Gestão de Nomes

Gestão de Nomes: Objetivo

- Associar **nomes** a **objetos**
- Objetos podem ser computadores, serviços, objetos remotos, ficheiros, utilizadores, etc.
- Nomes facilitam comunicação e partilha de recursos
  - Necessários quando de faz pedido a um sistema para atuar sobre um determinado recurso (de entre vários)
    - Exemplo: URL para abrir página
  - Permitem partilhar recursos
    - Exemplo: objeto remoto
  - Permitem comunicar
    - Exemplo: endereço email permite a utilizadores trocarem mensagens
  - Permitem associar atributos descritivos a recursos, e fazer procuras baseadas em atributos
    - Exemplo: procurar impressora a cores na rede local

Gestão de Nomes: Objetivo

- Associar **nomes** a **objetos** para:
  - Identificar os objetos
  - Localizar os objetos
  - Partilhar os objetos
  - Obter atributos associados ao objeto
  - Simplificar a interface com os utentes
  - Simplificar a gestão do sistema

Exemplos de Nomes

- Nome ficheiro
- URL
- UUID

- Número de Telefone
- Matrícula de Automóvel
- Número do Cartão do Cidadão

- Nome de uma Empresa
- Nome de um Produto

Exemplos de Utilização

- Resolver para encontrar o objeto
  - Endereço IP
  - Nome DNS
  - Número de Telefone
  - URL
  - Nome de Ficheiro
- Resolver para obter um atributo do objeto
  - Servidor de email de um domínio DNS
  - Morada associada a uma entrada UDDI
  - Nome da Empresa
- Resolver para verificar se o objeto é o mesmo
  - Número do Cartão do Cidadão
  - Nome de um Produto

Conceitos Base

- **Espaço de Nomes** → conjunto de regras que define um universo de nomes admissíveis

- **Autoridade** → gere o recurso que suporta a implementação do objeto
  - Define as regras de gestão dos identificadores
  - Deve garantir que as regras de gestão de nomes são cumpridas
  - Autoridade pode ser delegada (hierarquias)

Gestão de Nomes: Nomes e Autoridades

Nome	Autoridade
Endereço IP	IANA (*Internet Assigned Numbers Authority*)
Endereço Ethernet	Xerox e fabricante da placa
Endereço do controlador de disco	Configuração do computador
Nome de um ficheiro	Sistema de ficheiros
UUID	DCE; IETF standard
Nome DNS	IANA/ICANN + delegação (FCCN em Portugal)

Gestão de Nomes: Conceitos Base

- **Identificador** → mecanismo de discriminação de um objeto
  - Sob controlo do **sistema**
    - Sem carga semântica para os humanos
    - Sequências de bits
  - Se o identificador permitir encontrar diretamente o objeto é normalmente designado por **endereço** (um endereço pode deixar de referenciar o objeto se este mudar de localização)

- **Nome** → mecanismo de discriminação de um objeto
  - Usado por **humanos**
    - Programadores, utentes, etc.
    - Com carga semântica para os humanos
    - Sequências legíveis de caracteres
  - Permite normalmente obter um identificador para o objeto

- Nomes vs. Identificadores: nomes representam marcas → gera conflitos
  Exemplo: quem detém o nome nissan.com?

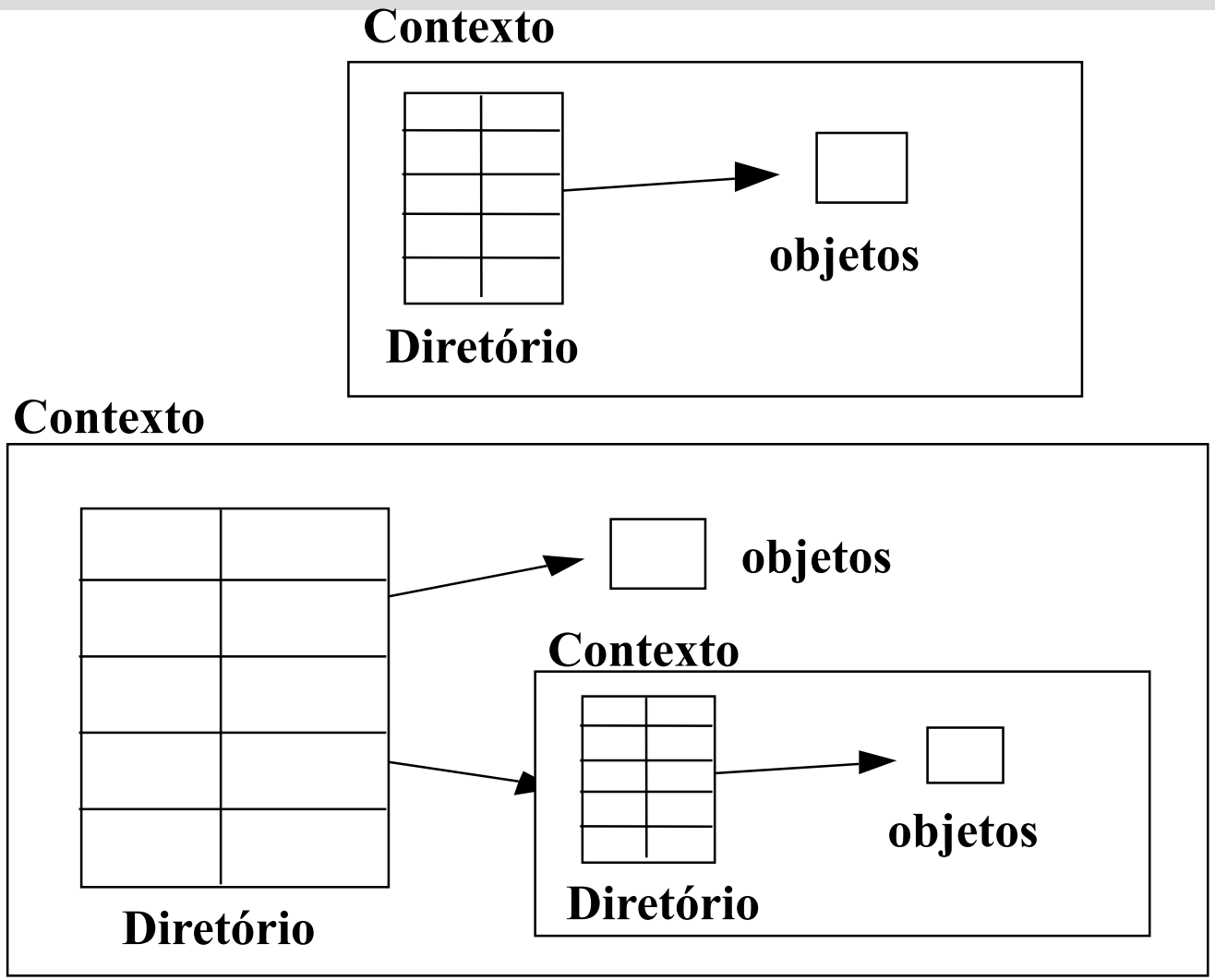
Associações nome→nome e nome→objeto (*bindings*)

- O nome que identifica um objeto raramente é o identificador que permite aceder-lhe
- A associação nome→objeto é **lógica**
- A partir do nome de um objeto existe uma cadeia de associações entre nomes, geralmente de espaços de nomes diferentes
  - Exemplo:
    - Nome de ficheiro UNIX
      a/b/c → i-number → inode → partição e bloco de disco
    - Nó da rede Internet
      www.tecnico.ulisboa.pt → endereço IP → endereço Ethernet

Gestão de Nomes: Conceitos Base

- **Contexto** → conjunto de associações pertencentes a um determinado espaço de nomes
  - Define domínio em que se consideram válidos um determinado conjunto de nomes
- **Diretório** → tabela (ou conjunto de tabelas) que materializa(m) as associações entre nomes e objetos de um contexto
  - Um diretório também é um objeto que tem de ter um nome associado

Contexto vs. Diretório



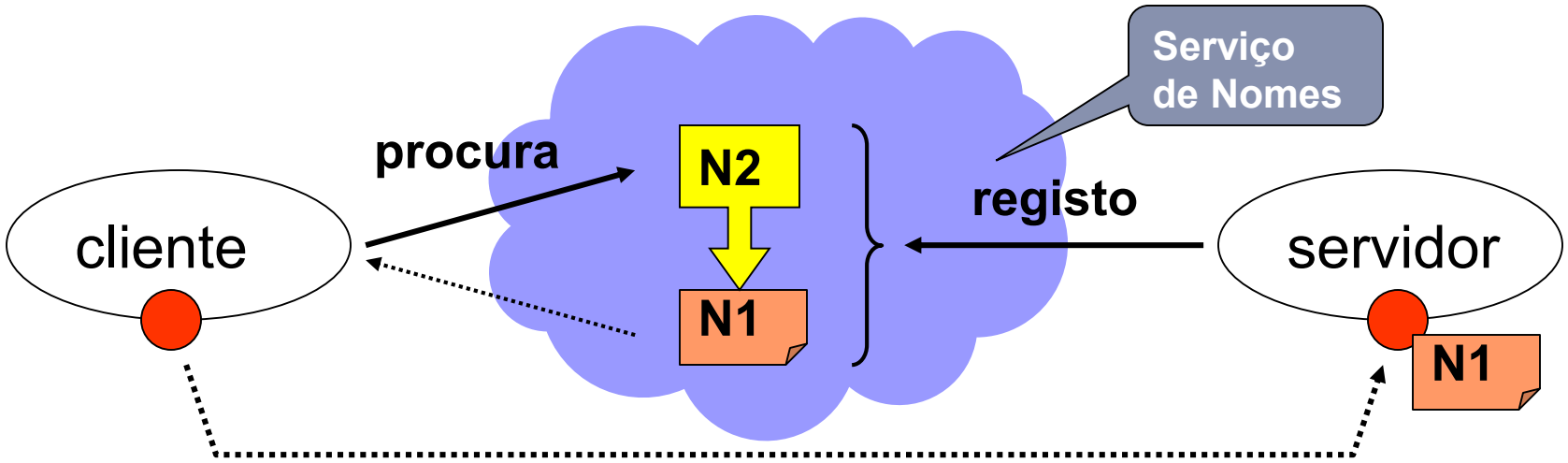
Exemplos de Serviços de Nomes

Sun RPC, Java RMI, Web Services

RPCBIND

RPC: Serviço de Nomes

- Permite que o servidor registe um nome de um serviço
  - Que tem de estar associado ao identificador de um porto de transporte
- Permite que um cliente consiga encontrar o servidor através do nome do serviço.
  - Obter o nome do seu porto de transporte



O registo tem de ser efectuado pelo servidor

- Registo
  - Escolha da identificação do utilizador
    - Nome do porto de transporte
    - Outros nomes alternativos
  - Registo dessa identificação
- {Esperar por pedidos de criação de sessões}*
  - Estabelecimento de um canal de transporte
  - Autenticação do cliente e/ou do servidor
- {Esperar por invocações de procedimentos}*
  - Enviados pelos clientes ligados
- Terminação da sessão
  - Eliminação do canal de transporte

O binding tem de ser efectuado pelo cliente

- Estabelecimento da sessão – vinculação ao servidor (binding)
  - Localização do servidor
  - Autenticação do cliente e/ou do servidor
  - Estabelecimento de um canal de transporte

{Chamada de procedimentos remotos}* - efetua os RPC necessários

- Terminação da sessão
  - Eliminação do canal de transporte

Referências de sessão – binding handles

- Cliente
  - Criação do *binding handle* no extremo cliente
    - Identifica um canal de comunicação ou
      um porto de comunicação para interatuar com o servidor
- Servidor
  - Possível criação de um *binding handle* no extremo servidor
    - Útil apenas se o servidor desejar manter estado entre diferentes RPCs do mesmo cliente
    - Um servidor sem estado não mantém *binding handles*

Exemplo Binding : Cliente – Sun RPC

```
void main (int argc, char *argv[]){
  CLIENT *cl;
  int a, *result;
  char* server;
  if (argc < 2) {
      fprintf(stderr, "Modo de Utili
         servidor\n", argv[0]);
      exit(1);
  }
  server = argv[1];
  cl = clnt_create(server, BANCOPROG, BANCOV
  if(cl == NULL) {
      clnt_pcreateerror(server);
      exit(1);
  }
  sresult = saldo_1(nconta, cl);
}
```

A função retorna um binding handle

A chamada ao procedimento remoto explicita o binding handle

RMI Registry

Divulgação e descoberta de objectos

- O cliente precisa de obter referência remota para um primeiro objeto no servidor
  - A partir da qual poderá invocar métodos e receber outras referência remotas para outros objetos remotos
- O *binder* é um serviço distribuído que permite:
  - Servidores registarem objetos remotos neles instanciados
    - Cada objeto registado tem um nome
  - Clientes podem consultar o *binder* por nome de objeto
    - Se registado, o *binder* devolve referência remota

RMI Registry

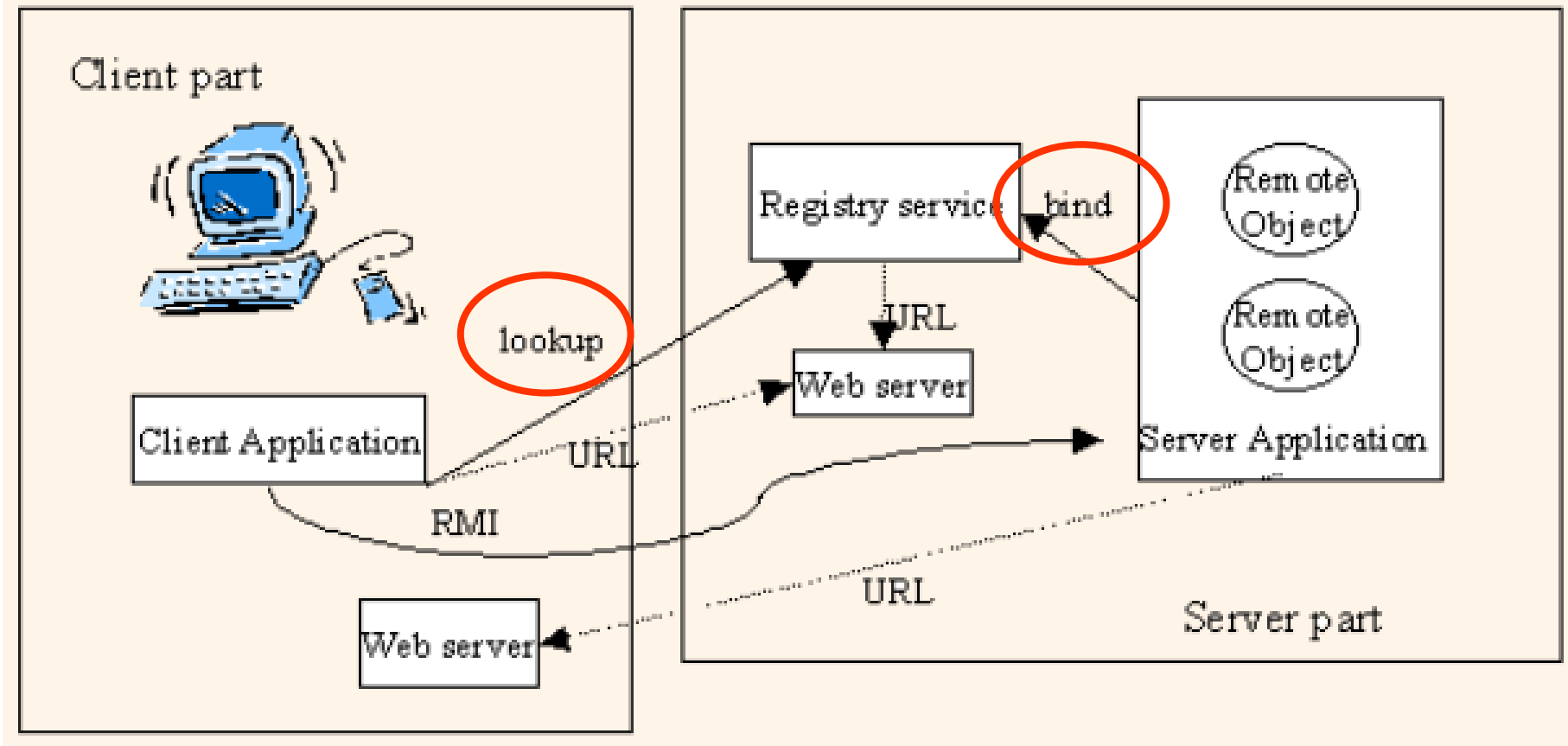
- Corresponde ao *binder* em Java RMI
  - Utilizado para associar nomes a recursos e objetos de forma portável
  - Permitindo identificação, localização, partilha
- Normalmente corre um processo RMI registry em cada máquina
  - Gere os objetos remotos registados dessa máquina
- Nome é do tipo *//hostname:port/nomeObjecto*
  - Em que hostname:port se refere à máquina e ao porto onde o nome está registado

RMI Registry

- Implementa a interface JNDI (*Java Naming and Directory Interface*)

void rebind (String name, Remote obj)
This method is used by a server to register the identifier of a remote object by name.
void bind (String name, Remote obj)
This method can alternatively be used by a server to register a remote object by name, but if the name is already bound to a remote object reference an exception is thrown.
void unbind (String name, Remote obj)
This method removes a binding.
Remote lookup(String name)
This method is used by clients to look up a remote object by name. A remote object reference is returned.
String [] list()
This method returns an array of Strings containing the names bound in the registry.

Bind, lookup


Client part
Registry service
bind
Remote Object
Remote Object
lookup
URL
Web server
Client Application
URL
RMI
Server Application
URL
Web server
Server part

Universal Description Discovery & Integration (UDDI)

Universal Description Discovery & Integration (UDDI)

- Definição de um conjunto de serviços que suportam a descrição e a localização de:
  - Entidades que disponibilizam Web Services (empresas, organizações)
  - Os Web Services disponibilizados
  - As interfaces a serem usadas para aceder aos Web Services
- Baseada em standards Web: HTTP, XML, XML Schema, SOAP
- Norma definida por um consórcio alargado: Accenture, Ariba, Commerce One, Fujitsu, HP, i2 Technologies, Intel, IBM, Microsoft, Oracle, SAP, Sun e Verisign
- Versão actual: UDDI Version **3.0.2**, http://uddi.org/pubs/uddi_v3.html

Informação representada na UDDI

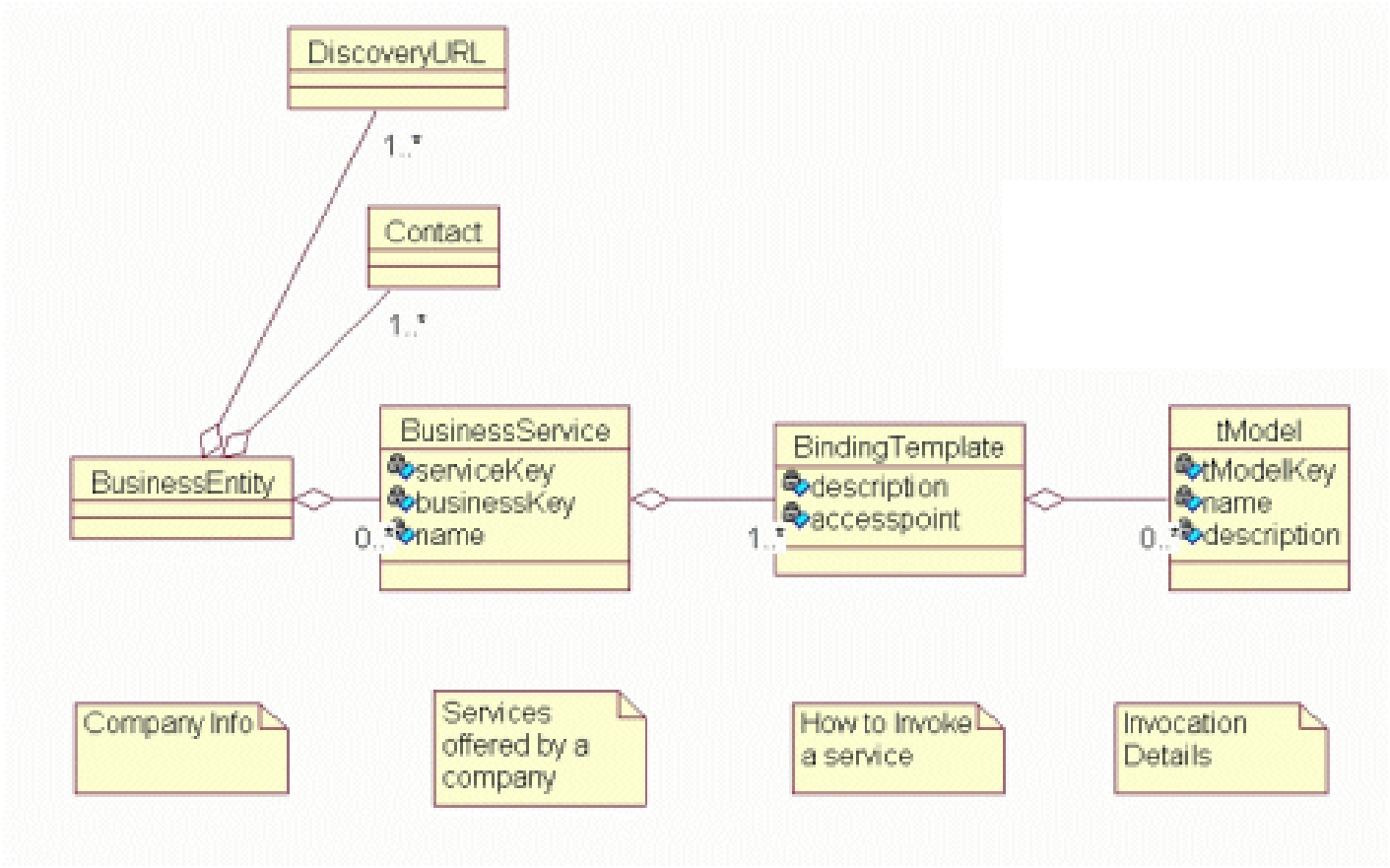
- A UDDI permite pesquisar informação muito variada sobre os Web Services. Ex:
  - Procurar Web Services que obedeçam a uma determinada interface abstracta
  - Procurar Web Services que estejam classificados de acordo com um esquema conhecido de classificação
  - Determinar os protocolos de transporte e segurança suportados por um determinado Web Service
  - Procurar Web Services classificados com uma palavra-chave

- O acesso é via as API definidas mas os operadores também disponibilizam *sites* para acesso via web

Modelo estrutural da informação

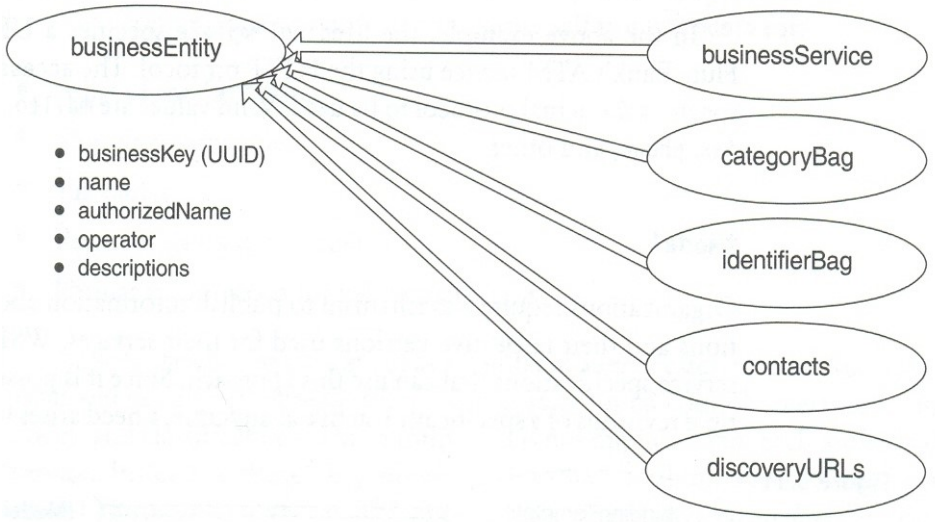
- **businessEntity**: descreve uma empresa ou organização que exporta Web Services
  - A informação encontra-se conceptualmente dividida em:
    - Páginas brancas – Informação geral de contacto
    - Páginas amarelas – Classificação do tipo de serviço e localização
    - Páginas verdes – Detalhes sobre a invocação do serviço
- **businessService**: descreve um conjunto de Web Services exportado por uma businessEntity
- **bindingTemplate**: descreve a informação técnica necessária para usar um determinado serviço
- **tModel**: descreve o "modelo técnico" de uma entidade reutilizável, como um tipo de Web Service, o binding a um protocolo usado por um Web Service, etc.

Modelo Estrutural da Informação



BusinessEntity

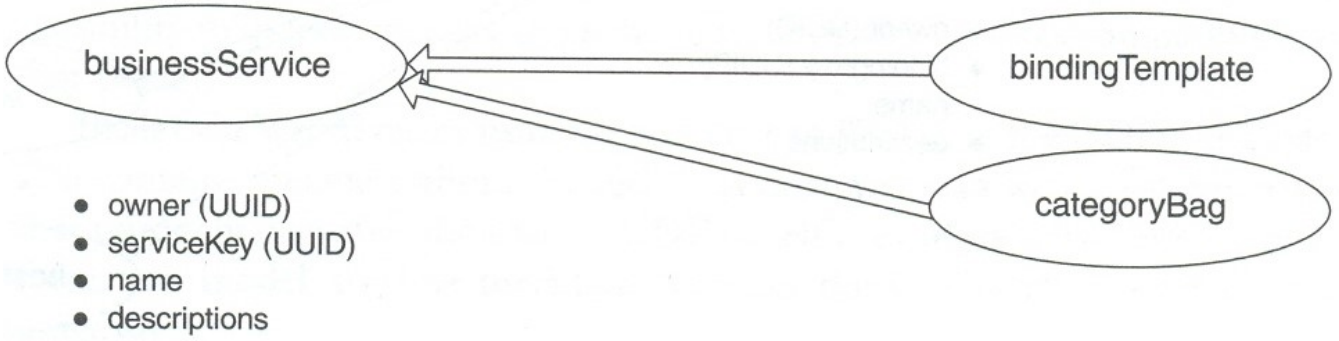
- Conceito mais alto na hierarquia contém informação descritiva sobre a empresa
  - É identificado por uma businessKey que é definida no registo ou criada nessa altura pelo *registry*
  - Tem as referências para os serviços que disponibiliza
  - CategoryBag permite descrever a entidade de acordo com várias taxonomias




```
<categoryBag>
   <keyedReference
        tModelKey="uddi:ubr.uddi.org:categorization:geo3166—2"
         keyName="Connecticut. USA"
         keyVa] ue="US-CT" />
</categoryBag>
```

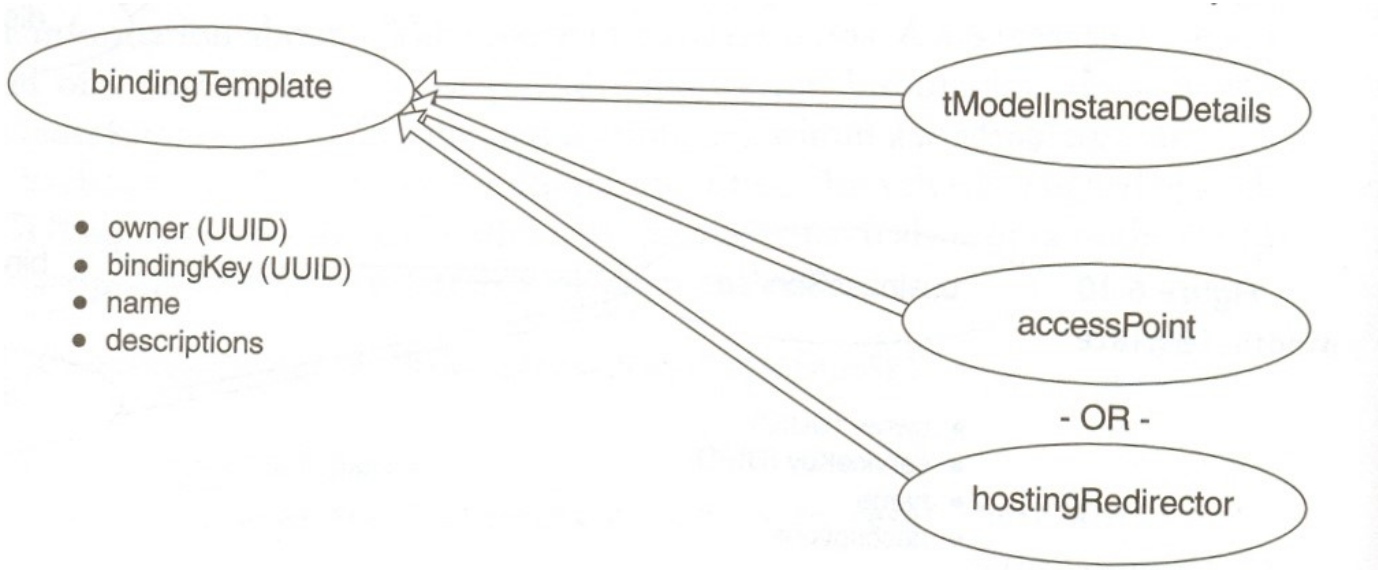
businessService

- Referência um serviço lógico tem a descrição de um Web Service em termos de negócio
  - Tem uma chave própria e uma chave para o businessEntity de que descende
  - CategoryBag serve para descrever o serviço de acordo com múltiplas taxionomias



bindingTemplate

- Contém a informação necessária para que o cliente possa invocar o serviço




```
<bindingTemplates>
 <bindingTemplate bindingKey="" serviceKey="">
    <description>Flute ATM Service via HTTP</description>
    <accessPoint URLType="http">
        http://www.flutebank.com/services/accesspoint>
</bindingTemplates>
```

Taxonomia

- Os serviços registados devem ser categorizados em taxonomias que os permitam pesquisar
- Existem várias taxonomias normalizadas, ex.:
  - ISO 3166 – Geografias
  - D-U-N-S – Data Universal Numbering System – Dun&Bradstreet
  - UN/SPSC – Produtos e serviços – ONU
- O UDDI permite que todas as entidades sejam classificadas
  - As pesquisas podem usar múltiplas classificações
- Para uso interno das organizações podem ser definidos os seus esquemas de classificação
  - Ex.: qualidade de serviço do fornecedor do Web Service

Arquitetura UDDI - *Registries*

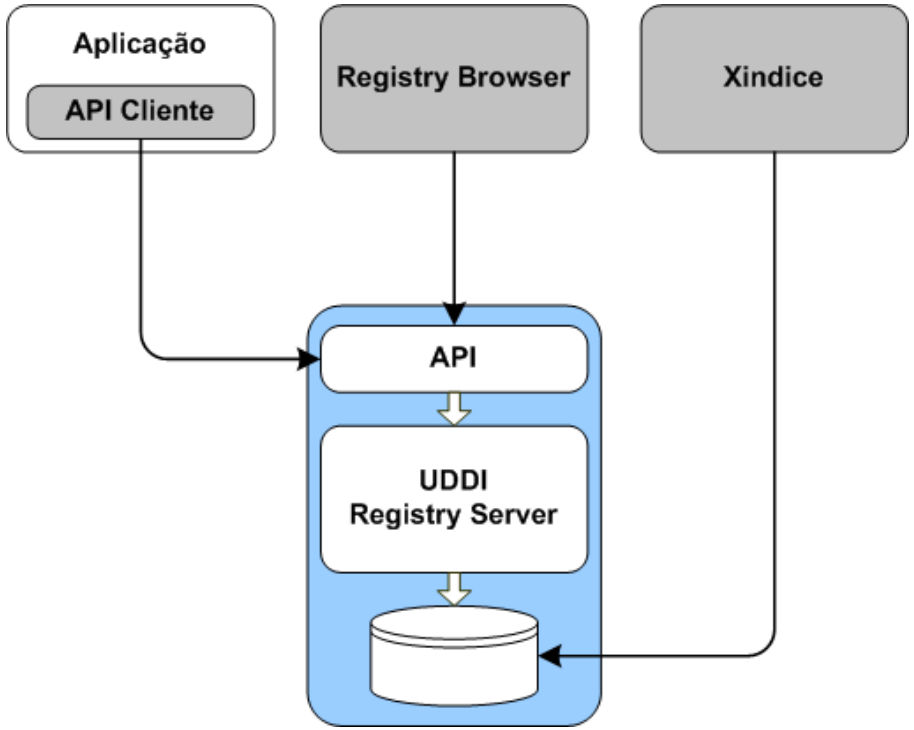
- Um *Registry* é composto por um ou mais nós UDDI
- Os nós de um *Registry* gerem colectivamente um conjunto bem definido de dados UDDI.
  - Tipicamente, isto é suportado com replicação entre os nós do *Registry*
- A representação física de um *Registry* é deixada à escolha das implementações

API UDDI

- API de Mensagens para as funções CRUD (Create, Retrieve, Update, Delete) definidas pelas UDDI Spec

	Business	*Service*	*Binding*	*tModel*
Save/Update	save_business	save_service	save_binding	save_tModel
Delete	delete_business	delete_service	delete_binding	delete_tModel
Find	find_business	find_service	find_binding	find_tModel
GetDetail	get_businessDetail	get_serviceDetail	get_bindingDetail	get_tModelDetail

Registry Server privado

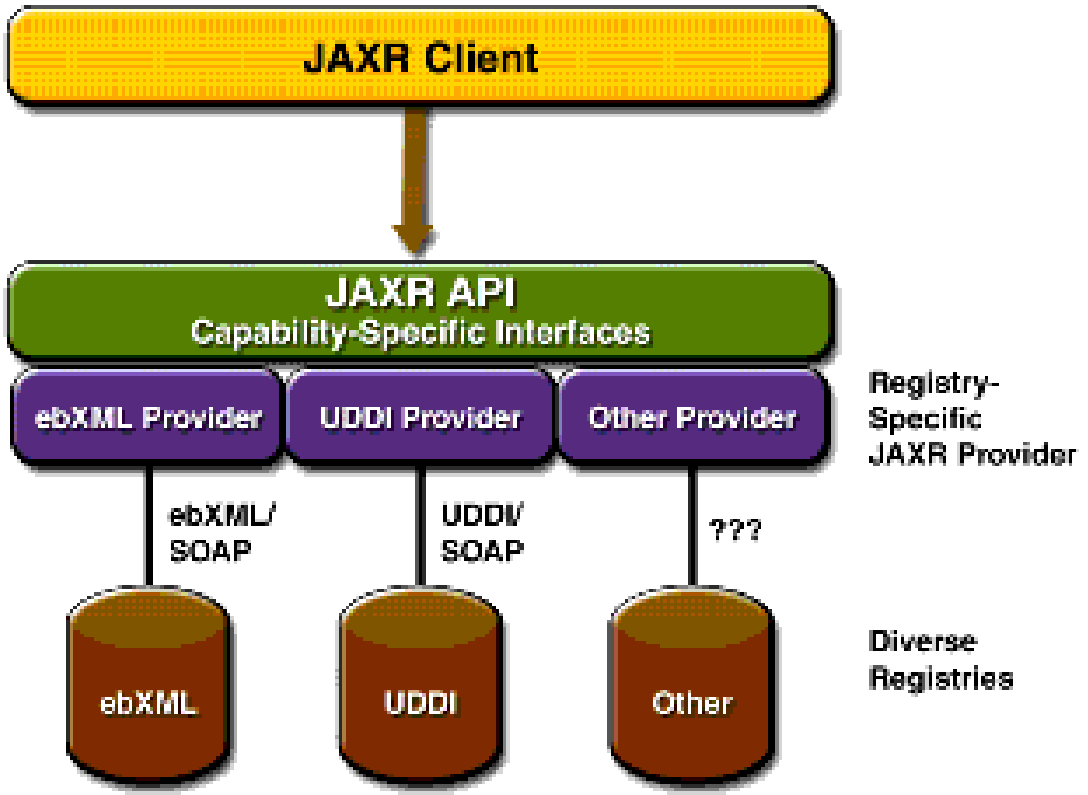


- Gere a base de dados com os registos UDDI
- Modos de acesso
  - Aplicações JAX-R
  - Registry Browser
  - Xindice
- Interfaces
  - API
  - Acesso directo aos registos

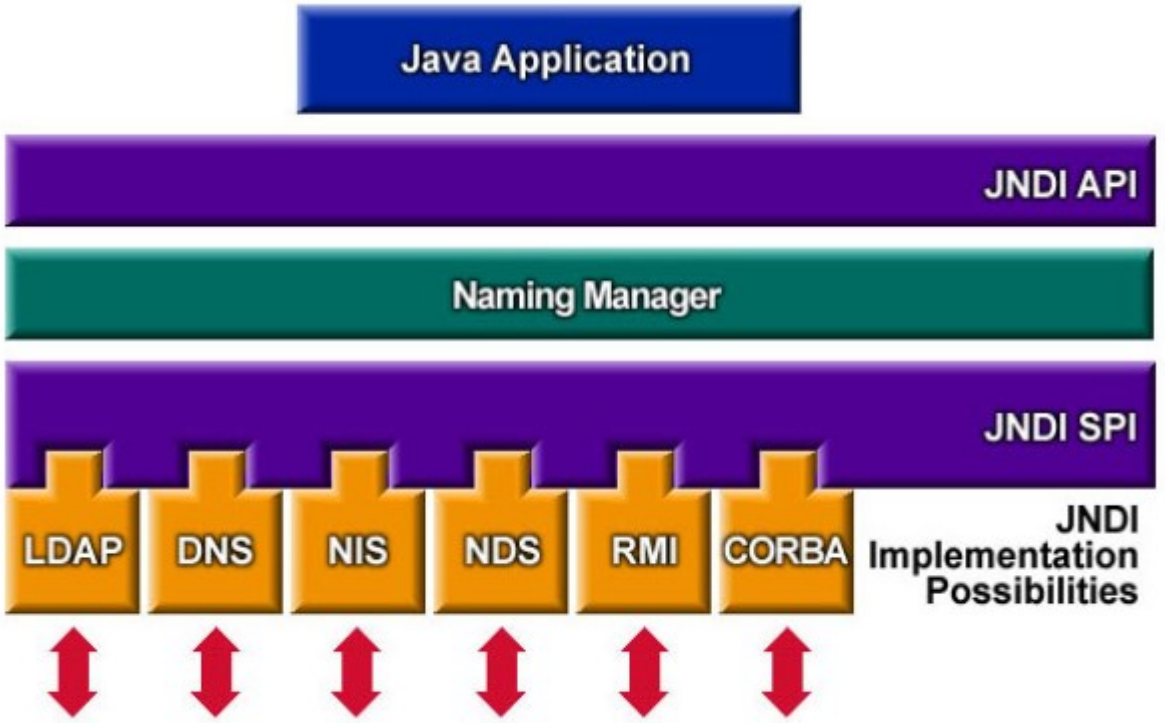
JAX-R

Arquitetura dos clientes

Interface do agente em Java para os diretórios baseados em XML

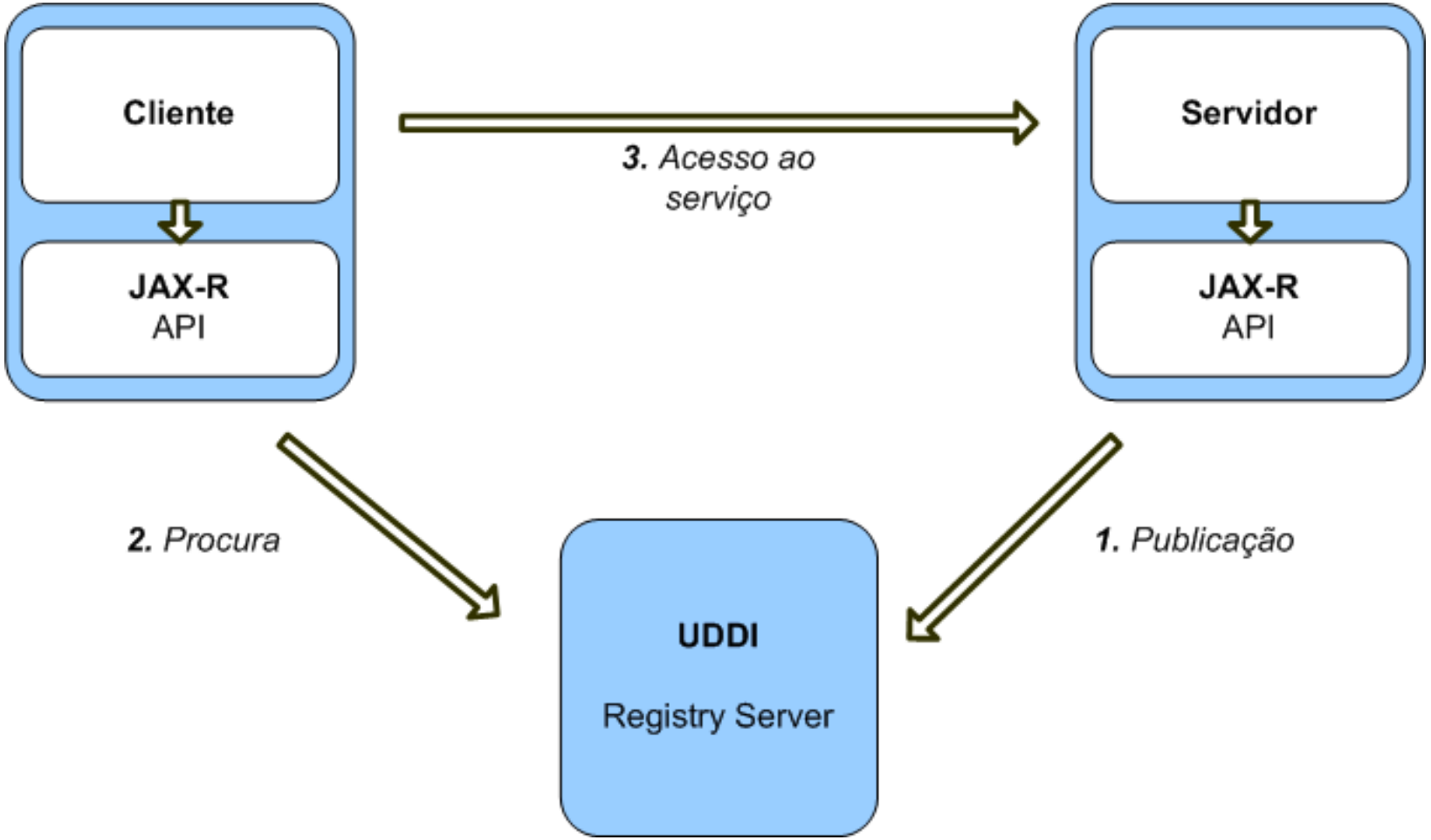


Serviço de nomes Java para outros Espaços de nomes



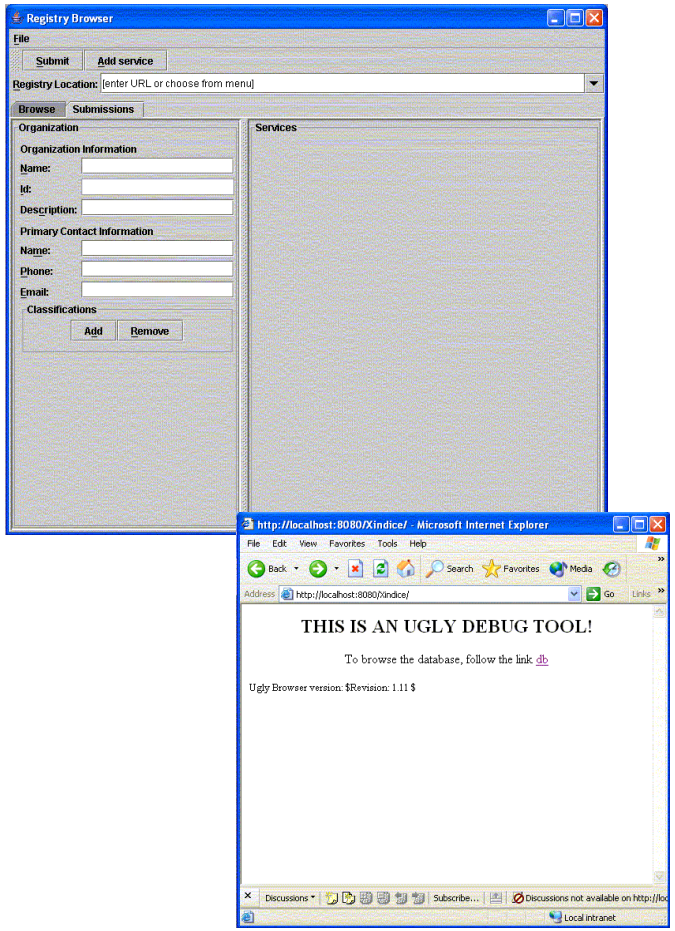
JAX-R

Arquitetura



Registry Server

Ferramentas de Acesso



- Registry Browser
  - Utiliza a API JAX-R

```
$> jaxr-browser
```

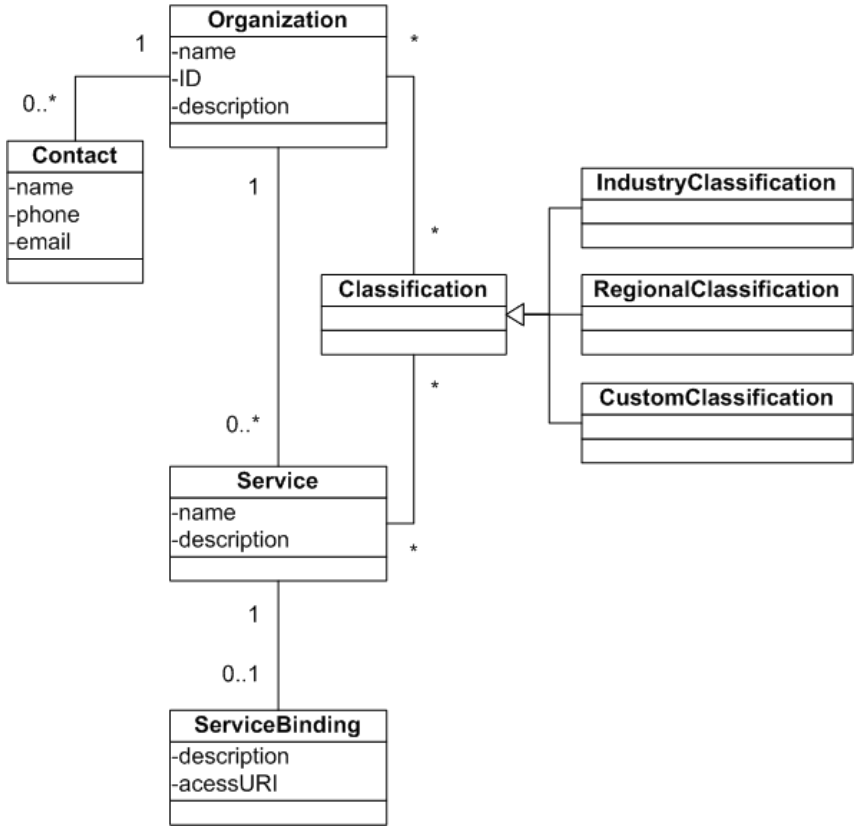
ou

```
Start -> ... -> jwsdp ->
JAX-R Registry Browser
```

- Xindice
  - Acesso directo aos registos da BD

```
http://localhost:8080/Xindice/
```

JAX-R API
Estruturas de dados



- O JAX-R tem um modelo de dados muito próximo do *standard* UDDI
  - O mapeamento de um para outro é quase direto
  - Só mudam alguns nomes
- Pacote de classes que implementam o modelo de dados:
  - `javax.xml.registry.infomodel`

JAX-R API
Classes Principais

- Connection
  - Representa a ligação entre o cliente JAX-R e o servidor JAX-R (Registry Server)
  - Efetua a autenticação necessária
- RegistryService
  - Publicação
    - BusinessLifeCycleManager
      - Adicionar e remover registos
      - Tipicamente, é feito sobre uma ligação autenticada
  - Pesquisa
    - BusinessQueryManager
      - Consultar registos
      - Tipicamente, não precisa de autenticação

JAX-R API

Publish

- Regista no *Registry* os vários tipos de informação associados a uma organização
  - contactos, serviços e localizações de serviços
- Devolve uma chave que identifica univocamente o serviço
- Exemplo:

```
BusinessLifeCycleManager blcm;
Organization org = blcm.createOrganization("my.org");
// preencher org com os seus dados
Collection orgs;
orgs.add(org);
String key = blcm.saveOrganizations(orgs);
```

JAX-R API

Inquiry

- Procura serviços que respeitem um dado critério de seleção:
  - `findQualifiers`
  - `namePatterns`
  - `classifications`
  - `specifications`
  - `externalIdentifiers`
  - `externalLinks`
- Exemplo:

```
BusinessQueryManager bqm;
// passar como argumentos os critérios de procura
BulkResponse response = bqm.findOrganizations(…);
```

Biblioteca UDDINaming

- Pacote:
  - `pt.ulisboa.tecnico.sdis.ws.uddi`
  - Código-fonte disponível, podem modificar no projeto, se necessário
- Biblioteca para simplificar a utilização do UDDI
  - Inspirada no JNDI Naming
    - `bind(), lookup()`
  - Limitação: apenas suporta relações 1-para-1
- 1 Organização
  - 1 Serviço
    - 1 Endereço

Propriedades dos Nomes

- Unicidade referencial
- Âmbito
- Homogeneidade/heterogeneidade
- Pureza
- Persistência

Propriedades dos Nomes

Unicidade referencial

Unicidade referencial

- Num determinado contexto,
  um nome só pode estar associado a um objeto
  - Caso contrário, haveria ambiguidade referencial
    - Não se poderia distinguir o objeto
    - Não se poderia endereçá-lo
  - A situação inversa não é verdadeira,
    um objeto pode estar associado a vários nomes
  - Os nomes simbólicos são normalmente nomes alternativos para um mesmo objeto no mesmo contexto

Gestão de Espaços de Nomes: Atribuição de Nomes Globais

Problema: garantir a unicidade referencial

- É preciso garantir que um dado nome é resolvido sempre para o mesmo objeto em todo e qualquer contexto

Soluções:

- Atribuição central
- Atribuição local e difusão para os outros contextos
- Nomes não estruturados com grande amplitude referencial
- Nomes hierárquicos → nomes globais compostos pela concatenação de nomes locais

Gestão de Espaços de Nomes: Atribuição de Nomes Globais

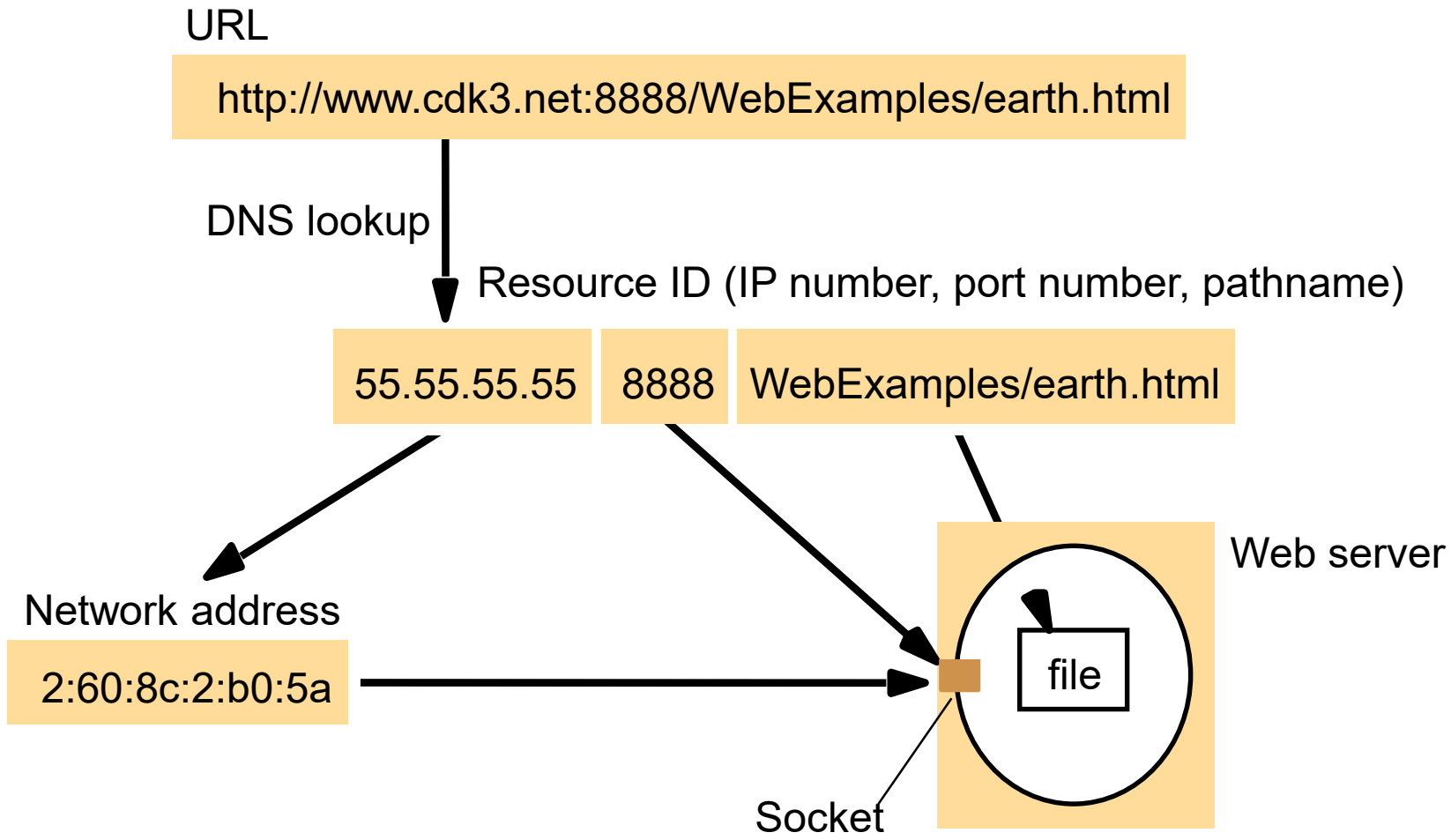
- Atribuição central
  - Grande latência, ponto único de falha
    - Exemplos:
      - Endereços IP oficiais (públicos)
      - Endereços Ethernet (de fábrica)
- Atribuição local e difusão para os outros contextos
  - Impraticável em larga escala
  - Simples e prático em redes locais
    - Exemplo:
      - Nomes no NetBios

Gestão de Espaços de Nomes: Atribuição de Nomes Globais

- Nomes não estruturados com grande amplitude referencial
  - Podem ser atribuídos independentemente por qualquer contexto
  - Podem ser gerados de forma pseudoaleatória ou mesmo totalmente aleatória
    - Identificadores com 128 bits ou maior amplitude referencial

- Nomes hierárquicos → nomes globais compostos pela concatenação de nomes locais
  - Exemplos
    - Números de Telefone (ex. +351 21 3100000)
    - Nomes DNS (ex. www.tecnico.ulisboa.pt)
    - Nomes de Ficheiros (ex. /a/b/b)

URL – Exemplo de Nome Hierárquico



Namespaces: Solução hierárquica de nomes no XML

- Problema: troca de dados XML entre organizações
- **<banco>** pode referir-se a uma instituição bancária num documento e a uma peça de mobiliário noutro
- Solução: usar *tags* na forma
  "nome único : nome do elemento"

```
<bank xmlns:FB="http://www.FirstBank.com">
  …
   <FB:branch>
       <FB:branchname>Downtown</FB:branchname>
       <FB:branchcity>Brooklyn</FB:branchcity>
   </FB:branch>
  …
</bank>
```

Propriedades dos Nomes

Âmbito

Âmbito de um Nome

- Global (absoluto) → Um nome tem o mesmo significado em todos os contextos onde o espaço de nomes é válido
  - Independentes da localização do utilizador
  - Simples de transferir entre contextos
  - Difíceis de criar para garantir a unicidade referencial
- Local (relativo) → O contexto apenas engloba parte do sistema, os nomes são válidos só nesse contexto. Nomes são atribuídos independentemente em cada contexto.
  - Permite criação eficiente de nomes
  - Nomes têm que ser traduzidos quando transferidos para outros contextos
- Exemplo: Endereços IP?
  - Respostas diferentes consoante se considera existência de NATs

Propriedades dos Nomes

Homogeneidade/heterogeneidade

Homogeneidade / Heterogeneidade

- Homogéneo:
  - Formado por uma única componente
    - Endereço de uma placa Ethernet
  - Formado por várias componentes com igual estrutura e significado
    - Pathname UNIX: /a/b/c
- Heterogéneos
  - Formado por várias componentes com estruturas e significados diferentes
    - Pathname Windows: C:/a/b/c
    - URL: http://máquina:[porto]/a/b/c

Propriedades dos Nomes

Pureza

Pureza dos nomes

Puro: o nome não contém informação sobre a localização do objecto

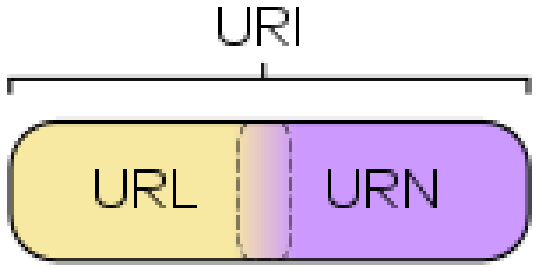
- O nome não contém identificadores
- O nome não reflecte os mecanismos de resolução do sistema
  - ☺ Flexibilidade, facilidade de reconfiguração
  - ☹ Impraticável em larga escala

Impuro → parcelas do conteúdo do nome são utilizadas na sua resolução

- O nome contém identificadores ou informação topológica
- O nome reflecte os mecanismos de resolução do sistema
  - ☺ Realização fácil, extensível, escalável
  - ☹ Reconfiguração difícil

Uniform Resource Identifiers (URIs)

- Standard de nomes de recursos na WWW
- Sintaxe: [prefixo]:[sufixo-específico-do-protocolo]
  - Exemplos:
  - urn:isbn:0-486-27557-4
  - https://www.tecnico.ulisboa.pt/index.html
  - mailto:info@tecnico.ulisboa.pt
- Duas funções distintas:
  - Identificar univocamente um recurso na internet
    - Chamados URNs (Uniform Resource Names)
  - e/ou
  - Localizar um recurso na internet
    - Chamados URLs (Uniform Resource Locator)



Exemplos de âmbito e pureza

		Pureza	
		Puro	Impuro
Âmbito	Global	UUID - DCE/IETF Endereço Ethernet Número de rede num endereço IP (público) URN	Porto TCP/IP ou UDP/IP URL Endereço IP (público) *Pathname* AFS *Pathname* UNIX (/XXX/)
	Local	Nome de um ficheiro num directório UNIX Servidor Sun RPC Tag XML	*i-numbers* num directório UNIX Endereço IP (qualquer) *Pathname* UNIX (XXX/) *Pathname* NFS

Propriedades dos Nomes

Persistência

Persistência de Nomes

- Uma referência é persistente se não estiver ligada a nenhum domínio administrativo ou entidade
  - Implica que o objecto possa mudar de domínio administrativo sem que a referência seja perdida

- Exemplos:
  - URLs: Problemático... Mudança de ISP implica um “HTTP redirect” permanente no ISP anterior
  - Números de telemóvel em Portugal: persistência foi imposta por legislação

Exemplo: UUID na Interface em IDL DCE

```
[
uuid(00918A0C-4D50-1C17-9BB3-92C1040B0000),
version(1.0)
]
interface banco
{
typedef enum {
    SUCESSO,
    ERRO,
    ERRO_NA_CRIACAO,
    CONTA_INEXISTENTE,
    FUNDOS_INSUFICIENTES
  } resultado
```

Identificador global, homogéneo, puro, persistente

Gerado por uma aplicação

Exemplo: Sun RPC

```
bancoprog_1(char *host)
{
  CLIENT *clnt;
  pedirExtratoIn pedirextrato_1_arg;
  #define MAXLINE 1024
  char comando[MAXLINE];

  clnt = clnt_create(host, BANCOPROG, BANCOVERS, "tcp");
  if (clnt == (CLIENT *) NULL) {
    clnt_pcreateerror(host);
    exit(1);
}
```

Resolução do nome

Local ao servidor de nomes da máquina identificada em host

Serviços de Diretório

Serviços de Diretório

- Os serviços de nomes
  - Têm por objectivo efectuar a tradução de nomes em identificadores de outros espaço de nomes
  - A sua estrutura era constituída por pares <nome, atributo>

- Serviços mais complexos podem armazenar relações entre nomes e múltiplos atributos e permitir a pesquisa por atributos
  - São normalmente designados **serviços de diretórios.**
  - Permite genericamente dois tipos de serviços de procura:
    - *White-pages*: procura por nome
    - *Yellow-pages*: procura por conteúdo semântico associado aos nomes

Serviços de Diretório

- Um diretório é constituído por:
  - Esquema – mapa lógico da base de dados. O esquema inclui quais os objetos que podem ser criados, os atributos dos objetos, e os tipos de dados
  - Classes – tipos abstractos que podem ser herdados
  - Atributos – define informação sobre objetos
  - Valores – para um atributo ter significado tem de ser instanciado por um valor
  - Objeto – instância de uma classe com os respectivos atributos

- Os serviços de diretório podem ser usados para diversos nomes utilizados pelo sistema ou por aplicações ex.: utilizadores, credenciais de segurança, etc.

- Não têm uma linguagem de interrogação (*query*) como as bases de dados

Arquitetura dos Serviços de Nomes e Diretório

Serviços de nomes: Funcionalidades

Registo das associações

- Verifica se a sintaxe do nome respeita o espaço de nomes
- Armazena a associação

Distribuição das associações

- Atualização dos diretórios nos contextos onde a associação deve ser válida

Resolução dos nomes

- Tradução do nome noutro nome ou num identificador
- Normalmente feita sem conhecimento da estrutura completa do nome
- Processo pode ser repetido recursivamente em vários níveis

Resolução inversa

- Dado um identificador, devolve o seu nome

Propriedades do espaço de nomes: Relevância consoante a ação

Relevantes para o **registo de nomes** (registo de associações nomes→objeto)

- Unicidade referencial
- Âmbito
- Homogeneidade
- Persistência

Relevantes para a **resolução de nomes** (obtenção de um objeto dado um nome)

- Pureza

Serviços de nomes:
Características dos sistemas distribuídos

- Larga escala
- Distribuição geográfica
- Heterogeneidade de nomes e protocolos
- Necessidade de grande disponibilidade
  - Uso de *caches*
- Estabilidade
  - Os nomes variam pouco
- Consistência fraca
  - Manutenção de *caches* com algum grau de erro

Arquitetura dos serviços de nomes:
Evolução da Arquitetura

1) Ficheiros replicados em todas as máquinas
   - **Ficheiros UNIX /etc/hosts, /etc/services, etc.**
   - **Ficheiro Windows LmHosts**

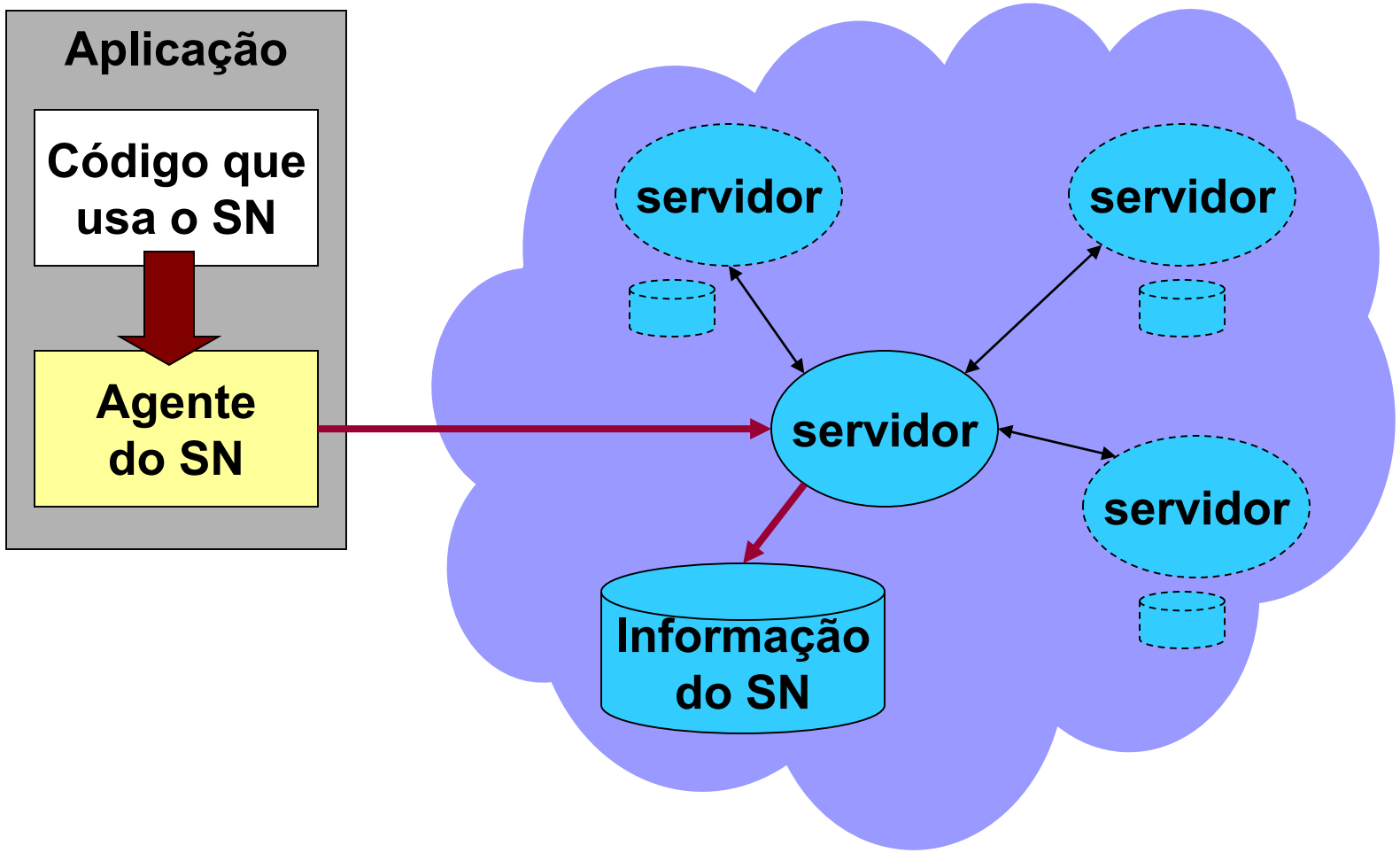
2) Pedido em difusão
   - Respondendo o nó que tem o objecto
     - **NetBIOS**
     - **IP ARP**

3) Arquitetura cliente-servidor (*solução habitual*)
   - Pergunta direta dos clientes a servidores específicos
     - **DNS, NIS, UDDI, Active Directory**

Arquitetura dos serviços de nomes:
Componentes

- Agente do serviço de nomes
  - Efetua o processamento do cliente
  - Oferece uma interface ao programador
- Servidores de nomes
  - Realizam o serviço de nomes
- Base de dados de nomes
  - Mecanismo de armazenamento persistente da informação nos servidores

Arquitetura dos serviços de nomes:
Diagrama de interações



Arquitetura dos serviços de nomes:
Agente

- Conjunto de utilitários e rotinas de adaptação (*stubs*)
  - Que efectuam os pedidos aos servidores
  - Exemplos: gethostbyname, gethostbyaddr,
    JNDI – Java Naming & Directory Interface,
    JAX-R – Java API for XML Registries

- Localização do(s) servidor(es):
  - Porto do servidor é fixo (*well-known*)
    - Ex. Sun RPC, DNS
  - Difusão periódica do endereço dos servidores
  - Pedido do cliente em difusão
    - Ex. NIS

Arquitetura dos serviços de nomes: Modelos de resolução de nomes

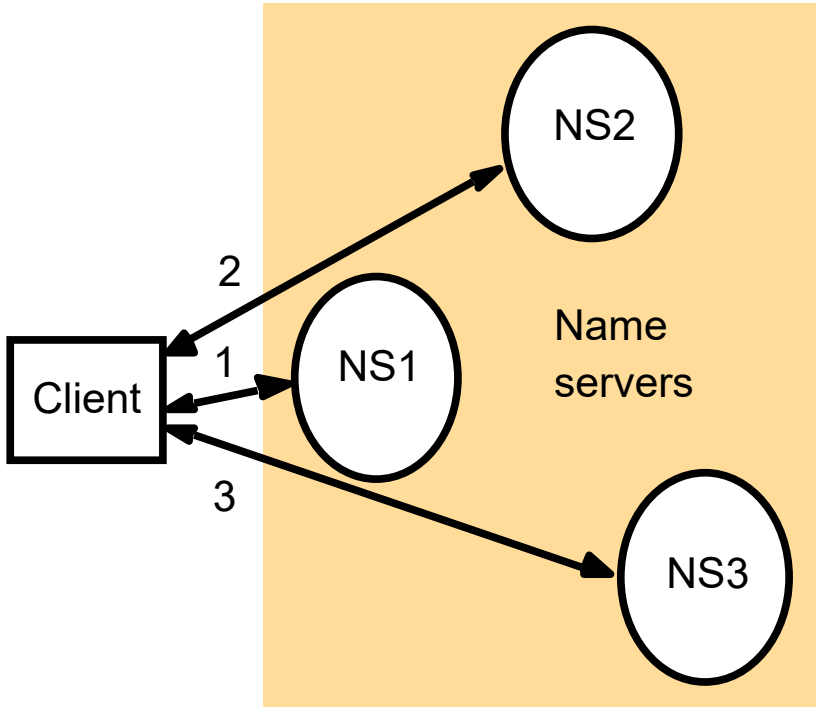
- Baseada em *multicast*
- Iterativo
- Iterativo controlado pelo servidor
- Recursivo

Arquitetura dos serviços de nomes: Modelo de resolução baseado em *multicast*

- Cliente envia nome a resolver em difusão para múltiplos servidores de nomes
- Quem souber, responde

- O que assumir se ninguém responde?
- Escalável para redes de grande escala?

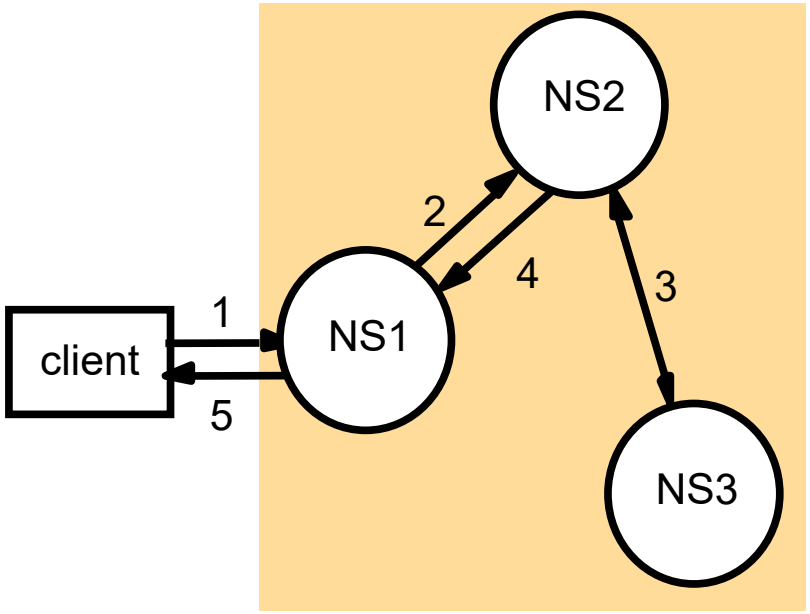
Arquitetura dos serviços de nomes: Modelo de resolução iterativo



Iterativo:

- A mais complexa para o agente
  - Pode interatuar com vários servidores
  - Tem que manter contexto de resolução
- Mais fácil em presença de falhas

Arquitetura dos serviços de nomes: Modelos de resolução controlada pelo servidor

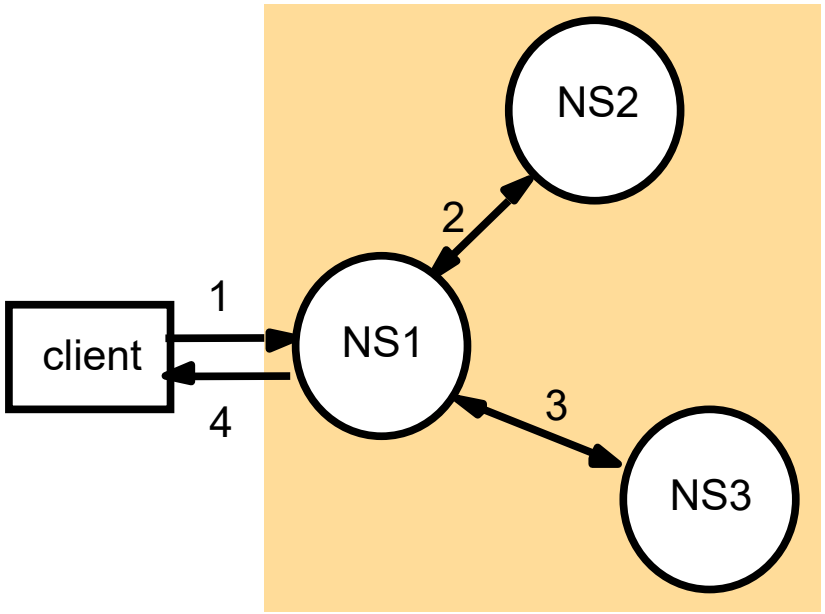


Recursiva

Recursivo:

- A mais simples para o agente
  - Apenas interactua com um servidor
- A mais complexa para os servidores
  - Têm que manter contexto de resolução
- Permite fazer *caching* nos servidores
- Simplifica o uso com barreiras de segurança

Arquitetura dos serviços de nomes: Modelos de resolução controlada pelo servidor



Iterativa
controlada pelo servidor

Iterativo controlado pelo servidor

- Combina algumas vantagens de ambos. Quais?

Arquitetura dos serviços de nomes:
Servidores

- Aproximação simplista: servidor centralizado
  - Ponto singular de falha
  - Excesso de informação
  - Estrangulamento no acesso
  - Complexidade do controlo de acesso

- Aspetos relevantes para a otimização:
  - Os nomes mudam com pouca frequência
  - Incoerências na resolução de nomes são normalmente toleráveis
    - Podem ser contornadas por repetição da resolução

É viável usar *caches* ou replicação em clientes e servidores intermediários

Arquitetura dos serviços de nomes:
Servidores e *caches*

- Um servidor centralizado, *caches* nos clientes
  - Não existe partilha das *caches* pelos clientes
  - Não facilita os registos

- Múltiplos servidores e *caches*
  - *Caches* em servidores intermédios podem ser usadas por vários clientes

Exemplos de Serviços de Nomes

Exemplos de Serviços de Nomes e Diretório

- Serviços de Nomes
  - DNS (Domain Name System)
- Serviços de Diretórios
  - NIS (Network Information System)
  - X500
  - Active Directory da Microsoft
  - DCE:
    - CDS (Cell Directory Service)
    - GDS (Global Directory Service)
  - UDDI

Características Genéricas

- Organização do espaço de nomes
  - Esquema de nomes – estrutura, atributos
  - Propriedades
- Arquitetura Cliente-servidor
  - Esquema de autoridades
  - Persistência
  - Disponibilidade
    - Replicação de Servidores
  - Desempenho
    - Cache em clientes e servidores

DNS (*Domain Name Service*): Introdução

- Arquitectura para registo e resolução de nomes de máquinas da Internet
  - Inicialmente proposta em 1983
- Concretizações:

  UNIX: BIND (*Berkeley Internet Name Domain*)

  Microsoft: Windows 2000 DNS
    - Integrado com o Active Directory

UC REDES

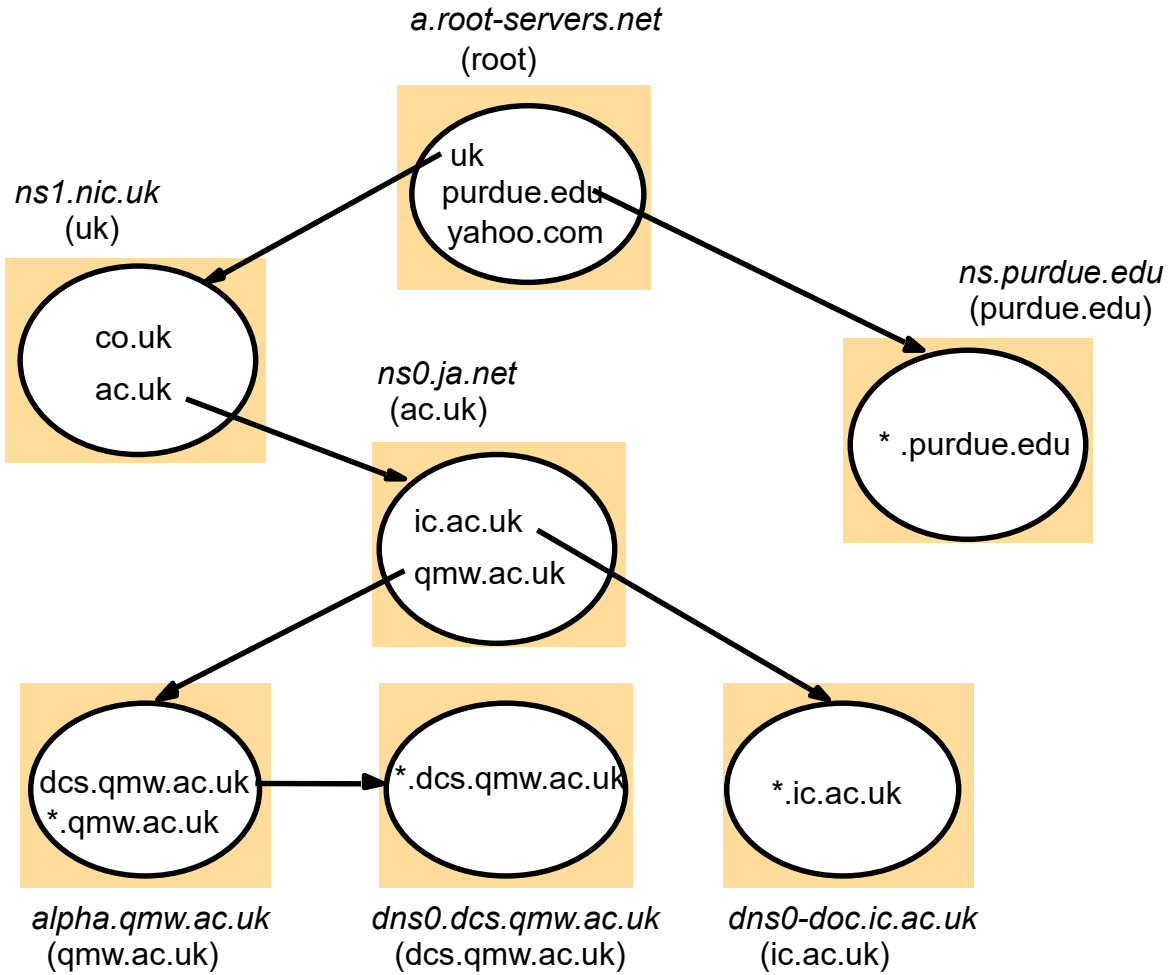
DNS: Caraterísticas

- Espaço de nomes hierárquico e homogéneo
- Âmbito dos nomes:
  - Global (*Fully Qualified Domain Name*)
    Ex. sigma01.tecnico.ulisboa.pt.
  - Local (resolvido no domínio corrente)
    Ex. sigma01
- Cada contexto designa-se por domínio
- Cada domínio é gerido por uma entidade (*authority*)
  - Pode criar e remover nomes
  - Pode delegar responsabilidades em subdomínios
- Nomes impuros

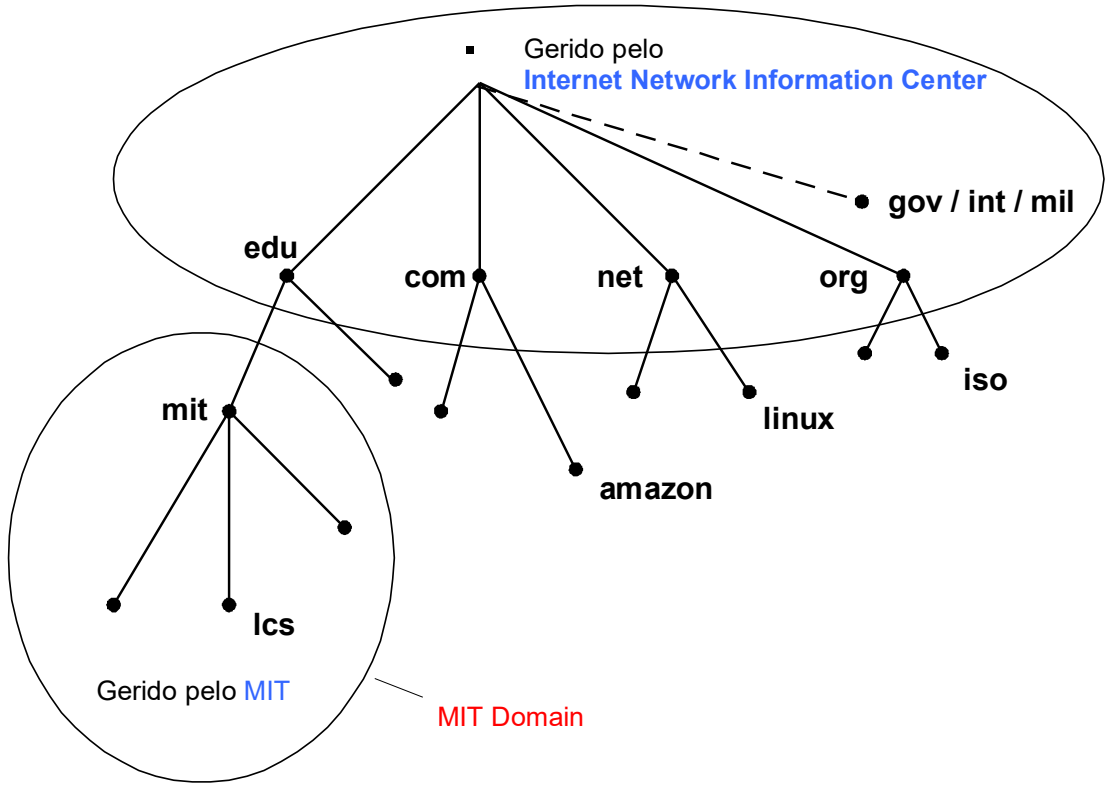
DNS: Estrutura

Note: Name server names are in italics, and the corresponding domains are in parentheses.

Arrows denote name server entries



DNS: Estrutura



Nome de domínio	Tipo de organização
com	Comercial
edu	Educação
org	Sem fins lucrativos
net	Redes
gov	Governamental (não militar)
mil	Governamental e militar
num	Números de telefone
arpa	Reverse DNS
xx	Código de país (2 letras) ISO 3166

DNS: Zonas

- Unidades de fracionamento da hierarquia de autoridades

- Uma zona é uma unidade de administração:
  - Cada domínio pertence a uma zona
  - Cada zona pode gerir um ou mais domínios
  - **A Zona constitui a autoridade**

Servidores DNS

- Associado a uma zona existe sempre um servidor
  - Contém a base de dados com os nomes desse conjunto de domínios
- Servidor sempre replicado
  - Primário: mantém a base de dados, onde se efectuam as atualizações
  - Secundário: contém uma cópia da informação do primário, atualizada periodicamente com um protocolo dedicado
- Todos os servidores mantêm *caches*
  - Validade indicada pelo parâmetro TTL
- Cada servidor indica a sua autoridade sobre os dados que fornece
  - Primário: autoridade total sobre os dados do domínio
  - Secundários: não possuem autoridade alguma

DNS: Esquema de Informação

- Registos RR (*Resource Register*)
  - Pares nome→valor tipificados
  - A tipificação exprime:
    Classe: família de nomes (ex. IN para endereços IP)
    Tipo: semântica de utilização do nome
  - Cada RR possui um TTL (*time to live*)
    - Serve para invalidar periodicamente RR em *cache*

- Informação estrutural (RRs do tipo NS)
  - Localização de servidores de zonas

DNS: Tipos de registos

Tipo de registo	Conteúdo
A	Endereço IP
CNAME	Nome simbólico para outro nome DNS
HINFO	Arquitectura e sistema operativo do nó
NS	Servidor de uma zona
MX	Máquina ou domínio do servidor preferencial de e-mail
SOA	Parâmetros que definem a zona
PTR	Nome DNS para resolução inversa de um endereço IP
TXT	Texto arbitrário
WKS	Descrição de um serviço com os respectivos nomes e protocolos

DNS:
Informação do domínio ist.utl.pt

```
$ORIGIN ist.utl.pt.
@                       1D IN SOA       ciistr1 root.ciistr1 (
                                        1999080607      ; serial
                                        4H              ; refresh
                                        2H              ; retry
                                        1W              ; expiry
                                        1D )            ; minimum

ciistr1                 1D IN TXT       "The_Domain_Name_Server_for ist

www                     1D IN CNAME     ci
ftp                     1D IN CNAME     ci
proxy                   1D IN CNAME     ci
samba                   1D IN CNAME     ci

@                       1D IN NS        ciistr1
                        1D IN NS        alfa
                        1D IN NS        ci
                        1D IN NS        ns.utl.pt.
                        1D IN NS        civil2.civil
                        1D IN NS        inesc.inesc.pt.
rnl                     1D IN NS        ns2.rnl
                        1D IN NS        ciistr1
                        1D IN NS        ns.rnl

@                       1D IN MX        5 ciistr1
secreta                 1D IN MX        5 seca

alfa                    1D  IN A        193.136.128.24
```

Definições da zona (obrigatório no ficheiro do primário)

Nomes simbólicos

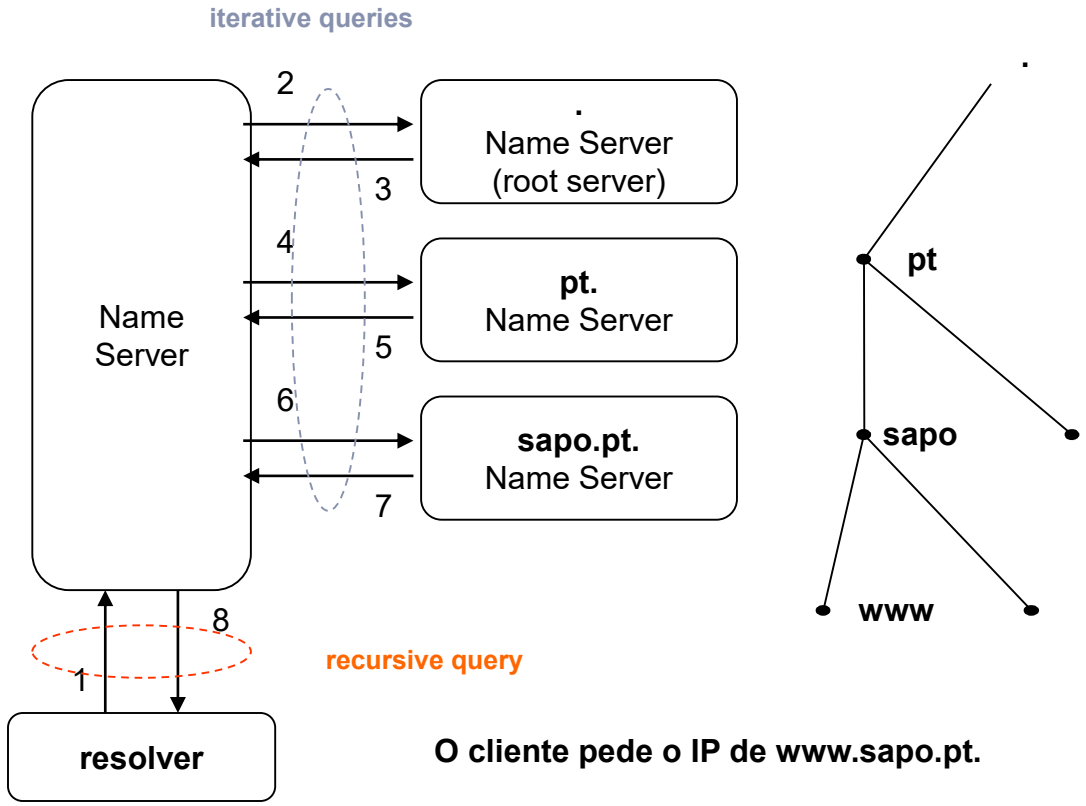
Servidores da zona ist.utl.pt

Servidores da zona rnl.ist.utl.pt

Servidores de email

Nome IP

DNS:
Resoluções recursivas ou iterativas

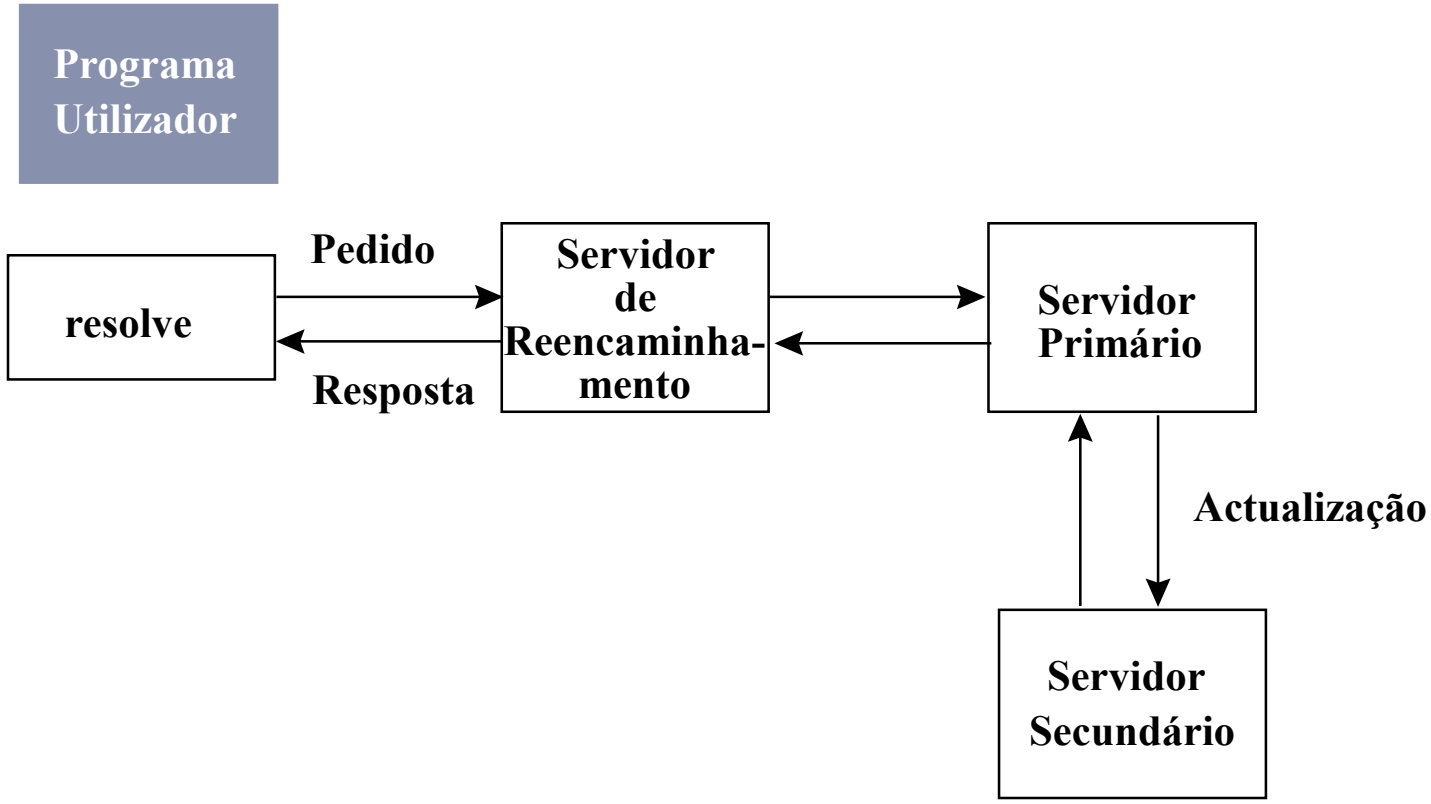


O cliente pede o IP de www.sapo.pt.

BIND
Berkeley Internet Name Domain

- Implementação do DNS para Unix
- Contém 2 componentes:
  - `resolver`: conjunto de rotinas cliente
    - Integradas na biblioteca de C (`/lib/libc.a`)
    - Usadas pelas rotinas de resolução de nomes (`gethostbyname, gethostbyaddr`)
  - `named`: (***name d**aemon*) servidor de nomes

Exemplo de Arquitectura do BIND



Norma X.500: Introdução

- Arquitetura para um diretório de nomes em aplicações informáticas à escala mundial
  - Inicialmente proposta em 1988
  - Foi desenhado para armazenar informação respeitantes a países, organizações, pessoas, máquinas, etc.
- Concretizações:
  - DCE GDS, NDS (Novell), Active Directory (Microsoft)
  - Protocolo de acesso LDAP

X.500: Características

- Espaço de nomes global, hierárquico e homogéneo
  - Nomes impuros
- Cada entrada é uma instância de uma classe (*object class*)
  - Cada classe define os atributos obrigatórios e opcionais dos objetos que os nomes podem referir
  - Um objeto pode ser referenciado por entradas pertencentes a diferentes classes
- Cada entrada da hierarquia é formada por um conjunto de atributos
  - Cada atributo tem um tipo e um ou mais valores
  - Um nome é uma seleção dentro desses atributos
- O conjunto de classes define o "esquema" do espaço de nomes

Um atributo que nasceu no X500 foi o certificado X509

X.500: Características

- Tipos de nomes:
  - *Relative Distinguished Name* (RDN)
    - Nome que identifica uma entrada concreta num dado contexto de resolução (Nome local)

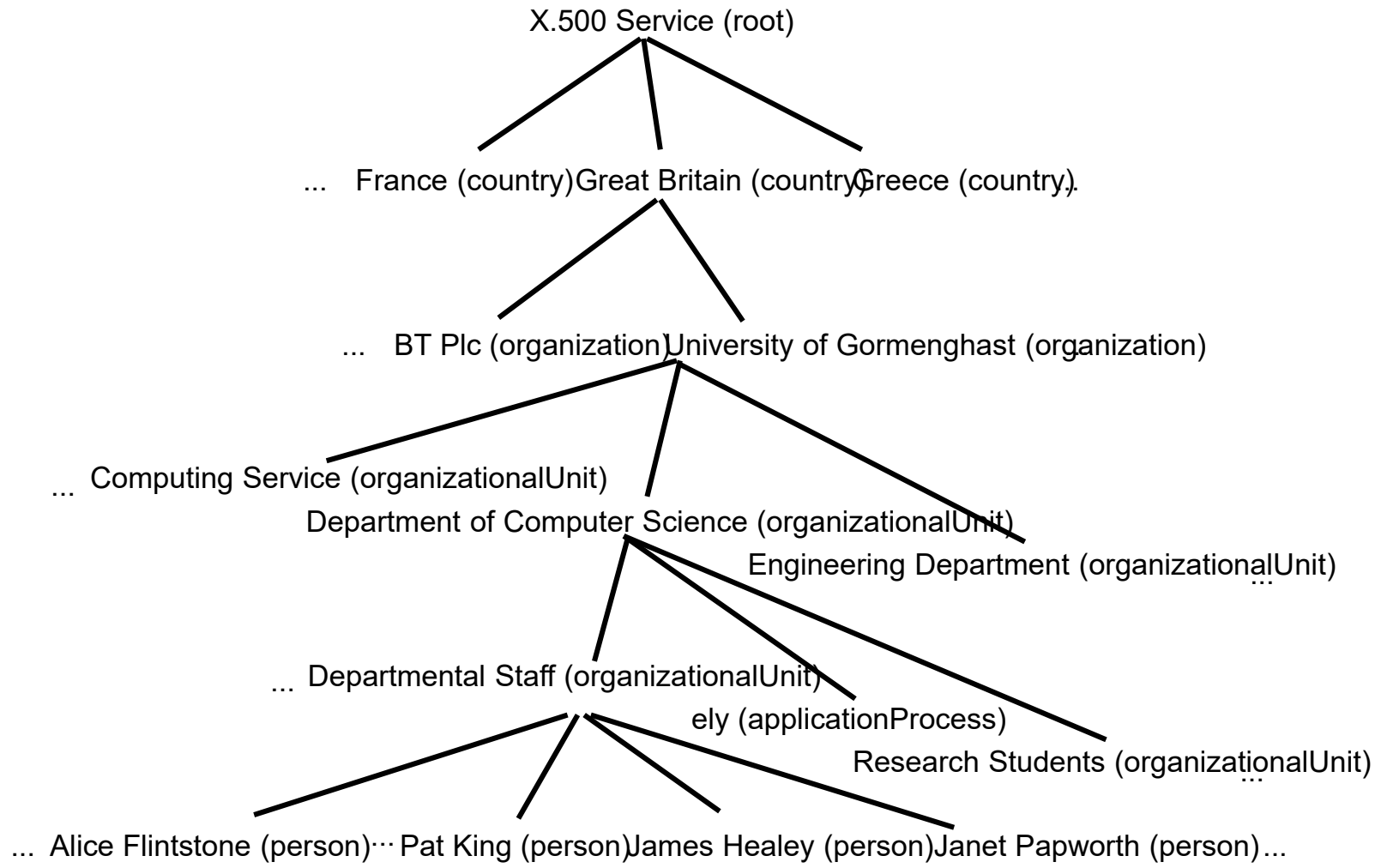
      C = PT O = IST OU = Secretaria
  - *Distinguished Name* (DN)
    - Concatenação não ambígua de RDNs desde a raiz (nome global)

      /…/C=PT/O=IST/OU=Secretaria

- Sintaxe dos nomes
  - O X.500 não define
    - Apenas define a sua estrutura
  - Cada concretização usa a sua sintaxe
    - A interacção é garantida por troca de informação estrutural

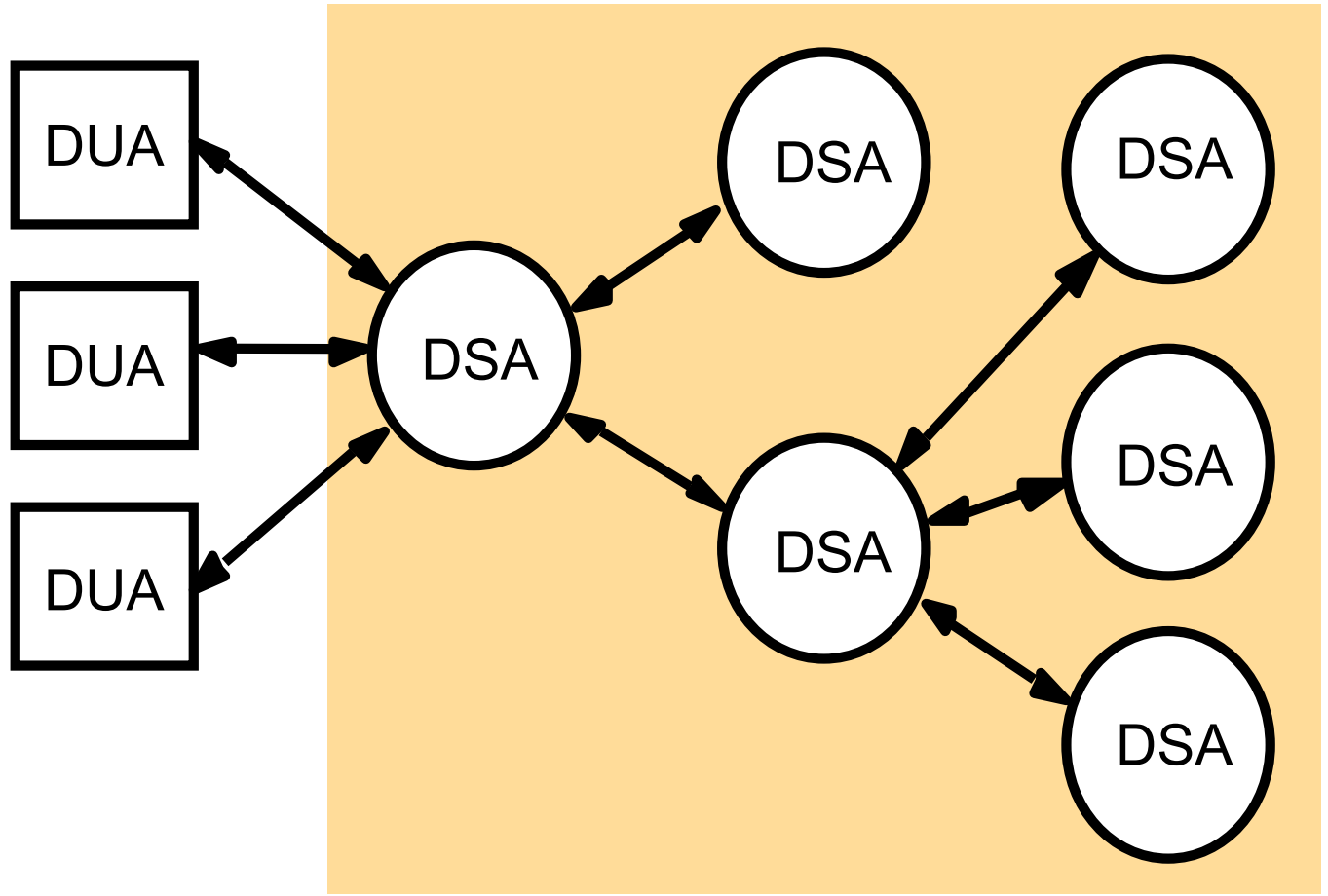
      LDAP → "OU=Secretaria, O=IST, C=PT"

X.500: Exemplo



X.500: Arquitetura

X.500: Arquitectura



- Servidores: DSAs (*Directory Service Agents*)
- Clientes: DUAs (*Directory User Agents*)

Lightweight Directory Access Protocol (LDAP)

- Baseado na proposta da Universidade de Michigan em que o acesso ao Directório X500 é efetuado diretamente sobre TCP/IP
- **Define o protocolo, não a estrutura do servidor**
- O LDAP substituiu a codificação ASN.1 por caracteres (texto)
- A API também é mais simples
- O LDAP pode ser usado com Diretórios que não sejam X500
  - O exemplo mais importante é o **Active Directory da Microsoft**

Active Directory

- Introduzido pela Microsoft em 2000
- Um directório é constituído por uma floresta que contém uma ou várias árvores do Active Directory
- Active Directory – algumas características:
  - Servidor DNS
  - Usa LDAP
  - Kerberos para autenticação
  - ACL para controlar o acesso aos objectos
  - Certificados X.509