

# **Распределенные вычислительные системы**

## **Лекция №7: Определение местонахождения распределенных объектов**

Алексей В. Бурдаков, к.т.н.  
[burdakov@usa.net](mailto:burdakov@usa.net)

# План лекций

---

№	Дата	Тема
1	04.09	Вводная лекция
2	11.09	Эволюция распределенных технологий
3	18.09	Принципы ПО среднего слоя
4	25.09	Стандарты OMG, CORBA и ORB
5	02.10	Пример приложения на CORBA
6	09.10	COM, Java/RMI
7	16.10	Определение местонахождения
8	23.10	Перенос (?)
9	30.10	Лекция + <u>Рейтинг по Л.1-7</u>
10	06.11	Лекция

# План лекций (продолжение)

---

№	Дата	Тема
11	13.11	Лекция
12	20.11	Лекция
13	27.11	Лекция
14	04.12	Лекция
15	11.12	Лекция
16	18.12	Лекция
17	25.12	Зачет

# Мотивация

---

- Важное свойство распределенной системы – прозрачность расположения
- Пример системы с защитой IP-адресом:
  - Невозможно перенести сервер на другой узел
- Объектная ссылка – универсальный механизм адресации объектов
- Каким образом получить объектную ссылку?
  - Именованное (нахождение объектов по логическим именам - «белые страницы»)
  - Трейдинг (нахождение объектов по функциональным и нефункциональным /качественным/ характеристикам – «желтые страницы»)

# План лекции

---

- Именованние объектов
- Объектный трейдинг

# Общие принципы именования

---

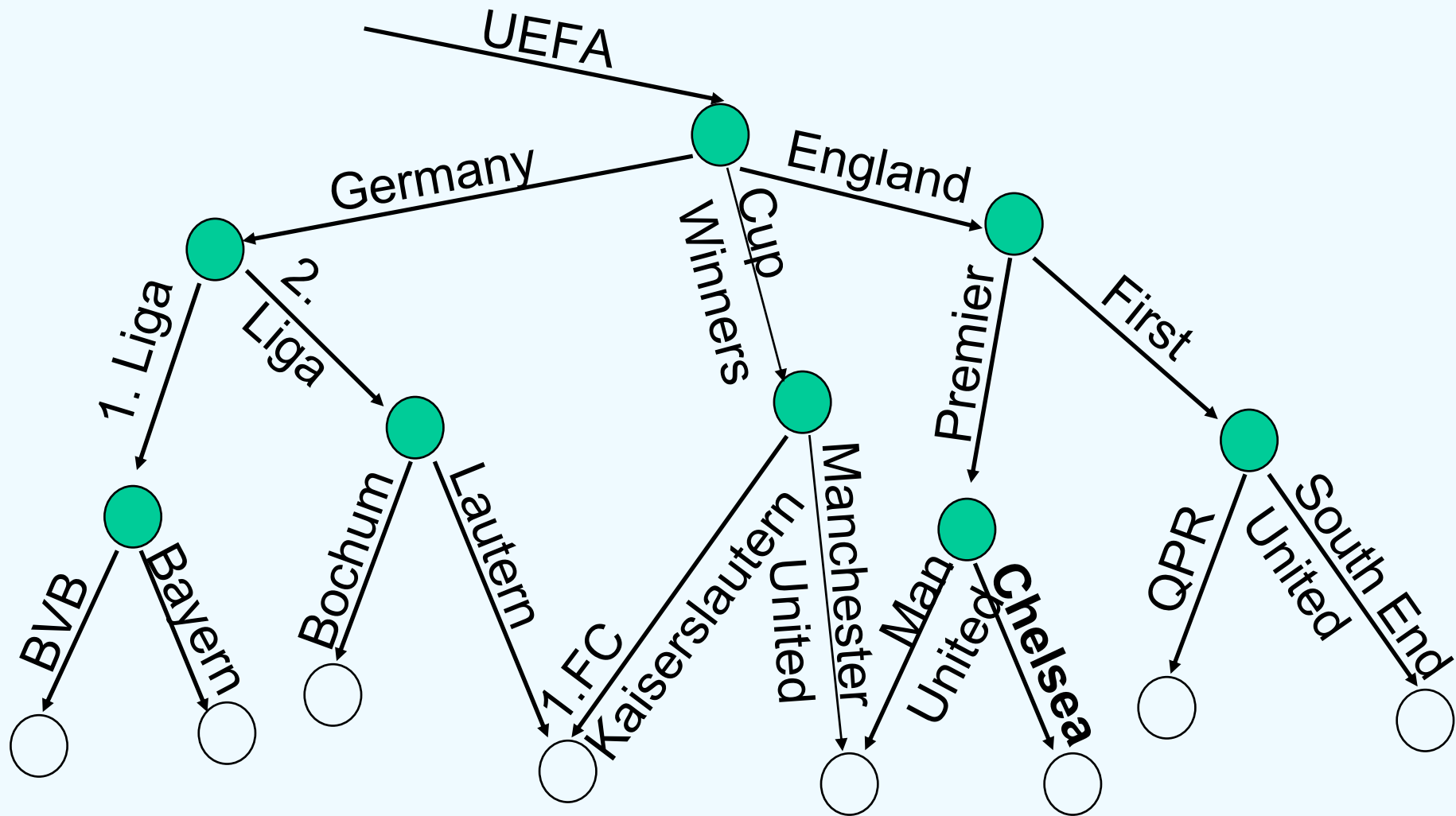
- ОО ПО среднего слоя использует объектные ссылки для адресации серверных объектов
- Необходим способ получения таких ссылок
- Имя (последовательность строк) могло бы быть привязано к объектной ссылке
- По имени могла бы быть получена ссылка

# Общие принципы именования

---

- В распределенной системе может быть множество распределенных объектов
- Серверные объекты могут иметь несколько имен
- Это приводит к большому объему связей имя-ссылка в системе
- Пространство имен д.б. организовано иерархически для того, чтобы избежать:
  - Конфликты имен
  - Деградацию производительности при привязке и получении ссылок
- Иерархия м.б. достигнута с помощью контекстов

# Контексты именования





# Общие принципы именования: композиционные имена

---

- Имена составлены из возможно более чем одного компонента
- Используется для описания навигации через несколько контекстов именования
- Пример:  
("UEFA", "England", "Premier", "Chelsea")

# **Общие принципы именования: поведение сервера именования**

---

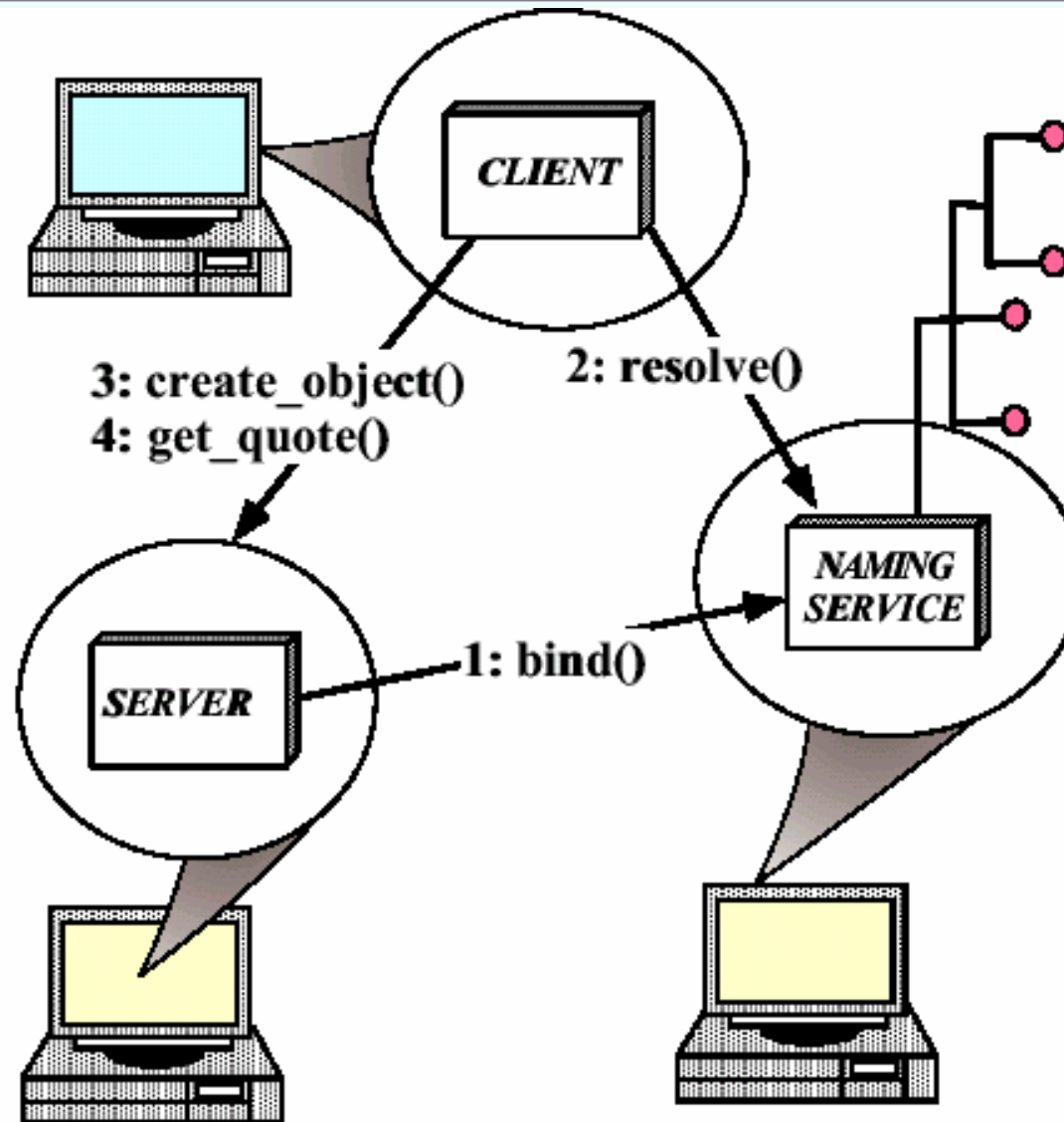
- Привязки имен к ссылке управляются серверами имен
- Не для каждого серверного объекта нужно имя
- Серверные объекты могут иметь несколько имен
- Серверы имен должны долговременно хранить привязку
- Эффективное получение ссылок по именам
- Серверы именования м.б. распределенными

# Сервис именования CORBA

---

- Поддерживает привязку имен CORBA к объектным ссылкам
- Имена сгруппированы по контекстам именования
- Для одной объектной ссылки может быть определено несколько имен
- Не все объектные ссылки нуждаются в именах

# Архитектура сервиса имен



# Имена CORBA

---

- Имена состояются из простых имен
- Простые имена представляют собой пары «значение-тип»
- Атрибут «значение» используется для разрешения имен
- Атрибут «тип» (kind) используется для предоставления информации о роли объекта

# IDL типы для имен

---

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

# Интерфейсы IDL

---

- Сервис именования определен двумя IDL интерфейсами
  - `NamingContext` определяет привязки имен к объектам и получения информации по привязке
  - `BindingIterator` определяет набор операций для итераций по набору имен, определенных в контексте именования

# Интерфейс 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);
};
```



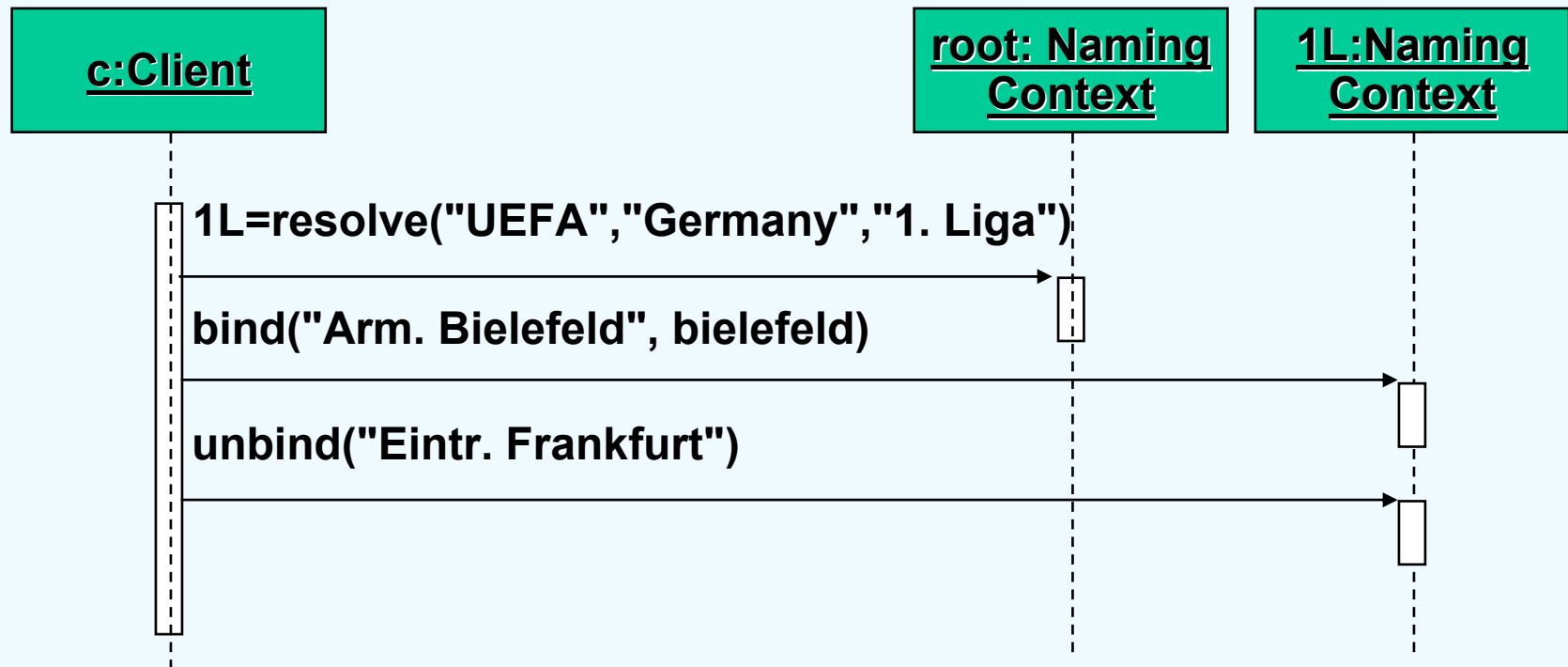
# Интерфейс BindingIterator

---

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

# Сценарии именования: привязка

- Привязать команду Bielefeld к первой лиги, а команду Frankfurt - отвязать



# Начало работы с сервисом ИМЕН

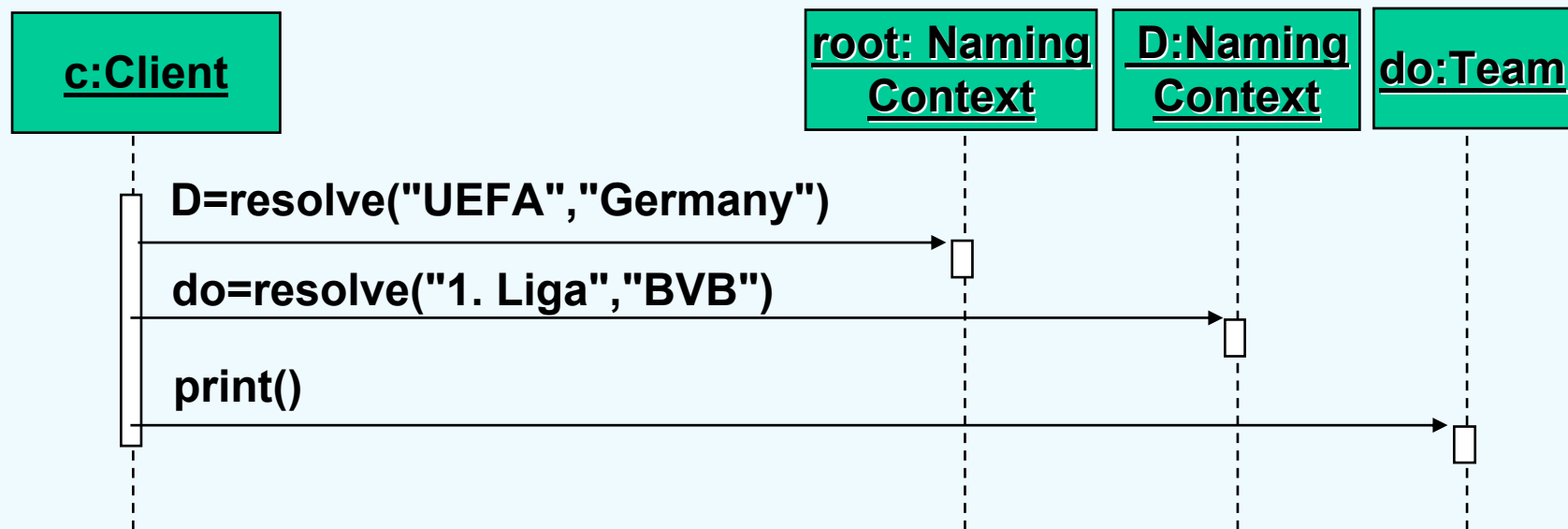
---

- Как получить корневой контекст именования?
- Интерфейс ORB обеспечивает для инициализации следующее:

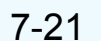
```
module CORBA {  
    interface ORB {  
        typedef string ObjectId;  
        typedef sequence <ObjectId> ObjectIdList;  
        exception InvalidName{};  
        ObjectIdList list_initial_services();  
        Object resolve_initial_references  
            (in ObjectId identifier)  
    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

# Сценарий именования: разрешение

- Распечатать состав Дортмундской Боруссии



- Распечатать все команды 1-й немецкой лиги



# Именованение в COM: клички

---

- Moniker – кличка
- Поддержка привязки и разрешения имен
- Пространство имен м.б. иерархически структурировано

# Именованное в COM: интерфейс IMoniker

---

```
interface IMoniker : IPersistStream {  
    HRESULT BindToObject([in] IBindCtx *pbc,  
        [in, unique] IMoniker *pmkToLeft,  
        [in] REFIID riid,  
        [out, iid_is(riid)] void **ppv);  
    ...  
}
```

# Создание мониторов: IParseDisplayName

---

- Для того, чтобы быть поименованным, серверный объект должен реализовывать интерфейс IParseDisplayName
- Создает объект-монитор разбирая внешнее (текстовое) имя, называемое отображаемое имя

```
interface IParseDisplayName {  
    HRESULT MkParseDisplayName([in] IBindCtx *pbc,  
                               [in,string] const OLECHAR *pwszName,  
                               [out] ULONG *ppchEaten  
                               [out] IMoniker **ppmk);  
}
```



# Оценка

---

- COM имеет внутренние и внешние имена (клички и отображаемые имена), что усложняет именование
- Именование COM переплетено с другими частями спецификации COM (контейнеры)
- Именование COM не прозрачно для разработчиков серверных объектов, т.к. они должны реализовывать `IParseDisplayName`

# Именованние в Java/RMI

---

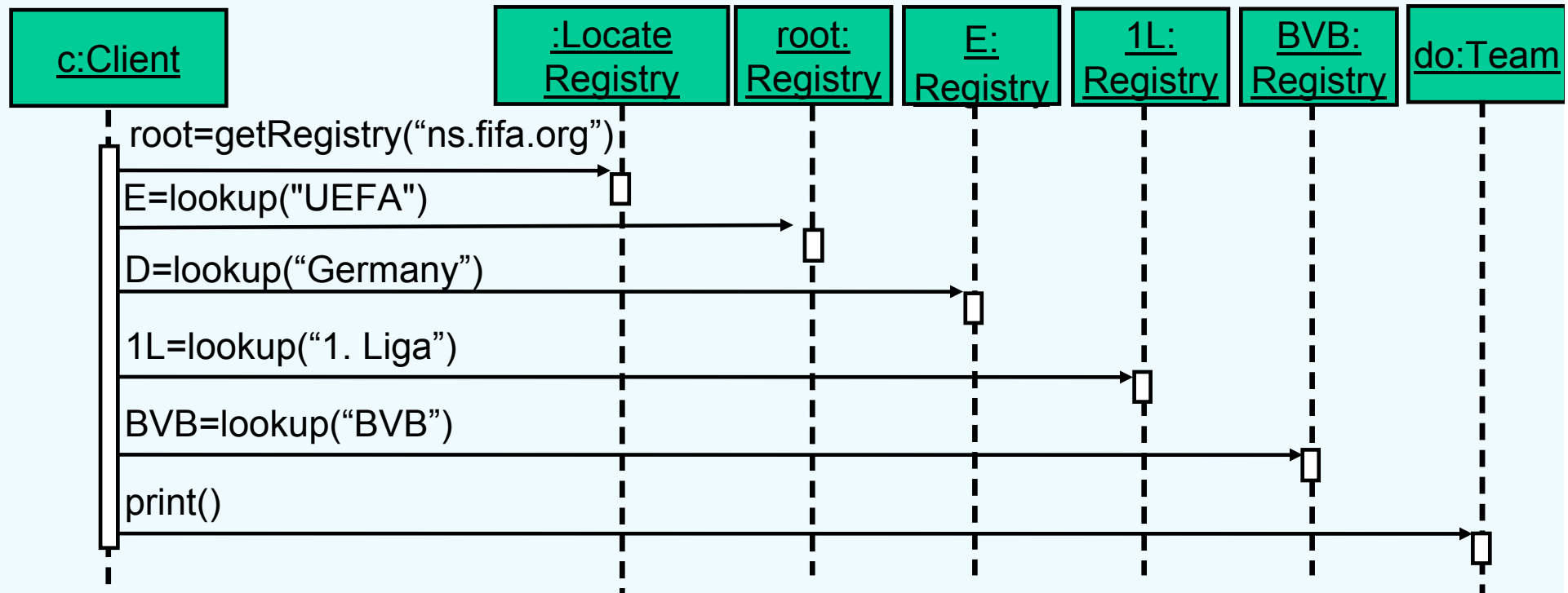
- Упрощенная версия именования в CORBA
- Нет композитных имен
- Ограничения по безопасности: привязки имен не могут быть созданы с удаленных узлов
- На каждом узле д.б. реестр
- Различные реестры должны быть интегрированы в федеративные пространства имен

# Именованное в Java/RMI: реестр

---

```
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;
}
```

# Именованное в Java/RMI: использование реестра



# Именованение в Java/RMI: оценка

---

- Отсутствие иерархии в именовании увеличивает кол-во удаленных операций необходимых для разрешения имен
- Поиск корневого узла не всегда прозрачно в отношении размещения
- Ограничения безопасности уничтожает прозрачность размещения

# Именованние: ограничения

---

- Клиент должен знать имя сервера
- Такой способ поиска нужных объектов не подходит, если клиент ищет сервис с определенными функциями и качеством, но не знает имя сервиса

# План лекции

---

- Именованние объектов
- Объектный трейдинг

# Характеристики трейдер-сервиса

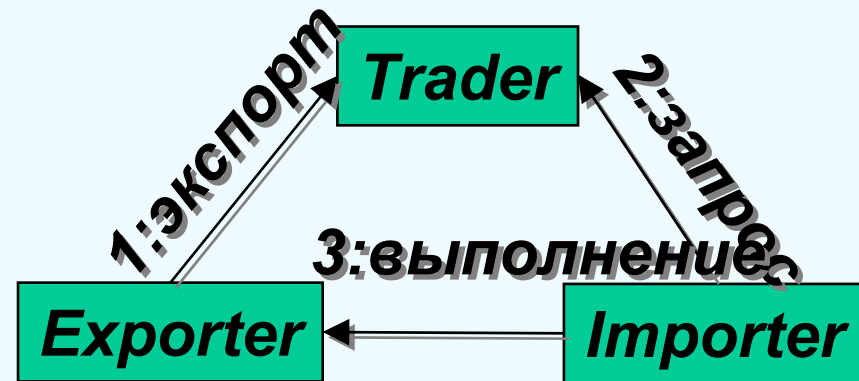
---

- Обнаружение объектов независимо от их физического расположения
- Именованное не приемлемо когда клиент:
  - Не знает имя сервиса
  - Существует несколько сервисов с одинаковыми функциями
- Трейдер-сервис поддерживает обнаружение серверов по функциональности и качеству предоставляемого сервиса
- Именованное ↔ Белые страницы
- Трейдер-сервис ↔ Желтые страницы



# Характеристики трейдер-сервиса

- Трейдер-сервис работает в качестве посредника между клиентом и сервером
- Позволяет клиенту менять перспективу с «кто?» на «что?»



- Та же самая идея:
  - биржевой брокер

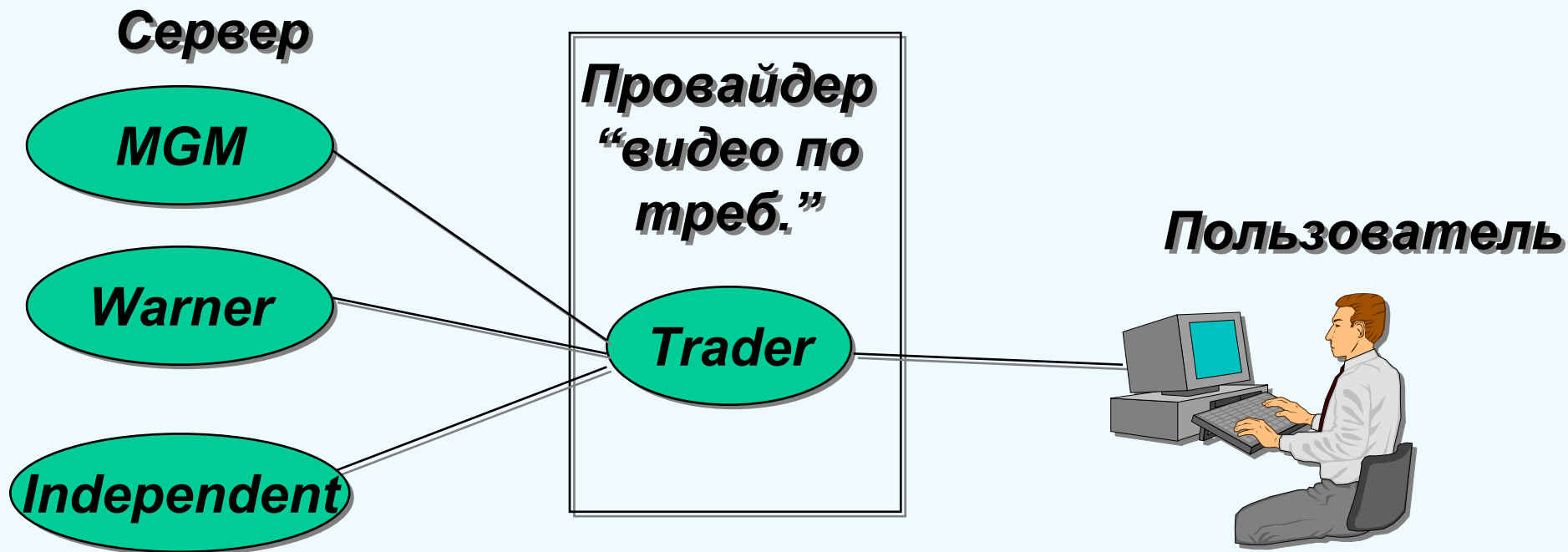
# Характеристики трейдер-сервиса

---

- Общий язык между клиентом и сервером:
  - Тип сервиса
  - Качество сервиса
- Сервер регистрирует сервис в трейдере
- Клиент запрашивает трейдер:
  - получить сервис определенного типа
  - сервис определенного качества
- Трейдер поддерживает:
  - сопоставление сервисов
  - «торговлю сервисами»

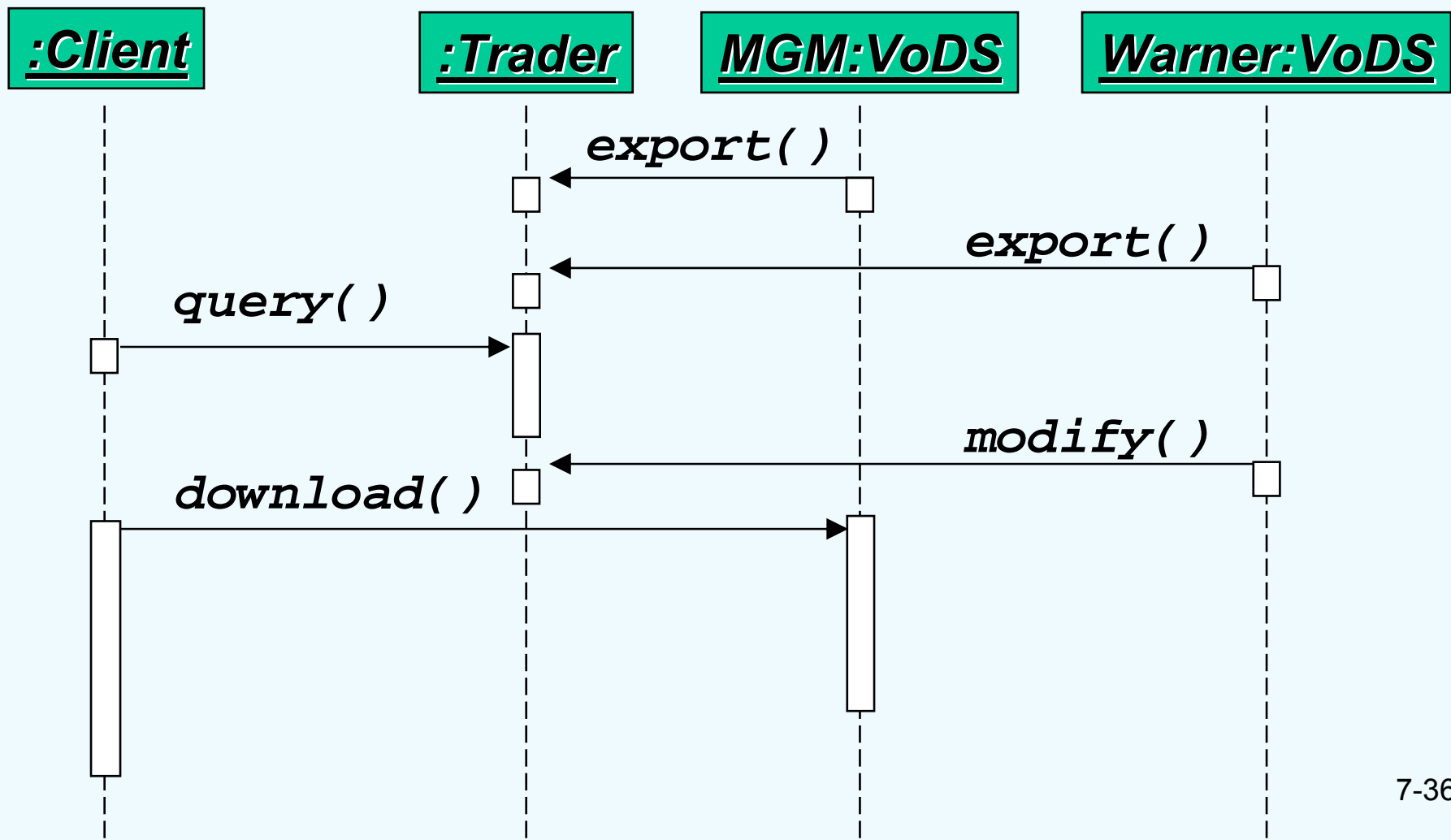
# Пример

- Сервер «видео по требованию»



# Процесс «трейдинга»

- Сервер «видео по требованию»



# Определение типа сервиса

---

- Определение типа сервиса
  - функциональность, определенная сервисом
  - характеристика качества сервиса (QoS)
- Функциональность определяется типом объекта
- QoS определена свойствами:
  - имя свойства
  - тип свойства
  - значение свойства
  - режим работы
    - обязательный/не обязательный
    - только для чтения/модифицируемый

# Пример: тип сервиса

---

```
typedef enum {VGA,SVGA,XGA} Resolution;  
service video_on_demand {  
    interface VideoServer;  
        readonly mandatory property float fee;  
        readonly mandatory property Resolution  
        res;  
        modifiable optional property float  
        bandwidth;  
}
```

# Определение ограничений

---

- Импортер сервиса определяет желаемые качества сервиса в качестве части запроса:
- Пример:

`fee<10 AND res >=SGA AND bandwidth>=256`

- Трейдер выдает в качестве ответа на запрос только те сервисы, которые удовлетворяют ограничениям

# Политики трейдинга

---

- В зависимости от ограничений и доступных сервисов, большой набор предложений может быть возвращен в качестве результата
- Политики трейдинга могут использоваться для ограничения количества найденных предложений
  - Указание максимального числа результатов
  - Ограничения по замещению сервисов
  - Ограничения по модифицируемым свойствам

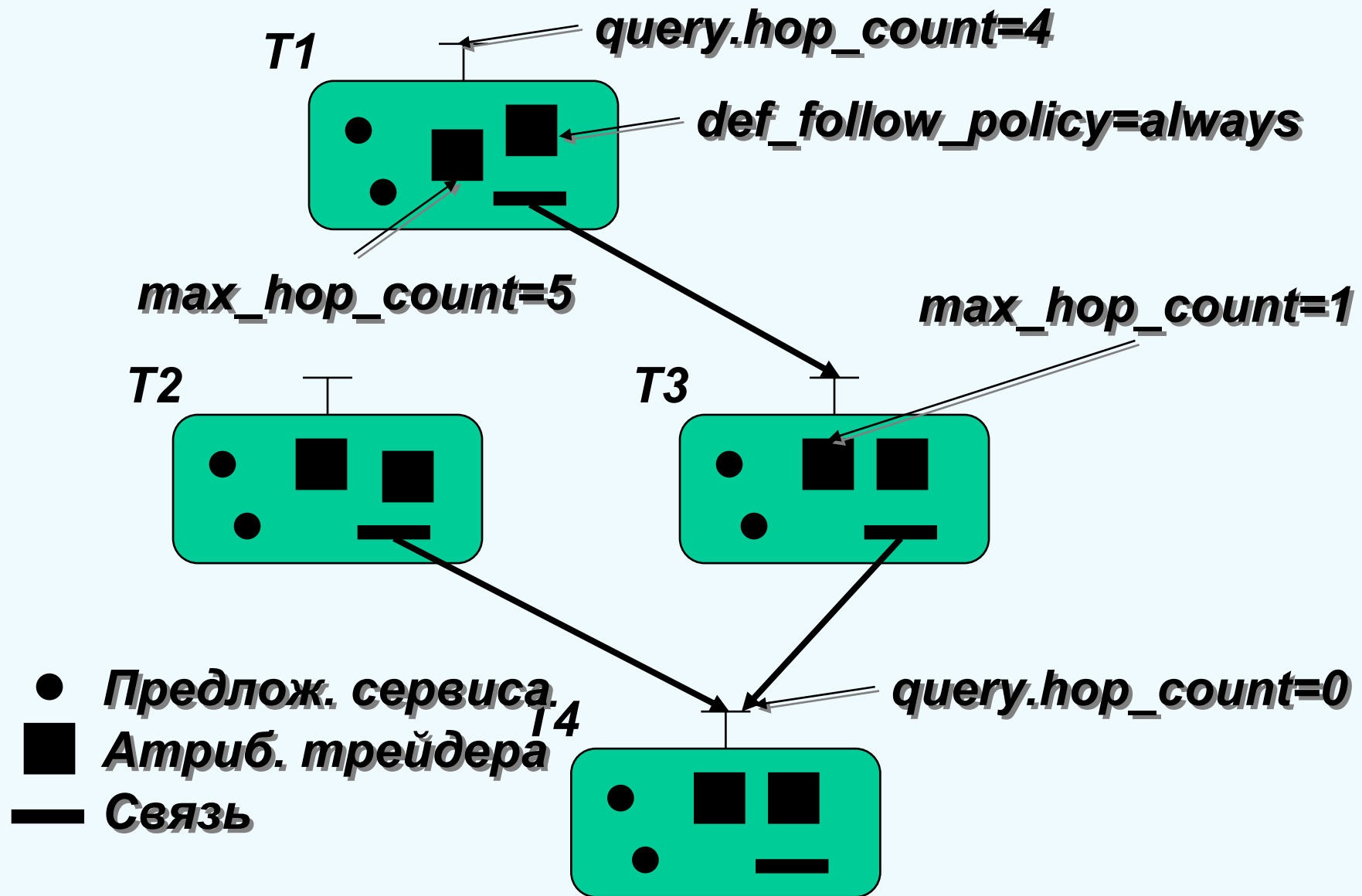


# Федерации трейдеров

---

- Масштабируемость приводит к необходимости создания федераций трейдеров
- Трейдер, участвующий в федерации:
  - предлагает информацию о сервисах другим трейдерам
  - запрашивает других трейдеров в случае если он не может удовлетворить запрос
- Проблемы
  - непрерывный импорт
  - дублирование предложений

# Граф процесса трейдинга



# Трейдинг и именование: ключевые моменты

---

- Распределенные объекты могут быть найдены с помощью именования и трейдинга
- Именование привязывает известные имена к объектным ссылкам и поддерживает разрешение имен для получения объектной ссылки
- Трейдинг поддерживает обнаружение объектов на основе их функциональности, которую они предлагают и на основе качества сервиса, которую они гарантируют

# Литература / Internet ИСТОЧНИКИ

---

- В. Эммерих *Конструирование распределенных объектов*. - М.:Мир. - 2002.
- Vinoksi S. *CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments*, IEEE'96.
- Schmidt D.C. And Vinoski S. *Object Adapters: Concepts and Technology*, SIGS C++ Report, Vol. 9, No. 11, Nov-Dec 1997.
- Ю.А. Григорьев, А.Д. Плутенко. *Жизненный цикл проектирования распределенных баз данных*. - Благовещенск. - 1999.
- [www.omg.org](http://www.omg.org)