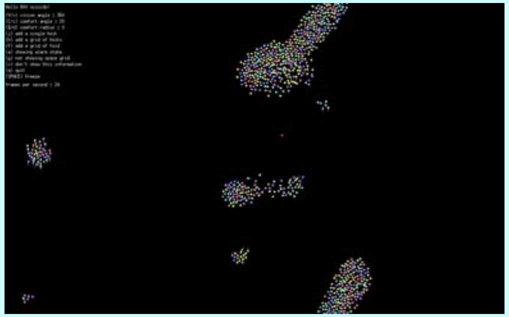
Modelling – Simulation – Validation cycle

Research and discovery of low-level processes from which observed complex behaviours emerge.

Discovering Unexpected Relations between Phenomena

Starlings and Hawk

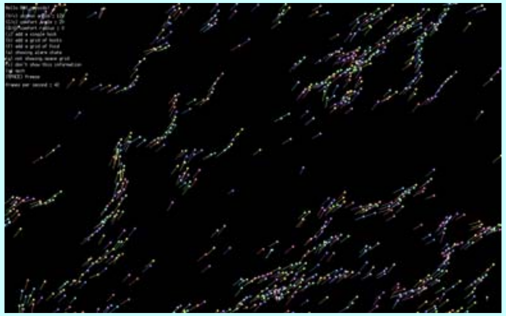
Scientist / Engineer



Simulation

Computer modelling and simulaton can show unexpected relationships between apparently different complex phenomena, operating with different physics and at different scales …

Discovering Unexpected Relations between Phenomena



Starlings and Hawk

Scientist / Engineer

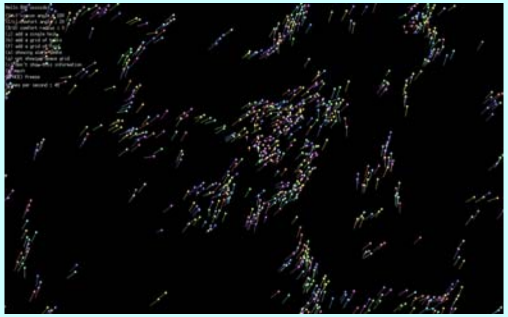
Simulation

Turbulent Flow

Computer modelling and simulaton can show unexpected relationships between apparently different complex phenomena, operating with different physics and at different scales …

Discovering Unexpected Relations between Phenomena

Scientist / Engineer



Simulation

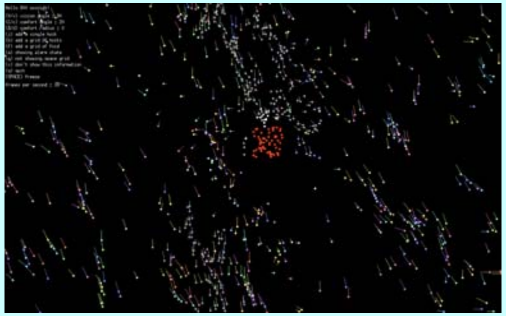
Starlings and Hawk

Turbulent Flow

Computer modelling and simulaton can show unexpected relationships between apparently different complex phenomena, operating with different physics and at different scales …

Discovering Unexpected Relations between Phenomena

Scientist / Engineer



Simulation

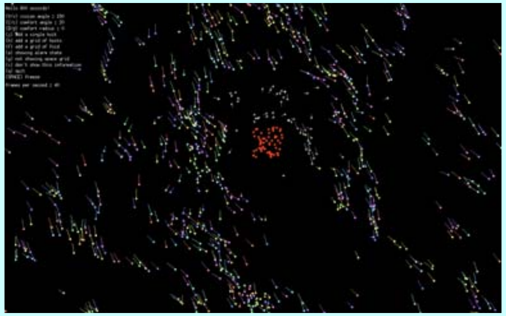
Starlings and Hawk

Turbulent Flow

Computer modelling and simulaton can show unexpected relationships between apparently different complex phenomena, operating with different physics and at different scales …

Discovering Unexpected Relations between Phenomena

Scientist / Engineer



Simulation

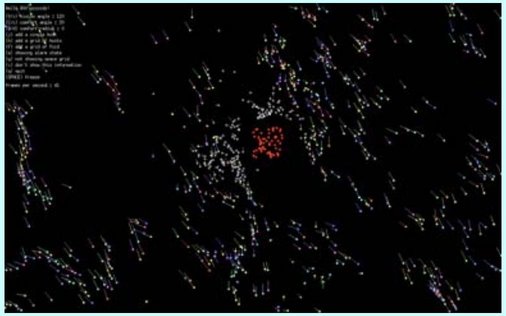
Starlings and Hawk

Turbulent Flow

Computer modelling and simulaton can show unexpected relationships between apparently different complex phenomena, operating with different physics and at different scales …

Discovering Unexpected Relations between Phenomena

Scientist / Engineer



Simulation

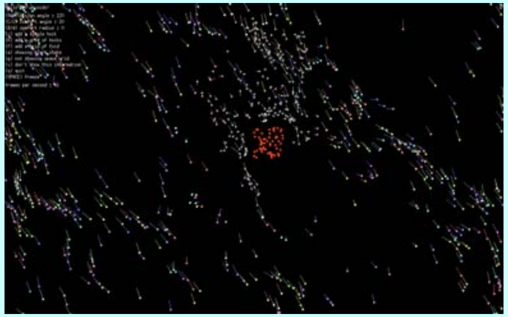
Starlings and Hawk

Turbulent Flow

Computer modelling and simulaton can show unexpected relationships between apparently different complex phenomena, operating with different physics and at different scales … *because their (differing) behavours emerge from agents following identical low-level rules, just with slightly different key parameters …*

Discovering Unexpected Relations between Phenomena

Scientist / Engineer



Simulation

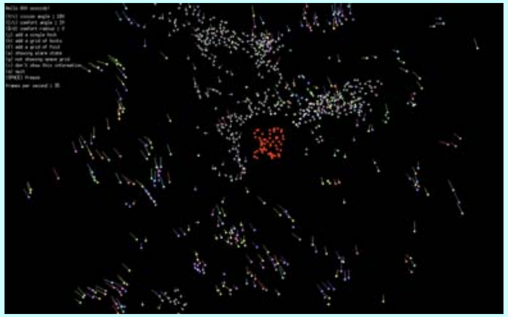
Starlings and Hawk

Turbulent Flow

Computer modelling and simulaton can show unexpected relationships between apparently different complex phenomena, operating with different physics and at different scales … *because their (differing) behavours emerge from agents following identical low-level rules, just with slightly different key parameters …*

Discovering Unexpected Relations between Phenomena

Scientist / Engineer



Simulation

Starlings and Hawk

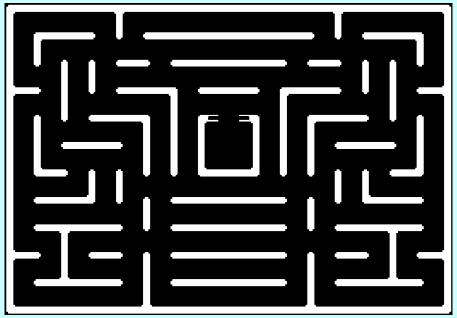
Turbulent Flow

Computer modelling and simulaton can show unexpected relationships between apparently different complex phenomena, operating with different physics and at different scales … *because their (differing) behavours emerge from agents following identical low-level rules, just with slightly different key parameters …*

Discovering and Experimenting with New Physics

???

Scientist / Engineer



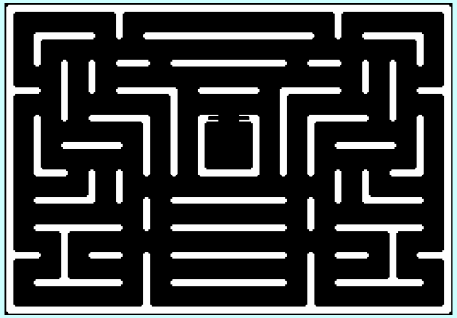
Simulation

???

Through computer modelling and simulaton, we can investigate the emergent properties of whole new worlds of materials, new states of physics, by experimenting with varieties of agent programmed with simple low-level rules.

Discovering and Experimenting with New Physics

Free-flow Traffic



???

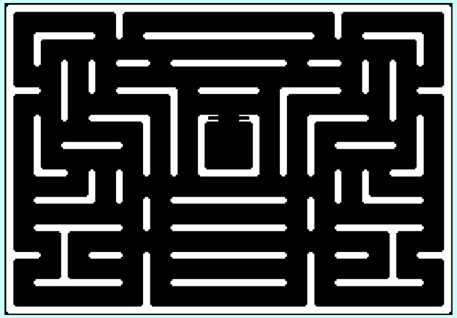
Scientist / Engineer

Simulation

Through computer modelling and simulaton, we can investigate the emergent properties of whole new worlds of materials, new states of physics, by experimenting with varieties of agent programmed with simple low-level rules.

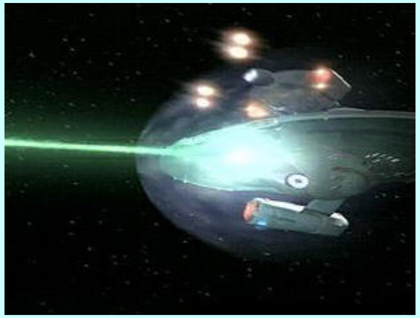
Discovering and Experimenting with New Physics

Scientist / Engineer



Simulation

Free-flow Traffic



Intelligent Plasma

Through computer modelling and simulaton, we can investigate the emergent properties of whole new worlds of materials, new states of physics, by experimenting with varieties of agent programmed with simple low-level rules. *Some of these may turn out to be interesting and useful … so that we might be motivated to find ways to build those agents for real!*

To Boldy Go – Summary

Research projects

cosmos-research.org
occam-pi.org
concurrency.cc
rmox.net

occam-pi course @ Kent

moodle.kent.ac.uk/external/course/view.php?id=31

To Boldy Go – Summary

Once more,
and this time with
feeling …

To Boldy Go – Summary

Future?

Drug design: try to build molecules with certain shapes *(to match the geometry of suspected weak spots of rogue cells)* …

Emergent behaviours: elimination *(or inhibition)* of tumours.

Autonomous driving: avoid collisions, head for the longest straight clear path *(with speed in proportion)*, add bias in general favour of destination *(if known)* …

Emergent behaviours: safe driving, efficient use of the road, faster completion of journey.


Any Questions?
COSTA