Tecnologias para Descoberta de Serviços

- Adelstein *et al*, **Fundamentals of Mobile and Pervasive Computing**, McGraw-Hill, 2005, Capítulo 7

- Miam, A.N.; Baraldi, R. & Baldoni, R. **Survey of Service Discovery Protocols in Mobile Ad Hoc Networks**, Technical Report, *Universitá degli Studi di Roma "La Sapienza"*, 2006

Serviços

Serviço é uma abstração:

- qualquer função de processamento, armazenamento, entrada/saída de dados, acionamento/controle de equipamento

em Ubicomp, toda aplicação consiste da interação entre serviços (*Service Oriented Architectures - SOA*)

A cada local, e para cada tarefa, serviços precisam ser:

- descobertos
- selecionados
- Interagir entre sí (requisição-resposta ou subscription/ notification)

Departamento de Informática

Descoberta de Serviços

- É elemento fundamental para auto-configuração (*zero conf*) em redes e sistemas distribuidos dinâmicos
- Serviço pode ser qualquer coisa disponibilizada por um nó e utilizada por outro nó. Por exemplo:
  - Capacidade de rotear pacotes
  - Acesso a informações, a uma base de dados, etc.
  - Dados de um sensor
  - API para controlar um atuador específico
  - Acesso a dados de cache
  - …
- A Descoberta é um mapeamento:

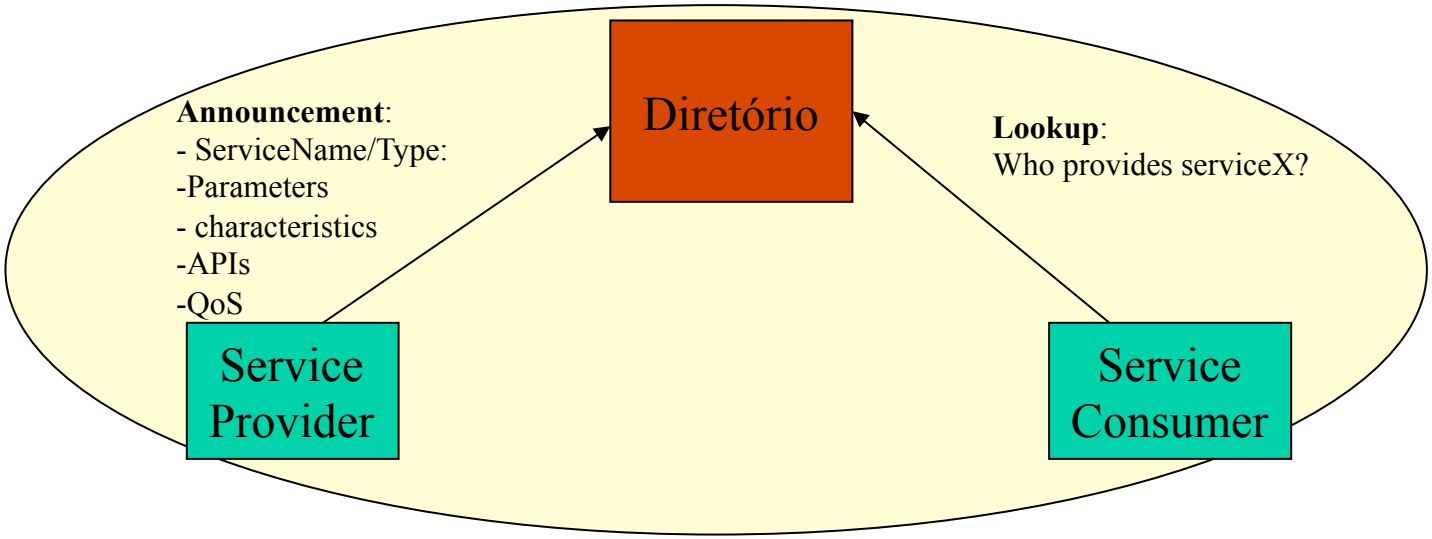
  Tipo/Nome do Serviço ➔ Endereço do/s nó/s que provêm esse serviço
  - Em alguns casos também inclui uma função de seleção (ou priorização) dos servidores encontrados

PUC RIO

Departamento de Informática

Principais Entidades

- Provedor de Serviço
- Consumidor de Serviço
- Diretório de Serviços
- Protocolos de Anúncio e Procura (lookup)



Obs:

- Diretório: pode ser servidor centralizado, federação de servidores, ou cache de anúncios recebidos
- Anúncio: unicast, broadcast, flooding, ou multicast
- Lookup: pode ser consulta a servidor, ou broadcast, ou flooding

PUC RIO

Principais Protocolos

Para redes infra-estruturadas / redes locais:

- Jini - Sun
- Service Location Protocol (SLP)
- Universal Plug & Play (UPnP) - Microsoft
- Salutation
- Bonjour (ex- Rendezvous) - Apple
- Ninja - UC Berkeley

Principais Protocolos

Para redes ad hoc:

- Single-Hop:
  - Service Discovery Protocol (SDP) p/ Bluetooth
  - JESA
  - Salutation Lite
  - Centaurus
- Multi-Hop:
  - Allia
  - Service Rings
  - Lanes
  - Konark
  - Varshavsky/Reid
  - Lima/Gomes/Ziviani/Endler

Departamento de Informática

Descoberta de Serviços em Redes Ad Hoc (MANETs)

Departamento de Informática

Motivação

Alguns nós em uma rede MANET possuem serviços (recursos) que os demais nós devem encontrar.

Por exemplo:

- Conectividade 2G/3G
- Maior capacidade de armazenamento/processamento
- Maior energia residual, ou energia ilimitada
- Um tipo de sensor ou atuador específico

PUC RIO

Roteiro

- Classificação
- Gerência de Informação sobre Serviços
- Métodos de Busca
- Escolha do Serviço
- Apoio à Mobilidade
- Descrição do serviço
- Conclusão

Protocolos para Descoberta de Serviços (Lookup)

Classificação:

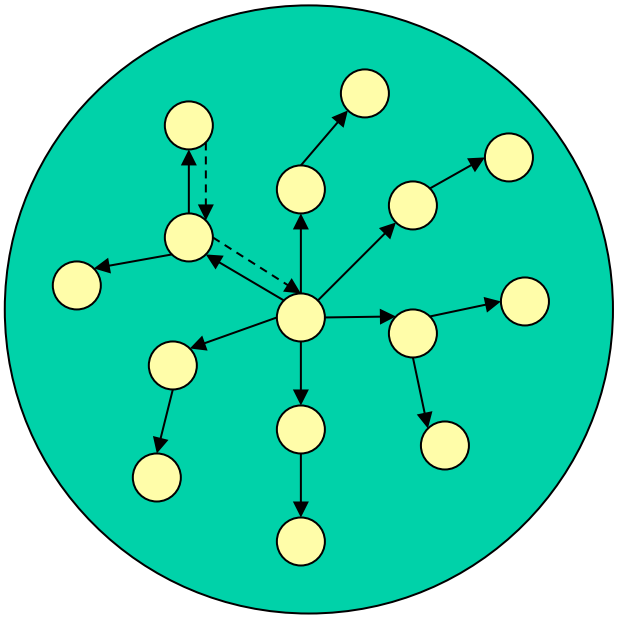
- Directory-less Architecture
- Directory-based Architecture
  - Centralized Directory Architecture
  - Distributed Directory Architecture
    - Infrastructure-less
    - Infrastructure-based

- Principais Características com Prós&Contras

Departamento de Informática

Arquitetura sem Diretório

Directory-less

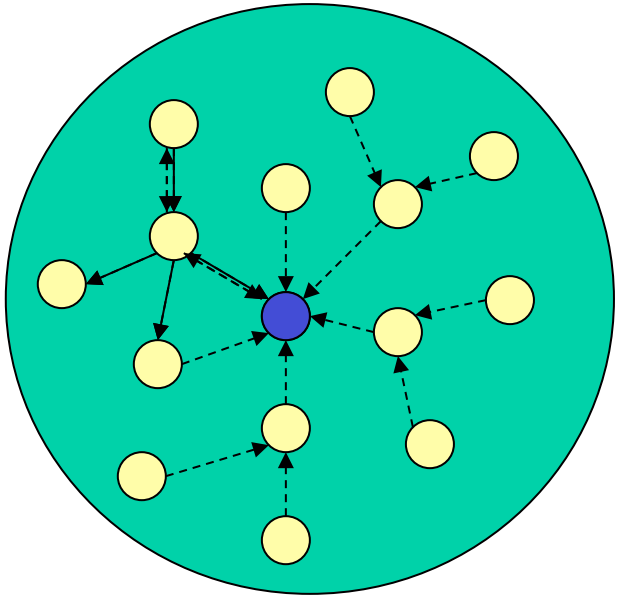


- Características
  - Consulta através de multicast, broadcast ou flooding
  - Resposta através de unicast
  - Adequado para redes ad hoc pequenas
  - Com baixa frequência de consultas
- Vantagens
  - Robusto a redes com alto grau de dinamismo
- Desvantagens
  - Consumo alto de largura de banda e energia

Departamento de Informática

Com Diretorio centralizado

Centralized Directory



- Características
  - Adequando para redes wireless infra-estruturadas
  - Lookup e resposta através de unicast
- Vantagens
  - Menor trafego na rede
- Desvantagens
  - Nó com o Diretório é um gargalo (*bottleneck*)
  - Ponto singular de falha

PUC RIO

Arquitetura de Diretórios Distribuídos

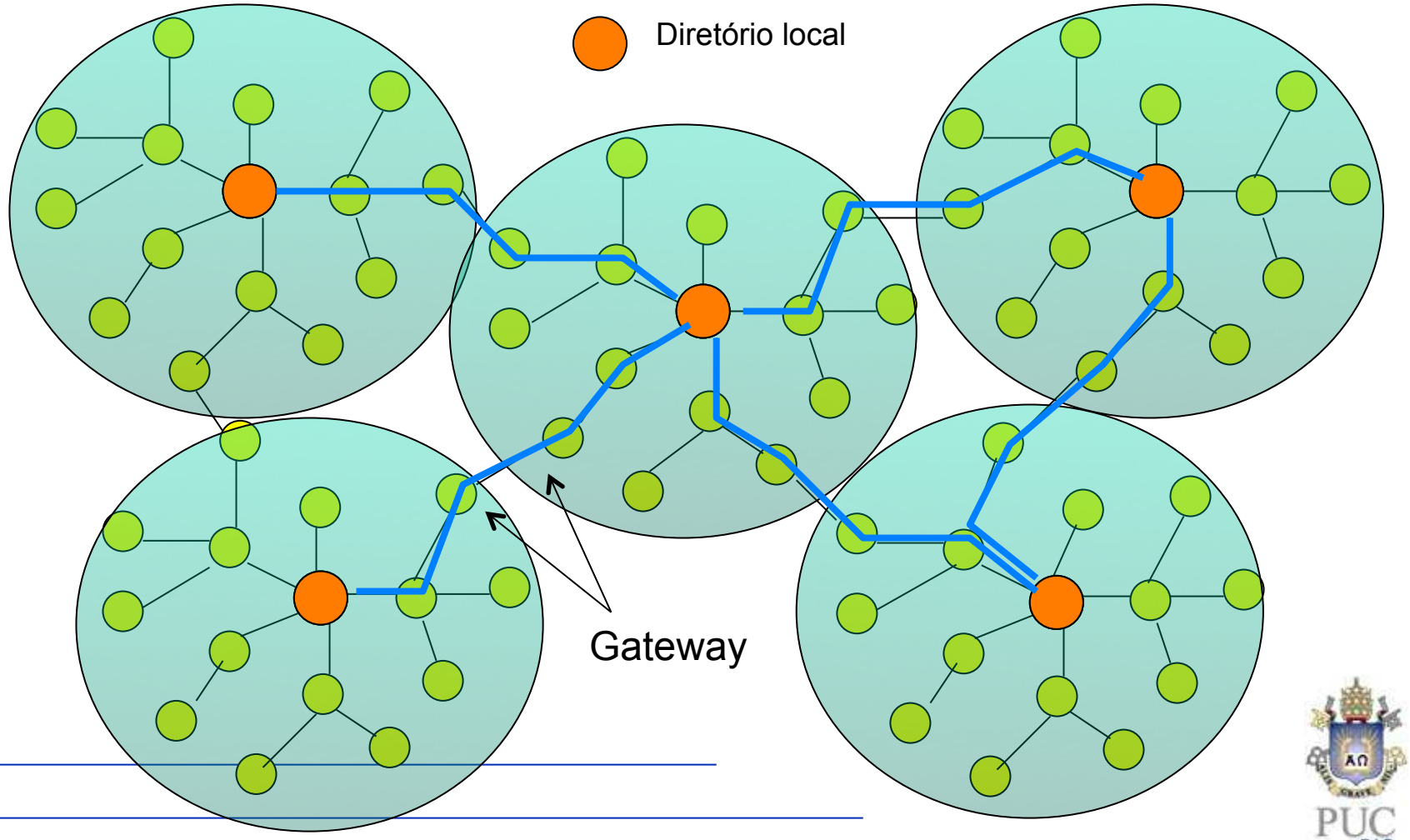
Distributed Directory Approach

- Sem Infra-estrutura
- Baseada em Infra-estrutura

Departamento de Informática

Arquitetura de Diretório Distribuída sem Infra-estrutura

Requer a formação de clusters



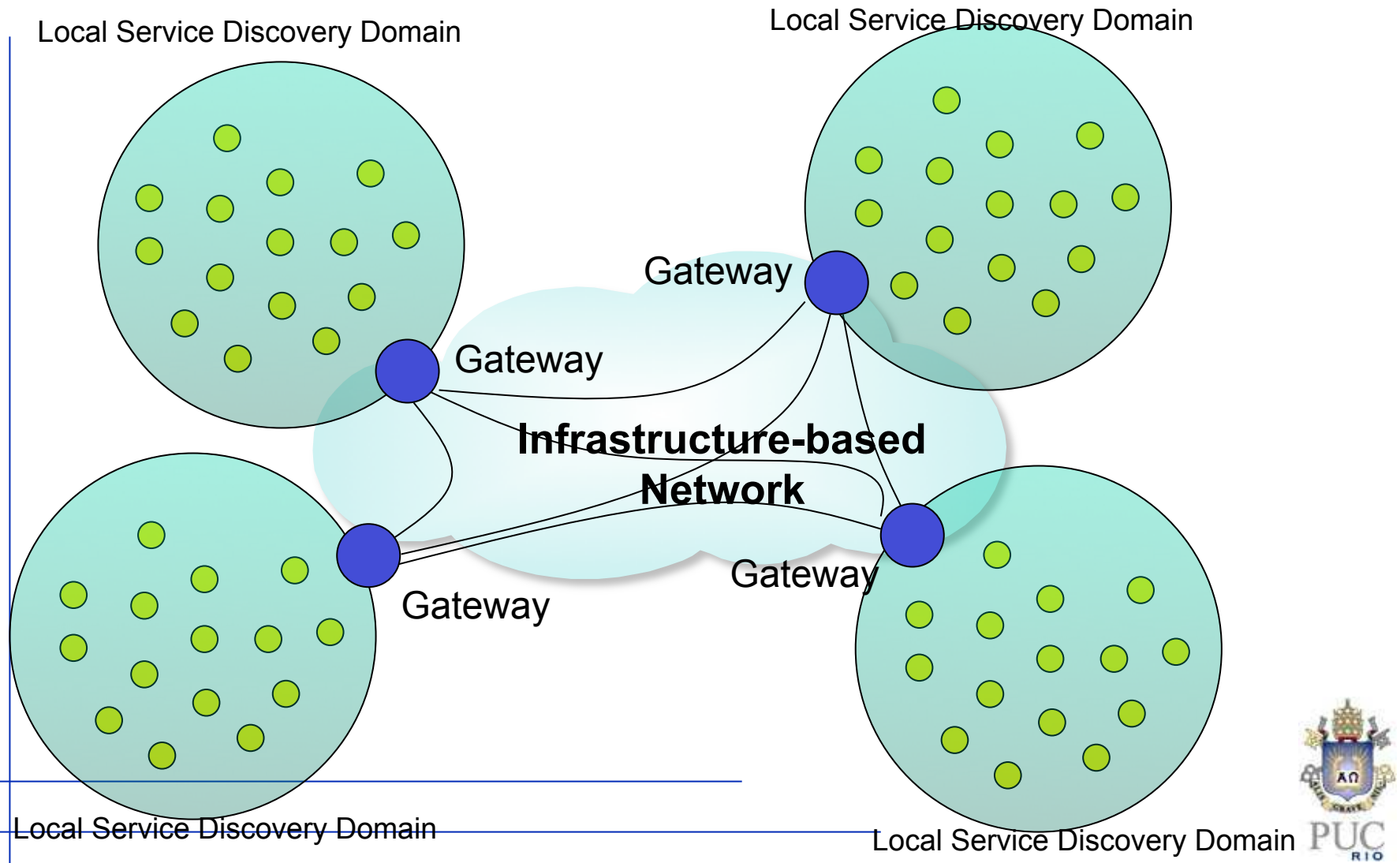
PUC RIO

Arquitetura de Diretórios Distribuída sem Infra-estrutura

- Características
  - Adequado para redes com grande número de nós
  - Nós Diretórios são selecionados dinamicamente dentre os nós que possuem maior capacidade/mais recursos
    - Exemplos: maior cobertura, energia residual, poder de processamento, memória, etc.
  - Nós Diretórios anunciam a sua presença periodicamente através de flooding (# hops limitado).
- Vantagens
  - Escalabilidade
  - Tempo de resposta para localização de serviço “próximo” é reduzida.
  - Pode-se usar técnicas de balanceamento de carga.
- Desvantagens
  - A procura por serviço em outro cluster envolve vários Nós Diretório
  - A escolha dinâmica dos Nós Diretórios pode:
    - causar sobrecarga na rede
    - deixar parte do diretório temporariamente inativo
  - Por isso, não é adequada para redes com toplogia muito variável (alto grau de dinamismo)

Departamento de Informática

Arquitetura de Diretorios Distribuida baseada em Infra-estrutura



PUC RIO

Arquitetura de Diretorios Distribuida baseada em Infra-estrutura

- Caracteristicas
  - Diretorios residem em **gateways** que conectam um **domínio de descoberta** à infra-estrutura fixa
  - Provedores anunciam serviços em um ou mais gateways, mas que tb precisam saber responder a lookups de outros domínios
  - Gateways trocam informações de diretório entre sí
  - Rede híbrida
- Vantagens
  - Altamente escalável
- Desvantagens
  - Necessidade de Infra-estrutura
  - Gateways se tornam o gargalo de comunicação e ponto central de falha

Departamento de Informática

Outra Categorização para Descoberta de Serviços

- Protocolos que não usam uma Rede Overlay
  - Única forma de restringir a quantidade de lookups e/ou anúncios é através do TTL (# hops)
  - Exemplos: GSD, Konark, Splendor, etc.
- Protocolos baseados em uma Rede Overlay (nó têm conhecimento e interage com nós não diretamente em sua cobertura)
  - Requer a criação da rede overlay e sua manutenção Principal vantagem: multicast controlado e eficiente ➔ causando pouco tráfego na rede
  - Exemplos: Allia, Service Rings, Lanes

PUC RIO

Departamento de Informática

Gestão de Informação sobre Serviços

- Um serviço anunciado é definido por:
  - Tipo
  - Descrição (ou características),
  - ID do serviço,
  - *service point:* endereço IP e número da porta,
  - protocolo a ser usado pelo cliente/consumidor para acessar o provedor de serviço

- Aspectos/opções da gestão:
  1. Onde armazenar esta informação de serviço (IS)?
  2. Tempo de validade da informação?
  3. número de hops dos anúncios?

PUC RIO

Departamento de Informática

Gestão de Informação sobre Serviços

Onde manter a informação do Serviço? Alternativas:

- Protocolo de Cheng/Marsic (sem diretório):
  - Etapa 1: provedor faz um broadcast de sua IS
  - Todos os nós que estão interessados no serviço respondem com um “service awareness”
  - Etapa 2: provedor envia (via multicast) a lista atualizada de seus serviços (apenas para os clientes interessados), que armazenam o IS
  - Problema: lookup para novos consumidores precisa achar um dos nós do grupo multicast

- Protocolo GSD [Chakraborty,et al], usa Peer-to-Peer Caching, e nós armazenam anúncios que vieram de provedores a uma distância máxima de N hops

D.Chakraborty, A. Joshi, Y. Yesha, T. Finin, GSD: A Novel Group-based Service Discovery Protocol for MANETS, IEEE MWCN, 2002

Métodos de Busca (Lookup)

Podem ser usados para descobrir diretórios ou provedores de serviço:

Duas formas canônicas:

- Com diretórios: busca é feita apenas no conjunto de nós diretórios
  - Exemplo: Service Ring, ou Virual Backbone nodes
- Sem diretórios:
  - busca é encaminhada para um grupo multicast (formado pelo provedor e potenciais clientes, em Chen/Marsic), ou
  - consulta-se o cache com anúncios próximos (em GSD) e depois para nós de grupos específicos.

Seleção do Serviço

Um cliente/consumidor que fez um lookup pode receber várias respostas, e precisa escolher o provedor que vai utilizar. Esta escolha pode ser feita pela aplicação, ou baseada em um **critério padrão de seleção**.

Exemplos de critérios de seleção:

- Escolher o provedor mais próximo (# de hops)
- Provedor com melhor capacidade de atender o serviço (e.g. Quality of Service)
- Provedor com menor carga (ou numero de clientes) momentânea
- Menor distância combinada com melhor capacidade do provedor de serviço.

Em [Varshavsky/Reid] e [Lima/Gomes/Ziviani/Endler] a seleção não é feita no consumidor, mas está mesclada no protocolo de descoberta, reduzindo assim a quantidade de respostas (i.e. Reply implosion)

Apoio à Mobilidade

Problema: em uma MANET a vizinhança de um nó e o caminho para alcançá-lo podem mudar a qualquer momento

Para protocolos de descoberta onde a informação sobre serviços (IS) só é armazenada no próprio provedor (e lookup é por flooding), não há problema.

Mas sempre que IS é armazenada em um diretório, ou no cache de outros nós, a mobilidade pode comprometer a corretude desta informação.

Apoio à Mobilidade

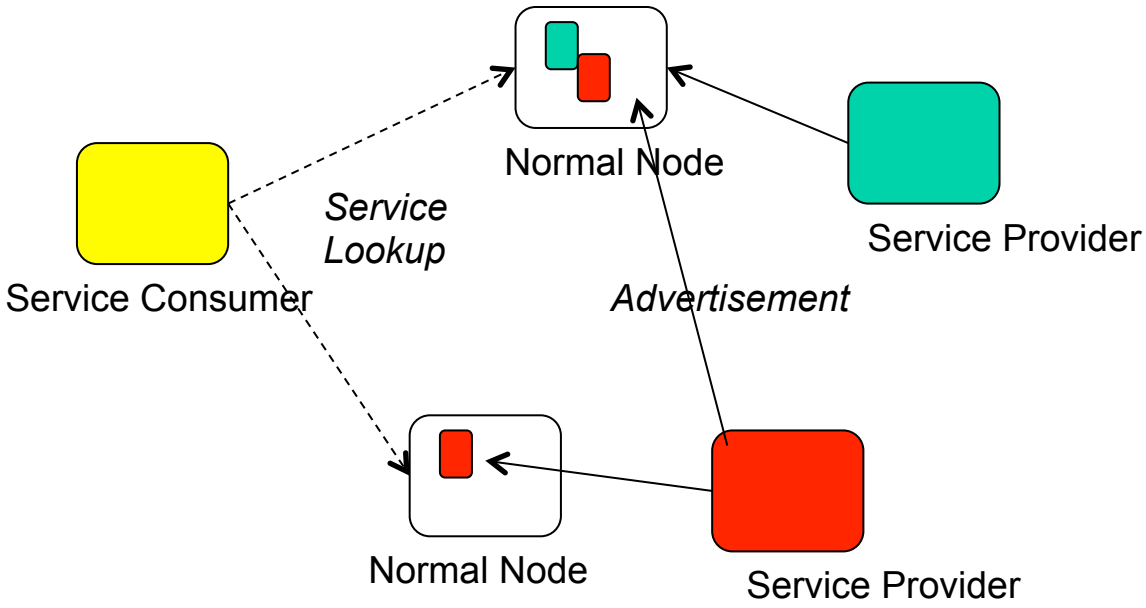
Fato: Se um nó armazena a IS de todos os membros de um grupo de provedores (GP), então esta informação precisa estar atualizada, independente da movimentação de qualquer nó (inclusive o próprio nó).

Existem 4 maneiras de lidar com mobilidade:

1) Atualizar a IS periodicamente (Exemplo: Konark, Splendor, Varshavsky/ Reid, etc.)
2) Atualizar a IS sempre que ocorre um evento (p.ex. rota do nó armazenador até determinado provedor deixou de existir (Exemplo: Varshavsky/Reid)
3) Controle dinâmico sobre anúncios: alcance (#hops) é reduzido e frequência é aumentada sempre que maior mobilidade dos nós é detectada (Exemplo: Allia e GSD)
4) Algoritmos para manter a estrutura da rede overlay
   - Os anúncios e lookups se baseiam no overlay
   - Exemplos: Service Rings, Lanes, Kozart/Tissiulas, etc.

Departamento de Informática

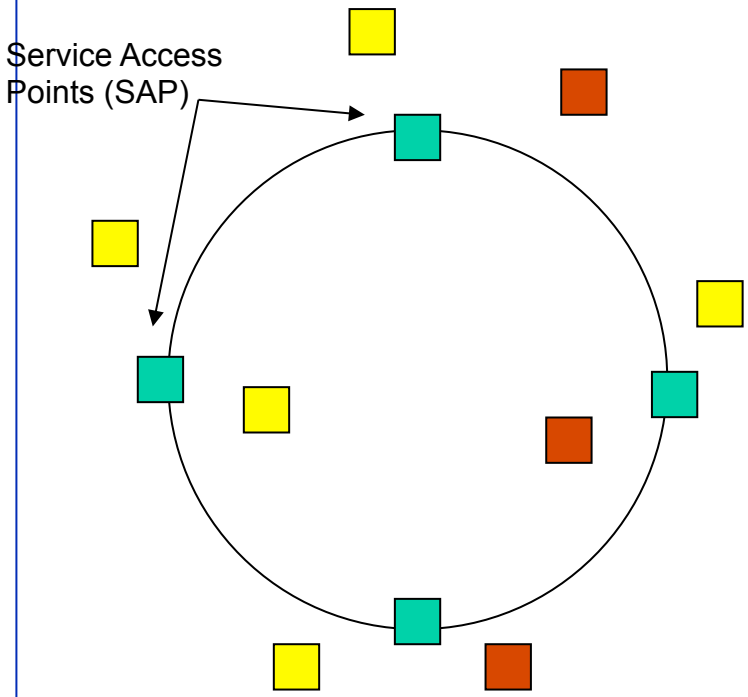
Apoio a Mobilidade



PUC RIO

Departamento de Informática

Service Rings: Exemplo de Mobilidade com Estrutura Overlay



- Lookups são feitos a um ServiceAccessPoint (SAP), que se comunica com outros SAPs
- Cada SAP só conhece o seu SAP predecessor e seu sucessor
- Periodicamente, circulam mensagens RingCheck entre os SAPs. Cada SAP encaminha a mensagem para seu sucessor, indicando seu ID e do predecessor
- Se um SAP não recebe tal mensagem em determinado período de tempo, verifica se consegue se comunicar com seu predecessor
- Possivelmente, incia um protocolo para re-criar o anel

Departamento de Informática

Descrição do Serviço

Como serviços são descritos?

linguagem utilizada?

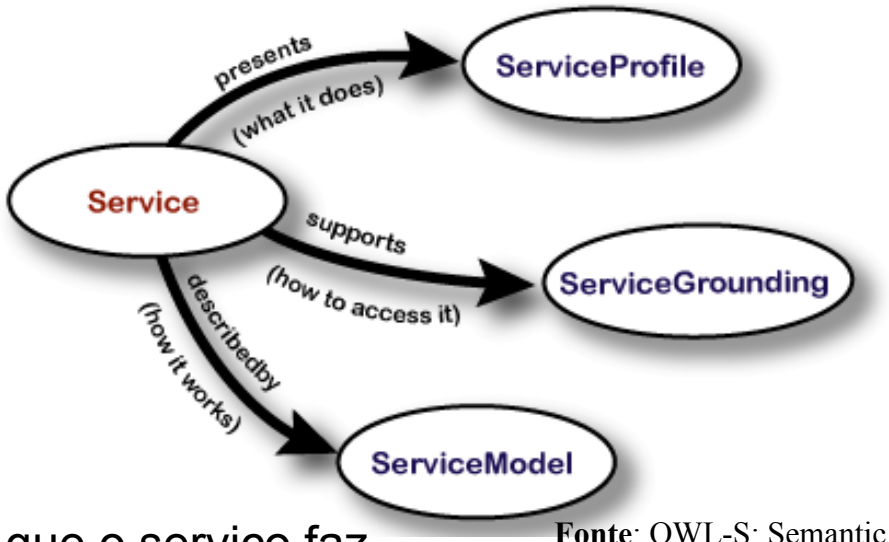
quais característcas?

Para MANETS, há 3 tendências:

1) Uso de uma linguagem estruturada (Exemplos: XML, DARPA Markup Language – DAML, Web Ontology Language – OWL-S)
   - Exemplos: Konark (XML), GSD (anúncios e requisições em DAML), DSD (OWL-S)
2) Formato livre (palavras-chave, pares atributo-valor, XML)
   - Exemplo: Allia, Varshavsky/Reid (matching anúncios/lookups feitos por um módulo de matching plugável)
3) Vários protocolos deixam esta questão em aberto

PUC RIO

Descrição do Serviço: ontologia básica



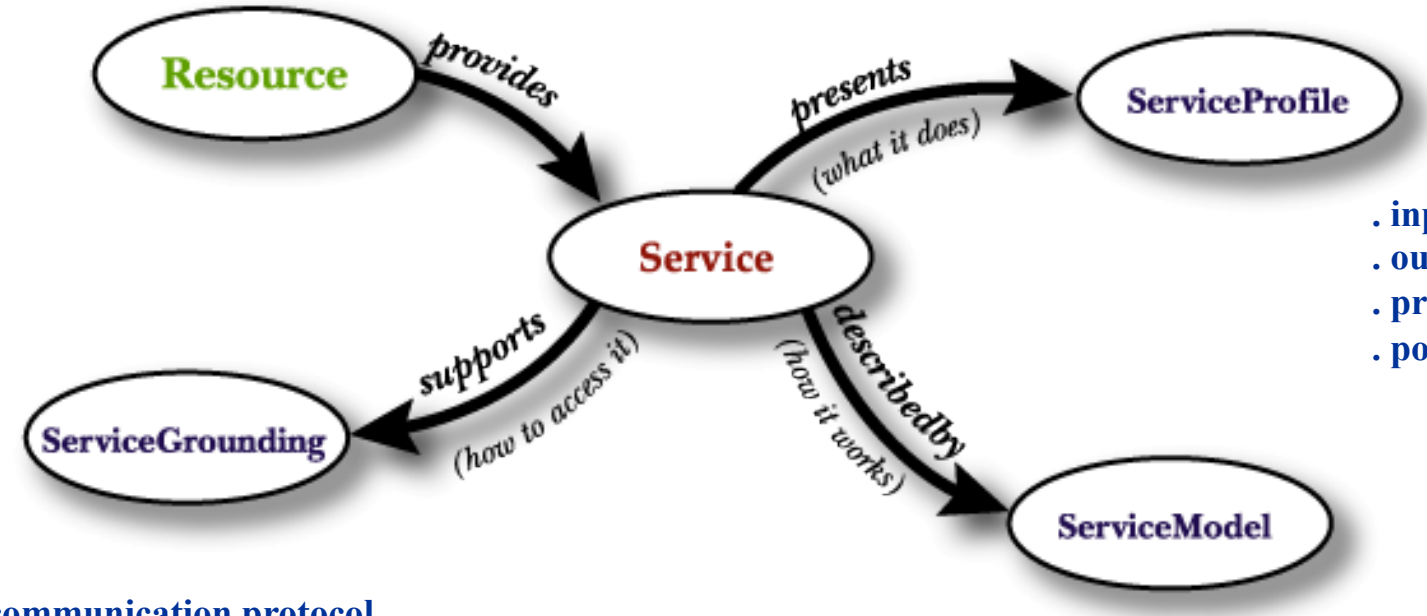
Fonte: OWL-S: Semantic Markup for Services, W3C

Service Profile: o que o serviço faz

Service Grounding: descreve o protocolo de comunicação, formatos de mensagem e outros detalhes específicos do serviço, tais como números de porta a serem usados

Service Model: como usar o serviço, detalhando o conteúdo semântico das requisicões, as condições em que ocorrerão resultados específicos, e a sequência de passos que levam a esses resultados

Exemplo



. input types
. output types
. preconditions
. postconditions

. communication protocol
(RPC, HTTP, …)
. port number
. marshalling/serialization

• process flow
• composition hierarchy
• process definitions

Service Profile

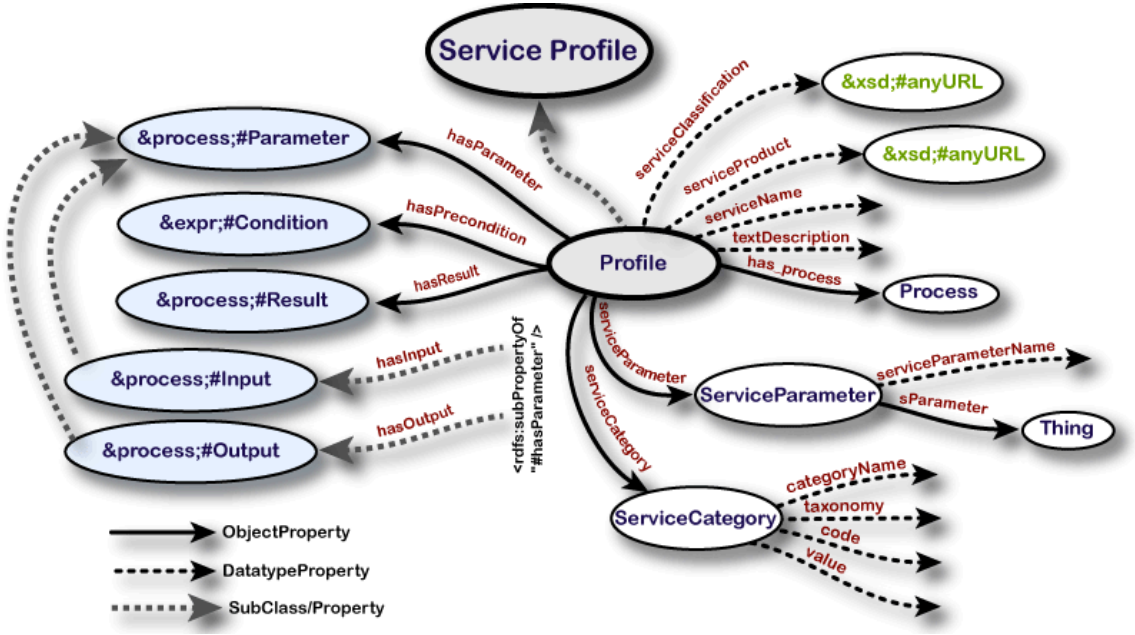
- É uma descrição do serviço em alto nível
- Usado para anúncios e requisições
- Um perfil coném:
  - Descrição em linguagem natural do serviço
  - Atributos funcionais
    - Inputs, outputs, pré-condições, efeitos
  - Atributos não-funcionais
    - Garantias sobre tempo de resposta, acurácia, custo do serviço, etc.
- Um perfil é uma visão abstrata do serviço

Service Profile: Funcionalidade

Especifica a funcionalidade do serviço e as condições que devem ser satisfeitas para obter o resultado esperado, além de possíveis exceções geradas do serviço.

Perfil OWL-S expressa dois aspectos da funcionalidade:

- a transformação de informação (representados por entradas e saídas) e
- a mudança de estado produzida pela execução do serviço (representada por pré-condições e pós-condições)



Fonte: OWL-S: Semantic Markup for Services, W3C

Departamento de Informática

Overlay usando DHT (Distributed Hash Tables)

- Muitos overlays para descoberta de serviço se utilizam de Distributed Hash Tables:
- Pares *chave/valor* são mapeados uniformemente nos nós de uma rede P2P
- Qualquer nó pode recuperar o valor a partir da chave
- A responsabilidade para manter o mapeamento é compartilhada por todos os nós
- Qualquer mudança no conjunto de nós causa uma re-organização do mapa, causando uma interrupção minimal na DHT (consistent hashing)
- Pode escalar paraum grande número de nós
- Existem muitos protocolos para DHT, p.ex. Chord Kademilia

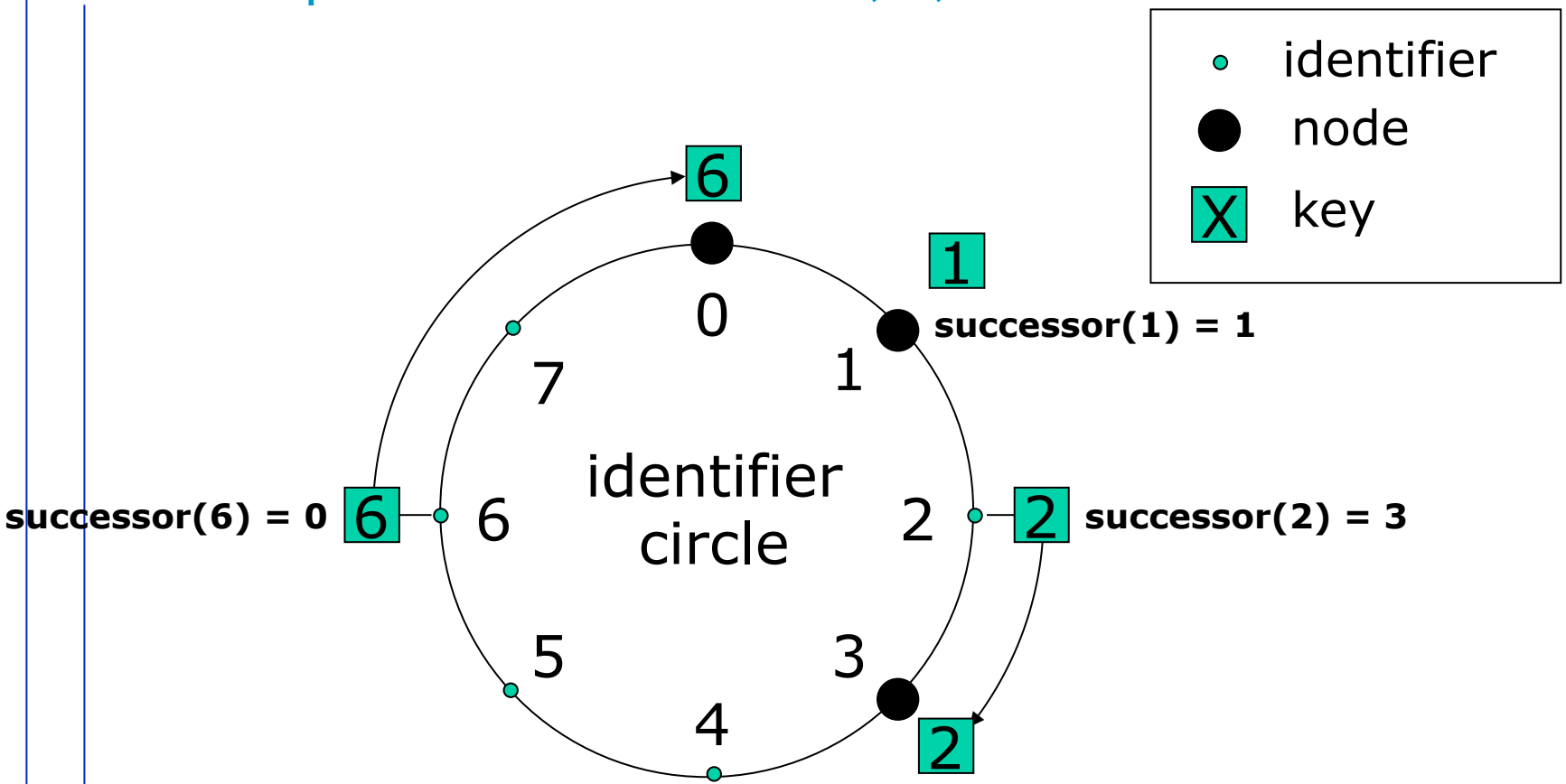
Chord: Consistent Hashing

- Atribui a cada nó e a cada chave um ID de m bits (permitindo 2^m identificadores)
- Usa SHA-1 como função de hash
  - ID(node) = hash(IP, Port)
  - ID(key) = hash(key)
- Identificadores são ordenados em um *círculo de identificadores*, modulo 2^m, chamado de *Chord Ring*.
- The identifier ring is called *Chord ring*.
- Chave k é posicionado no primeiro nó cujo ID é igual ou sucede ao ID de k no espaço de identificadores usados. Esse nó é chamado de *successor(k)*.

Departamento de Informática

Chord : Um exemplo

Exemplo: três nós com ID 0, 1, 3



PUC RIO

Departamento de Informática

Chord: Join and Departure

Operação de join:

- Quando um nó n entra no sistema (Chord Ring), algumas chaves k anteriormente atribuidas ao sucessor de n precisam ser movidas para n.

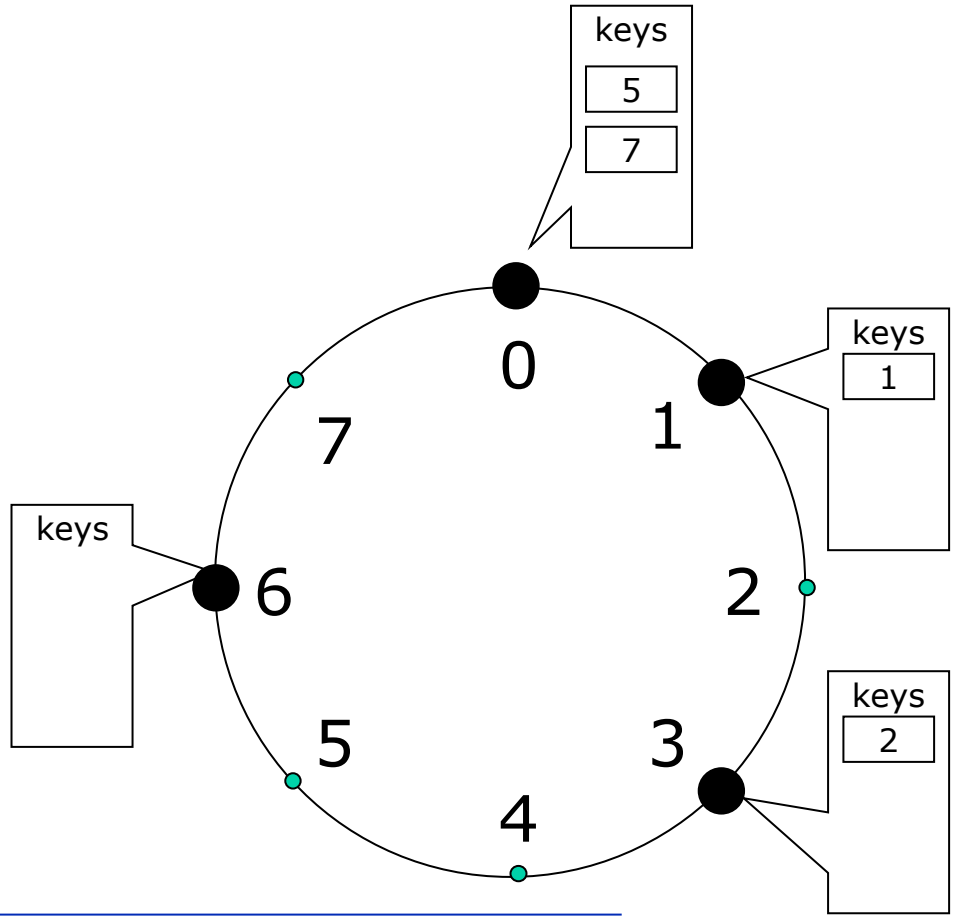
Operação de departure/leave:

- Quando um nó n sai do sistema todas as chaves que estavam com ele precisam ser movidas para o seu nó sucessor.

PUC RIO

Departamento de Informática

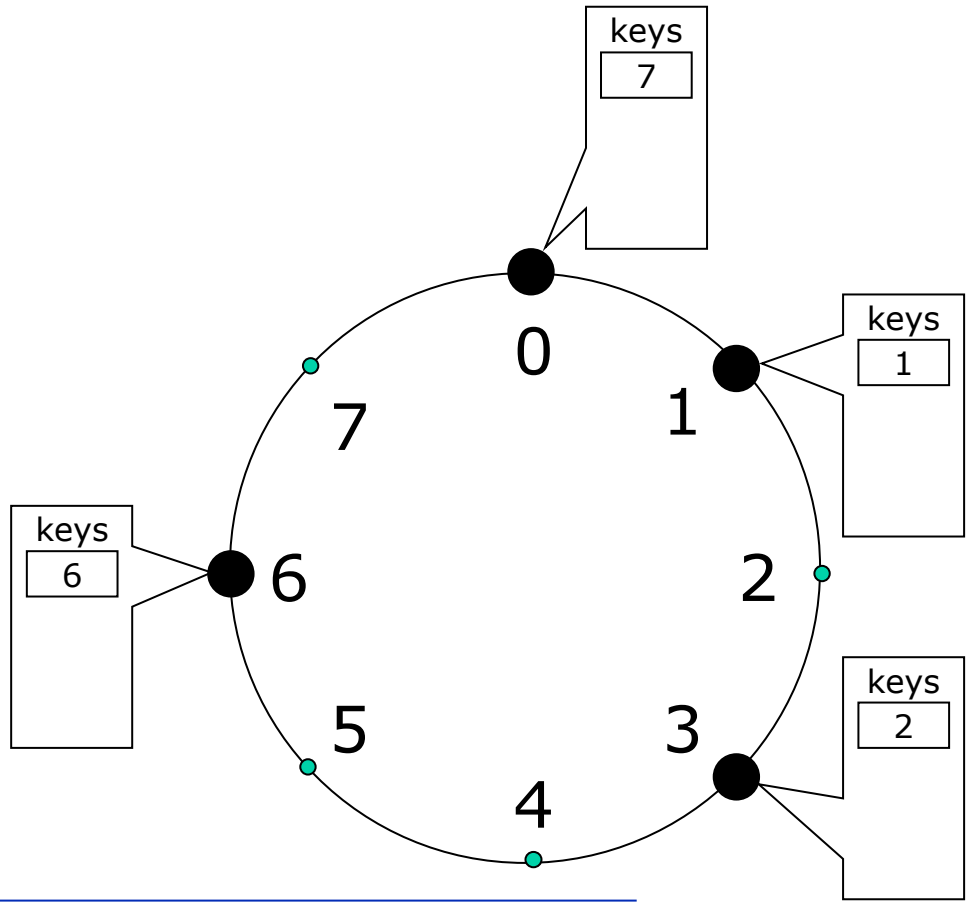
Chord: Exemplo
nó 6 entra no sistema



PUC RIO

Departamento de Informática

Chord: Exemplo
nó 1 sai do sistema



PUC RIO

Departamento de Informática

Lookup de chaves

- Basta pouca informação de roteamento para implementar hashing consistente em um sistema distribuído
- Se cada nó sabe como contactar o seu nó sucessor no círculo (Chord Ring) todos os nós podem ser visitados em órdem linear.
- Consultas a um certa chave podem ser passadas a diante no círculo até que chegue ao nó que contenha a chave.
- Pseudo-código para achar o nó sucessor:

```
n.find_successor(id)
        if (id ∈ (n, successor])
                return successor;
        else
                // forward the query around the circle
                return successor.find_successor(id);
```

PUC RIO

Descoberta de Serviços em Larga Escala (para IoT)

Em IoT têm-se alguns nós com mobilidade, mas alguns são fixos. Em compensação, têm-se o problema de larga escala.

Em [Cirani et al, 2014] IoT Gateways formam overlays P2P para descoberta de serviços em larga escala:

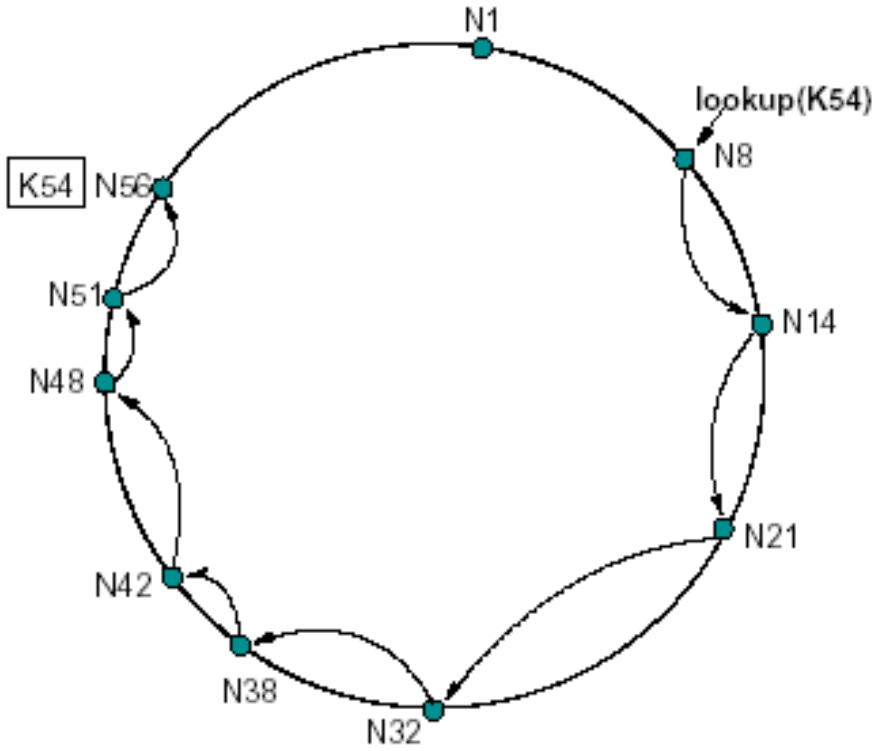
Overlay P2P de IoT gateways provê várias vantagens:

- **Escalabilidade**: overlays P2P podium attender a demandas crescentes à media que o número de nós/dispositivos aumenta.
- **Alta disponibilidade**: redes P2P são inerentemente robustas a falhas, pois não dependem de
- A**uto-configuração**: Protocolos P2P permitem que o overlay se reorganize automaticamente quando nós entram ou saem do system, não demandando qualquer intervenção manual.

Departamento de Informática

Exemplo de Lookup

- O caminho de um lookup de nó 8 para chave 54:



PUC RIO

Lookup Escalável

- Em alguns casos, um lookup de uma chave precisaria percorrer uma grande quantidade de nós.
- Para acelerar a descoberta, Chord mantém informação de roteamento adicional.
- Essa informação não é essencial para a corretude do protocolo (que é garantida se cada nó souber qual é o seu sucessor)…
- … mas permite um lookup muito mais eficiente

Departamento de Informática

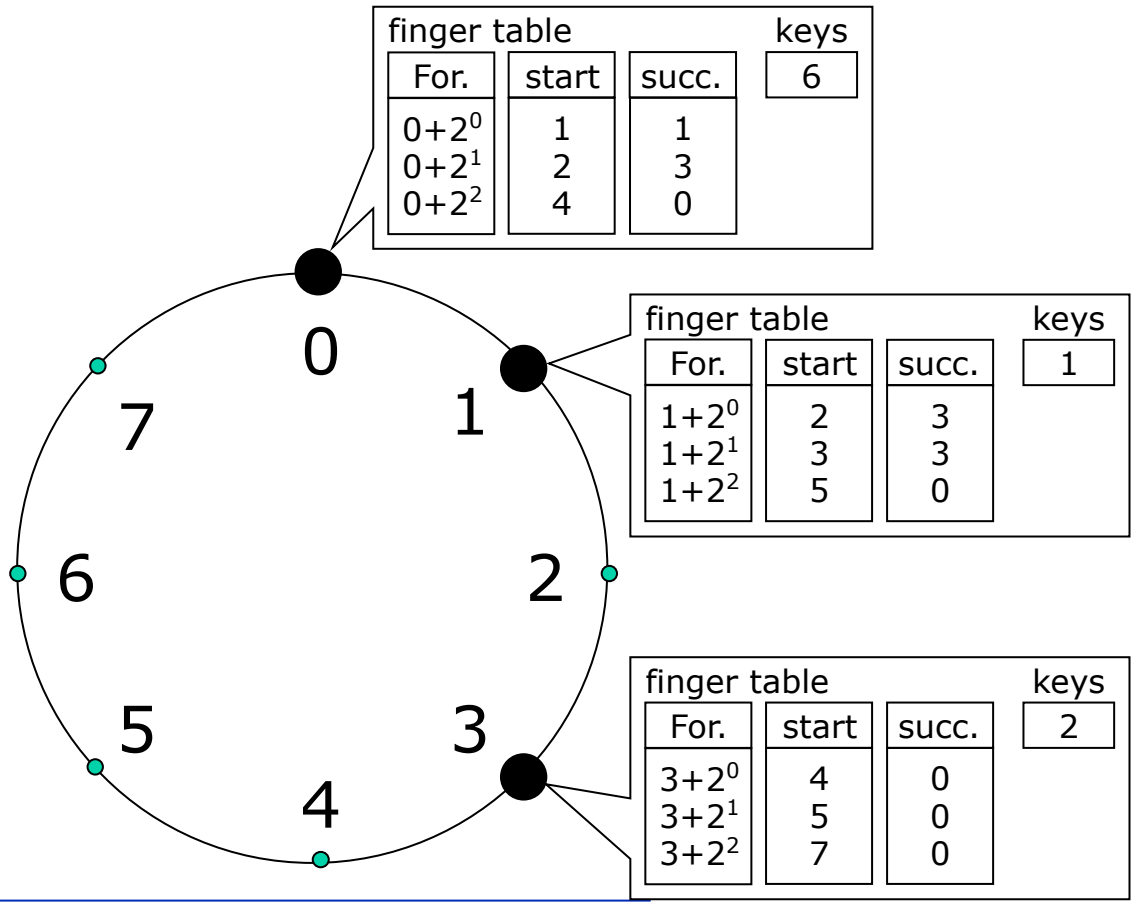
Lookup escalável – Finger Tables

- Cada nó n' mantém uma tabela de roteamento com até m entradas (o número de bits nos identificadores), chamada de *finger table.*
- A i-ésima entrada da tabela no nó n contém a identidade do primeiro nó s que sucede n em 2^{i-1} no circulo de identificadores. Ou seja:
- *s = successor(n+2i-1)*
- s é chamado de *i-ésimo finger* do nó n, ou *n.finger(i)*

PUC RIO

Departamento de Informática

Exemplo de Finger Tables

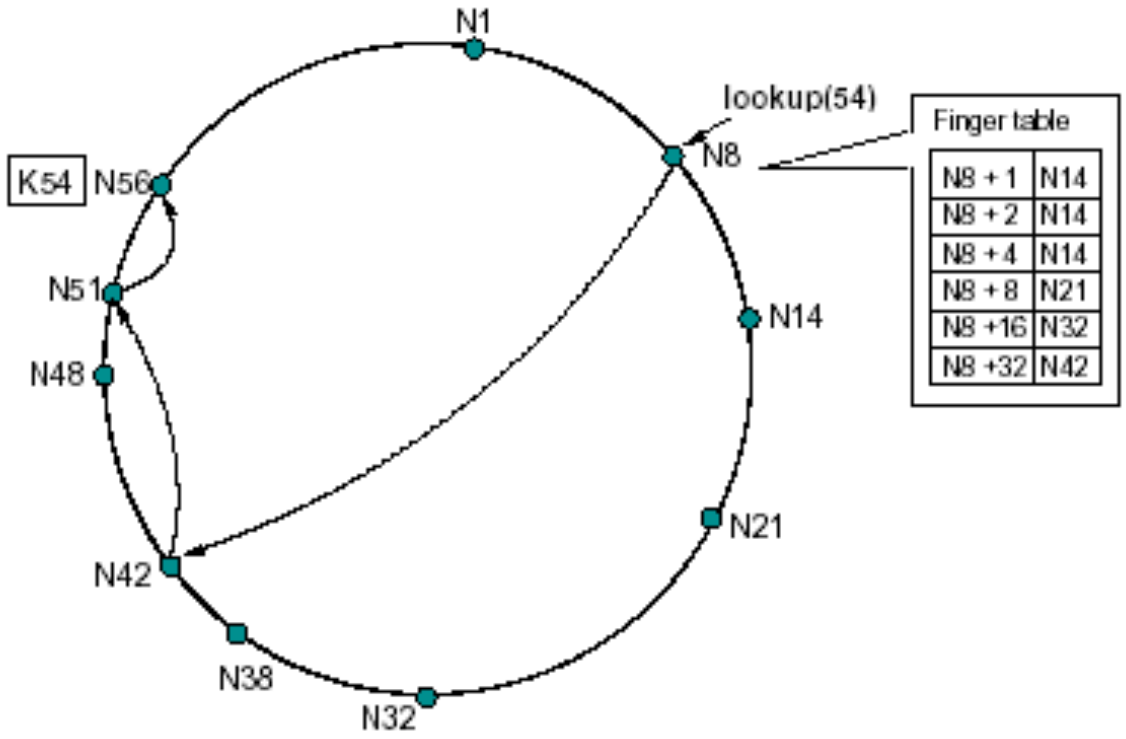


Lookup escalável – Finger Tables

- Uma entrada na finger table inclui tanto identificador Chord como o endereço IP (e port number) do nó correspondente.
- O primeiro finger de n é sempre o sucessor imediato de n no cículo.
- Como cada nó possui entradas em intervalos de potências de 2 em torno do cículo, este consegue dar um forward na msg de consulta em pelo menos metade da distância restante até o nó alvo (o que possui a chave).

Exemplo de Lookup usando a Finger Table

- O caminho de um lookup de nó 8 para chave 54



Departamento de Informática

Lookup Escalável: find_successor

- **Pseudo code:**

```
// ask node n to find the successor of id
n.find_successor(id)
    if (id ∈ (n, successor])
        return successor;
    else
        n' = closest_preceding_node(id);
        return n'.find_successor(id);

// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
    for i = m downto 1
        if (finger[i] ∈ (n, id))
            return finger[i];
    return n;
```

PUC RIO

Join de nós e estabilização

- A informação essencial em n é o ponteiro para o nó successor(n).
- Se esse ponteiro está atualizado (o que é suficiente para a corretude do lookup) então a finger table pode ser sempre verificada e reconstruida..
- Periodicamente, cada nó executa um protocolo de estabilização em background para atualizar o successor pointer e a finger table.

Departamento de Informática

Join de nós e estabilização

- O protocolo de estabilização contém 6 funções:
  - create()
  - join()
  - stabilize()
  - notify()
  - fix_fingers()
  - check_predecessor()

PUC RIO

Join de um nó

- Quando n inicia, este chama n.*join*(n'), onde n' é qualquer nó do Chord conhecido.
- *join*() pede para n' achar o sucessor imediato de n.
- *join*() não faz o restante dos nós ficarem cientes da existência de n!
  - Pseudo código:

```
// create a new Chord ring.
n.create()
        predecessor = nil;
        successor = n;

// join a Chord ring containing node n'.
n.join(n')
        predecessor = nil;
        successor = n'.find_successor(n);
```

Departamento de Informática

O Protocolo de estabilização

- Cada vez que nó n executa *stabilize*(), pede que seu sucessor s indique o seu predecessor p, e decide se p deve ser o seu successor em vez de s.
- *stabilize*() notifica o sucessor de n sobre a existência de n, possibilitando que este (s) troque predecessor(s) para apontar para n.
- O successor só faz essa troca se não conhecer outro predecessor mais próximo do que n

PUC RIO

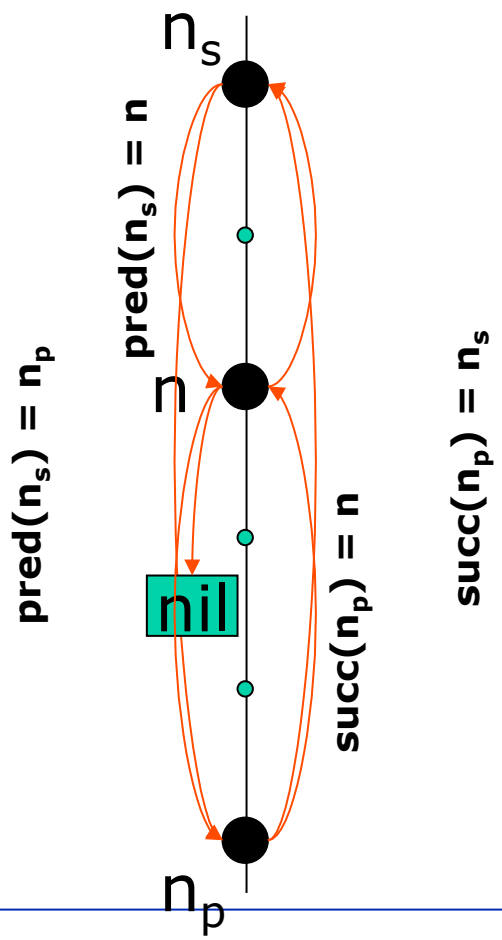
O Protocolo de estabilização

```
// called periodically. verifies n's immediate
// successor, and tells the successor about n.
n.stabilize()
    x = successor.predecessor;
    if (x ∈ (n, successor))
        successor = x;
    successor.notify(n);

// n' thinks it might be our predecessor.
n.notify(n')
if (predecessor is nil or n' ∈ (predecessor, n))
    predecessor = n';
```

Departamento de Informática

O Protocolo de estabilização



- **n faz join()**
  - predecessor = nil
  - n acquires n_s as successor via some n'
- **n executa stabilize()**
  - n notifies n_s being the new predecessor
  - n_s acquires n as its predecessor
- **n_p executa stabilize()**
  - n_p asks n_s for its predecessor (now n)
  - n_p acquires n as its successor
  - n_p notifies n
  - n will acquire n_p as its predecessor
- **todos os ponteiros predecessor and successor agora estão corretos**
- **Tabelas finger ainda precisam ser atualizadas, mas as entradas antigas ainda funcionam**

PUC RIO

Departamento de Informática

Entrada de nós - Fixfingers

- Cada nó periodicamente chama fix_fingers() para verificar se suas entradas na finger table estão corretas.
- Isso é feito tanto para nós que entram no sistema, quanto para nós antigos conhecerem novos nós entrantes

```
n.fix_fingers()
    next = next + 1 ;
    if (next > m)
        next = 1 ;
    finger[next] = find_successor(n + 2^(next-1));

// checks whether predecessor has failed.
n.check_predecessor()
    if (predecessor has failed)
        predecessor = nil;
```


PUC RIO

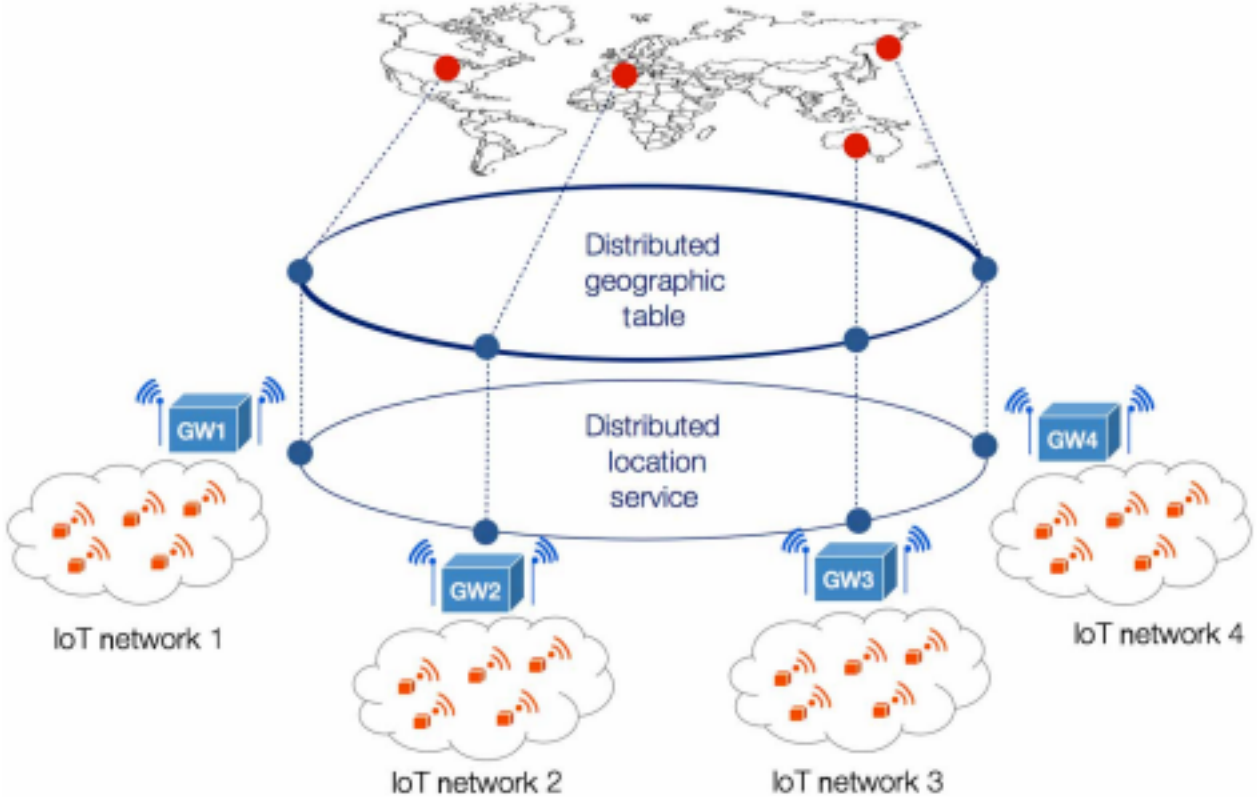
Departamento de Informática

Falha de nós

- O elemento principal de recuperação de falhas é manter um ponteiro correto para sucessor
- Para tal, cada nó mantém uma lista de sucessores (dos *r* sucessors mais próximos no círculo)
- Se un nó *n* percebe que seu sucessor falhou, troca este pela sua 1a. entrada nessa lista de um nó ativo
- Essa lista de sucessores precisa ser estabilizada assim:
  - Nó n atualiza sua lista com o seu sucessor s copiando a lista se sucessors de s, removendo a sua última entrada e colocando s como nó inicial de sua própria lista..
  - Se nó n percebe que seu sucessor falhou, troca ele pela primeira entrada de sua lista e atualiza a sua própria lista com a lista de sucessores do novo sucesor.

PUC RIO

Descoberta de Serviços para IoT



S. Cirani, L. Davoli, G. Ferrari, R. Léone,, P. Medagliani, M. Picone, and L Veltri,
A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things
IEEE INTERNET OF THINGS JOURNAL, VOL. 1, NO. 5, OCTOBER 2014

Departamento de Informática

DLS e DGS

Os *IoT Gateways* de [Cirani et al, 2014] participam de 2 overlays:

DLS é uma arquitetura baseada em DHT que implementa um serviço de localização, usado para armazenar e recuperar informações sobre serviços providos por servidores CoAP nas redes de cada IoT Gateway

- Exemplo: URI, tempo de expiração, prioridade de acesso, etc.
- URI do recurso é a chave (key do hash), e o valor é informação estruturada que pode incluir informação de localização.
- DLS funciona como uma resolução de nomes (para a URI), similar ao DNS para internet, só que distribuído e mais robusto a falhas

DGT é um overlay estruturado, criado usando diretamente a informação sobre a localização geográfica dos nós. A responsabilidade para manter a informação geográfica de cada peers ativos é compartilhada entre todos os nós.

- Cada peer consegue descobir quais são os nós ou serviços próximos de qualquer posição geográfica.
- Além, disso, cada IoT Gateway mantém um Local Node Directory.

M. Picone, M. Amoretti, and F. Zanichelli, "GeoKad: A P2P distributed localization protocol," in Proc. 8th IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PERCOM), 2010, pp. 800–803

Anúncio na entrada de novo dispositivo CoAP

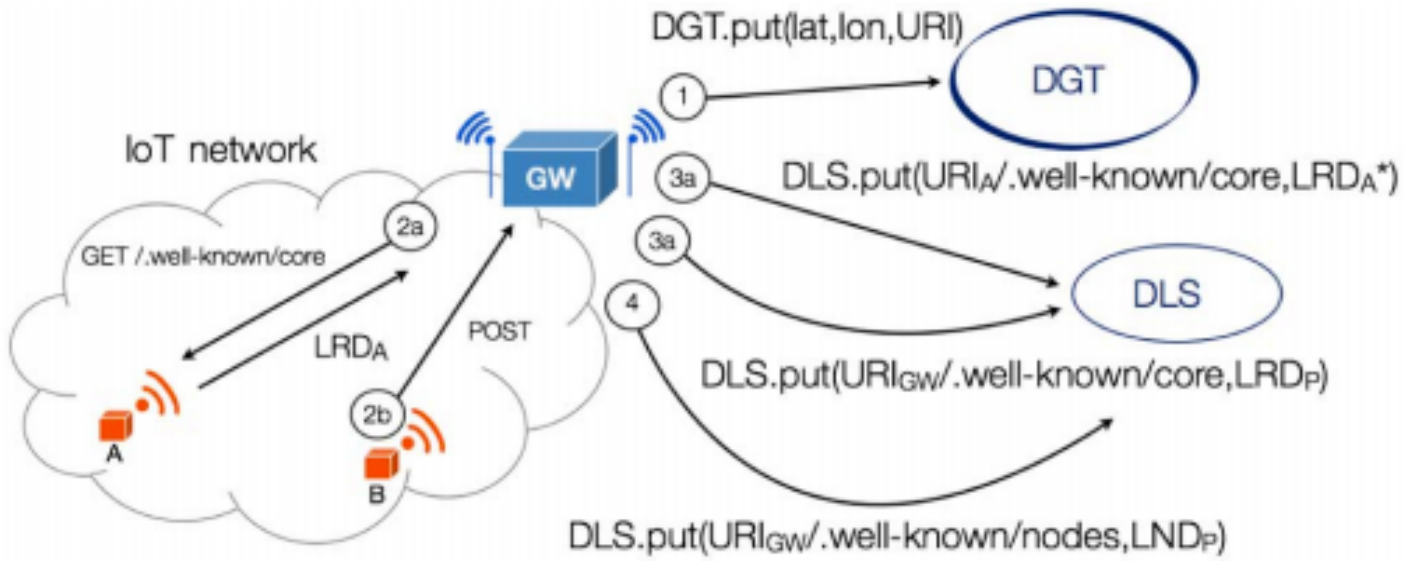


Fig. 4. Messages exchanged when a new node joins the network. First, the IoT Gateway discovers the resources of a new CoAP server or stores them on behalf of a CoAP client. Finally, DGT and DLS are updated with information about the new node.

Departamento de Informática

Anúncio na entrada de novo dispositivo CoAP

1. IoT Gateway joins the DLS and DGT overlays, and publishes its presence
2. When the IoT Gateway detects a new CoAP node in the network, through any suitable means (e.g., Zeroconf), it fetches the node's Local Resource Directory (LRD) through a CoAP GET request targeting the /.wellknown/core URI.
3. The LRD is filled with JSON-WSP3 documents containing the description of all the resources that are hosted by the CoAP node and the information to be used to access them. The resources included in the fetched node's LRD are added to the IoT Gateway's LRD.
4. It then publishes this information in the DLS and DGS
5. If the IoT Gateway is willing to let the resources be reachable through it, it will modify its LRD to include the references of the URLs to be used to reach the resources through the IoT Gateway
6. The IoT Gateway keeps track of the list of nodes that are in its managed network, by adding the node in a Local Node Directory (LND).

Departamento de Informática

SD para IoT: Outras propostas

- [Kaneko 2005] **P2P Interactive Agent eXtensions (PIAX)**, é uma platform P2P platform para localização geográfica de serviços, mas todos os nós são pares no overlay P2P, o que não é escalável.

Y. Kaneko, K. Harumoto, S. Fukumura, S. Shimojo, and S. Nishio, “A location-based peer-to-peer network for context-aware services in a ubiquitous environment,” Symposium on Appl. Internet Workshops, Jan. 2005, pp. 208–211

PUC RIO

Descoberta de Serviços: Conclusão

- A maioria dos protocolos para MANETs é single-hop
- Dos protocolos multi-hops, atualmente maioria é do tipo “sem diretório e sem overlay”, devido à própria natureza de uma MANET.
- Categoria promissora de protocolos é: com diretórios e com overlay, devido à escalabilidade e ao fato de que nós móveis possuem perfis de mobilidade distintos
- Em overlays, DHTs são uma ótima opção
- Vários outros aspectos são igualmente relevantes:
  - Segurança, Autenticação Mútua
  - Sistemas de Reputação e Compensação de serviços
  - Continuidade de Serviço

Jini

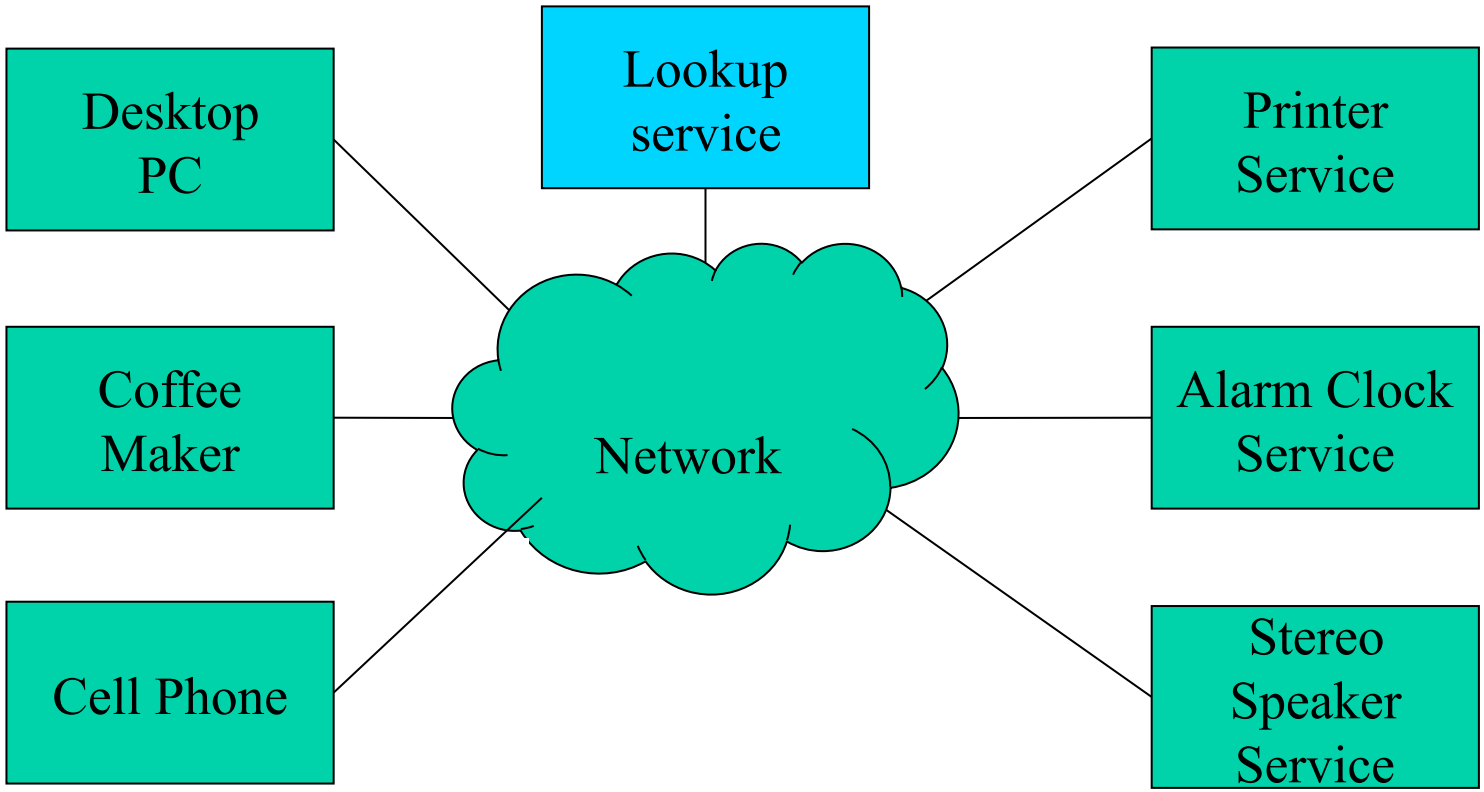
- Jini foi desenvolvido pela Sun com a filosofia "A rede é o computador"
- Jini é um middleware para "network plug and work" onde serviços, dispositivos e clientes podem se descobrir espontaneamente a fim de interagir
- Implementado em Java e usando RMI
- Assume que todos os componentes executam uma JVM (podem ser wrappers Java de objetos/código não-Java)

Motivação

- Ser um framework para interoperabilidade simples, invisível e escalável
  - Exigir o mínimo de administração “network plug and play"
  - Software e hardware em rede provêm serviços
  - Qualquer dispositivo pode achar e usar serviços existentes

- Exemplos
  - “Ache todos as impressoras coloridas próximas”
  - “Iniciar a máquina de fazer café 5 minutos antes do despertador tocar"
  - “Transfira o som do celular para os auto-falantes do carro”

A Proposta Jini



Federação Jini: o grupo de clientes e serviços que podem se encontrar

Componentes

Os principais componentes:

- serviço (impressora, torradeira, ou qualquer serviço na rede)
- cliente
- o Lookup Service (broker/trader/locator)

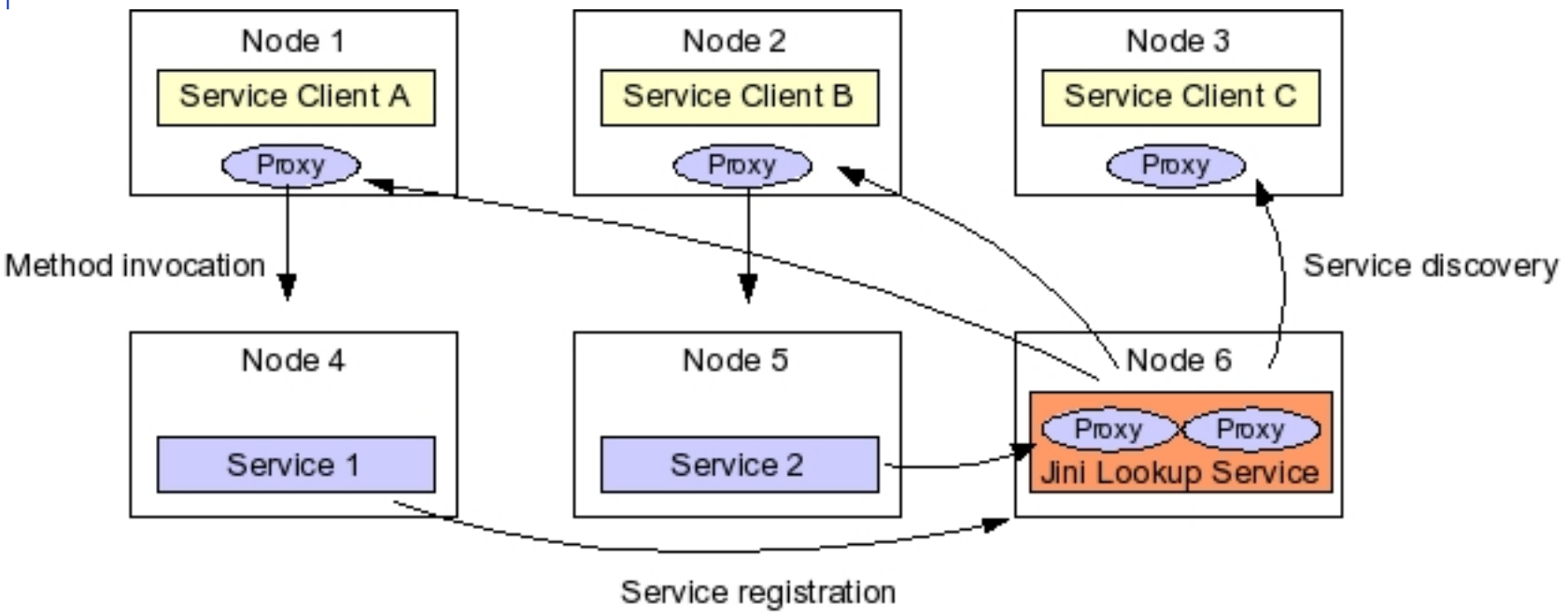
Além disto, assume-se a existência de uma rede (geralmente TCP/IP)

Em Jini, serviços são sempre acessados através de Proxies (um objeto fornecido pelo serviço), onde o proxy:

- pode realizar o serviço (“fat proxy”)
- É um RMI stub para interagir com um objeto remoto (“thin proxy”)
- Usa um protocolo proprietário para interagir com o serviço (“smart proxy”)

Um proxy é carregado no cliente quando este deseja usar o serviço.

Jini Proxy

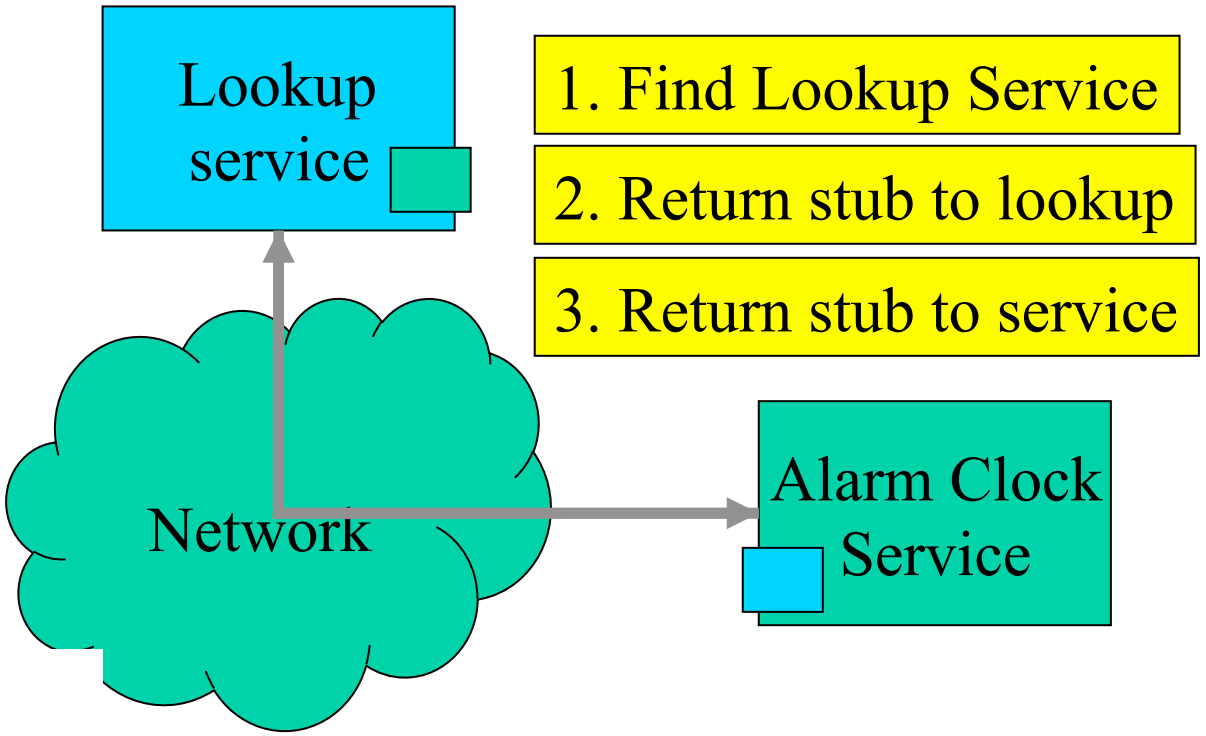


O que Jini provê?

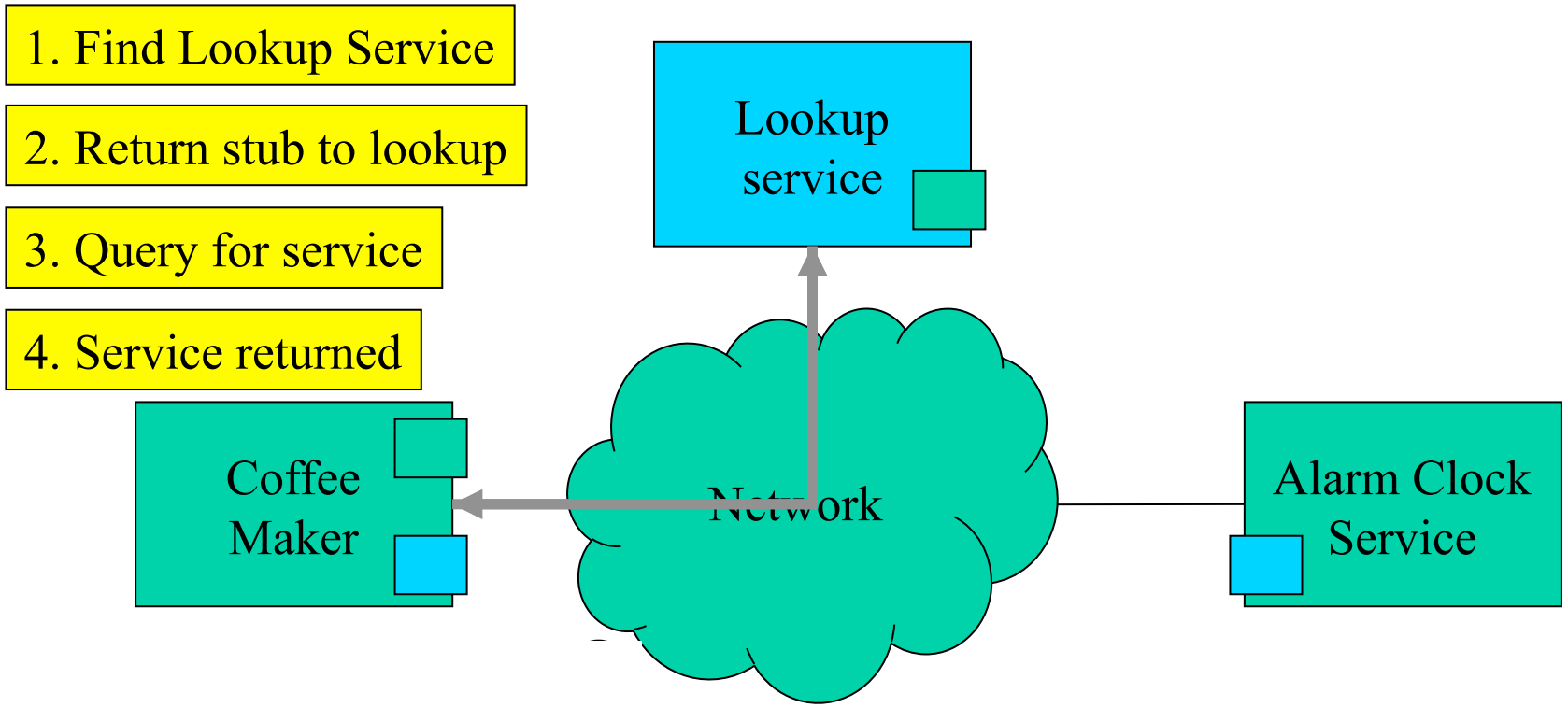
- Interfaces e implementações de funções de MW
- Um modelo de programação para serviços distribuídos
- A federação e a facilidade de acesso transparente a um serviço

- O *lookup service* é o elemento central
  - Para registro, descoberta, e uso temporário de serviços (leasing)
  - Usa um endereço IP multicast pré-definido

Registro de um serviço

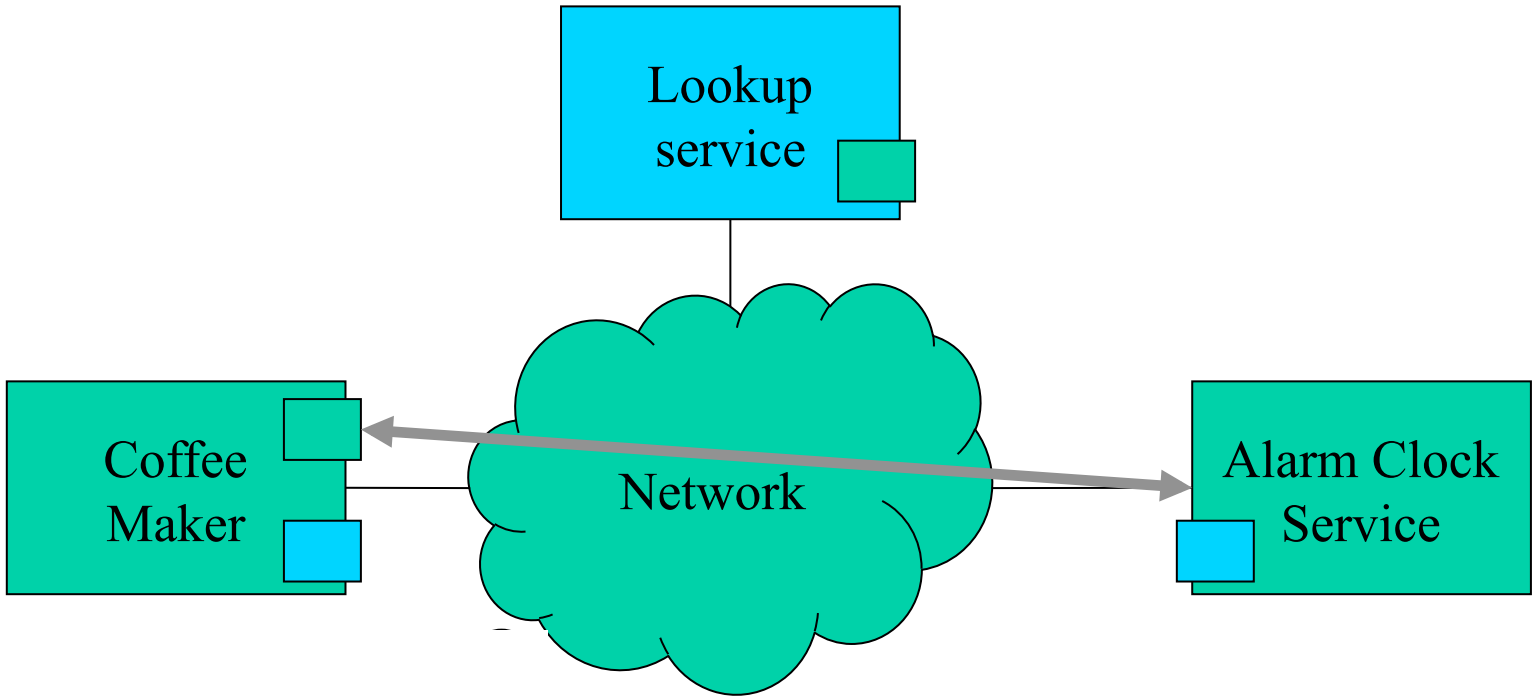


Leasing



Find interface Printer (duplex=yes, color=yes)

Usando um serviço



Pode usar qq protocolo para comunicação/acesso ao serviço (ou o próprio stub provê o serviço)

Vantagens do Jini

- Filosofia Jini: "Write Once Run Anywhere" (usando download de código)
- Permite a formação de composições hierárquicas de lookup services (como DNS)
- O modelo de Leasing trata de falhas do serviço
- Consulta flexível (por serviços) baseada em propriedades
- Suporte a transações distribuídas (two-phase)
- Eventos Distribuídos

Alguns Problemas do Jini

- Requer conhecimento *a priori* da interface do serviço
- Há necessidade de uniformizar as interfaces para um mesmo tipo de serviço
  - Exemplo: “Interface Printer" and “Interface Speakers"
- Requer Java VMs em todos os nós, demandando assim muita memória e processamento
- Complexidade do código

Alguns Problemas do Juni (cont.)

- Dificuldade de acoplar interfaces de usuário
- Ausência de mecanismos de segurança
- Service lookup é ponto central de falha (não há réplicas de registros de serviços)

Outras Tecnologias

Universal Plug and Play (UPnP)

- UPnP usa formato XML e protocolos baseados em HTTP
- Filosofia UPnP: ser independente de linguagem, uma API para cada plataforma
- Não faz download de código (como Jini)

UPnP

É uma arquitetura para conectividade P2P entre aparelhos diversos em uma rede, que:

- Possibilita aparelhos disponibilizarem interfaces para controle por computadores (ou outos dispositivos)
- Implementa registro totalmente descentralizado
- É baseado em padrões IETF

Principais Conceitos:

- Dispositivo (device)
- Ponto de controle (control point)
- Um device pode disponibilizar vários serviços

Exemplo UPnP: uma TV

- TV se registra como “device”
- Exporta o “Control Service” volume, power, channel
- Exporta o serviço “Imagem” para cor, contraste, brilho, etc.
- Um PC ou outro dispositivo pode ser o “controller” para esta TV

- “devices” & “controllers” se descobrem em uma rede não gerenciada (sem DNS ou DHCP)

Possíveis ações em UPnP

- **Descoberta** de devices & serviços
- Obter uma **descrição** dos devices
- **Controlar** os devices descobertos
- Ser informado de **eventos** indicando mudanças do device
- Usar uma **apresentação** preparada pelo device para apresentar um controle???

Descoberta UPnP

- Baseado em SSDP (simple service discovery protocol – IETF draft)
- Sempre que novo device é adicionado na rede, este pode anunciar seus serviços para os "control points"
- Quando um control point é adicionado, este pode procurar por dispositivos existentes
- Durante a descoberta, informa-se o tipo de dispositivo, um identificador e uma URL (apontando para mais informações)
- Como parte da descoberta, um control point obtém a descrição através da URL
- Um documento XML descreve o device e seus serviços
  - Informação específica sobre produtor, tipo, modelo, versão, número serial, e URL do Web site do produtor

Controle & Eventos UPnP

Controle UPnP

- Control point pode enviar mensagens de controle para um serviço de um device
  - Mensagens codificadas em XML usando o Simple Object Access Protocol (SOAP)

Eventos UPnP

- Descrição do serviço inclui variáveis que modelam o estado do serviço
- Usando Pub/Sub os control points são notificados a cada vez que uma variábel muda
- Um evento especial incial informa os valores iniciais
- Eventos são formatados em XML e usam GENA (General Event Notification Architecture)

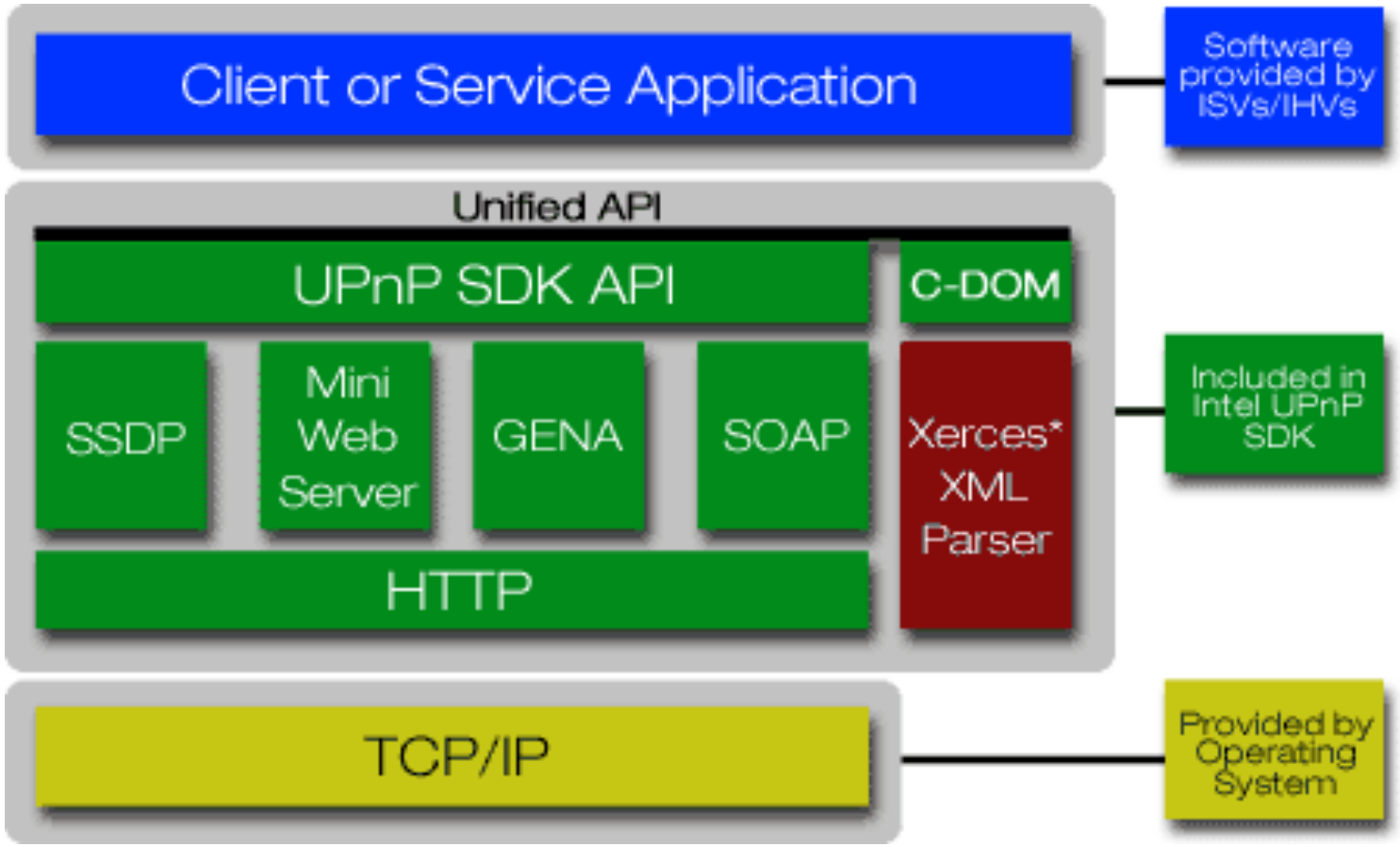
Apresentação UPnP

Apresentação:

- Um device pode oferecer uma URL para apresentação
- Control point pode obter a página e mostra-la em um browser, de forma a observar o estado do dispositivo

- UPNP somente trata de obter a página

Arquitetura UPnP



Outras Tecnologias

Salutation

- Aim is to be independent of vendors and network architectures (ex. Java, UDP, TCP/IP, HTTP)
- Aim for platform, OS, and network independence
- All service functions mediated by Salutation Manager
- Completely open and nonproprietary architecture
- Small footprint
- Point-to-point service discovery (but can also do directory-based too)

Algumas URLs

- Jini Specification
  http://www.sun.com/jini/specs
- Jini Developers
  http://jini.org
- Universal Plug and Play
  http://www.upnp.org
- Salutation
  http://www.salutation.org