KES AMSTA 2012

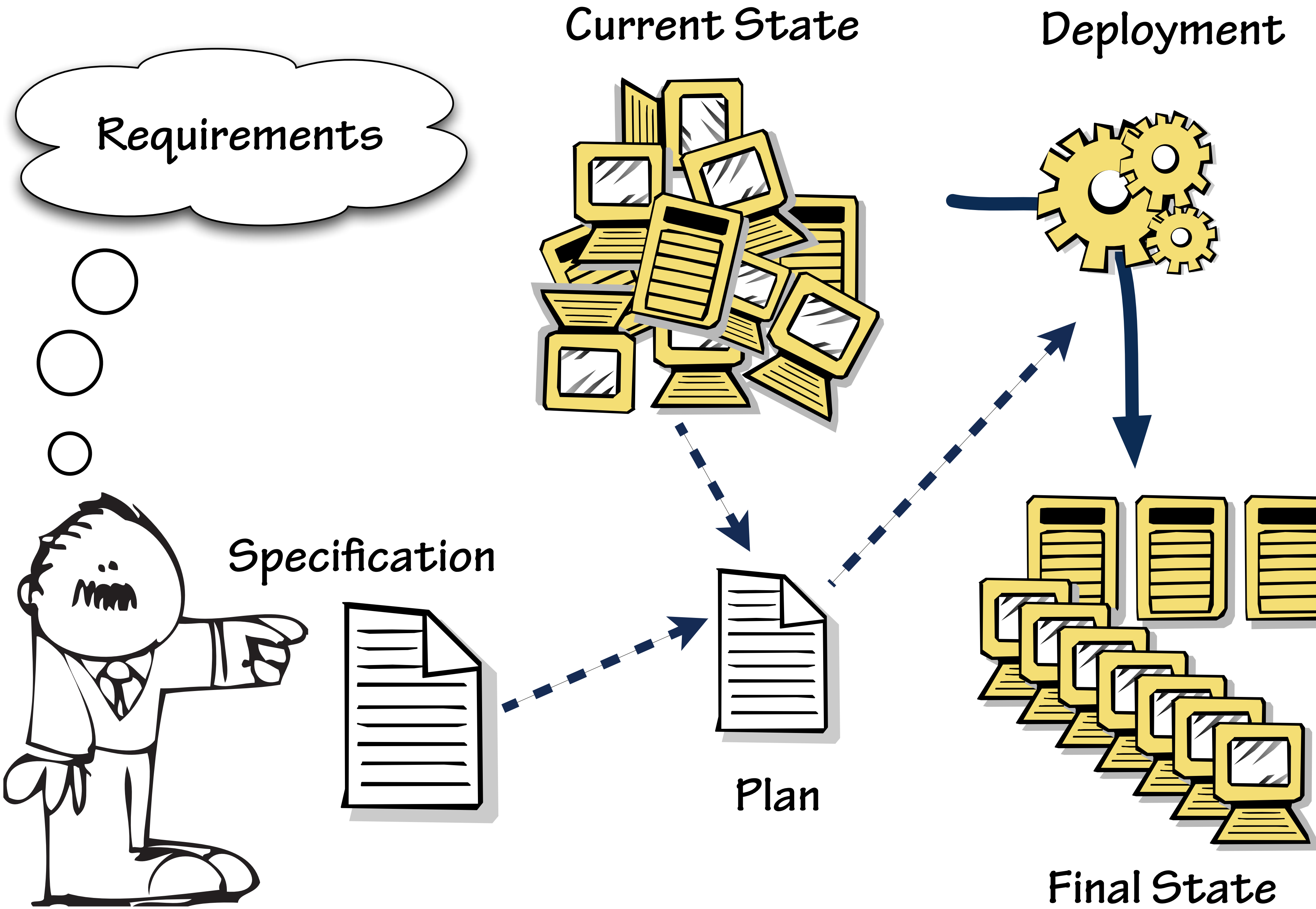
Multi-Agent Negotiation of Virtual Machine Migration

Using the Lightweight Coordination Calculus

Paul Anderson <dcspaul@ed.ac.uk>
Shahriar Bijani <S.Bijani@sms.ed.ac.uk>
Alexandros Vichos

THE UNIVERSITY of EDINBURGH
informatics

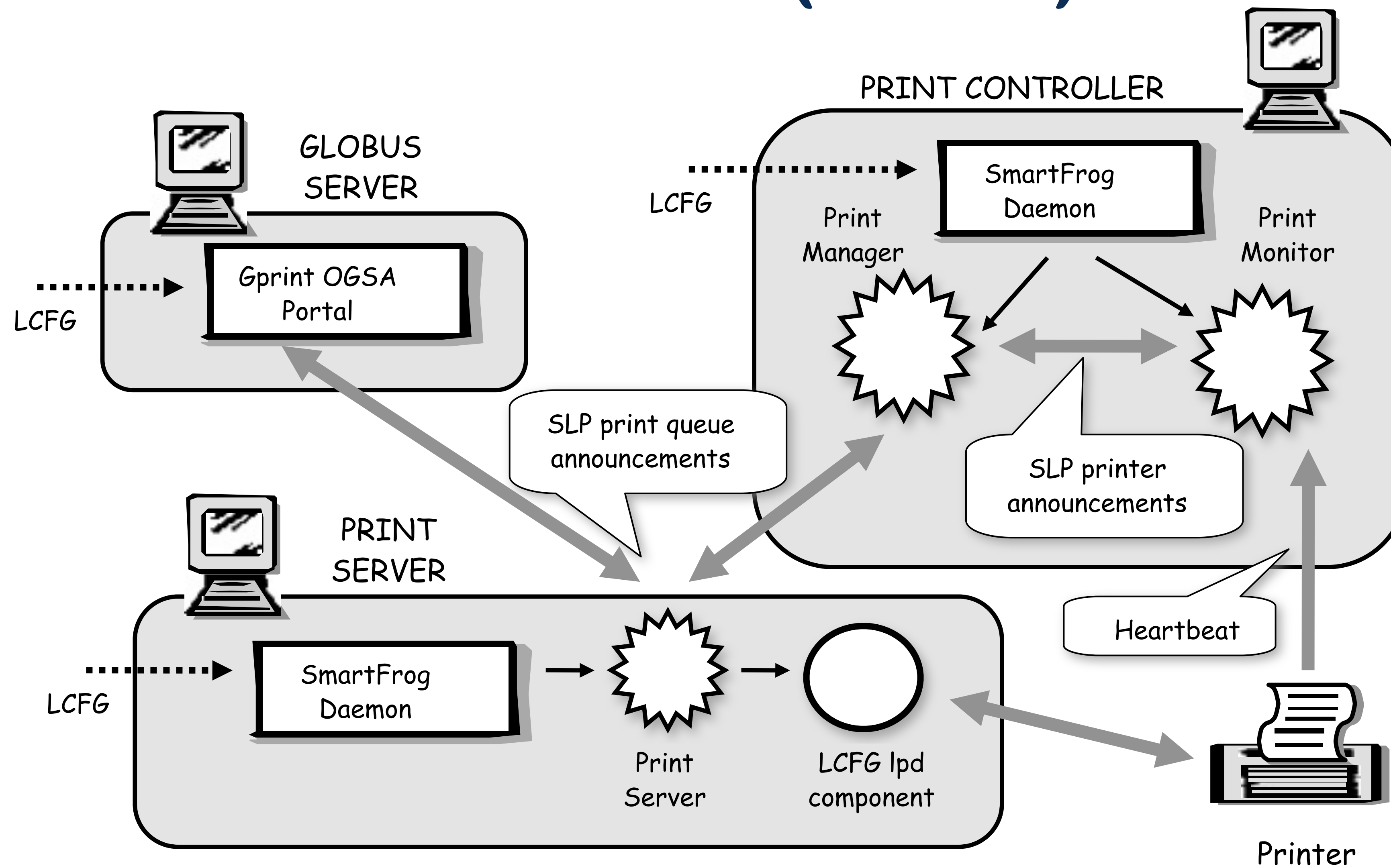
cisa
Centre for Intelligent Systems and their Applications


Requirements
Current State
Deployment
Specification
Plan
Final State

Centralised Configuration?

- Centralised configuration
  - allows a global view with complete knowledge
- But ...
  - it is not scalable
  - it is not robust against communication failures
  - federated environments have no obvious centre
  - different security policies may apply to different subsystems
- The challenge ...
  - devolve control to an appropriately low level
  - but allow high-level policies to determine the behaviour

GPrint (2003)



- Distributed configuration with centralised policy
- Subsystem-specific mechanisms

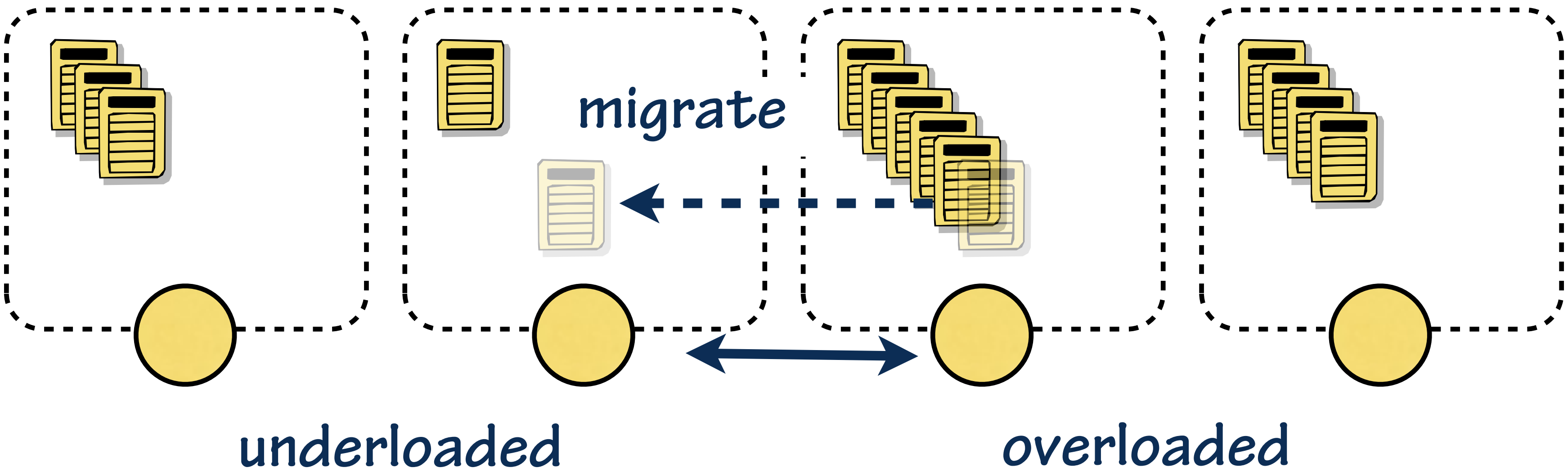
“OpenKnowledge” & LCC

- Agents execute “interaction models”
- Written in a “lightweight coordination calculus” (LCC)
- This provides a very general mechanism for doing distributed configuration
- Policy is determined by the interaction models themselves which can be managed and distributed from a central point of control
- The choice of interaction model and the decision to participate in a particular “role” remains with the individual peer
  - and hence, the management authority

A Simple LCC Example

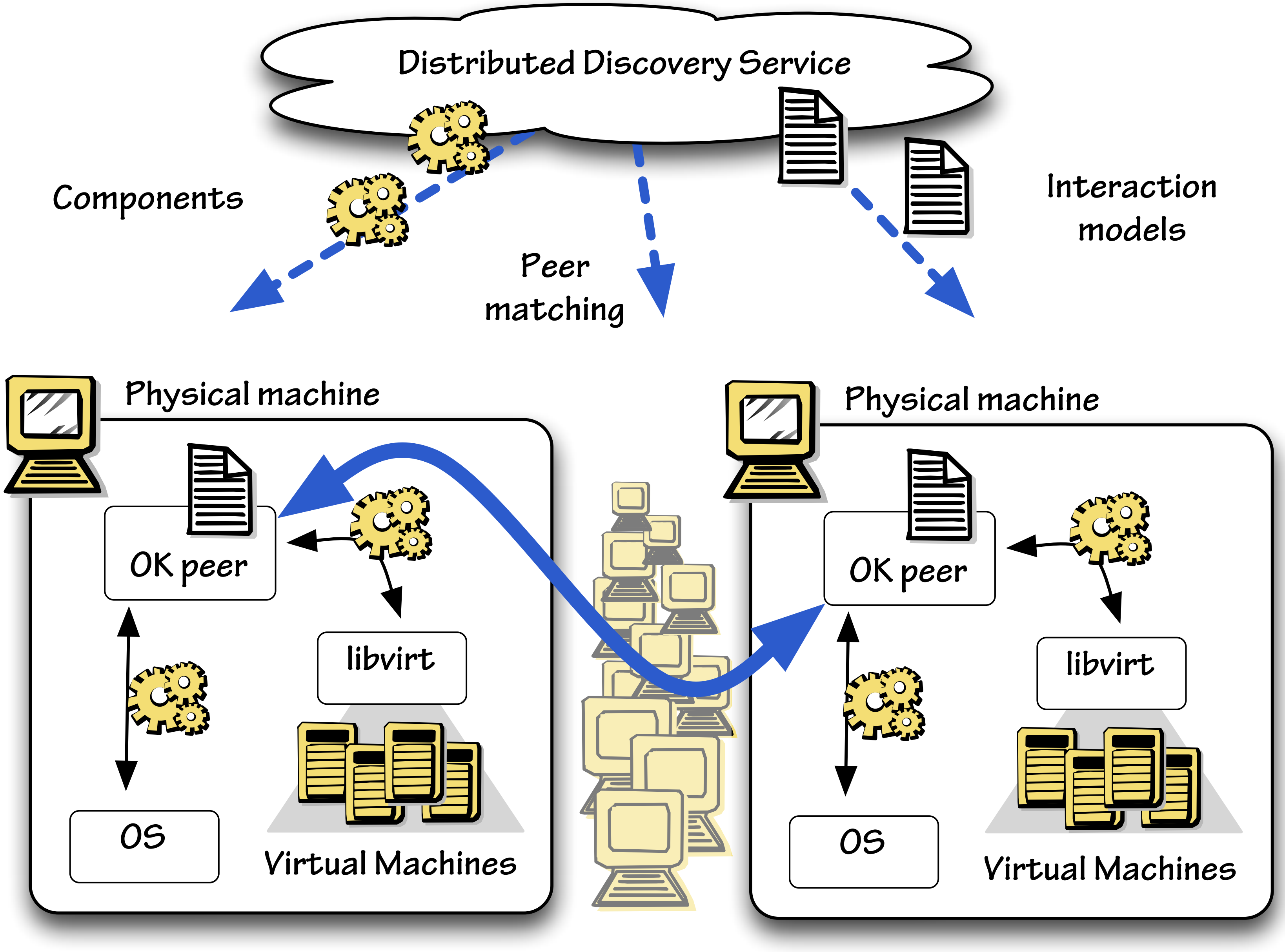
```
a(buyer, B) ::
 ask(X) => a(shopkeeper, S) then
 price(X,P) <= a(shopkeeper, S) then
 buy(X,P) => a(shopkeeper, S)
                ← afford(X, P) then
 sold(X,P) <= a(shopkeeper, S)

a(shopkeeper, S) ::
  ask(X) <= a(buyer, B) then
  price(X, P) => a(buyer, B)
                   ← in_stock(X, P)then
  buy(X,P) <= a(buyer, B) then
  sold(X, P) => a(buyer, B)
```

An Example: VM Allocation



- Policy 1 - power saving
  - pack VMs onto the minimum number of physical machines
- Policy 2 - agility
  - maintain an even loading across the physical machines

A Prototype


Distributed Discovery Service
Components
Peer
matching
Interaction
models
Physical machine
OK peer
libvirt
OS
Virtual Machines
Physical machine
OK peer
libvirt
OS
Virtual Machines

An Idle Host

```
a(idle, ID1) ::
        null
        ← overloaded(Status)
     then
        a(overload(Status), ID1)
   ) or (
        null
        ← underloaded(Status)
     then
        a(underload(Status), ID1)
   ) or (
     a(idle, ID1)
   )
```

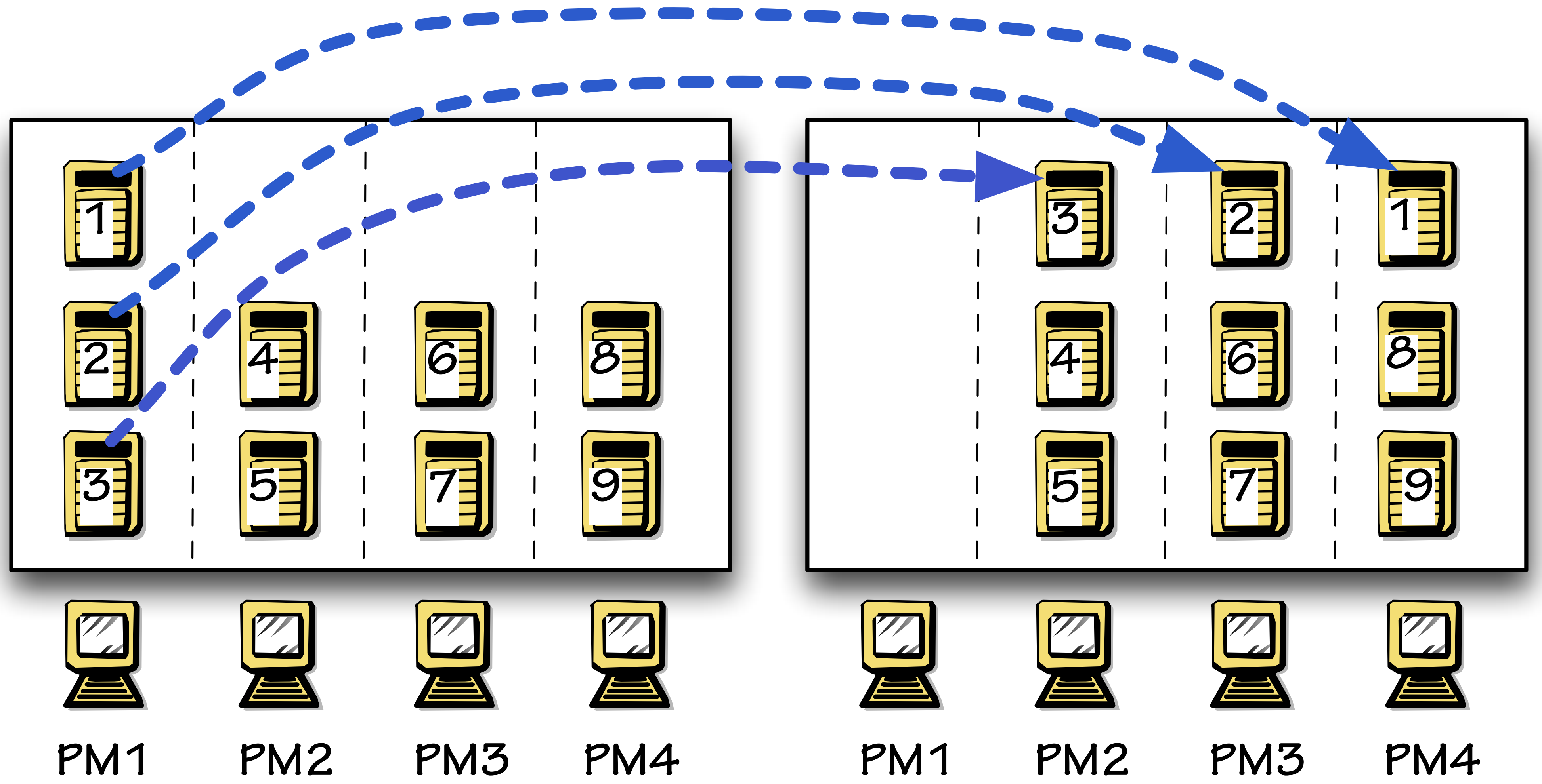
An Overloaded Host

```
a(overloaded(Need), ID2) ::
        readyToMigrate(Need)
        => a(underloaded, ID3)
    then
        migration(OK)
        <= a(underloaded, ID3)
    then
        null
        ← migration(ID2, ID3)
    then
        a(idle, ID2)
```

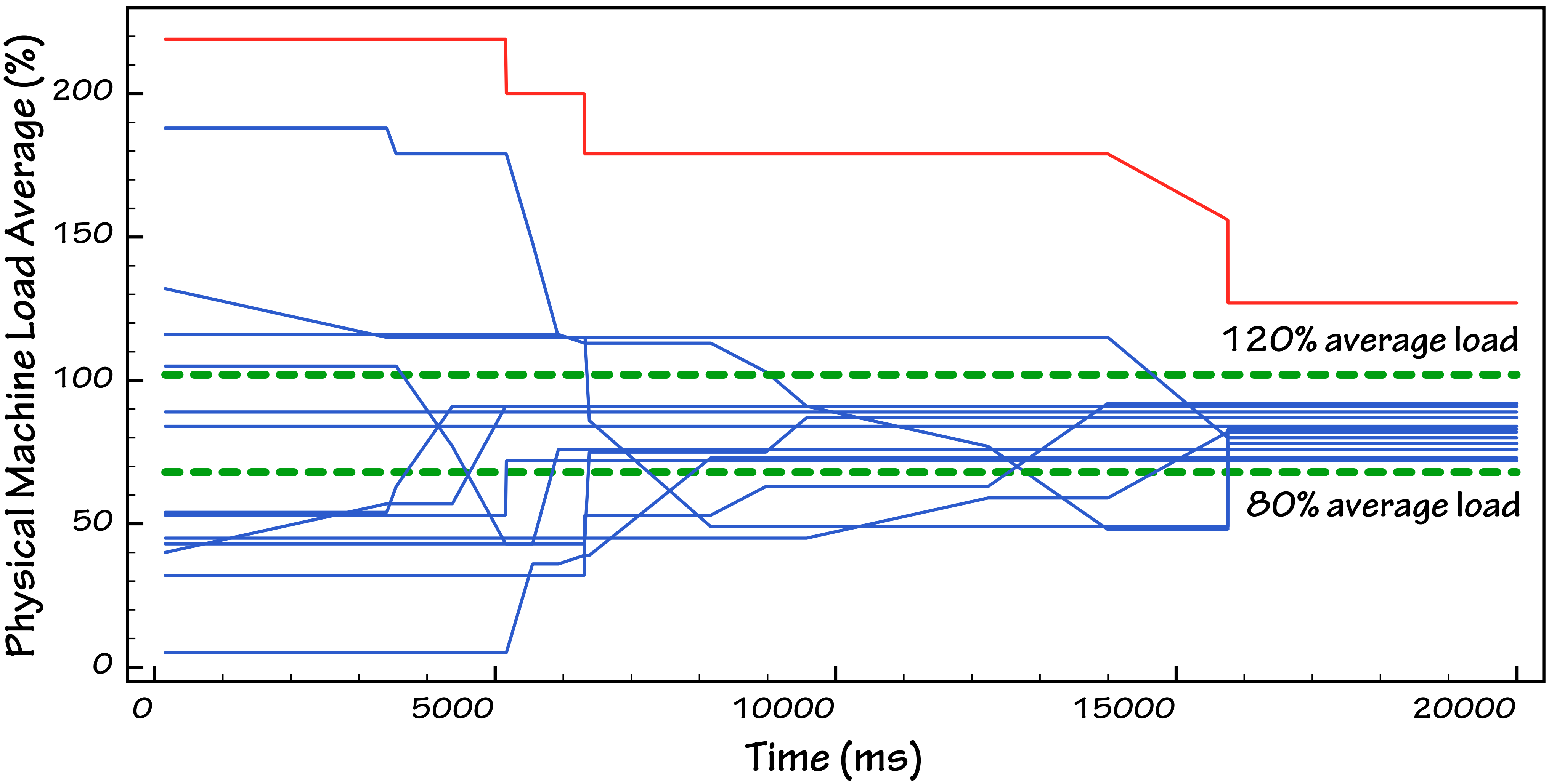
An Underloaded Host

```
a(underloaded(Capacity), ID3) ::
    readyToMigrate(Need)
    <= a(overloaded, ID2)
  then
    migration(OK)
    => a(overloaded, ID2)
    ← canMigrate(Capacity, Need)
  then
    null ← waitForMigration()
  then
    a(idle, ID3)
```

Migration Example



Simulation



Physical Machine Load Average (%)
200
150
100
50
0
0
5000
10000
15000
20000
Time (ms)
120% average load
80% average load

Protocol Issues

- The simple protocols described here are very naive
  - But the flexibility of the framework is the real contribution
  - LCC can easily be used to implement more sophisticated protocols - such as "auctions" which are ideal for many configuration scenarios
- The protocols so far rely rather heavily on the "discovery service"
  - This seems against the spirit of peer negotiation
  - We are investigating other protocols which (for example) search for suitable exchange candidates by passing requests to a local peer group who will then forward them if they cannot satisfy them

Measurement Issues

- Unpredictability of virtual machine performance is a significant problem
  - Latency is high for machine migrations
  - We are looking at machine learning techniques to identify (for example) "stable" and "unstable" machines
- In practice, machine "load" is multi-dimensional
  - We may want to consider cpu usage, memory usage and network (for example)
  - we are looking at ways of incorporating different factors

General Issues

- We would like to evaluate the approach in a more production environment, but ...
  - Handling errors and timeouts in an unreliable distributed system is hard
  - We have been using a research implementation of lcc which is not very robust
- Some things are hard to do without global knowledge
  - balance the system so that all the machines have exactly the same load?

KES AMSTA 2012

Multi-Agent Negotiation of Virtual Machine Migration

Using the Lightweight Coordination Calculus

Paul Anderson <dcspaul@ed.ac.uk>
Shahriar Bijani <S.Bijani@sms.ed.ac.uk>
Alexandros Vichos

THE UNIVERSITY of EDINBURGH
informatics

cisa | Centre for Intelligent Systems and their Applications