UNIVERSITÉ DE GENÈVE

The Lana Approach to Autonomous Distributed Systems

Ciarán Bryce

University of Geneva

Why program in Java?

- "Object-oriented"
- "A language for robust, secure distributed computing"

The language designers

Why program in Java?

- For the application programmer
  - Security, robustness and distribution are important qualities
- But what is the programming environment?
  - *Robust with respect to what failures?*
  - *Secure with respect to what attacks?*
  - *Distributed over what architecture?*

The Java programming environment

- Distribution
  - A network of single user machines
  - Machines can exchange objects and code
    - →*Serialization, class loaders*
  - Machines can be heterogeneous
    - →*Bytecode interpretation*
  - Machines can be small (e.g., embedded devices)
    - →*Micro-editions, remote code loading*
  - Programs communicate over channels
    - →*Synchronous method invocation*

The Java programming environment

- Security
  - Do not trust remote code as much as “local” code
    - →*Sandbox model*
- Failure model
  - Network connections can break but generally do not
    - →*Exceptions*
  - When connections do break, things have to be fixed
    - →Program recovery protocol into the application

Is this model still good enough?

What are the trends in computing environments?

The trends

- Wireless networks are here
  - Long Distance Wireless
    - →E.g., Satellite, GSM, UMTS, ….
  - Short Distance Wireless - SDW
    - →E.g., Wireless LAN (IEEE 802.1), Bluetooth
      - Communication up to 100 metres; no operator needed

- *Wireless will help us to program and control embedded devices*

The trends

- Personalised devices
  - Personal Device Assistants
  - Mobile telephones
  - Smart-cards

} *same thing !*

- Embedded devices
  - 98% of all processors
- Devices can be heterogeneous in size, functionality, etc.

The trends

- Peer-to-peer computing on the Internet
  - Sharing of resources
    - →e.g., disk space, CPU space, music files, video
  - But without the use of dedicated and centralised services
- Peer to peer encourages the participation of end user machines as equals

UNIVERSITÉ DE GENÈVE

Where are these trends leading?

Global Computing Systems

Global Systems Characteristics

- Distribution
  - The nodes of the system are autonomous
  - There need not be a centralised control in a network
  - The number of nodes can be large
  - The configuration of a network can vary dynamically
    - →Networks are *spontaneous* or ad hoc

Global Systems Characteristics

- Failure : connection is never guaranteed
  - Mobile networks meet physical obstacles
    - →E.g., tunnels, walls, etc ….
  - Users leave their network
    - →e.g., Peer community members can just switch off
    - →e.g., Bluetooth user leaves his piconet
  - *So, failure does not mean that something is broken*

Global Computing Programming Model

- Autonomy for devices
  - A device's set of network neighbours continuously evolves
    - ⇨Avoid use of connections and synchronous communication
    - ⇨A device should be able to leave a network, then rejoin it, and continue running an application from where it left off
  - Applications should avoid dependence on specific sites
    - →Jini is unsuitable for GC since it requires a coordinator site
    - ⇒Associative information search
      - A telephone number for Marcel is found by broadcasting the message "Marcel's phone number?" and not by connecting to www.phones.ch

Global Computing Programming Model

- Program and Data mobility
  - Tolerating disconnection requires being able to download programs and data before disconnection
  - The overhead of network communication can be reduced by moving programs close together
  - Small devices may need to delegate programs and data to more powerful devices on their network

⇒An application must be able to exchange code, data and programs between devices

Global Computing Programming Model

- Security
  - For data exchanged over the networks
    - →No one should be able to intercept messages for a device
    - →Device can detect modifications of messages it receives
  - From data and programs downloaded
    - →Protection against viruses and Denial of Service attacks
  - Between programs running on a device
    - →Standard memory protection
  - From other devices on the network
    - →Detect and eliminate masquerading servers

UNIVERSITÉ DE GENÈVE

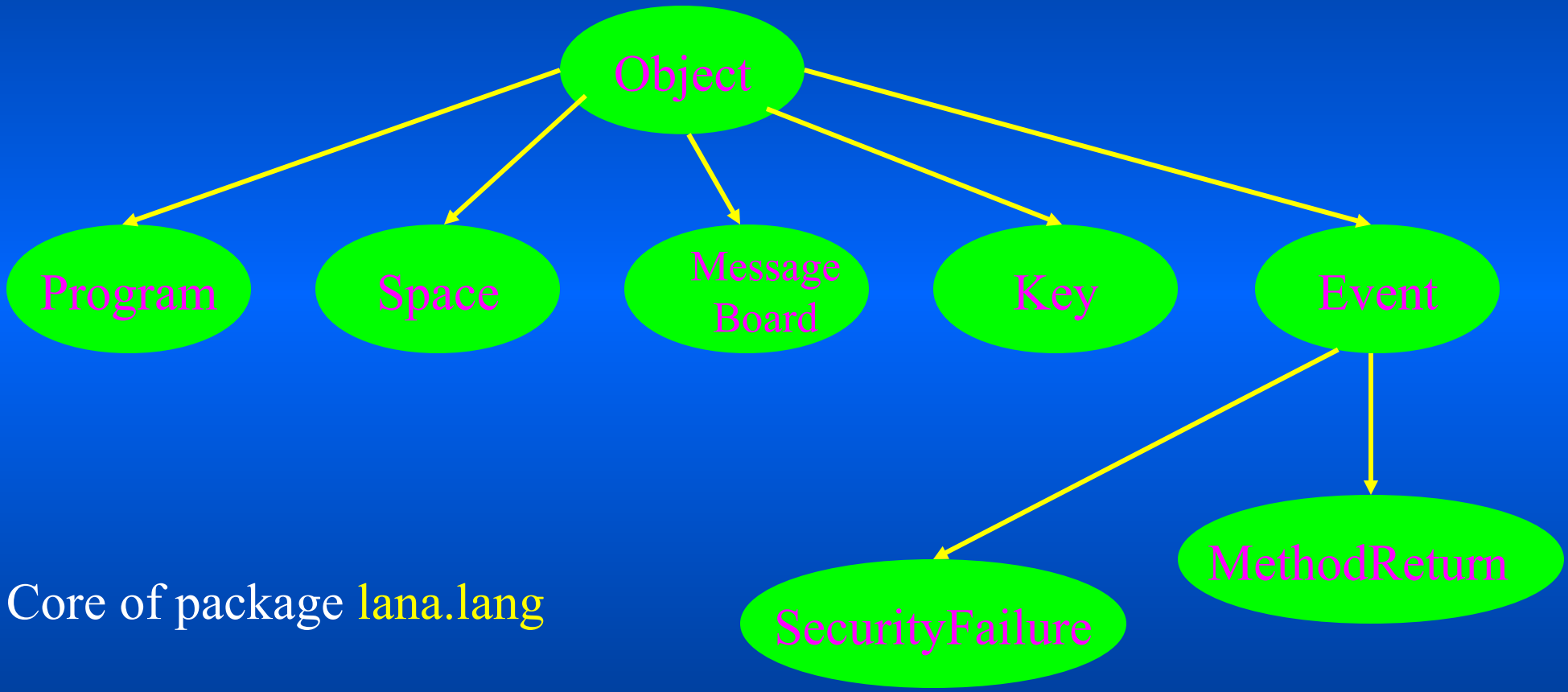
The Lana Language

Lana : Languages for Advanced Network Architectures

Lana in a Nutshell

- Extension to Java
  - Object-oriented, (classes, single inheritance, interfaces, packages)
- Designed for Global Computing environments
  - Multi-programmed language
  - Supports device autonomy
    - →Device can leave network application and rejoin later
    - →Allows applications to adapt to loss of network
  - Secure information access

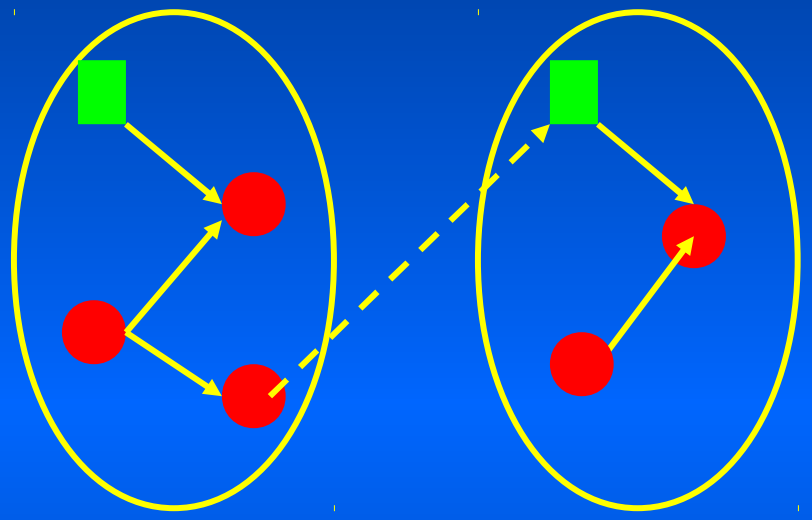
Lana Class Hierarchy



Core of package lana.lang

Programs in Lana

- Unit of accounting
  - An object belongs to only one program
- Unit of mobility
  - A program moves with all of its contained objects
- Unit of protection
  - A program is unable to call methods on objects in another program
  - Each method call on another program is verified by a security policy



Protection + Accountability + Mobility + …. ⇒ Autonomy

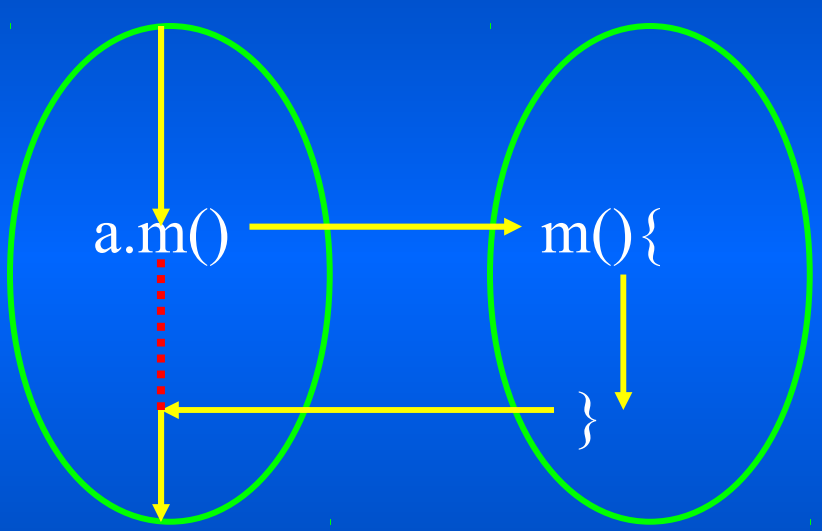
Asynchronous Method Calls

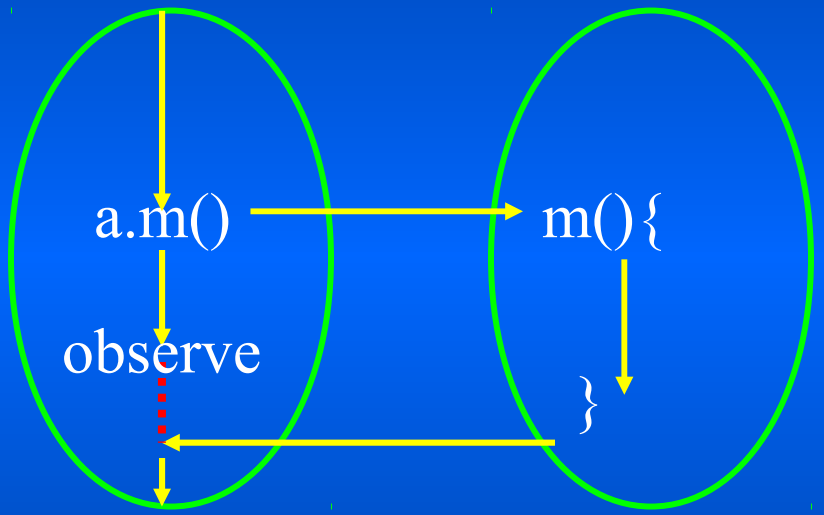
- Method calls on strong references are *synchronous*
- Method Calls on weak references are *asynchronous*
  - Calling program places method call request in the called program's mailbox
  - And continues its execution
- Each method call return message is identified by a unique *key*

```
class Me extends Program{
Key k;
Event e;
Program p = new You();

k = p!yourName();
Device.print("I asked for a name\n");
observe[k](e); // await method return
Device.print("and got the reply " +
    e.extract());
}
```

Asynchronous Method Calls





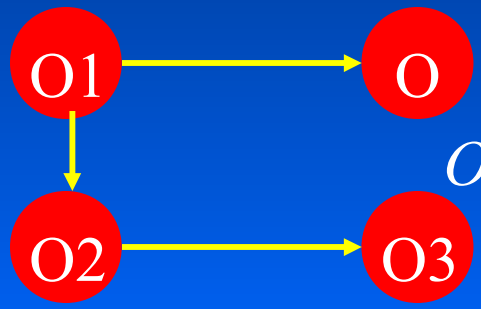
- Each program possesses a thread that reads its mailbox and executes the requested method

Keys

- Keys are objects whose values are unique in time and space
- Keys cannot be fabricated
- A fresh key is generated at each method call
  - A method call return message can only be read (observed) by a program that possesses a copy of the key
  - Any exception linked to a method call is locked with the same key
    - →E.g., security violation exception, device unreachable exception
  - Keys can be exchanged between programs (as method call parameters)
  - In this way, a program may delegate the handling of a method to another program

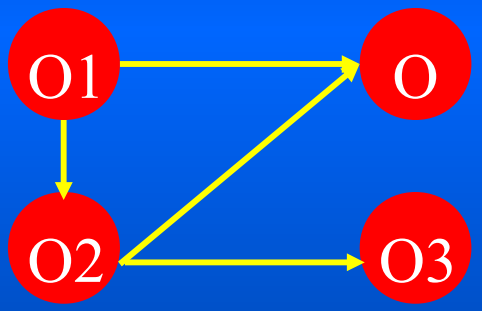
Spaces

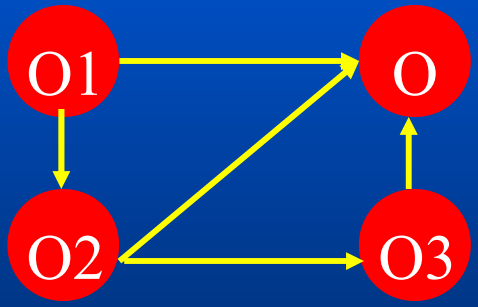
- An object is *aliased* when more than one copy of a reference for that object exists
- Reference passing is how information is dispersed in an OO system
- But uncontrolled aliasing leads to security leaks
- Aliasing is the source of very many documented security flaws in Java and other OO systems



O1 references O

Passes a copy to O2, ...

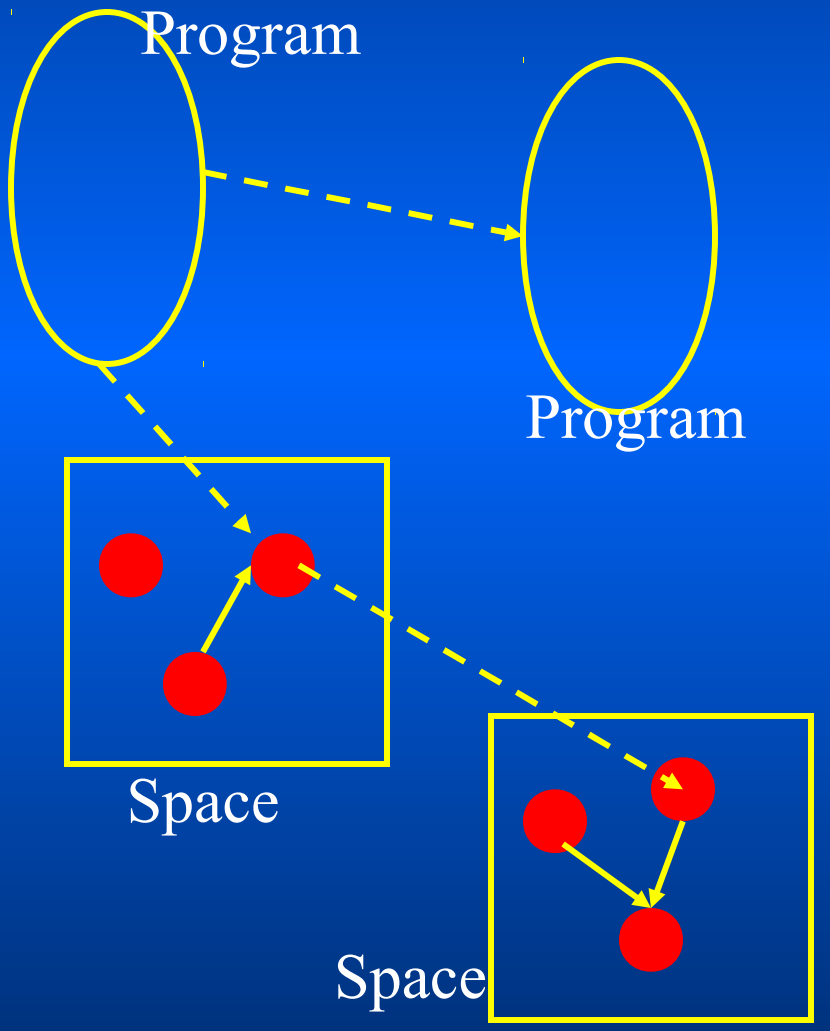




Which passes a copy to O3, without O1's knowledge

Spaces for Secure Aliasing

- Programs can share spaces of objects
  - Objects in the same space name each other using strong references
  - Objects in a space are named from outside using weak references
  - Method calls on objects in other spaces are asynchronous
- The fact that you hold a reference for an object does not mean that you can call methods on it
  - The security policy might refuse
  - Or the object's space may be moved to another device



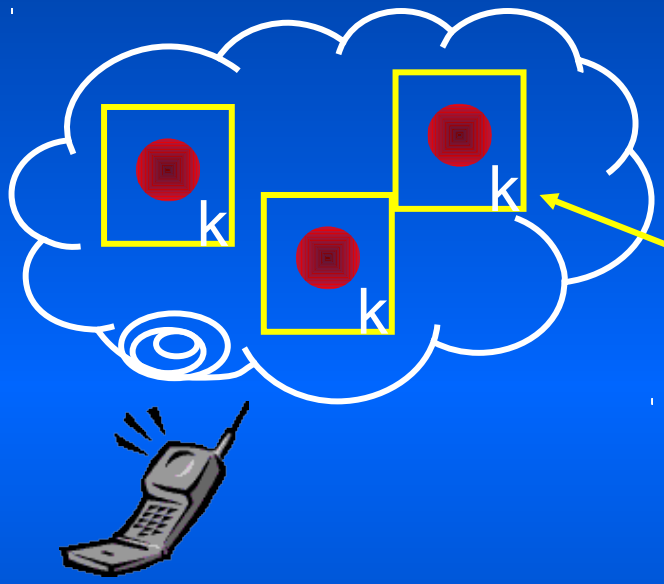
Message Board

- Devices that meet need to get to know each other
  - I.e. exchange references for their objects
- Each device possesses a message board
  - A device's message board may be accessed by any other device in the network
  - Each object stored in a message board is locked with a key
    - →An object can only be recovered from a message board if the calling device possesses the key that locks the object

API: void out(Device, Key, Object)

and Object in(Device, Key)

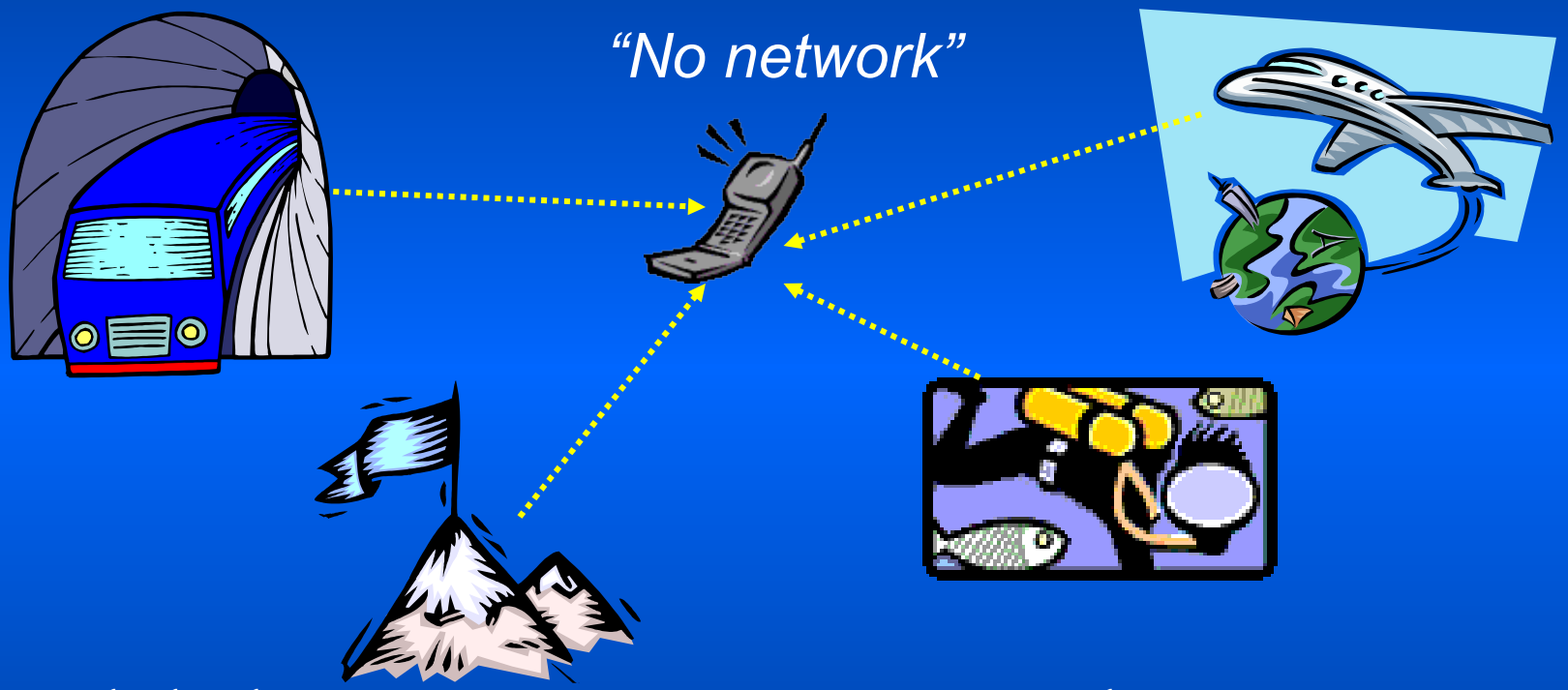
Message Board



Protection + Accountability + Mobility + Message Boards + Asynchronous Messaging ⇒ Autonomy

- Other roles of Message Board
  - Support copy-by-value information exchange
  - Whenever a method call cannot be returned to calling object
    - →The result is stored in the local message board
  - Used for exchanging data and programs between devices

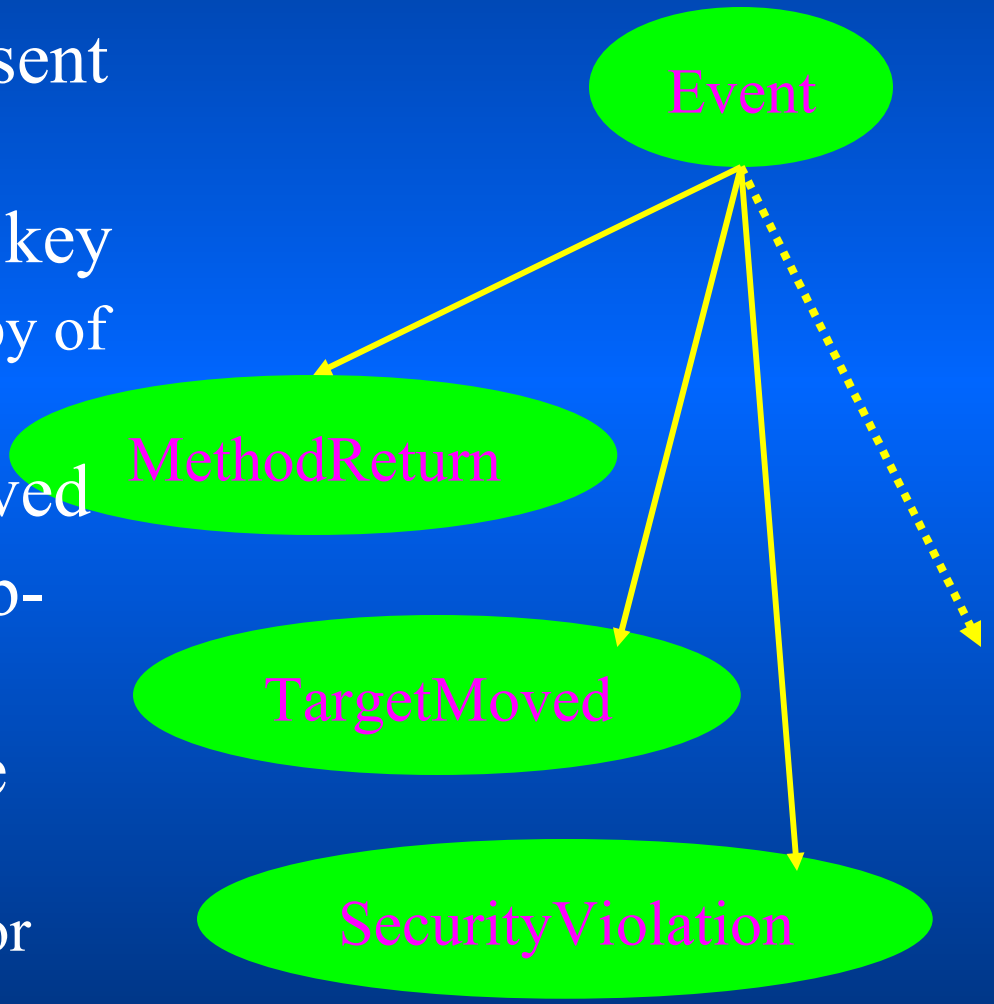
Events



- In a global system, a program must be autonomous
  - It must be able to recognize and to adapt to different event kinds

Events

- Events are asynchronously sent between objects
- Each event is locked with a key
  - An object must possess a copy of this key to observe the event
- An event need not be observed
- Programmers can define sub-classes of events
- Lana defined events include
  - Method return, failure of a method call due to mobility or security



Lana Project Status

- Java Library
  - 1st Prototype written in Java programming language
- Minimal VM
  - The design of a minimal virtual machine that can run both Lana and Java programs
    - →Cooperation with OVM