Reusable Mobile Agents for Cluster Computing

National Institute of Informatics

Ichiro Satoh

E-mail: ichiro@nii.ac.jp

Outline

- Motivation
- Tradeoff problem: reusability and network dependency
- Separation of itinerary and tasks in a mobile agent
- Specification language for agent itinerary
- Applications
- Conclusion

Motivation

System and network management is an important issue in cluster and grid computing systems.

Mobile agent-based management systems for cluster and grid computing have many advantages in comparison with C/S-based systems.

- the reduction of network traffic and latency,
- non-centralized administration, and
- direct manipulation at remote hosts

Mobile Agents

- Each mobile agent can migrate to and continue its execution at the destination.

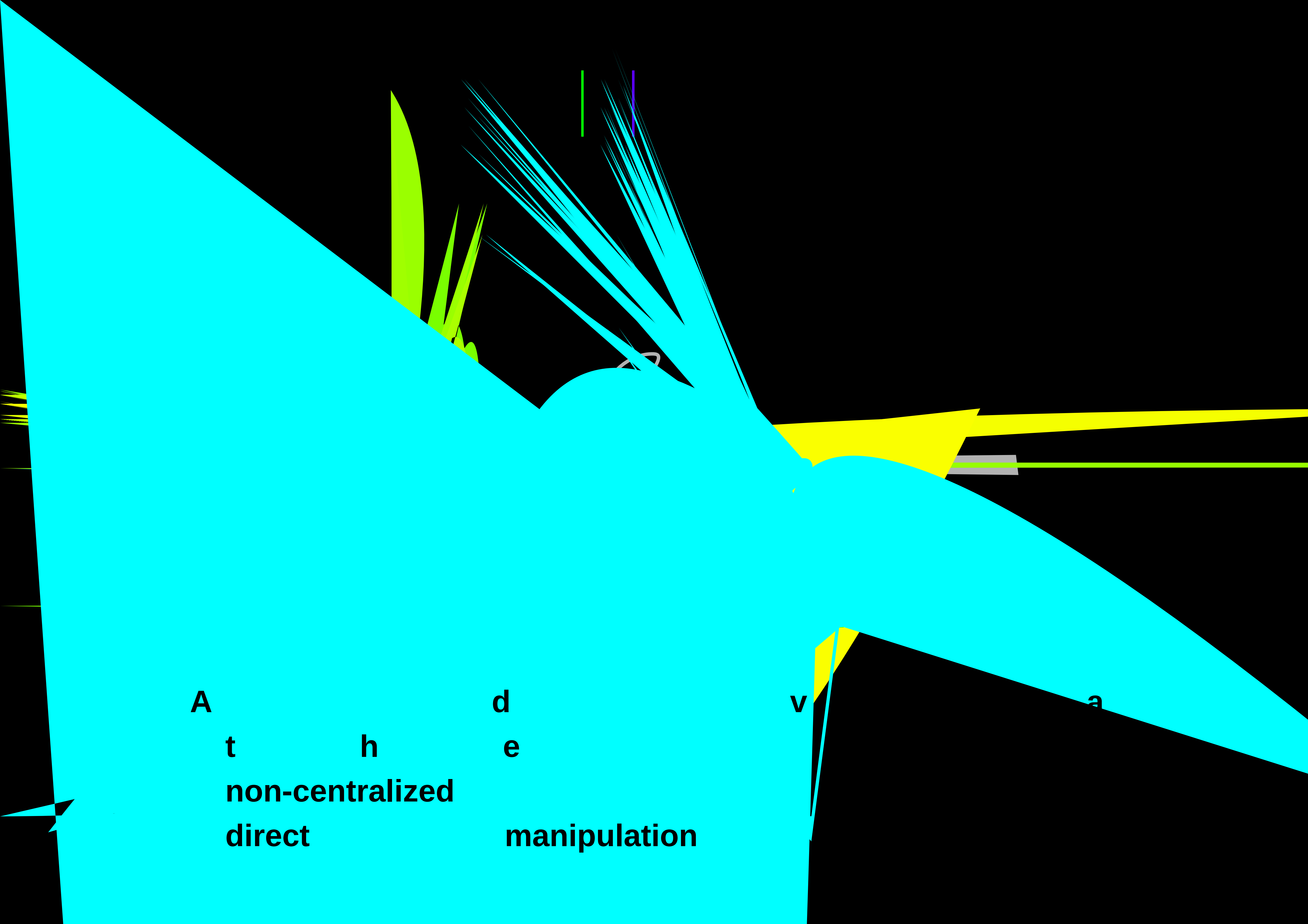
network

computer A

computer B

network

Ichiro Satoh



A d v a

t h e

non-centralized

direct manipulation

MA’s Applications

Mobile agents can provide many useful applications for cluster and grid computing systems, e.g.

- monitoring and controlling equipments at remote hosts,
- dynamic installation and upgrading of software at remote hosts, and
- dynamic load balancing

They become important in large-scale computing systems.

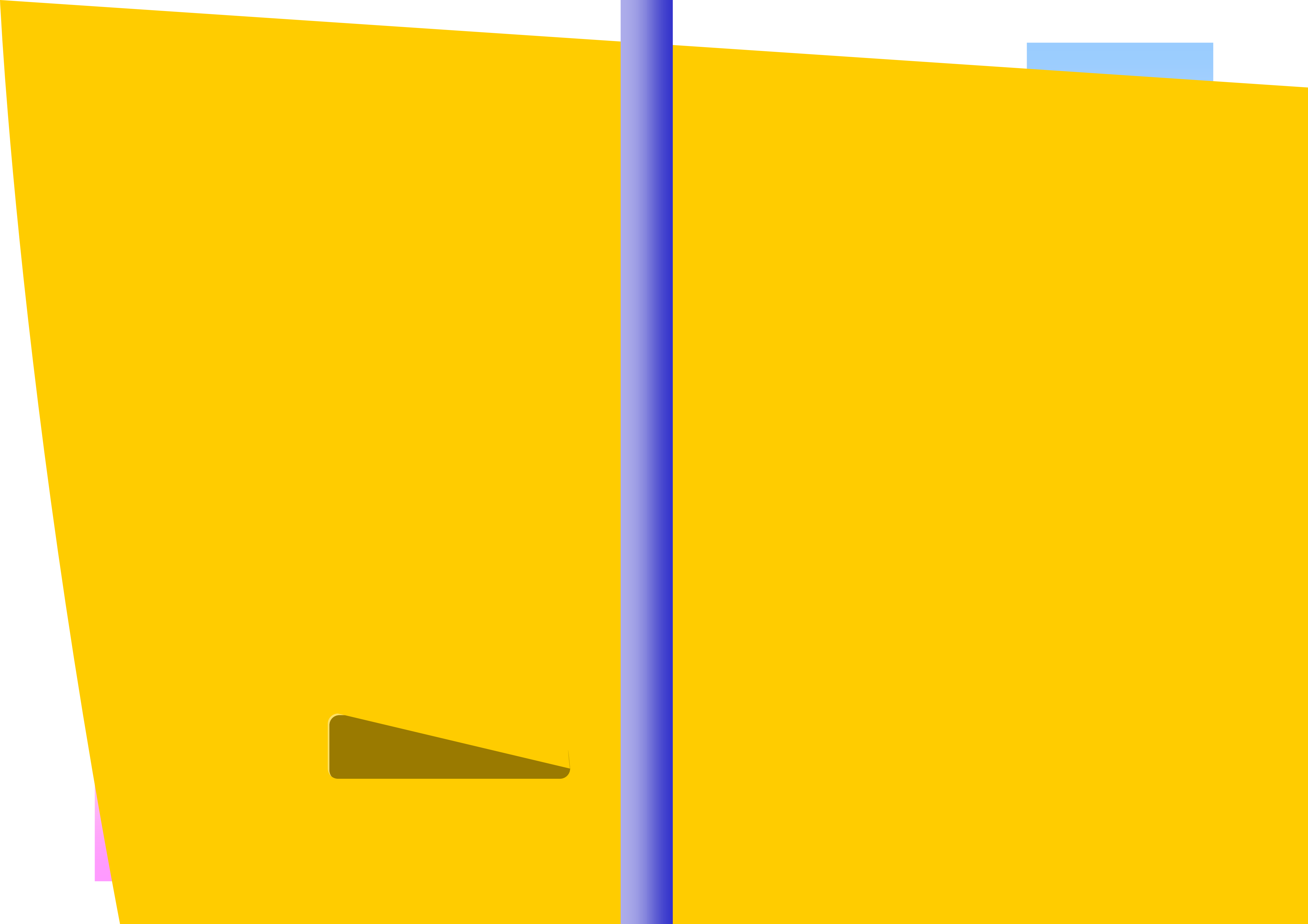
There have been many attempts for applying mobile agent technology to system and network management for cluster computing systems.

Efficiency vs. Reusability

Smart agents, which can dynamically generate their efficiently itineraries, tend to be large and complex.

Stupid agents, whose itineraries are statically designed for particular networks, can travel efficiently in the networks but cannot be reused in other networks.

This problem becomes a tradeoff between reusability and performance.



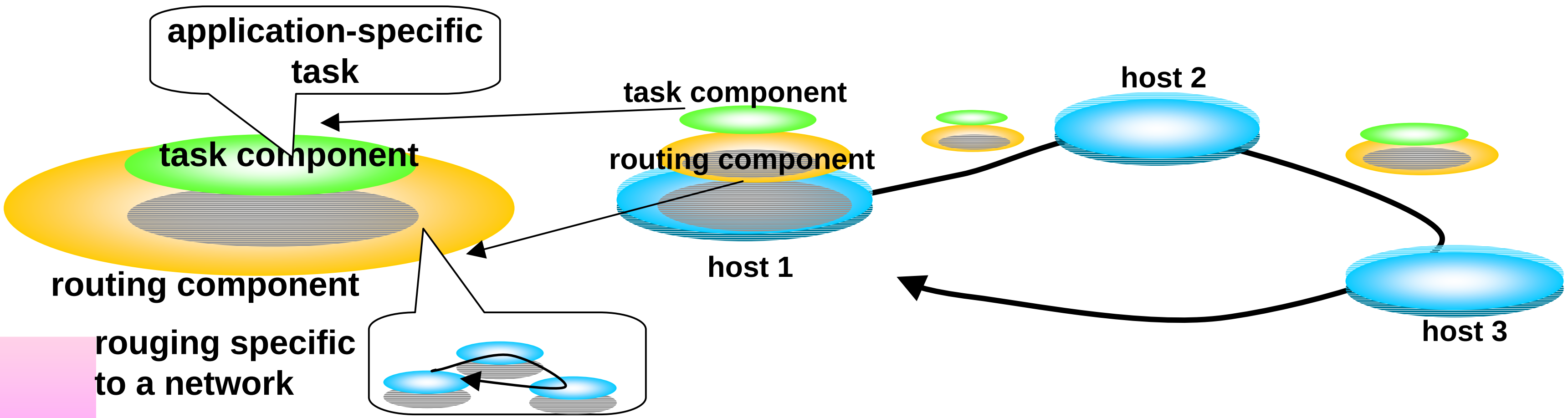
Separation Concerns

Each routing component is implemented as a mobile agent.

- it has its own itineraries, and
- it can carry task components among multiple hosts.

Each task component defines application-specific behaviors.

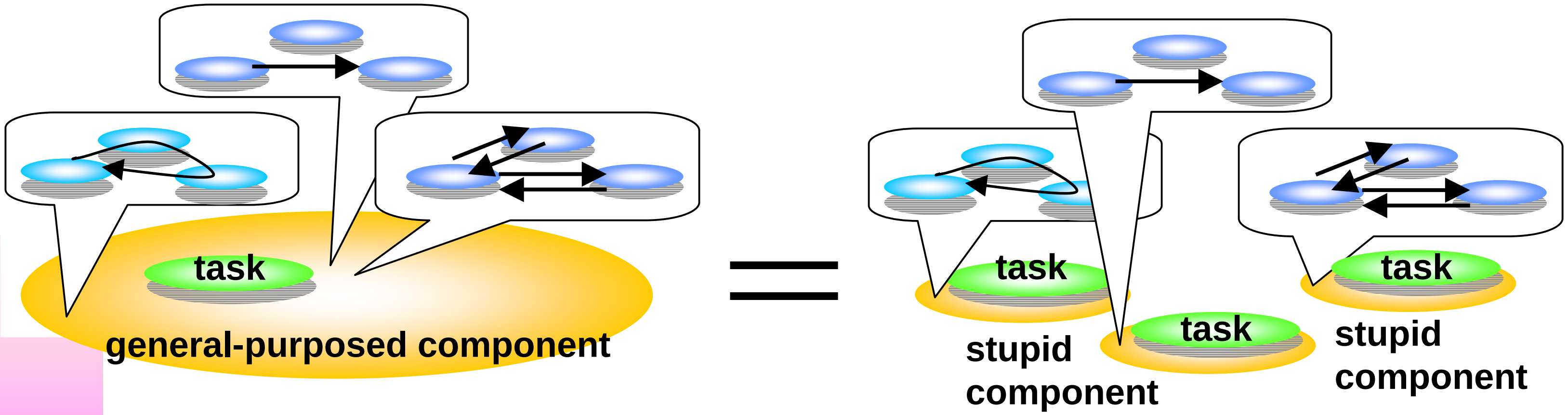
The both components can be dynamically composed into a mobile agent.



Variety vs. Generality

This framework provides a variety of small routing-components, which are statically optimized to particular itineraries, instead of any few general-purposed components.

Instead, the framework selects a suitable routing component among a variety of routing components according to the requirements of a task component.



Mobile Agent Selection

Mobile agents should be generally selected according to their itineraries as well as their application-specific behaviors.

Approach

- **A domain-specific language for describing the itineraries of routing components.**
- **An algebraic order relation for selecting suitable routing components whose itineraries can satisfy the requirements of task component.**

Formalism of Agent Itinerary Specification Language

- **Agent itineraries are specified based on a process algebra-based language:**

$$
\begin{array}{lllllllll}
E & ::= & 0 & \mid & \ell & \mid & E_1 \,;\, E_2 & \mid & E_1 + E_2 \\
 & \mid & E_1 \# E_2 & \mid & E_1 \mathbin{\%} E_2 & \mid & E_1 \mathbin{\&} E_2 & \mid & E^*
\end{array}
$$

- **The operational semantics of the language is defined as a labeled transition system:**

$$\frac{-}{\ell \xrightarrow{\ell} 0} \qquad \frac{E_1 \xrightarrow{\ell} E_1'}{E_1 \,;\, E_2 \xrightarrow{\ell} E_1' \,;\, E_2}$$

$$\frac{E_1 \xrightarrow{\ell} E_1'}{E_1 + E_2 \xrightarrow{\ell} E_1'} \qquad \frac{E_2 \xrightarrow{\ell} E_2'}{E_1 + E_2 \xrightarrow{\ell} E_2'}$$

$$E_1 \xrightarrow{\ell} E_1' \qquad E_2 \xrightarrow{\ell} E_2'$$

$$E_1' \mathbin{\&} E_2 \qquad E_1 \mathbin{\&} E_2 \xrightarrow{\ell} E_1 \mathbin{\&} E_2' \qquad E_1 \mathbin{\&} E_2 \xrightarrow{\ell} E$$

$$\frac{E_1 \xrightarrow{\tau} E_1'}{E_1 \,;\, E_2 \xrightarrow{\tau} E_1' \,;\, E_2} \qquad \frac{-}{E_1 \# E_2 \xrightarrow{\tau} E_1} \qquad \frac{-}{E_1 \# E_2 \xrightarrow{\tau} E_2}$$

$$\frac{-}{E_1 \mathbin{\%} E_2 \xrightarrow{\tau} E_1 \,;\, E_2} \qquad \frac{-}{E_1 \mathbin{\%} E_2 \xrightarrow{\tau} E_2 \,;\, E_1}$$

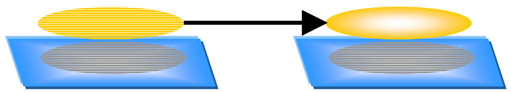
$$\frac{E_1 \xrightarrow{\tau} E_1'}{E_1 + E_2 \xrightarrow{\tau} E_1'} \qquad \frac{E_2 \xrightarrow{\tau} E_2'}{E_1 + E_2 \xrightarrow{\tau} E_2'}$$

$$\frac{E_1 \xrightarrow{\tau} E_1'}{E_1 \mathbin{\&} E_2 \xrightarrow{\tau} E_1' \mathbin{\&} E_2} \qquad \frac{E_2 \xrightarrow{\tau} E_2'}{E_1 \mathbin{\&} E_2 \xrightarrow{\tau} E_1 \mathbin{\&} E_2'}$$

Expressiveness

The language for specifying the required and possible itineraries.

- **sequential migration**



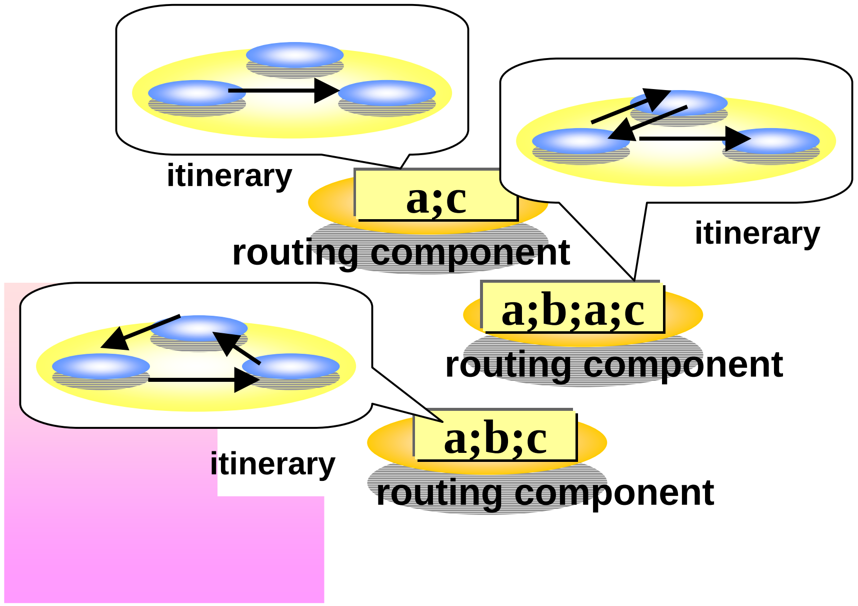
- **selective migration**

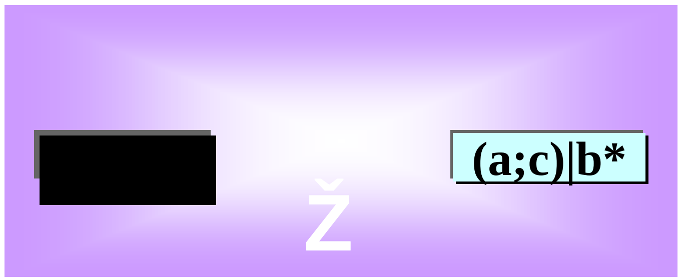
- **parallel migration**
- **commutable migration**
- **interrupted migration**

Agent Itinerary Selecti[redacted]

- The possible itinerary of a routing co[redacted] required by a task component are def[redacted] language.
- The algebraic relation compares the t[redacted] not routing components can satisfy th[redacted] task component.

task componentsatisfy





Agent Itinerary Selection

- **Algebraic order relations can judge whether the itineraries of routing components can satisfy the itinerary required by a task component.**

Definition 3.3: A binary relation $\mathcal{R}^n$ ($\mathcal{R} \subseteq (\mathcal{E} \times \mathcal{S}) \times \mathcal{N}$) is an n-*itinerary* prebisimulation, where $\mathcal{N}$ is the set of natural numbers, if whenever $(E, S) \in \mathcal{R}^n$ where $n \geq 0$ (n is 0 if $n < 0$), then the following hold for all $\ell \in \mathcal{L}$ or τ.

(i) if $E \xrightarrow{\ell} E'$ then there is an S' such that $S \xrightarrow{\ell} S'$ and $(E', S') \in \mathcal{R}^{n-1}$

(ii) $E\,(\xrightarrow{\tau})^*\,E'$ and $(E', S) \in \mathcal{R}^n$

(iii) if $S \xrightarrow{\ell} S'$ then there exist E', E'' such that $E\,(\xrightarrow{\tau})^* E' \xrightarrow{\ell} E''$ and $(E'', S') \in \mathcal{R}^{n-1}$

where $E \sqsupseteq_n S$ if there exist some n-itinerary prebisimulations such that $(E, S) \in \mathcal{R}^n$. We call $\sqsupseteq_n$ n-*itinerary* order. □

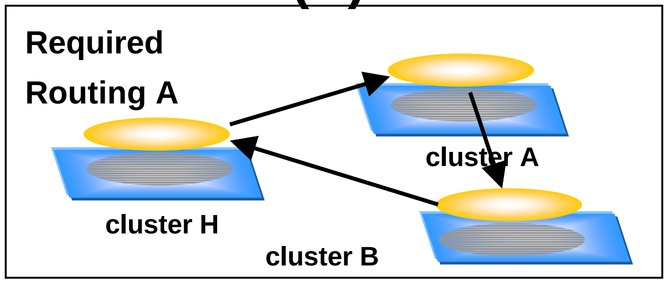
Agent Itinerary Selection

The language can specify loose requirements of agent itinerary.

The algebraic relation provides a reasonable itinerary-selection.

routing required by a given task

H → A → (H) → B → H

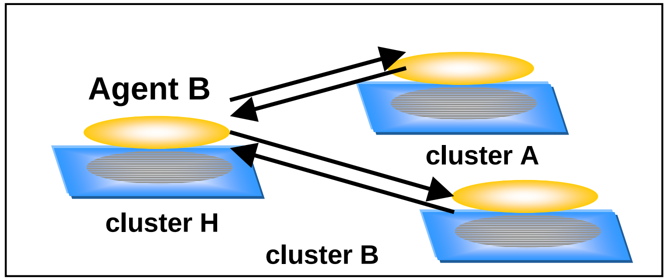


H;(A;B|H);H

⊇

the possible routing of a task component

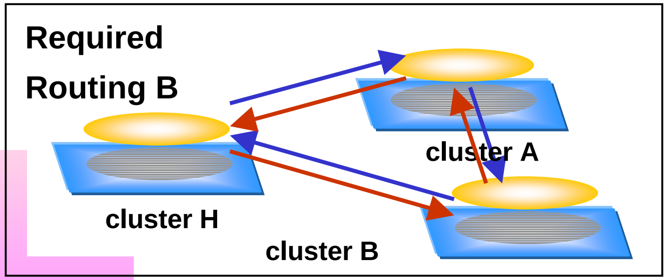
H → A → H → B → H



H;A;H;B;H

routing required by a given task

H → A → B → H or H → B → A → H

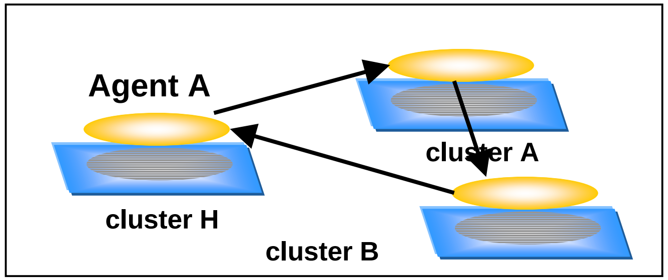


H;(A%B);H

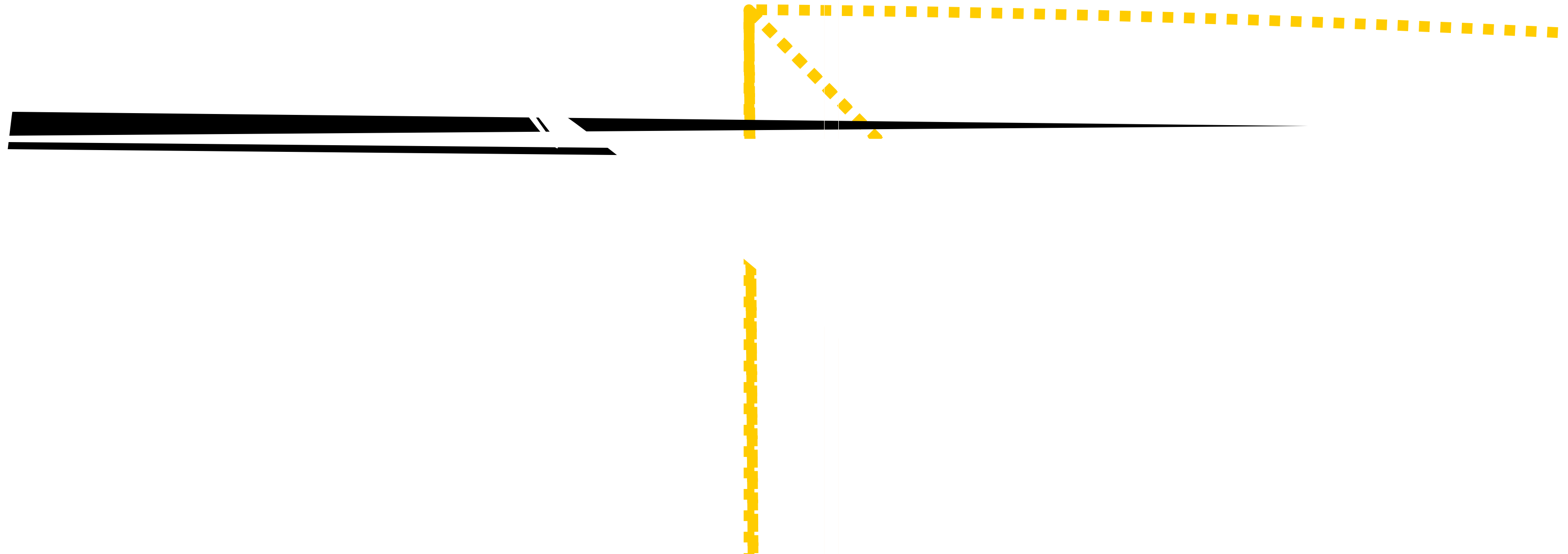
⊇

the possible routing of a task component

H → A → B → H

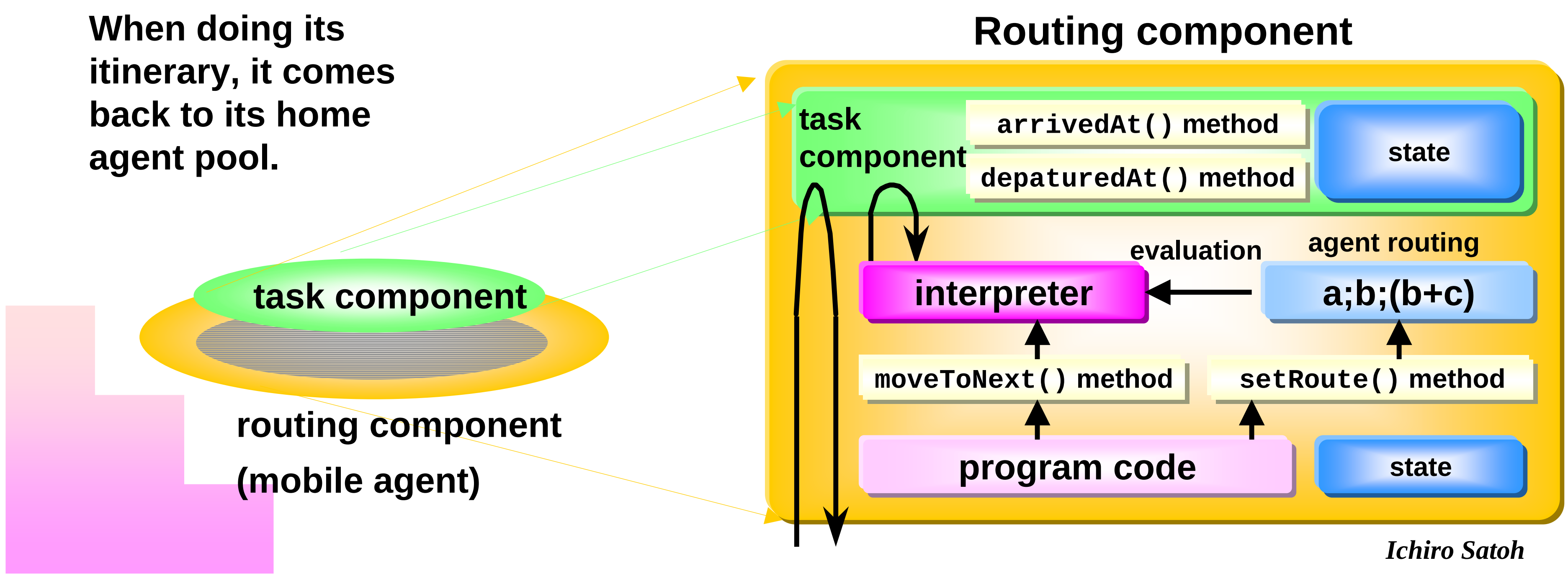


H;A;B;H



Routing Component

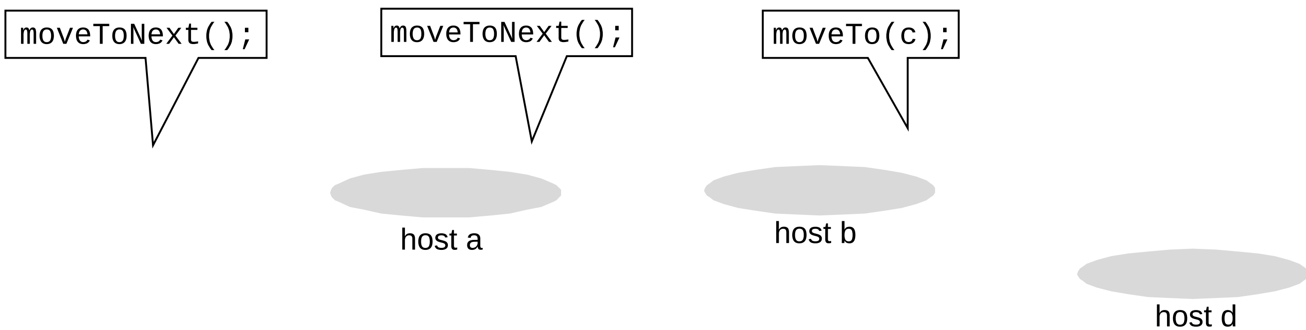
- Each routing component is implemented as a mobile agent.
- It is a carrier of task components to the hosts the task components must visit according to its own routing.
- When arriving at a destination, it invokes certain methods of its task components to do something at the destination.



Task Component

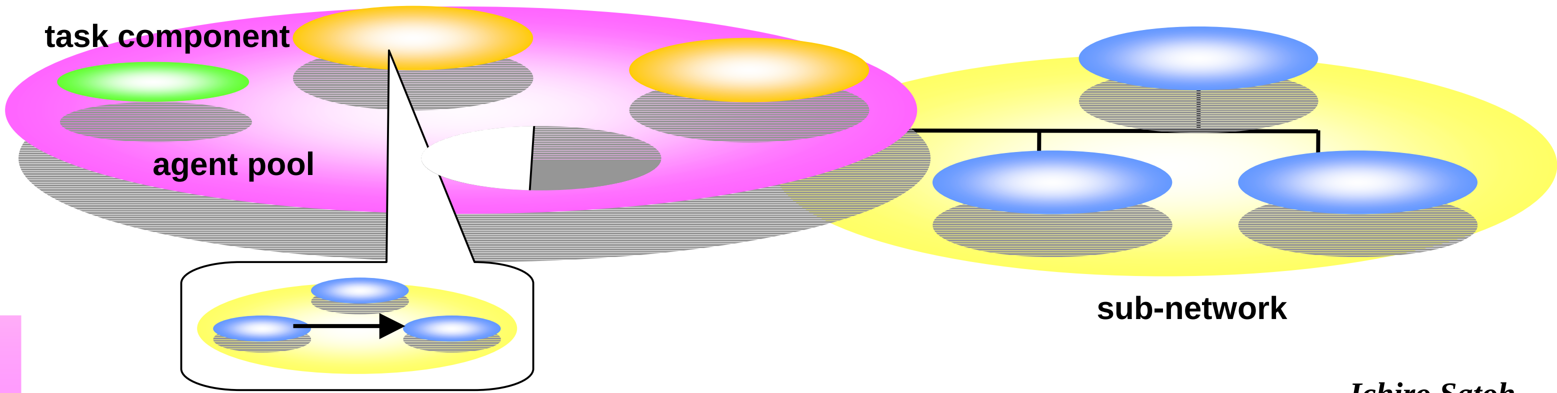
- **Each task component defines the application-specific tasks that should be executed at each of the visiting hosts.**
- **It can be implemented as Java Beans and Applets only when it supports specified callback methods.**

```
class Task extends TaskAgent {
  // after arriving the destin   on
  void arrive(AgentURL dst) {
    ... management task progra    .
    try {
      moveToNext();
    } (Exception e) { ... }
  }
  // before leaving to the des       on
  void departure(AgentURL to)        }
}
```



Agent Pool

- Agent pools are allocated on sub-networks and can store multiple routing components, whose itineraries are different and optimized to their target sub-networks.
- When receiving a task, an agent pool selects a suitable routing component that can satisfy the requirements of the task by using the algebraic relations.

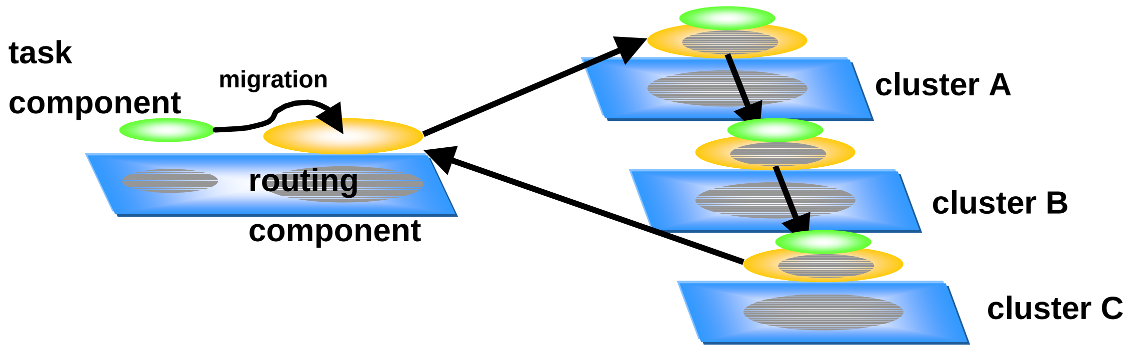


Application: Migration Patterns

- We have developed a variety of routing components, for example

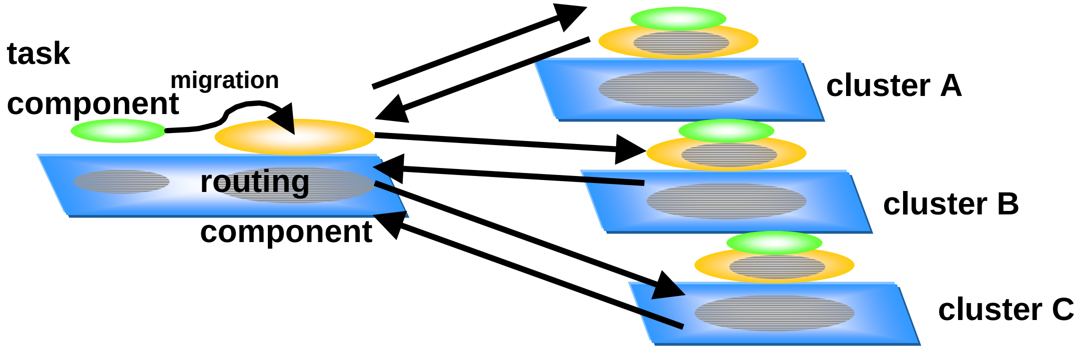
Example:

Round-robin migration



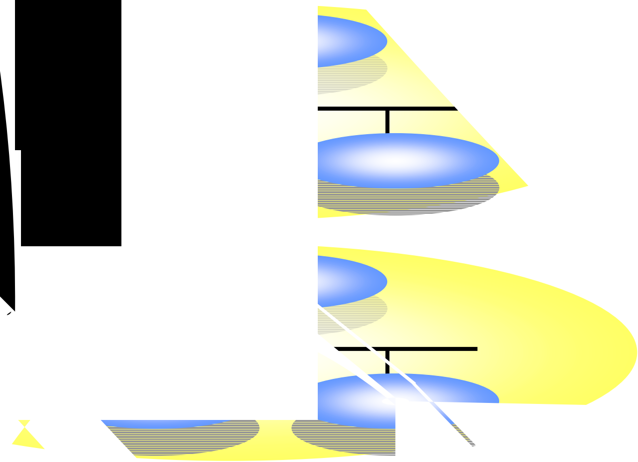
Example:

Star-shaped migration



Each task component can change its carrier according to its requirements and its current network.

ystem



Application: A Monitoring System for Cluster Computing

Reusability in software engineering

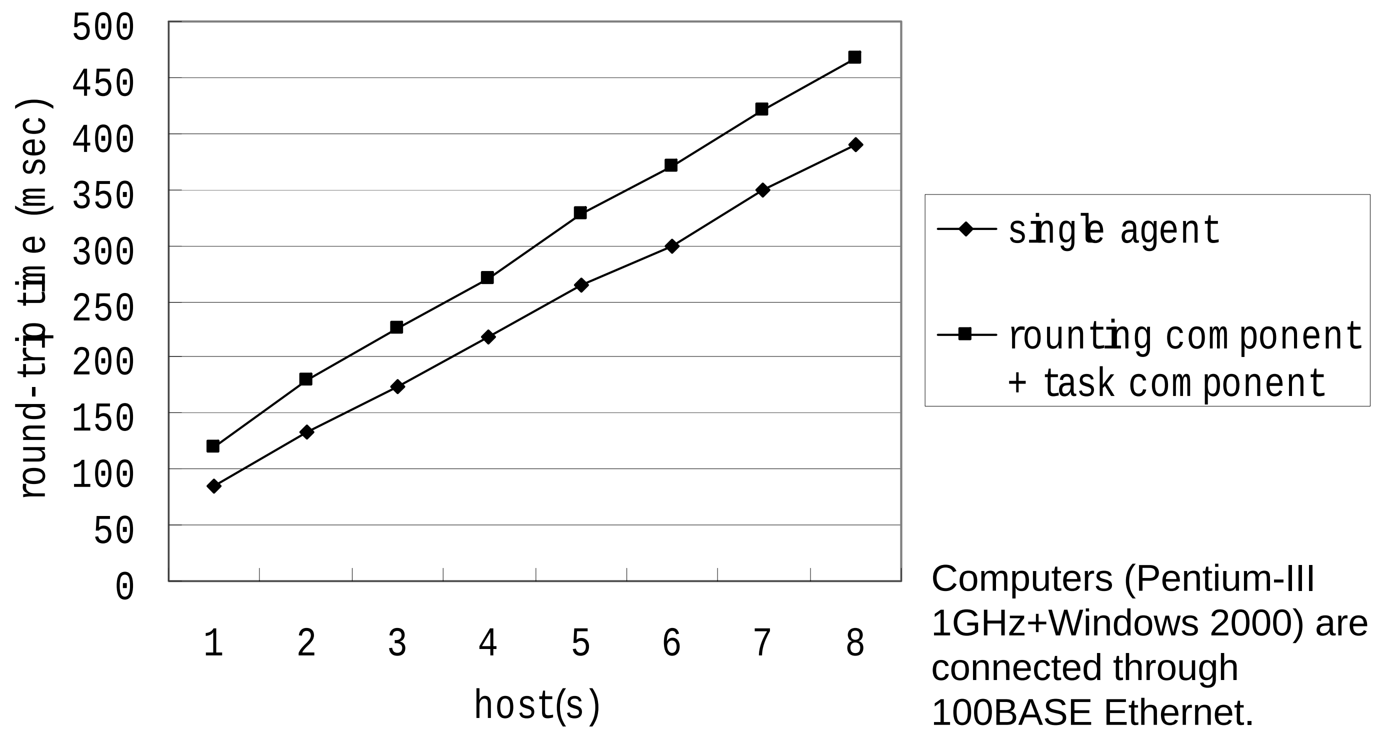
- Task components consists of only application specific tasks and can be reused in different networks.
- Routing components can be optimized for particular itineraries so that they can efficiently carry their task components among their destinations.

Performance:

- The cost of agent selection among 100 routing components is less than 1 sec. (dependent on the lengths of itineraries)
- The cost of migrating a pair of routing and task components is 45 msec between two hosts connected through 100 BASE-T.(15%-larger than single-layered agent)

Performance

- The cost of migrating a routing component, including a task component, in a cluster computing system.
- The task component gathers information from the MIB table of SNMP at each of the hosts that it visits.



Computers (Pentium-III 1GHz+Windows 2000) are connected through 100BASE Ethernet.

Conclusion

This framework can compose a mobile agent from its application-specific part and its mobility control part.

- Task components can be reused in other networks and routing components can efficiently travel over their target networks.

Agent selection mechanism compensates the variety and generality of small and optimized mobile agents.

A prototype implementation of the framework was built on a Java mobile agent system and was used for managing cluster computing systems.

Future Work

1. Evaluation from software engineering points
2. Higher-level specification language for agent routing
3. Enhancement of security mechanism
4. Performance improvement

Q&A

Please contact to:

- **E-mail: ichiro@nii.ac.jp**

Thank you very much