Specifying Runtime Functionality of Downloadable Components under the Sandbox Model

Haruhiko Kaiya & ***Kenji Kaijiri***

Sinshu University, JAPAN

Nov. 1, 2000

Sorry, misNaming!

Specifying Runtime *Environment* *for* Downloadable Components

Haruhiko Kaiya & ***Kenji Kaijiri***

Sinshu University, JAPAN

Nov. 1, 2000

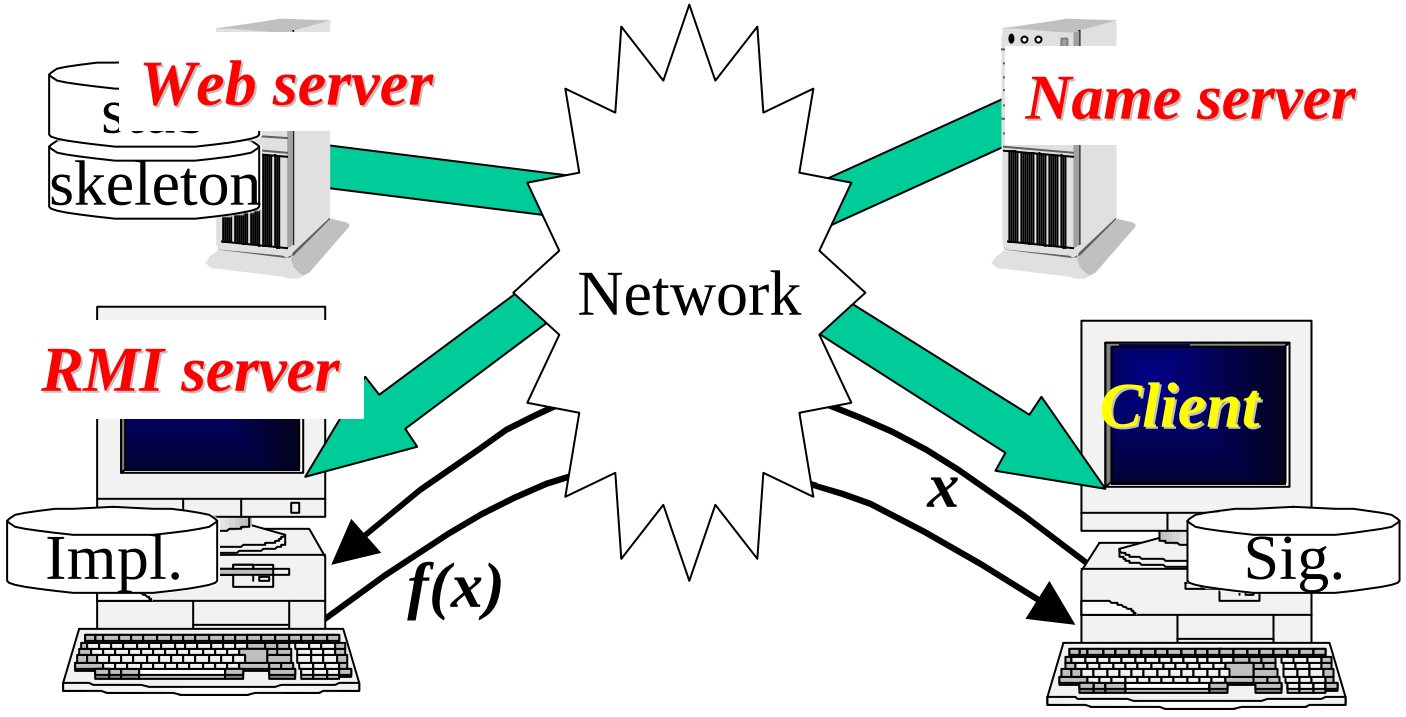
Outline

- Why we specify the environments?

  Confusion for using mobile code system.
- Objective of our work.
- How to specify the environments?
  - Overview of our specification.
  - Tools for our specification.
- Advantages and Disadvantages of our Spec

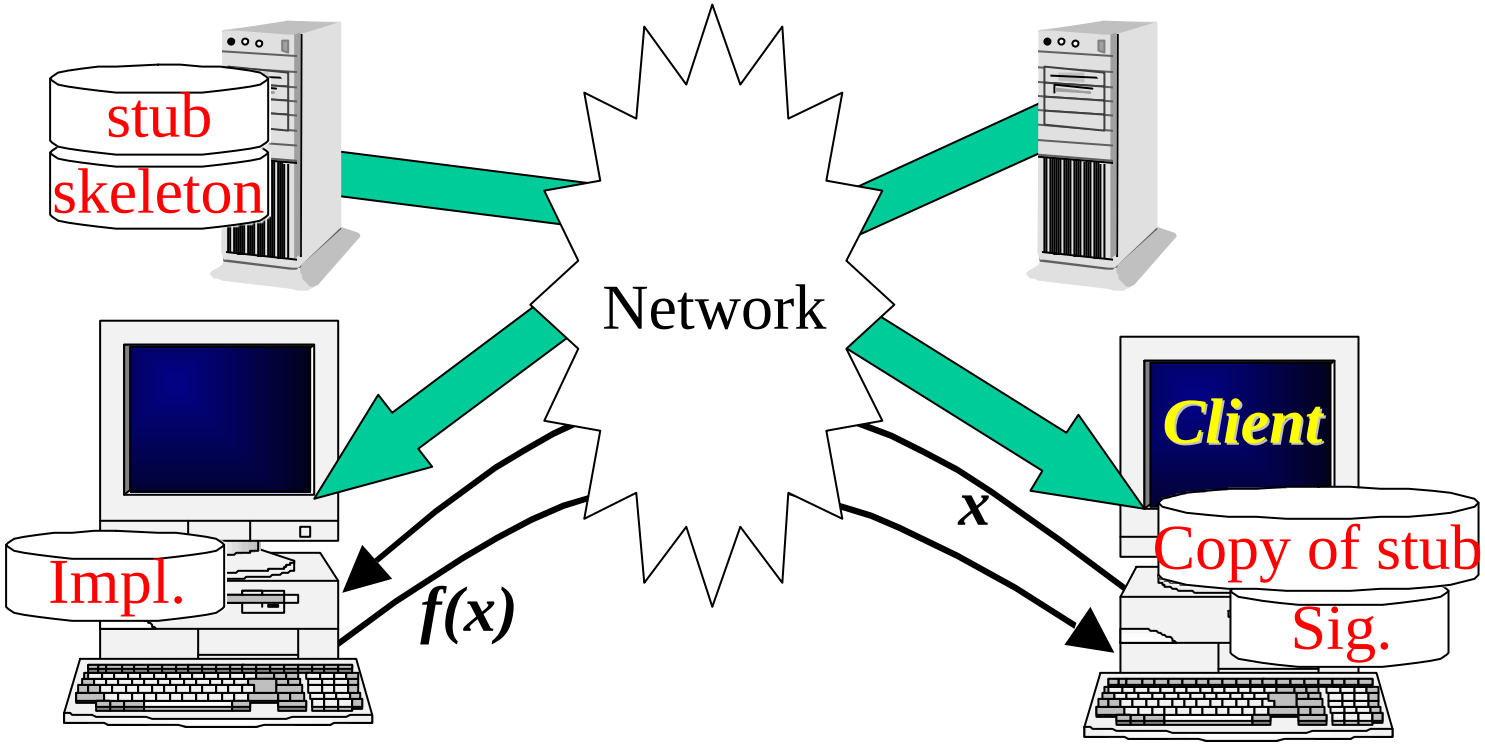
Confusion 1

- Confusion for setting the ***many services*** for mobile codes.



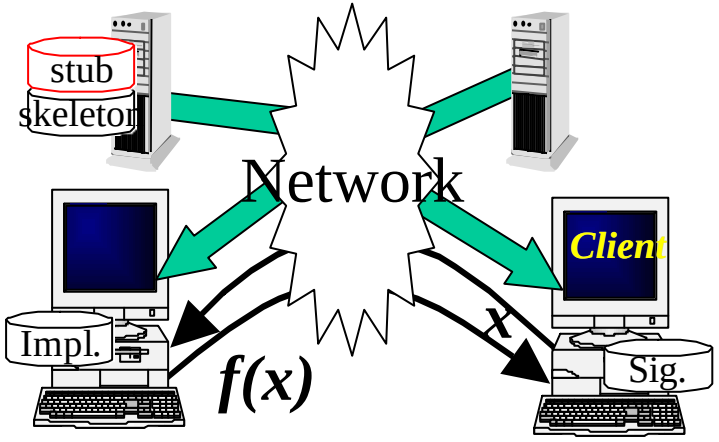
Confusion 2

- Confusion for ***deploying codes*** for mobile.



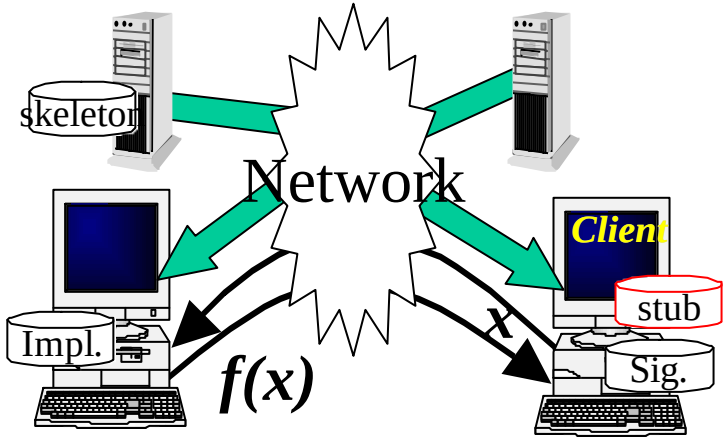
Confusion 3

- Confusion for ***identifying current security***, Security is sensitive to the environment.



Deployment changed,

Security is changed.



Objective of our Work

- **Decreasing** above **confusions** of mobile code users.
  - Support for checking whether **servers work adequately** or not.
  - Support for checking whether **the code deployment is suitable** or not.
  - Support for identifying whether **the security works good** or not.

Currently, not support

Overview of our Spec.

- Specification of runtime **environment**
  - Spec. for code **deployment**.
  - Spec. for **security policy** for the machine, where the codes are running.
- Specification of mobile **code**
  - **Search path** for loading codes used by the code.
  - Relationship between the **code and its birthplace**.
  - **Functionality** of the code.

Tools for our Spec.

- Z notation: state based formal notation.
  - well known.
  - simple predicate calculus and sets.
  - match for OO system.
  - formal reasoning.

Example of our Spec.

Env. Spec.

$$| \; deploy : Loc \pfun \mathbb{P}\, ByteCode$$

$$\begin{array}{|l}
\hline
SysRes \\
res : R \pfun Bool;\ limit : \mathbb{P}\, R;\ here : Loc \\
\hline
limit \subseteq \mathrm{dom}\ res \\
\forall\, x : limit \bullet (x, \mathrm{false}) \in res \\
here \in \mathrm{dom}\ deploy \\
\hline
\end{array}$$

Comp. Spec.

$$\begin{array}{|l}
\hline
Class \\
birth : Loc;\ byte : ByteCode \\
lslctr : \mathrm{seq}\ Loc \\
\hline
birth \in \mathrm{ran}\ lslctr \\
birth \in \mathrm{dom}\ deploy \\
\hline
\end{array}$$

$$\begin{array}{|l}
\hline
SetLoader \\
sl?;\ \mathrm{seq}\ Loc;\ \Delta Class \\
\hline
lslctr' = sl? \\
\forall\, x, y : \mathbb{N} \bullet byte \in deploy\ lslctr'\ x \land \\
\qquad x \in \mathrm{dom}\ lslctr' \land lslctr'\ y = birth' \Rightarrow y \leq x \\
\hline
\end{array}$$

For more detail, in proceedings.

Example of Formal Reasoning

- This schema also tells `***Cracking is established even if the security manager is set***'

$$SetLimit \fatsemi (SetLoader \land \Xi SysRes \fatsemi Crack \fatsemi Func \land \Xi Class \land \Xi SysRes) \setminus Class \mid pas! \in l? \land sl? = \langle here, there \rangle$$

- The above becomes consistent under the deployment:

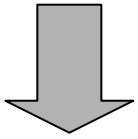
$$deploy = \{(here, \{byte, \cdots\}), (there, \{byte, \cdots\}) \cdots\}.$$

Advantages of our Spec.

- Reasoning whether your intended **security** and application **functionalities** are **preserved or not**.
- **Traceability for security** and functionality change.

Disadvantages of our Spec.

- Hard to read, formal notation.
- Breaking the modularity of spec; environments and codes.



- We should explore the **parameterization** for the properties of **environment among the component** spec.