Gerenciamento de Localização (Mobilidade)

Referências:
- Cap. 5 do Livro de Pitoura, E. & Samaras, G.
- Nacif01: Simulação e Gerenciamento de Unidades Móveis em Ambientes de Computação sem Fio, Tese de Doutorado, DCC/UFMG, 2001
- Artigos 1.6 e Artigo 1 de GM (link "Mais Artigos")
- Aljadhai99: A Framework for Adaptive and Predictive QoS Support in Wireless and Mobile Networks (PhD Thesis, U. of Pittsburgh
- Chan, Zhou, Seneviratne98: A QoS Adaptive Mobility Prediction Scheme for Wireless Networks, GlobeCom98, Sidney, 1998.

Gerenciamento de Mobilidade

Gerenciamento de Mobilidade (GM):

- trata da atualização da informação sobre localização de usuários móveis.
- permite que usuários com dispositivos móveis (Unidades Móveis - UM) tenham acesso a serviços, tais como recebimento de chamadas, mensagens/ pacotes/ fluxos de dados, enquanto se movimentam.

Introdução

Tipos de mobilidade:

- Mobilidade de Terminais: capacidade de manter a conectividade com a rede independente de movimentação, de forma a manter ativos quaisquer serviços da rede;
- Mobilidade de usuários: prover serviços independente do terminal; Distingue-se entre usuário e terminal (p.ex. GSM + smart card com Subscr. Identification Module- SIM)
- Mobilidade de serviços: provisão de serviços a clientes móveis independente do domínio administrativo da rede sem fio.

Gerenciamento de Mobilidade

Gerenciamento de Mobilidade (GM) é um problema intrínseco à Computação Móvel, e é encontrado nos mais diversos tipos de redes:

- telefonia celular
- IP Móvel
- Redes Ad-hoc (MANET)
- Comunicação via satélite
- Personal Area Networks (p.ex. Bluetooth)

- Onde cada tipo de rede têm objetivos e restrições próprias, por exemplo:
  - telefonia: completar uma chamada para um usuário
  - IP Móvel: roteamento de datagramas
  - MANET: manter/descobrir rotas
  - Satélites: manter/descobrir rotas entre satélites
  - PANs: descobrir disp. com energia/recursos suficientes

Gerenciamento de Mobilidade

Fatos:

- o custo computacional associado ao GM é um overhead inevitável
- tem consequências sobre o desempenho de protocolos/serviços na rede móvel
- o GM pode ocorrer em diferentes níveis do sistema.

  Exemplos:
  - físico: qual o ponto de acesso (MSS) mais apropriado para atender o dispositivo móvel?
  - físico (telefonia): o usuário se encontra na região de assinatura ou em "roaming"?
  - Rede (Mobile IP): qual é o Care-of-Address corrente?
  - Rede (Wireless ATM): qual é o melhor switch para garantir QoS?
  - sessão/aplicação: em qual região/domínio o usuário se encontra?

Gerenciamento de Mobilidade

Gerenciamento de Mobilidade consiste de:

- Gerenciamento de Localização
  - manter atualizada a informação sobre a localização e UMs
  - identificar a localização exata antes de completar uma chamada, ou entregar uma mensagem
  - tratar chamadas (entrega de mensagens) pendentes quando localização não é possível
- Gerenciamento de Handover (Handoff)
  - transferência automática de conexões ou reencaminhamento de mensagens com o mínimo de interrupção (ou degradação de QoS)
  - reserva antecipada de recursos & renegociação
  - transferência do estado de execução e re-estabelecimento do contexto de execução (p.ex: descoberta de servidores)

A partir de agora daremos atenção especial ao aspecto de Gerenciamento de Localização

Localização de Unidades Móveis

Tendências em redes móveis/celulares:

- aumento do número de dispositivos/usuários móveis
  ➔ precisam ser localizados
- serviços baseados em localização
  ➔ usam informação sobre localização como parâmetro adicional
- células/regiões de localização cada vez menores ➔ atualização mais frequente da informação sobre localização

Estes fatores requerem estratégias eficientes para o monitoramento, a consulta, a atualização e o armazenamento da informação sobre a localização.

Localização de Unidades Móveis

Pode-se entender GL como um problema de Diretórios (base de dados contendo endereços), que tem características específicas :

- devido à migração/movimentação, a taxa de atualização pode ser elevada (e variar muito)
- a taxa de consulta (busca por uma UM) pode variar muito e ter requisitos de eficiência (urgência) diferenciados
- taxas de migração e consulta podem variar muito para diferentes tipos de usuários, regiões e horas do dia

A razão entre a taxa de consultas e a taxa de migração (Razão Chamada/Mobilidade - *Call-to-Mobility Ratio*) é um parâmetro importante na avaliação do desempenho de uma estratégia de GL.

Estratégias para Gerenciamento de Localização

As estratégias são compostas de:

- Estratégias de Atualização (Update) e de Consulta (Lookup)
- Particionamento da área de cobertura em Áreas de Localização - AL (p.ex. Conjunto de células)
- Utilizam Registros de Localização (em Servidores de Localização) organizados de certa forma, que armazenam os endereços dos dispositivos/usuários.
- Update é implementado através de sinais de controle na interface sem fio (p.ex. Beacon), e através de mensagens de sinalização na rede fixa.

➔ Eficiência GL = Eficiência Update + Eficiência Lookup

Possíveis Estratégias para Atualização de uma UM

- UM nunca emite sinal para Update
  - ➔ isto requer uma consulta a todas as AL (*blanket paging*)
- Update por tempo
  - ◆ Periodicamente UM envia sinal sobre sua localização (AMPS)
  - \- UM precisa estar ativa
  - \- há gasto de energia
  - \+ pode ser eficiente se periodicidade de migração ≈ periodicidade de sinalização
- Update por distância
  - ◆ UM envia sinal sempre que tiver percorrido uma distância limite desde o último envio
  - \- UM precisa ter um sensor de localização (GPS), que consome energia continuamente
  - \+ se raio da AL for conhecido, UM consegue sinalizar precisamente a mudança de AL

Possíveis Estratégias para Atualização de uma UM

- Update por número de hand-offs
  - UM envia sinal sempre que atravessar um determ. número de células
  - \+ não requer sensor de localização, pois UM pode contar o nº de hand-offs realizados
  - \- nº de handoffs ≈ distância percorrida, mas não a distância de deslocamento efetivo fim-a-fim (p.ex: percorrimento circular)
  - \+ fácil localizar UM através de paging hierárquico
- Update na Transposição de Limite de AL (IS-41em 2G)
  - cada ERB conheçe a AL a qual pertence; a cada hand-off entre células de ALs distintas, é gerado um Update
  - \+ não requer participação da UM (➔ economia de energia)
  - \- pode gerar muitas mensagens Update caso todas as UMs tenham que atravessar frequentemente os limites de ALs vizinhas (estrada em "zigue-zague")

Possíveis Estratégias para Atualização de uma UM

- Update por Categoria de usuário
  - Ideia central: usar a estratégia mais eficiente de acordo com o padrão de mobilidade de cada categoria:
    - movimentação pouco frequente e regular (ex: empregado, servidor)
    - movimentação frequente e irregular (ex: taxi, motoboy)
    - movimentação pouco frequente e irregular (ex: vendedor ambulante, corretor de imóveis, etc.)
  - Além disto: um histórico de mobilidade por categoria permite prever futura movimentação (e localização eficiente) com certa probabilidade (*Mobile User Profile*)
  - \+ para certas categorias de usuários, permite otimizar atualizações

  P.ex. Categoria1: atualização por transposição de AL

  Categoria2: atualização periódica
  - \- pode ser caro ou impossível manter o histórico de mobilidade

  trata-se de uma estratégia complexa, porém eficiente

Possíveis Estratégias para Atualização de uma UM

Segundo [Imielinksi & Badrinath] um Mobile User Profile deveria conter:

- probabilidades de locomoção entre quaisquer duas ALs
- número médio de chamadas (localizações) por unidade de tempo
- número médio de movimenções por unidade de tempo

… e tudo isto registrado para os diferentes períodos do dia:

- 6 - 8 hs - morning rush
- 8 - 12 hs - morning work
- 12 - 14 hs - lunch
- 14 - 17 hs - afternoon work
- 17 - 20 hs - afternoon rush
- 20 - 6 hs - night

Localização de uma UM

Ao contrário das estratégias de atualização, cujo custo é:

- proporcional ao #UMs e à frequência de movimentação; e
- inversamente proporcional ao tamanho das ALs

O Custo da localização (LookUp) depende de:

- frequência de chamadas (ou envios para UMs)
- grau de precisão (e atualidade) da informação de localização
- # de ERBs em uma AL (ou tamanho das células)

Principais Objetivos da Localização:

- localizar uma UM o mais rápido possível
- usar a menor quantidade possível de recursos (canais/ largura de banda) para a sinalização
- obter a localização precisa de uma UM

Possíveis Estratégias para Localização de uma UM

"Blanket Paging" (ou Flooding):

- ◆ Todas as ERBs enviam um sinal de paging e esperam por resposta da UM correspondente
- \+ não requer que as UMs enviem informação sobre localização atual
- \- difusão gasta muita largura de banda do meio sem fio (canal de controle)
- \- quando existe apenas 1 canal de controle permite somente localização sequencial de UMs

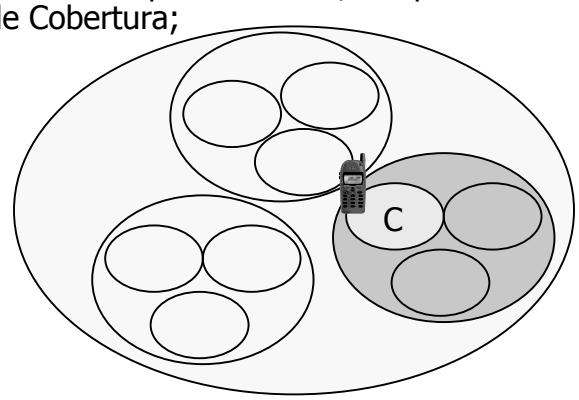
ERBs na atual AL:

- ◆ Somente as ERBs das células da AL mais recente fazem o paging
- \+ menor uso de recursos do que no Blanket paging
- \- com 1 canal, localização sequencial ainda dentro de AL
- \- se UM já migrou para outra AL, uso desnecessário de largura de banda
- \- requer Estratégia de Atualização de Posição (AL) precisa e eficiente

Possíveis Estratégias para Localização de uma UM

Busca Hierarquica:

- Define-se uma hierarquia de ALs aninhadas; primeiro busca-se na AL corrente (digamos C); depois em ALs cada vez maiores que incluam C; até possivelmente toda a Área de Cobertura;



Busca Hierárquica

Prós e Contras:

+ Flexibilidade: dependendo da urgência da busca, pode-se pular algumas etapas e fazer diretamente o Blanket Paging
+ estratégia otimista, que visa minimizar o gasto da largura de banda
+ dependendo da frequência e da velocidade de movimentação das UMs, descoberta da UM pode ocorrer já em ALs menores (níveis maiores da hierarquia) ➔ eficiência variável
- a latência da localização tende a ser maior do que a do Blanket Paging
+ pode ser otimizada a partir de informação sobre direção e velocidade de deslocamento das UMs

Possíveis Estratégias para Localização de uma UM

Busca Diferenciada por Categoria:

- A estratégia de localização é definida de acordo com o Mobile User Profile (MOP), levando em conta
  - direção, velocidade e frequência de mobilidade e
  - frequência de chamadas
- e geralmente é combinada com uma busca hierarquica e com algoritmos de predição de mobilidade
  - requer manter um MOP para cada UM
  + o histórico de mobilidade para "UMs com padrões regulares" permite uma busca eficiente
  - complexidade: requer gerenciamento de recursos mais complexo e definiçao de prioridades entre chamadas para diferentes categorias (busca hierarquica vs. blanket paging direto)

Possíveis Estratégias para Localização de uma UM

Predição de Mobilidade:

- Utiliza-se algoritmos sub-ótimos baseados na análise do Histórico de Movimentação
- Possíveis critérios:
  - Direção: Assume-se que UM manterá a direção global
  - Localização: Deduz-se próxima localização a partir da posição corrente (áreas vizinhas)
  - Temporal: Critério de Direção em função da hora do dia
  - Segmento: Deduz-se próxima etapa de um itinerário a partir da direção corrente
  - Correlação: quando não há registro para determinada AL, então usa-se histórico de movimentação de uma população de UMs

  Predição baseada no histórico naturalmente tem suas limitações pois movimentação recente pode não estar refletida no histórico (predição probabilistica)

➔ [Chan,Zhou,Seneviratne98] mencionam que o Criério de Direção fornece os melhores resultados

Arquiteturas para Gerenciamento de Localização

Pitoura & Samaras distinguem 3 níveis de redes:

- rede fixa: interconecta ERBs, servidores e clientes
- rede de acesso: conjunto de ERBs (e seus domínios)
- "Redes de Dados": rede de servidores/repositórios de informação sobre localização (registros de localização -RL)

Informação sobre localização é volátil (tem um tempo de validade), e é descartada para UMs que não geram sinais, por exemplo, estão desativadas.

Principais Classes de Arquiteturas de Redes de Dados:

- Two-Tier (➔ Home Location & Visitor Location Register)
- Hierarquia de RLs

Critérios para Escolha da Estratégia

- Objetivo Central:

  Minimizar custo total = localização + atualização

Critérios/Restrições adicionais:

- Capacidade do serviço: taxa máxima de atualizações/consultas suportadas nos RL
- Limite de tamanho dos Registros
- Tipos & Frequência relativa de Chamadas e Migrações
  - CMRij = #chamadas para UMi na ALj em T seg / #migrações de UMi entre ALs em T seg

Nestas estratégias de atualização/localização podem ser:

- fixas ou adaptativas e
- uniformes (para todos usuários) ou seletivas

Arquitetura Two-Tier (Básica)

Usa dois tipos de bases de dados:

- Home Location Register (HLR): cada UM está associado a um HLR (na rede origem), que contém a localização atual da UM como parte de seu perfil
- Visitor Location Register (VLR): BD em cada AL contém uma cópia do perfil de todas as UMs atualmente posicionadas na AL

Chamada originada em ALi para UMx:

- procura por UMx em VLRi; só se não for achada, consulta-se HRL de UMx

Migração de UMx de ALi para ALj:

- atualização de HRL de UMx
- registro em ALi é apagado e novo registro é criado em ALj

Arquitetura Two-Tier ➢ Usos

Devido à sua simplicidade, vários padrões de redes celulares/móveis utilizam esta arquitetura.

Exemplos:

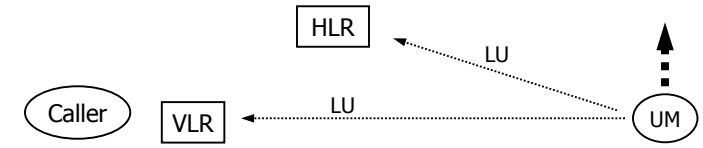
- Eletronics Industry Association Telecommunications Industry Association Interim Standard 41 (EIA/TIA IS41)
- Global Systems for Mobile Communications (GSM)
- IETF Mobile IP

Two-Tier ➢ Otimizações

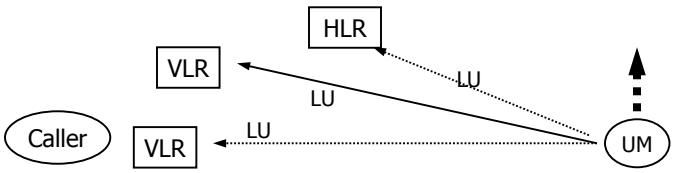
Três propostas canônicas de otimização:

- Caching por UM [Jain et al,94]
  - \+ reduz custo de localização
  - \- aumenta custo de atualização
  - \+ explora localidade espacial e temporal das chamadas
- Replicação (estática)
  - ◆ Replicação do User Profile [Shivakumar&Widom, MobiCom'95]
  - ◆ Replicação do Working Set [Rajagopalan et al., MobiCom'95]
- Ponteiros de Redirecionamento (Forwarding Pointers)
  [Jain & Lin, Wireless Networks 1, '95]
  - \+ reduz o custo de atualizações
  - \- aumenta o custo da localização

Two-Tier ➢ Otimizações



a) Caching



b) Replication

Two-Tier ➢ Caching por UM

Trata-se de um esquema de replicação dinâmica.

Ideia Central:

- Cada vez que UMx é chamada de uma ALi, o endereço de x (ou um ponteiro para o endereço) é armazenado no VLRi
- Para qualquer chamada subsequente originada em ALi pode-se usar esta informação. Portanto, primeiro é consultado o cache local, so depois é consultado o HLRx.

Discussão:

- vale a pena se
  - freq. chamadas de uma AL para UMx > freq. Migrações UMx
  - há localidade espacial e temporal de chamadas
- adota-se esquemas LRU para remover entradas obsoletas
- Atualização do cache pode ser preventivo (Eager) ou por demanda (Lazy).
  - Eager: cada movimentação causa uma atualização
  - Lazy: cache é atualizado somente como consequência de um envio

Two-Tier ➢ Caching por UM

- Eager Caching
  - Cada vez que um usuário se move para outra AL, todas as entradas nos caches são atualizadas
  - Aumenta em muito o custo de atualização para UMs que tem as suas informações em caches
    - precisa-se manter a informação sobre quais são os VLR com a informação cacheada
    - a atualização dos caches envolve mensagens adicionais na rede fixa
  - Em compensação, localização é muito eficiente
  - Exemplo: LocUpdate nos proxies do protocolo RDP

Two-Tier ➢ Caching por UM

- Lazy Caching
  - em toda migração para ALi, somente HLR e o VLRi são atualizados
  - demais caches só são atualizados quando ocorre um "cache miss", e UM é de fato localizado (através de consulta ao HLR)
  - overhead só quando há cache miss (consulta ao VLR local)
  - lazy é melhor do que Two-Tier básico se $p \geq C_H/C_B$ onde p é a hit ratio e C_H e C_B são os custos de localização com cache hit e no esquema basico, respectivamente. Estes dependem dos custos de acesso ao HLR e dos caches.

  Relacionando hit ratio p e CMRij:

  Assumindo uma Distribuição de Poisson para geração de novas chamadas λ e uma duração entre migrações exponencial μ, temos $p = \lambda/ (\lambda + \mu)$
  - Então o $CMR_{Min} = p/(1 - p)$ é o mínimo para que caching seja vantajoso ➔ Se CMRij > CMR_{Min} vale a pena fazer caching por UMi

Two-Tier ➢ Replicação

- Idéra Central: Copiar o perfil de uma UMx para o VLR de várias AL, a fim de diminuir o custo da localização
- Ao contrário de caching, é uma estratégia que leva em conta custos globais
  - + custo de atualização a cada migração é limitado e pode ser estimado
  - - esquema menos adaptativo do que o caching
  - - assume o conhecimento das frequências relativas de geração de chamadas em cada AL
  - - a escolha do posicionamento das réplicas é decisivo para a eficiência do esquema
- Atualização é mais cara ➔ só vale a pena se a redução do custo da busca for maior do que o custo de manter as réplicas atualizadas

Two-Tier ➢ Replicação

Formulação precisa do critério: "vale a pena replicar perfil de UMi em ALj ?"

- Sejam:
  - α: economia de custo quando LookUp se utiliza de inform. da réplica Alj (diferença entre cache hit e cache miss)
  - β: custo de atualizar a réplica em ALj
  - Cij: # estimado de chamadas para UMi provenientes de ALj em período T
  - Ui: # de migrações de UMi no período T
- Então manter réplica em ALj vale a pena se:

$$\alpha * Cij \geq \beta * Ui$$

Existem várias estratégias para decidir onde replicar perfil de UMi, mas veremos apenas duas "estratégias por UM":

- ◆ baseada no perfil do usuário
- ◆ baseada no Working Set (WS)

Two-Tier ➢ Replicação baseada no Perfil do Usuário

Objetivo: minimizar o custo total de atualizações e LookUps, sujeito às seguintes restrições:

- ◆ exsite um número máximo ri de réplicas por usuário i
- ◆ existe um número máximo pj de entradas no VLR de cada ALj

Sejam M = # de usuários e N = # de ALs

- Problema combinatório clássico: "Posicionamento de Recursos"
- escolher o conjunto R(i) de VLRs (ALs) tal que custo total

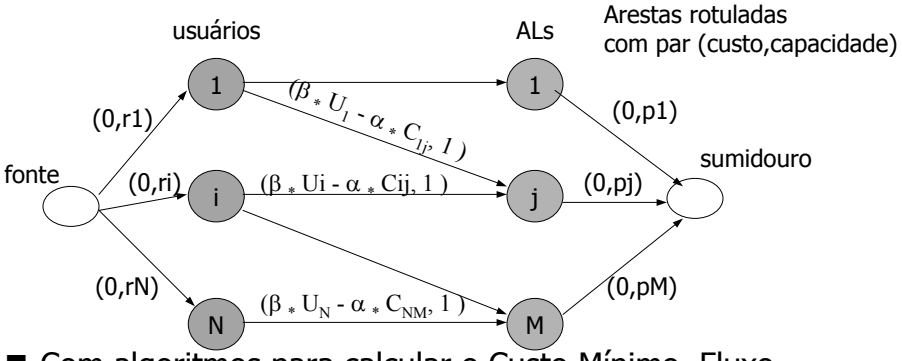
$$\sum^{N}_{i=1}\sum^{M}_{j\in R(i)} (\beta * Ui - \alpha * Cij)$$

é minimizado e as restrições acima permanecem satisfeitas.

A solução pode ser computada usando um Grafo de Fluxo.

Two-Tier ➢ Replicação baseada no Perfil do Usuário

- Solução baseada em Grafo de Fluxo:



- Com algoritmos para calcular o Custo Mínimo, Fluxo Máximo (Min-Cost-Max-Flow) de fonte p/ sumidouro

➔ acha-se o posicionamento ideal das réplicas

Two-Tier ➢ Replicação baseada no Working Set

Motivação Central do WS [Rajagopalan,Badrinath95]:

- Normalmente, cada usuário móvel se comunica e recebe chamadas/mensagens de um número reduzido de fontes (working set)

➔ Obtém-se melhor performance se cópias do perfil da UMx são mantidas somente nos VLR das ALs nas quais se encontram os elementos do WS(x)

- Não são consideradas restrições como:
  - capacidade limitada dos RLs
  - número máximo de réplicas por UM

A decisão sobre a replicação do perfil no VLR da ALj pode:

- ser tomada independentemente para cada UM
- adaptada (ajustada) dinamicamente

[Rajagopalan,Badrinath95] An Adaptive Location Management Strategy for Mobile IP, MobiCom'95

Two-Tier ➢ Adaptação do Working Set

A desigualdade Q: $\alpha * Cij \geq \beta * Ui$ é reavaliada em cada UMi sempre que:

1) uma nova chamada/ mensagem é recebida
2) UM migra para outra AL

- No caso 1: só é avaliado se AL do nó correspondente F (fonte) não pertencer ao WS(x)
  ➔ se Q é satisfeita, então F é incluído no WS
- No caso 2: desigualdade é reavaliada para todo j $\in$WS(x), e remove-se os elementos do WS para os quais Q não é satisfeita
- Custo computacional:
  1) todos os 4 termos de Q precisam ser medidos/ estimados
  2) somente Ui precisa ser reavaliado

Two-Tier ➢ Adaptação do Working Set

Simulações mostram que:

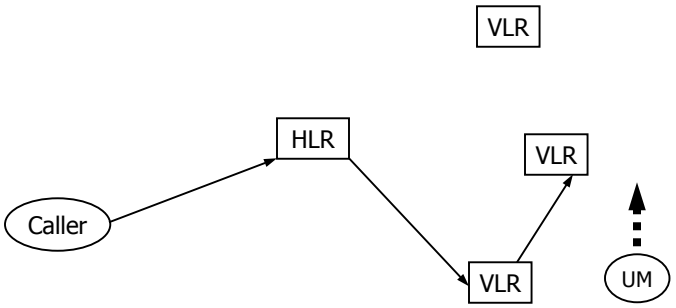
- Quando o CMRij é baixo, esta técnica funciona com o mesmo desempenho de um esquema sem replicação
- Quando o CMRij é alto, esta técnica funciona como uma replicação estática em que o WS(x) é fixo para cada usuário
- O desempenho da Replicação baseada em Working Sets essencialmente depende do CMR de cada usuário, e é independente do número de usuários, portanto:
  - + é uma técnica escalável e adaptativa
  - - a maior dificuldade está na obtenção de Cij e Ui (estimativas de # de chamadas e migrações)
  - - técnica deve levar em conta o histórico de interaçãoes a fim de evitar uma atualizações muito frequentes do WS (para garantir estabilidade)

Two-Tier ➢ Ponteiros de Redirecionamento

Idéa Central:

- Cada vez que UMx move de uma ALi para uma nova área, digamos ALj, é colocado no VLRi uma referência para VLRj, em vez de atualizar o HLR
- A cada nova chamada, consulta-se o HLR para obter a referência para o primero VLR, e depois segue-se a lista encadeada de VLRs
  - + Atualização é muito simpes e barata
  - - LookUp aumenta a medida que uma UM acumula migrações
- Para limitar o custo do LookUp, limita-se o comprimento máximo da cadeia
- E usa-se algorítmos de compressão de caminhos para eliminar loops

Two-Tier ➢ Ponteiros de Redirecionamento



c) Forwarding Pointer

Two-Tier ➢ Ponteiros de Redirecionamento

Esta técnica só vale a pena CMRs baixas (usuários que migram muito e recebem poucas chamadas/mensagens)

Pois:

- ◆ Atualização é barata (Update)
- ◆ Consulta é custosa (LookUp)

O benefício da técnica tambem depende do custo de modificar o HLR e de percorrer uma sequência de VLRs

Nas Otimizções de Rota do Mobile IP, usa-se ponteiros de redirecionamento para garantir o "smooth handover" (minimizar o # de datagramas perdidos durante a atualização do binding no *Home Agent*)

Two-Tier ➢ Resumo

Para a Arquitetura Two-Tier (HLR/VLR) vimos as seguintes técnicas:

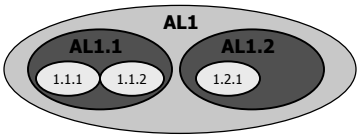
- Caching: ➔ *CMR alto*
  - Eager
  - Lazy
- Replicação: ➔ *CMR alto*
  - Perfil do Usuário
  - Working Set
- Ponteiros de Redirecionamento: ➔ *CMR baixo*

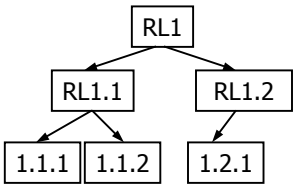
Em muitos sistemas, usa-se uma combinação das técnicas e/ou uma técnica para cada tipo de usuário.

Arquitetura Hierárquica

Princípio:

- Organizar os RLs de forma hierárquica (árvore), onde RLs de níveis superiores correspondem a ALs maiores, que englobam várias ALs de níveis inferiores



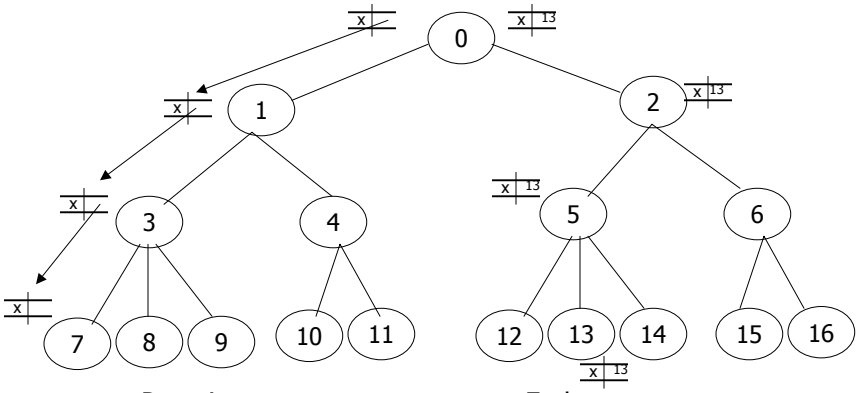


Motivação:
- Explorar melhor localidade de chamadas e migrações
- Evitar que cada UM precise ter um endereço fixo (HLR)

Consequências:
- Aumenta o # de acessos a RL e # de mensagens
- RL de níveis mais altos precisam ter capacidades maiores de armazenamento e processamento

Arquitetura Hierárquica

Informação armazenada nos RLs pode ser o endereço concreto (de uma RL), ou uma referência para um outro RL no "caminho" para o RL atual.



Arquitetura Hierárquica

Os RLs estão distribuídos na rede fixa e formam a Rede de Dados para o GL. Em telefonia celular denominada de: "Common Channel Signalling Network"

Para consultas e atualizações (LookUp/Update) precisa-se conhecer a AL comum entre quaisquer duas ALs.

Em termos da árvore, o nó ancestral comum de menor altura entre quaisquer dois RLs i e j ➔ *Least Common Ancessor LCA(i,j)*

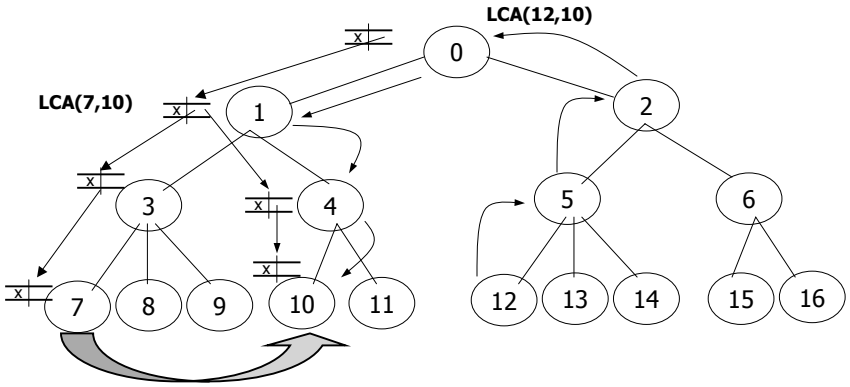
Na arquitetura hierarquica, tambem se redefine o CMRij: engloba-se o CMR de todas as ALs contidas na ALj

$CMRij = \sum_{k \in K} CMRi,k$ onde ALk é sub-area de ALj

Arquitetura Hierárquica: LookUp e Update com Ponteiros

- Quando UMx migra de área i para j, os seguintes RL são atualizados:
  1) no caminho de i para LCA(i,j), as entradas para UMx são removidas
  2) no caminho de LCA(i,j) para j, é incluida uma nova entrada para UMx
- Quando uma nova chamada é gerada na ALi para uma UMy localizada na ALj, então no LookUp:
  1) consulta os RLs iniciando do nó i, e percorre os nós ancestrais de i até encontrar a primeira ocorrência para y (tipicamente em LCA(i,j))
  2) percorre os ponteiros entre os RL de LCA (i,j) até a ALj

Arquitetura Hierárquica: Exemplo com Ponteiros



Migração de UM de AL7 para AL10:
- remoção das entradas em RL7 e RL3
- mudança do ponteiro em RL1
- inclusão das entradas em RL4 e RL10

Chamada para UMx originada em AL12:
- consulta a RL12, RL5, RL2, RL0
- percorrer cadeia de ponteiros até RL10

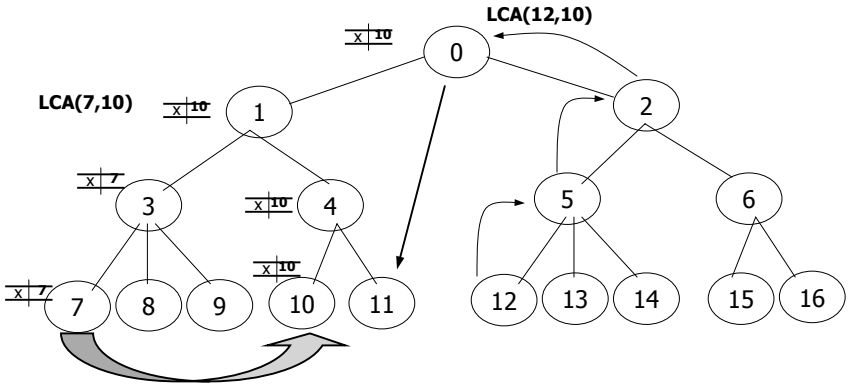
Arquitetura Hierárquica: LookUp e Update com Endereços

- Quando UMx migra de área i para j, os seguintes RL são atualizados:
  1) no caminho de i para LCA(i,j) - exceto LCA(i,j) - as entradas para UMx são removidas
  2) o novo endereço de UMx é atualizado em todos os RLs da raiz até nó folha j

- Quando uma nova chamada é gerada na ALi para uma UMy localizada na ALj, então no LookUp:
  1) consulta os RLs iniciando do nó i, e percorre os nós ancestrais de i até LCA(i,j)

Comparando com uso de ponteiros:
- Atualizalções são mais caras
+ Consultas são mais eficientes
- maior volume de acessos (LookUps) a RLs superiores

Arquitetura Hierárquica: Exemplo com Endereços



Migração de UM de AL7 para AL10:
- remoção das entradas em RL7 e RL3
- atualização dos endereços em RL10, RL4, RL1, RL0

Chamada para UMx originada em AL12:
- consulta a RL12, RL5, RL2, RL0
- encaminhamento direto para AL10

Arquitetura Hierárquica: Considerações Gerais

- Para que esta Arquitetura seja vantajosa, é necessário que as migrações e chamadas sejam regionalizadas (localidade espacial), isto é entre "ALs próximas"
- Neste caso, fazem-se consultas/atualizações somente a RLs próximos, sem a necessidade de acesso a um HLR (eventualmente distante).
- Mas se não houver localidade espacial de migrações/chamdas, LookUps e Updates tipicamente envolvem consultas/atualizações a mais de um RL

➔ *Mais custoso do que Arquitetura Two-Tier*

- Alternativas para diminuir o # de RL acessados:
  - dados de localização armazenados somente em alguns níveis da hierarquia de ALs (armazenamento seletivo)
  - RLs em alturas superiores contém dados menos precisos (particionamento)
  - Uso de caching, replicação e/ou ponteiros de redirecionamneto

Arquitetura Hierárquica: Armazenamento Seletivo

Motivação:

- Evitar que para um Update, precise-se atualizar muitos RLs

Ideia Central:

1) Deixar de colocar entradas (endereços/ponteiros) para UMx em todos os RL da raiz até o nó folha.
   (Exemplo extremos: somente nos nós folha, ou somente no nó raiz)

2) Durante LookUp a partir de i:
   1) os RLs sem informação são ignorados e consulta-se até o RL comum mais próximo que contenha informação sobre UMx (pode ser mais alto do que LCA(i,j))
   2) árvore abaixo (até achar ALj), precisa-se usar consultas por difusão

Arquitetura Hierárquica: Armazenamento Seletivo

Três possíveis estratégias de consulta por difusão:

- consulta plana (flat search): a partir do nó raiz faz-se uma difusão recursiva nos diferentes níveis até encontrar um RL com uma entrada para UMx
- consulta expansionista: a partir do nó RLi consulta-se sucessivamente os seus nós ancestrais, até encontrar uma entrada para UMx. A partir deste nó, faz-se uma difusão recursiva para todos os RL filhos até achar ALj
- consulta híbrida: começa como uma consulta expansionista, mas se transforma em uma consulta plana se UMx não for localizado em certo número de expansões. ➔ Motivação: ou UM está perto, ou então está em qualquer lugar

Arquitetura Hierárquica: Particionamento

Motivação: Explorar localidade de migração para reduzir o custo dos LookUps e ao mesmo tempo limitar a replicação de entradas de um UMx em muitos RLs

Partição :: é um conjunto de ALs nas quais uma UM costuma estar

Seja ALg a área que engloba todas as ALs da partição de UMx.

No RLg (Representante da Partição) mantém-se apenas a informação se no momento UMx está presente na partição (ou seja, um "VLR para várias áreas")

Esta informação é usada na difusão plana para delimitar a "largura da consulta" por difusão plana: ➔ se representante da partição indica que UMx não está na respectiva partição, pode-se "podar" esta parte consulta

Arquitetura Hierárquica: Particionamento

Normalmente, define-se um conjunto de partições por usuário para cada período do dia. Exemplo:

- Partição ida-volta-trabalho
- Partição horário comercial
- Partição noite-manhã

Particionamento:

+ na média, reduz custo do LookUp (consulta por difusão), mas se UM estiver for a da partição não há ganho
- aumenta o custo do update, pois quando UM troca de partição, é necessário atualizar tambem os RLg (representantes de partição)

Agentes de redirecionamento (uma otimização):

- Cada RLg contém também informação sobre AL atual da UM, e
- quando UM vai para outra partição, RLg anterior aponta para RLg nova

Arquitetura Hierárquica: Caching

Na arquitetura hierarquica, pode-se usar caches em diferentes níveis da árvore (level caching).

➔ Quanto mais alto o nó com o cache, maior é a região que se beneficia da informação cacheada.

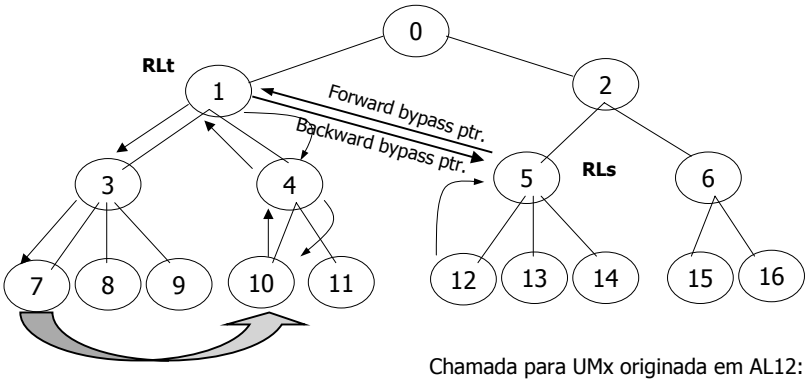
Assume-se:

- cada chamada/mensagem gerada de ALi para ALj, gera também um Ackn. de j para i.
- Seja RLs ancestral de RLi e RLt ancestral de RLj

Idéia Central:

- Durante o primeiro LookUp
  - Cria-se um atalho para frente (forward bypass pointer) de RLs para RLt
  - Cria-se um atalho para trás (backward bypass pointer) de RLt para RLs
- Em LookUps subsequentes (a partir de AL $\subseteq$ ALi)
  - Sobe-se até RLs e desce-se a partir de RLt

Arquitetura Hierárquica: Exemplo Level Caching



Migração de UM de AL7 para AL10:
- inclusão das entradas em RL4, RL10, RL1
- remoção das entradas em RL7 e RL3

Chamada para UMx originada em AL12:
- consulta a RL12, RL5
- forward bypass aponta para RLt
- percorrer cadeia a partir de RLt → RL10

➔ Level caching tenta explorar localidade de migração e chamadas

Arquitetura Hierárquica: Level Caching

Os parâmetros do Level Caching são as alturas dos caches s e t:

+ quanto maior a altura de RLs, maior a área na qual chamadas se beneficiam de LookUp mais eficiente
+ quanto maior a altura de RLs ➔ mais utilizada é a informação cacheada ➔ menor a chance de ficar obsoleta
- quanto maior a altura de RLs e RLt, maior é o número de RL que precisam ser percorridos em LookUp
+ quanto maior a altura de RLt, menor a chance do atalho para frente se tornar obsoleto
- quanto maior a altura de RLt, maior é o número de RLs que precisam ser atualaizados em uma migração

Um extremo: todos os caches estão nos nós folha (caching simples)

Arquitetura Hierárquica: Level Caching

Level Caching também pode ser lazy (só quando ocorre cache miss), or eager (a cada migração).

No tipo eager usa-se atalho para trás para atualizar todos os ponteiros em RLs.

Define-se uma extensão do CMRij:

➔ Regional Call-to-Mobility-Ratio (R-CMRstx) :: razão entre o # médio de chamadas originadas em ALs e o # médio de migrações de UMx dentre as AL $\subseteq$ ALt

Simulações [Jain96] com level caching mostraram que (considerando somente o # de operações sobre as RL, e desprezando o custo de comunicação), para UM com R-CMR > 5 pode-se obter um ganho de eficiência de até 30% (Lookup+Update)

[Jain96] Reducing Traffic Impacts of PCS Using Hierarchical User Location Databases, IEEE ICC, 96

Arquitetura Hierárquica: Replicação

Critério idêntico ao do Two-Tier:

"só vale a pena replicar inf. sobre localização de UMx se o custo de manter a réplica não ultrapassa o ganho de eficiência do Lookup" ➔ valor alto de CMR é desejável

Na estrutura hierárquica, o quanto mais alto estiver um RL, maior será o CMR.

Seguindo este raciocínio, a raiz seria o nó ideal, mas precisa-se também considerar:

- capacidade de processamento e armazenamento de um RL

Para evitar a escolha dos nós superiores da árvore p/ as réplicas, define-se um limite superior (nível da árvore) para todas as réplicas.

Para a escolha dos locais mais apropriados para cada réplica pode-se usar os mesmos métodos ja vistos no Two-Tier

Arquitetura Hierárquica: Ponteiros de Redirecionamento

Na arq. hierárquica, tem-se a possibilidade de escolher nível da árvore onde colocar os ponteiros.

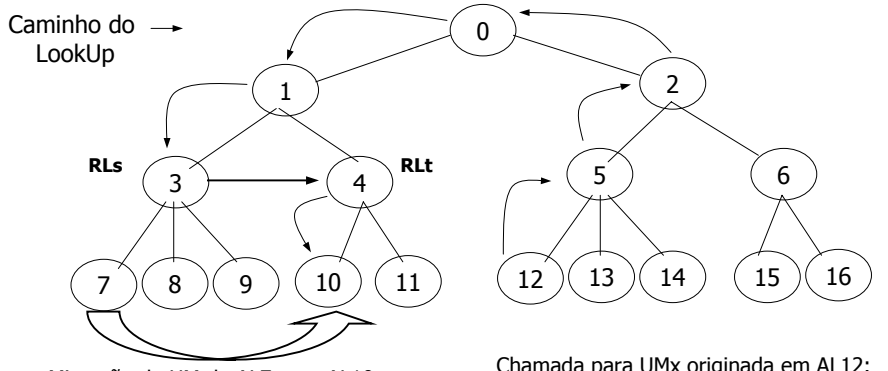
Idéia central: parecida com Level Caching, só que para Updates (migração: ALi → ALj)

- em vez de remover entradas de UMx de i até LCA(i,j) e adicionar entradas de LCA(i,j) até j, faz-se isto somente até nós RLs e RLt (antecessores de RLi e RLj, respectivamente, em determinado nível).
- Se RLs e RLt forem nós folha: redirecionamento simples; senão redirecionamento em nível

Prós e Contras:

+ reduz o custo de Updates
+ o quanto mais alto for o nível de redirecionamento, menos frequentes serão as migrações que necessitem de ponteiro
- encarece os Lookups, que precisam não só localizar RLs, mas seguir os ponteitos até RLt

Arquitetura Hierárquica: Exemplo Ponteiros

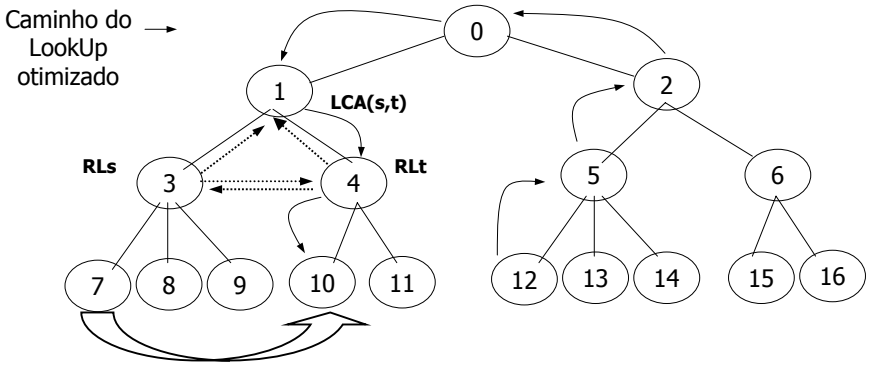


Migração de UM de AL7 para AL10:
- remoção da entrada em RL7
- criação do ponteiro em RL3
- inclusão da entrada em RL4 e RL10

Chamada para UMx originada em AL12:
- consulta a RL12, RL5, RL2, RL0
- percorrer cadeia a partir de RL0 → RLs
- percorrer a cadeia de redirecionamento

➔ Level forwarding visa limitar o custo de Updates

Arquitetura Hierárquica: Compressão de Caminhos



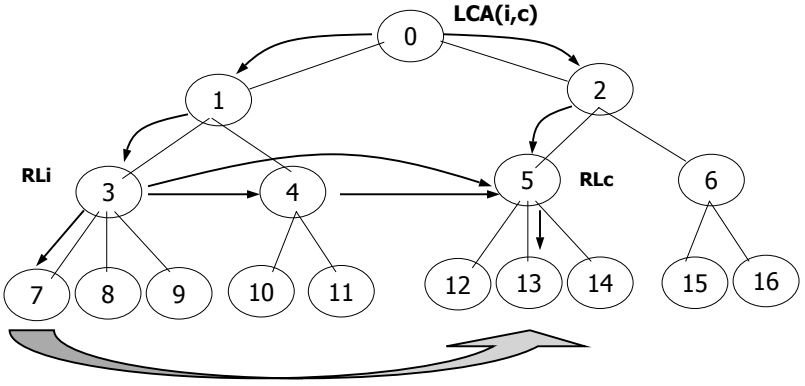
Para evitar que cadeias se tornem longas e aumentem o custo de LookUps:
• quando uma UM tiver sido localizada, um Ackn pode ser usado para atualizar os ancestrais de s e t (até o nó LCA(s,t)) com novo endereço de UMx

Arquitetura Hierárquica: Evitando longas cadeias

Técnicas para evitar longas cadeias de ponteiros (usadas para "simple" e "level forwarding"):

- Poda direta:
  - 1º Elemento da cadeia passa a apontar diretamente para o RL corrente e
  - remove-se todos os ponteiros intermediários
  - Exemplo: RLi $\rightarrow$ RL4 $\rightarrow$ RL5 $\rightarrow$ RL6 $\rightarrow$ RL4 $\Rightarrow$ RLi $\rightarrow$ RL4
- Poda Completa:
  - remove-se todos os ponteiros (inclusive no RLi)
  - Seja i local incial e c local corrente de uma UM:
    - remove-se todas as entradas de UMx nos nós internos de i até LCA(i,c)
    - adiciona-se as entradas para UMx nos nós de LCA(i,c) até RLc
- Obs: Ambos os tipos de poda requerem o conhecimento (ou percorrimento) da cadeia para as atualizações.

Exemplos de Poda



Arquitetura Hierárquica: Evitando longas cadeias

Há também variações com relação ao momento da poda:

1) a cada LookUp bem sucedido, que detecta a presença de uma cadeia muito longa
   - Desvantagem: é necessário mais um percorrimento dos RL

2) a cada migração a UM verifica se já deixou uma cadeia de ponteiros muito longa para trás
   - a UM saberia qual é o RLi e poderia solicitar uma poda direta ou completa
   - o ideal é ter um Update flexível, que dependendo da situação poderia criar ponteiros, fazer poda direta, ou poda completa

Principais aplicações para uso de ponteiros:
   - em sistemas onde CMR é baixo
   - onde o # de migrações tende a ser limitado (p.ex: Ambientes de programação de Objetos Móveis)

Gerenciamento Hierárquico: Resumo

Método	Questões/Variações	Melhor cenário
Caching	Até que nível manter os caches? Quando atualizar caches?	CMR alto Estabilidade de Chamadas
Replicação	Escolha das réplicas	CMR alto Estabilidade de Chamadas
Partições	Escolha das partições de forma que migração intra-partição seja frequente.	Estabilidade de migrações
Ponteiros de Redirecionamento	Escolha do nível. Quando e como fazer a poda?	CMR baixo

Gerenciamento de Localização Conclusão

Desempenho da Estratégia/Arquitetura para gerenciamento de Localização depende dos seguintes parâmetros:

- Padrão de Mobilidade:
  - se apresenta localidade (chamadas/migrações)
  - se existe um perfil para cada UM
  - se existem epicentros
- Padrão de Chamadas/Envios de Msgs:
  - se cada UM tem um conjunto fixo/estável de nós correspondentes
  - se existe um padrão de repetição de envios (de um mesmo nó), em uma sessão de comunicação

Propriedades Relevantes

- Estabilidade de
  - chamadas = maioria de chamadas originadas do mesmo conjunto de ALs
  - migrações = usuários tendem a se movimentar no mesmo conjunto de Als
- Localidade: o custo de um Lookup e de um Update aumentam com a distância (pois possivelmente envolvem mais RLs)
- Frequência relativa entre chamadas e migrações (CMR)

Gerenciamento de Localização Conclusão

Se padrão ...

- apresenta localidade de chamadas ou de migração use
  → Arquitetura Hierarquica
- de chamadas é estável → Cache nos RLs com maior frequencia de chamadas
- de migrações é estável → partição das ALs
- apresenta baixo CMR → usar ponteiros de redirecionamento
- apresenta alto CMR → replicação ou caching

Formas de Estimar/Avaliar o CMR

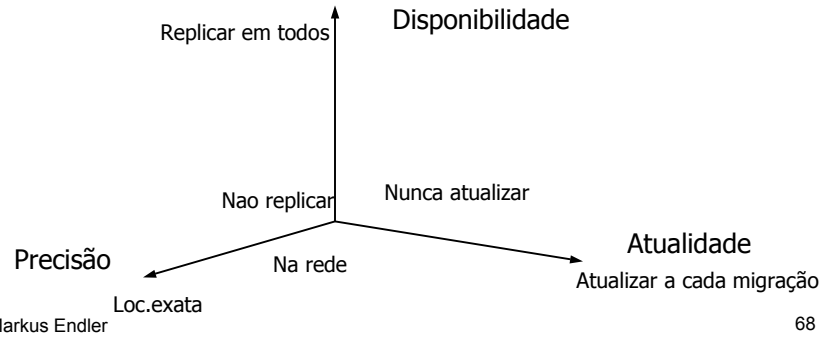
- Algoritmo iterativo de Médias[Jain94]: para cada usuário manter um registro atualizado do #de chamadas recebidas e # de migrações
- Estimativa analítica: modelagem do fluxo de chamadas como Processo de Poisson com taxa de chegada λ e tempo de permanência em uma AL com distribuição com média $1/\mu$. Então LCMR = λ/μ
- Log de chamadas e migrações de categorias de usuários

[Jain94] Jain, Ling, Lo, Mohan. A Caching Strategy to Reduce Network impacts of PCS, IEEE Journal on Selected Areas in Communications, 12(8), 1994

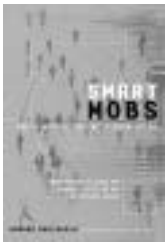
Gerenciamento de Localização: Conclusão

Gerenciamento de Localização permite soluções em um espaço 3-dimensional:

- precisão da informação
- atualidade da informação
- disponibilidade da informação



Leitura Recomendada



Howard Rheingold,
Smart Mobs: The Next Social Revolution

(Transforming Cultures and Communities in the Age of Instant Access)

Algumas Passagens

- "Mobile Internet, when it really arrives, will not be just a way to do old things while moving. It will be a way to do things that couldn't be done before"
- "Groups of people using these [wireless, always-on, global positioning devices] will gain new forms of social power, new ways to organize their interactions and exchanges just in time and just in place."
- "...mobile communications and pervasive computing technologies, together with social contracts that were never possible before, are already beginning to change the way people meet, mate, work, fight, buy, sell, govern, and create."