UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Sistemas Distribuídos

Prof. Sérgio T. Carvalho

Serviço de Nomes

Roteiro

Conceitos básicos de serviços de nomes

Exemplos de serviços de nomes

- CORBA *Naming Service*
- RMI *Registry*

Serviço de *Trading* (descoberta de serviços)

Bibliografia

Wolfgang Emmerich, *Engineering Distributed Objects*, Wiley, 2000.

Disponível na Biblioteca Central

Motivação

Evitar o uso de endereços físicos para a localização de componentes

Naming

- Localização de componentes por meio de nomes externos
- Similar às *páginas brancas* de um catálogo telefônico

Trading

- Localização de componentes por meio de características de serviço
- Similar às *páginas amarelas*

Serviços de Nomes

Princípios Básicos

Plataformas de middleware orientadas a objetos utilizam-se de *referências de objetos* para endereçar objetos servidores

É necessária uma forma de obter tais referências de objetos sem a necessidade de suposições sobre localização física

Um *nome* é uma sequência de cadeias de caracteres que pode ser "amarrada" a uma referência de objeto

name binding

Um nome pode ser resolvido para se obter a referência de objeto correspondente

Princípios Básicos (cont.)

Podem haver muitos objetos servidores em um sistema de objetos distribuídos

Objetos servidores podem ter vários nomes

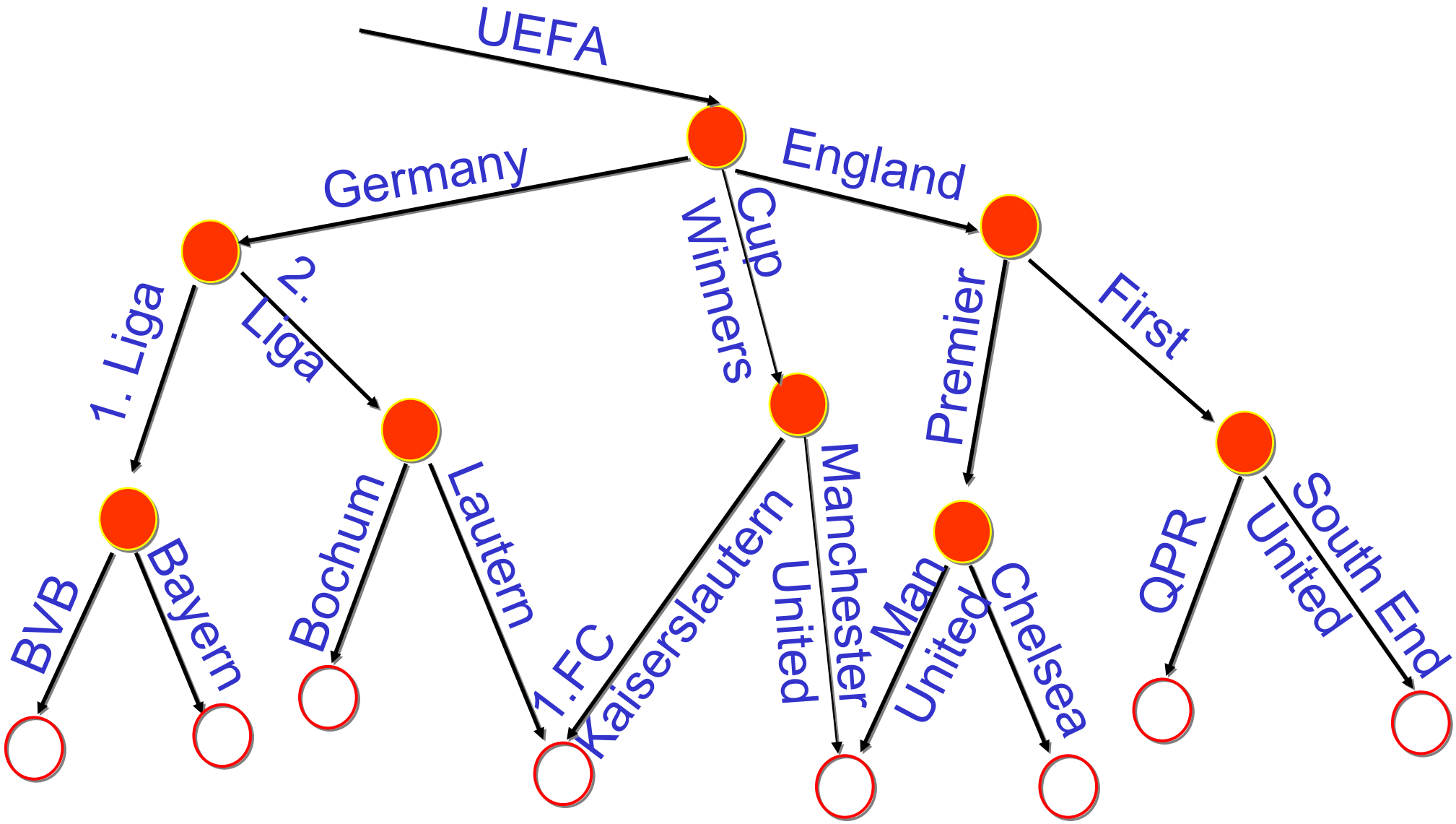
Isto leva a um grande número de *"name bindings"*

O espaço de nomes deve ser organizado em uma *hierarquia* para evitar

- conflitos de nomes
- baixo desempenho na resolução de nomes

Esta hierarquia pode ser construída através de *contextos de nomes*

Princípios Básicos: Contextos de Nomes



Princípios Básicos: Nomes Compostos

Nomes podem ser formados através da composição de vários nomes simples (*strings*)

A sequência de componentes de um nome descreve um caminho através do grafo de contextos de nomes

Exemplo: Nome do objeto representando o time Chelsea:

("UEFA", "England", "Premier", "Chelsea")

Princípios Básicos: Servidor de Nomes

"Amarrações" de nomes (a referências de objetos) são administradas por servidores de nomes

Nem todo objeto servidor precisa de um nome

Alguns objetos servidores podem ter vários nomes

Os servidores de nomes devem armazenar as amarrações de nomes persistentemente

Os servidores de nomes devem ser "calibrados" com vistas à eficiência na resolução de nomes

O próprio servidor de nomes pode ser distribuído

O Serviço de Nomes de CORBA

Provê suporte para a amarração de nomes (*name binding*) a referências de objetos CORBA

Escopo de um nome: contexto de nomes

Múltiplos nomes podem ser definidos para uma mesma referência de objeto

Nem todas as referências de objetos precisam de nomes

Nomes em CORBA

Nomes de objetos são compostos por *nomes simples*

Nomes simples (componentes): pares *valor-tipo*

- O atributo *valor* é usado para resolução de nomes
- O atributo *tipo* é usado para fornecer informação a respeito do papel do objeto

Tipos IDL para Nomes

```
module CosNaming {
  typedef string Istring;

  struct NameComponent {
     Istring id;
     Istring kind;
  };
  typedef sequence <NameComponent> Name;
  ...
};
```

Interfaces IDL do Serviço de Nomes

O Serviço de Nomes é especificado através de duas interfaces:

NamingContext define operações para ligar objetos a nomes e para a resolução de nomes

BindingIterator define operações para iterar em um conjunto de nomes definido em um dado contexto de nomes

Interface *NamingContext*

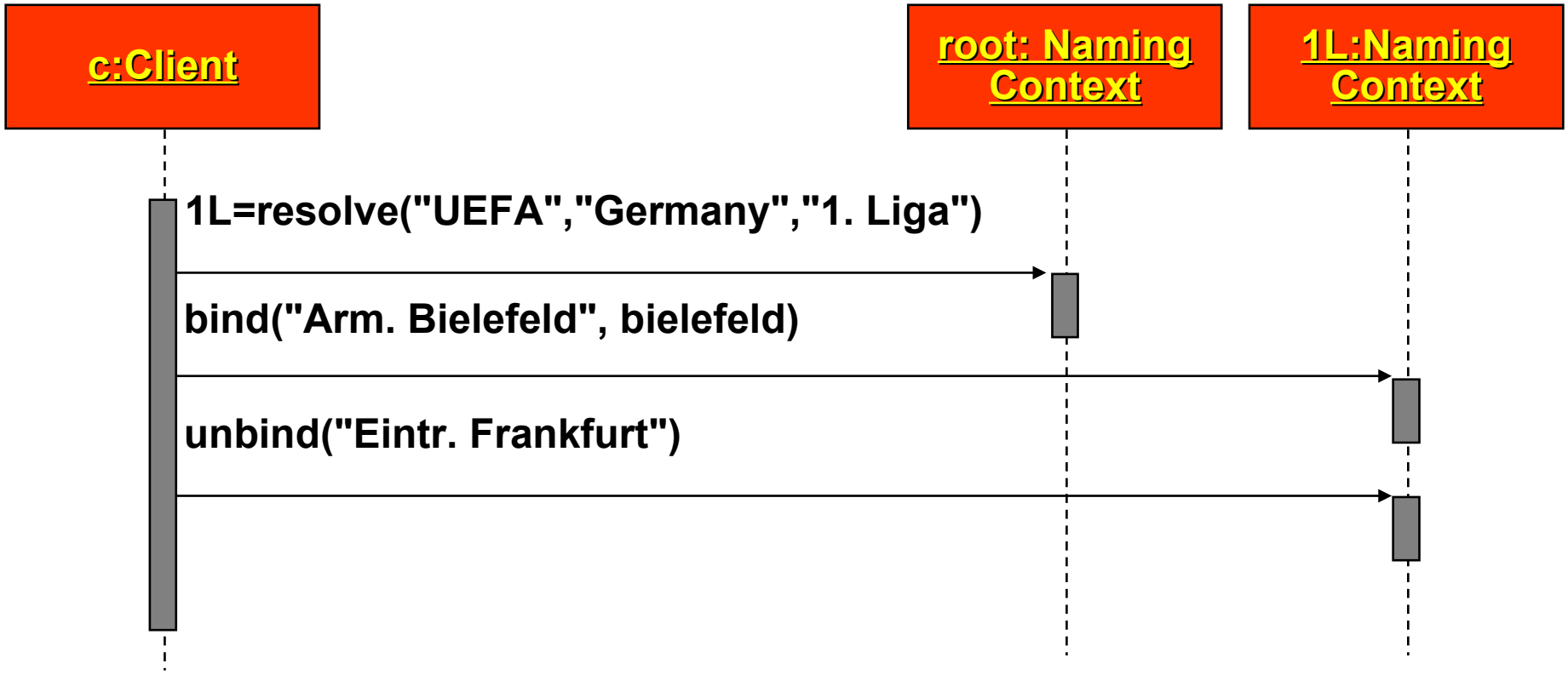
```
interface NamingContext {
    void bind(in Name n, in Object obj)
            raises (NotFound, ...);
    Object resolve(in Name n)
         raises (NotFound,CannotProceed,...);
    void unbind (in Name n)
         raises (NotFound, CannotProceed...);
    NamingContext new_context();
    NamingContext bind_new_context(in Name n)
                  raises (NotFound, ...)
    void list(in unsigned long how_many,
              out BindingList bl,
              out BindingIterator bi);
};
```

Interface *BindingIterator*

```
interface BindingIterator {
  boolean next_one(out Binding b);
  boolean next_n(in unsigned long how_many,
                 out BindingList bl);
  void destroy();
}
```

Cenário de Uso do Serviço de Nomes: Registro de Nomes

Exemplo: Promover um time para a 1a. divisão e rebaixar outro



Inicialização do Serviço de Nomes

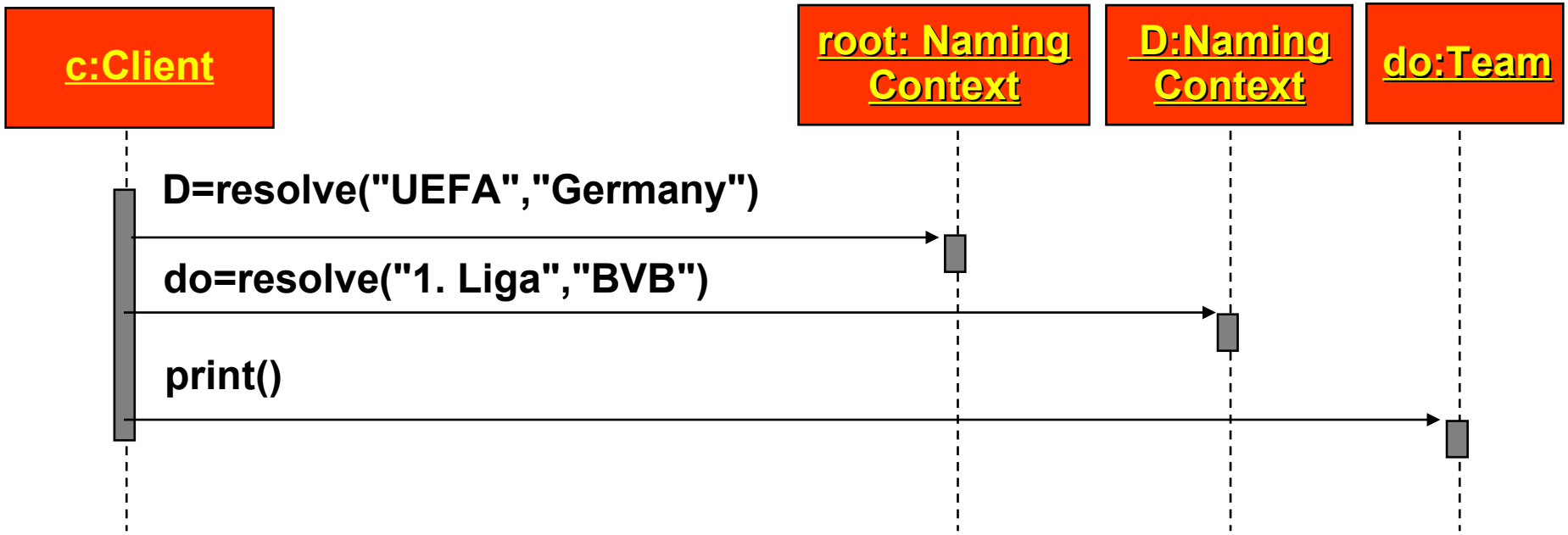
Como obter o Contexto de Nomes Raiz?

Através da Interface do ORB

```
module CORBA {
 interface ORB {
   typedef string ObjectId;
   typedef sequence <ObjectId> ObjectIdList;
   exception InvalidName{};
   ObjectIdList list_initial_services();
   Object resolve_initial_references
                    (in ObjectId identifier)
            raises(InvalidName);
 }
}
```

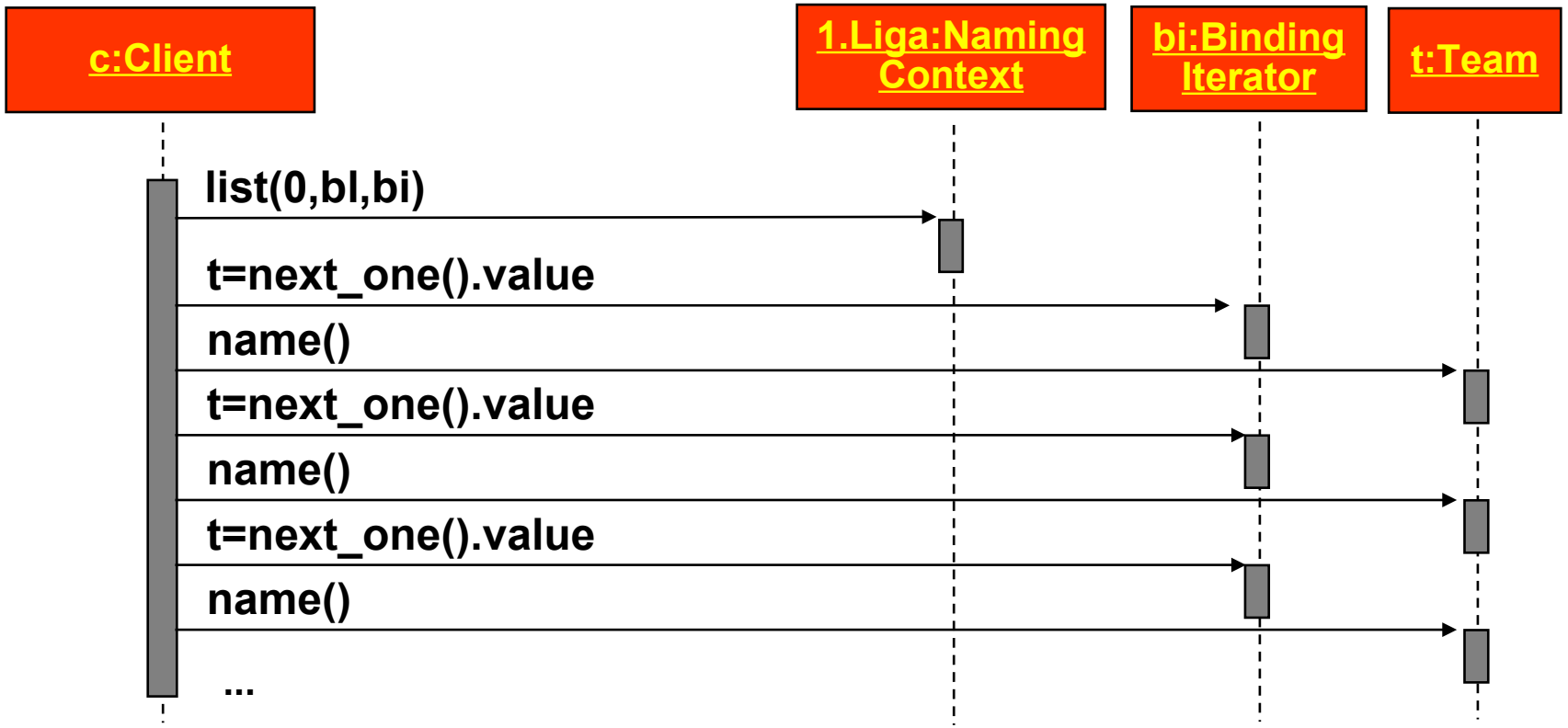
Um Cenário para o Serviço de Nomes: Resolução

Exemplo: Imprimir o elenco de um time



Cenário de uso do Serviço de Nomes: Iteração

Exemplo: Imprimir os nomes de todos os times do campeonato



O Serviço de Nomes em Java RMI: *Registry*

Versão simplificada do Serviço de Nomes de CORBA

Sem suporte para nomes compostos

Restrição de segurança: ligações de nomes não podem ser criadas a partir de máquinas remotas

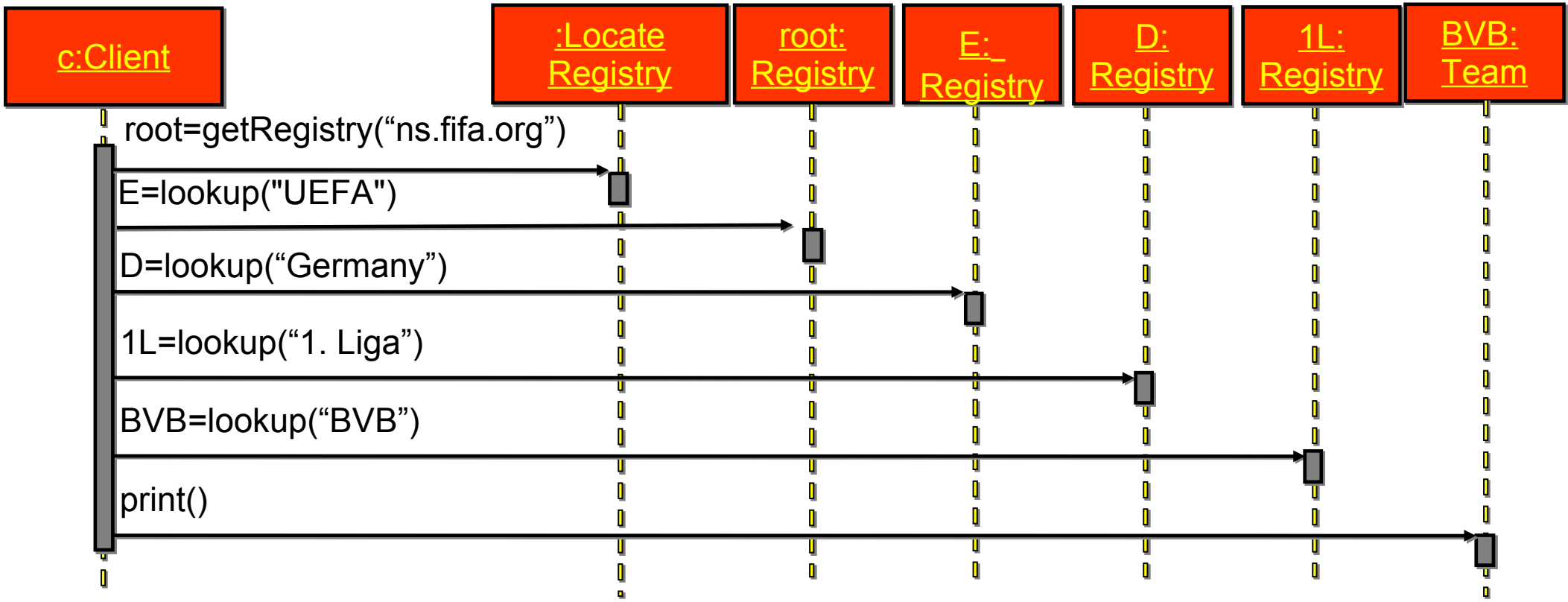
Deve haver um *registry* em cada máquina

Diferentes *registries* devem estar integrados dentro de um *espaço de nomes federado*

RMI Registry

```
package java.rmi.registry;
public interface Registry extends java.rmi.Remote {
 public static final int REGISTRY_PORT = 1099;
 public java.rmi.Remote lookup(String name)
                  throws java.rmi.RemoteException,
                         java.rmi.NotBoundException,
                         java.rmi.AccessException;
 public void bind(String name, java.rmi.Remote obj)
                  throws java.rmi.RemoteException,
                         java.rmi.AlreadyBoundException,
                         java.rmi.AccessException;
 public void rebind(String name, java.rmi.Remote obj)
                  throws java.rmi.RemoteException,
                         java.rmi.AccessException;
 public void unbind(String name)
                  throws java.rmi.RemoteException,
                         java.rmi.NotBoundException,
                         java.rmi.AccessException;
 public String[] list()
                  throws java.rmi.RemoteException,
                         java.rmi.AccessException;
```

Usando o RMI Registry



Avaliação

A ausência de nomes hierárquicos aumenta o número de operações remotas necessárias para a resolução de nomes

A descoberta do *registry* raiz não é necessariamente transparente de localização

Limitações de Serviços de Nomes

Em todas as abordagens mencionadas:

Os clientes sempre devem identificar o servidor específico por meio de um nome

Inapropriado se o cliente apenas deseja usar um *serviço* com certas características e qualidades, mas não sabe (ou não precisa saber) em que servidor:

Comércio eletrônico

Vídeo sob demanda

Venda automática de bilhetes de cinema