ACH 2147 — Desenvolvimento de Sistemas de Informação Distribuídos

Aula 15: Coordenação (parte 5)

Prof. Renan Alves

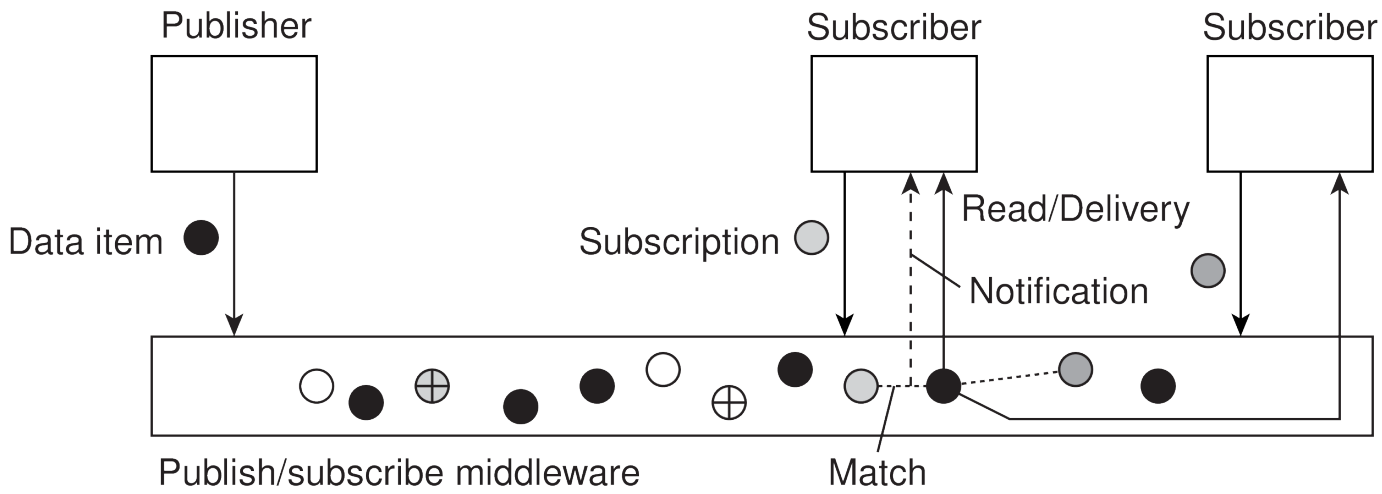
Escola de Artes, Ciências e Humanidades — EACH — USP

26/04/2024

Na aula passada...

- Eleição por prova e trabalho e prova de participação
- Usos do gossip
  - Agregação
  - Amostragem de pares

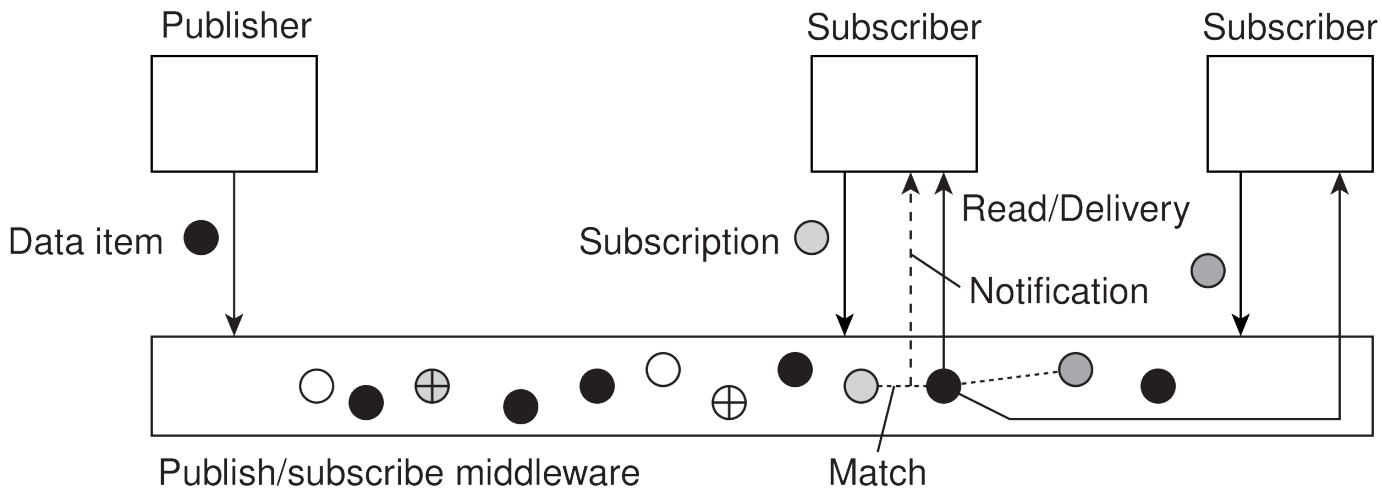
Correspondência (matching) de eventos distribuída



Princípio

- Um processo especifica em quais eventos está interessado (inscrição *S*)
- Quando um processo publica uma notificação *N*, precisamos verificar se *S* corresponde a *N*.

Correspondência (matching) de eventos distribuída



Princípio

- Um processo especifica em quais eventos está interessado (inscrição S)
- Quando um processo publica uma notificação N, precisamos verificar se S corresponde a N.

Parte difícil

Implementar a função de correspondência de maneira escalável.

Abordagem geral

O que é necessário

- *sub2node*(S): mapear uma inscrição S para um subconjunto não vazio **S** de servidores
- *not2node*(N): mapear uma notificação N para um subconjunto não vazio **N** de servidores

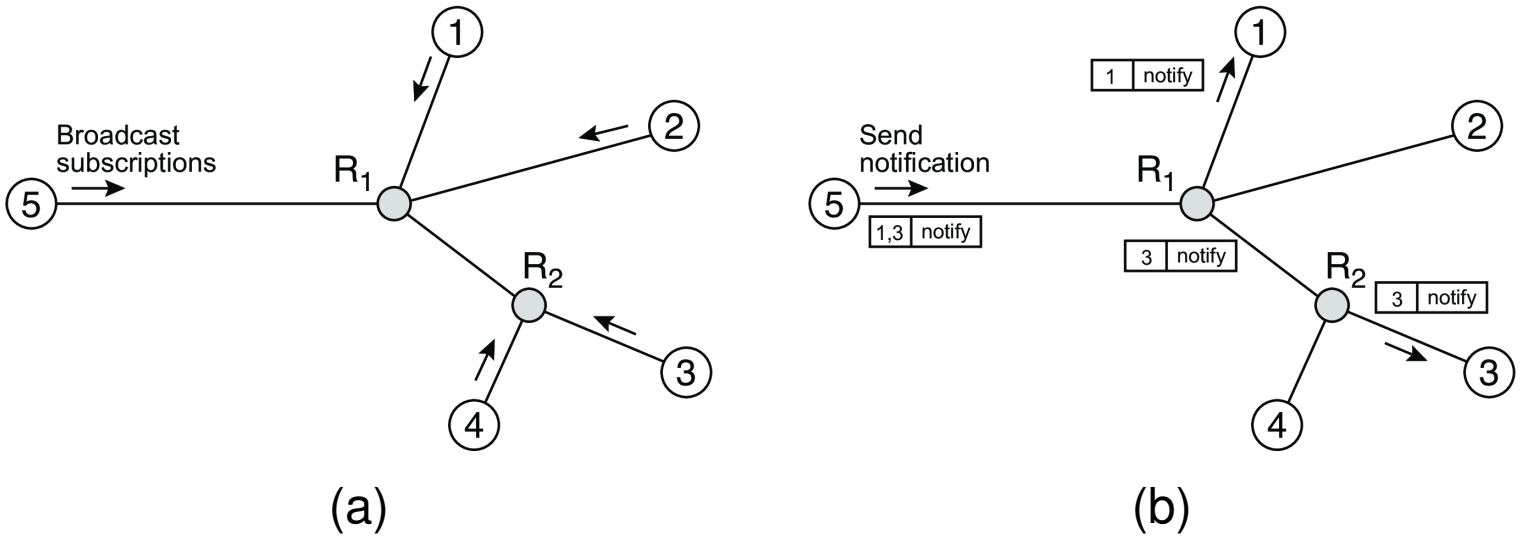
Certifique-se de que $\mathbf{S} \cap \mathbf{N} \neq \emptyset$.

Abordagem geral

Observações

- A solução centralizada é simples: $\mathbf{S} = \mathbf{N} = \{s\}$, ou seja, um único servidor.
- Publish-subscribe baseada em tópicos também é simples: cada S e N é marcado com um único tópico; cada tópico é tratado por um único servidor (um nó de encontro ou rendezevous node). Vários tópicos podem ser tratados pelo mesmo servidor.
- Publish-subscribe baseada em conteúdo é difícil: uma inscrição assume a forma de um par (*atributo*, *valor*), com valores de exemplo:
  - intervalo: " $1 \leq x < 10$ "
  - pertencimento: " $x \in \{red, blue\}$ "
  - expressões de prefixo e sufixo: `url.startswith("https")`

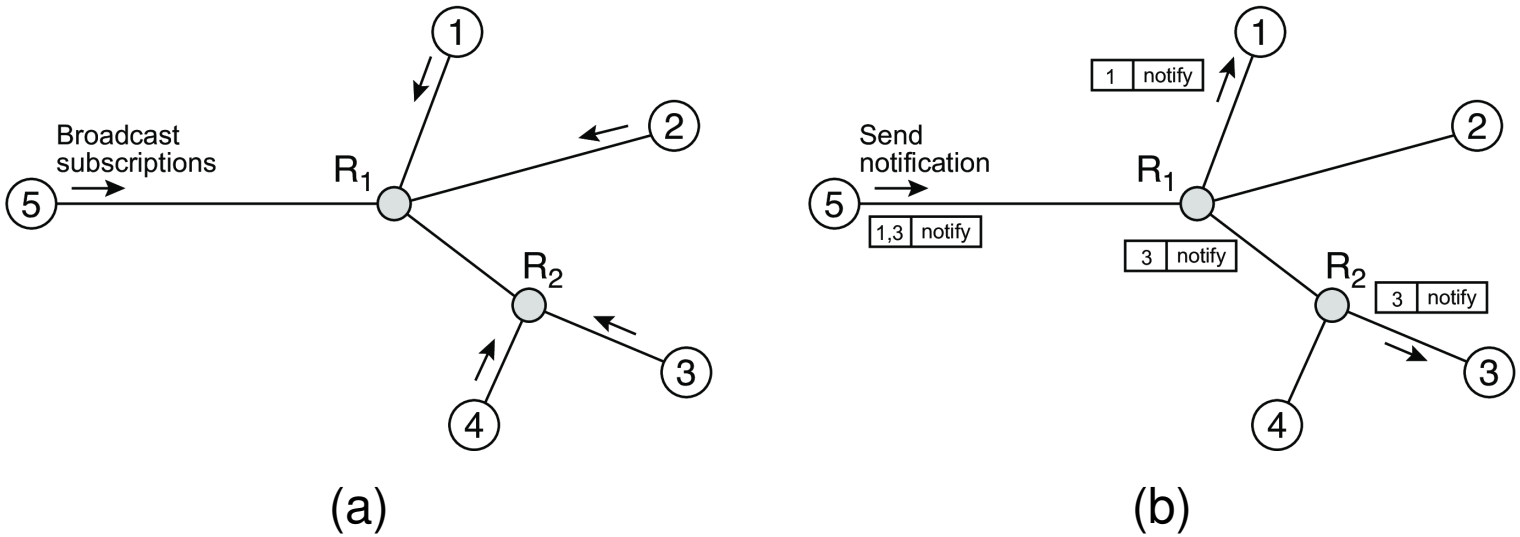
Roteamento Seletivo



(a) primeiro divulga as inscrições

(b) encaminha notificações apenas para nós de encontro relevantes

Roteamento Seletivo



(a) primeiro divulga as inscrições

(b) encaminha notificações apenas para nós de encontro relevantes

Exemplo: tabela de roteamento (parcialmente preenchida) do nó R_2

Interface	**Filtro**
Para o nó 3	$a \in [0,3]$
Para o nó 4	$a \in [2,5]$
Para o roteador R_1	(não especificado)

Gossiping: Sub-2-Sub

Noções básicas

- Objetivo: Ter escalabilidade, garantir que os assinantes com os mesmos interesses formem um único grupo
- Modelo: Existem N atributos $a_1, \ldots, a_N$. Um valor de atributo é sempre (mapeável para) um número de ponto flutuante.
- Inscrição: Assume formas como $S = \langle a_1 \rightarrow 3.0, a_4 \rightarrow [0.0,\ 0.5) \rangle$: a_1 deve ser 3.0; a_4 deve estar entre 0.0 e 0.5; outros valores não importam.

Observações

- Uma inscrição S_i especifica um subconjunto $\mathbf{S_i}$ em um espaço N-dimensional.
- Estamos interessados apenas em notificações em $\overline{\mathbf{S}} = \cup \mathbf{S_i}$.

Gossiping: Sub-2-Sub

Noções básicas

- Objetivo: Ter escalabilidade, garantir que os assinantes com os mesmos interesses formem um único grupo
- Modelo: Existem N atributos $a_1, \ldots, a_N$. Um valor de atributo é sempre (mapeável para) um número de ponto flutuante.
- Inscrição: Assume formas como $S = \langle a_1 \to 3.0, a_4 \to [0.0,\ 0.5) \rangle$: a_1 deve ser 3.0; a_4 deve estar entre 0.0 e 0.5; outros valores não importam.

Observações

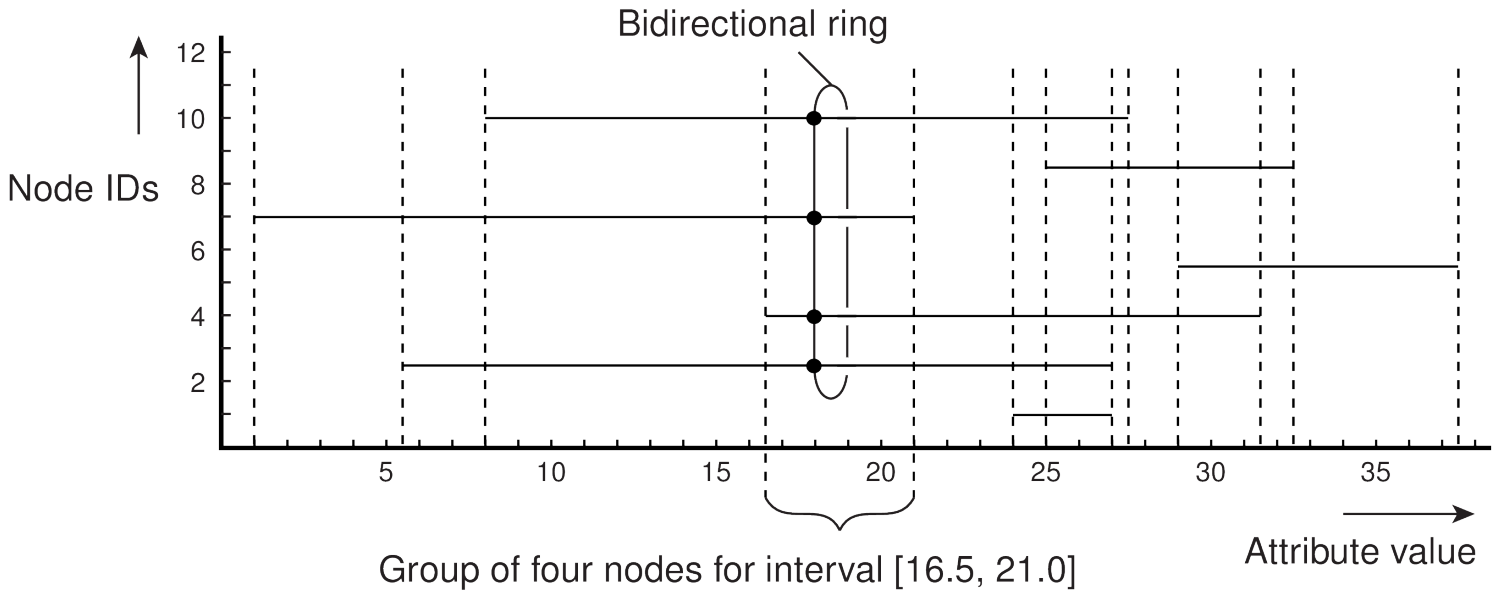
- Uma inscrição S_i especifica um subconjunto $\mathbf{S_i}$ em um espaço N-dimensional.
- Estamos interessados apenas em notificações em $\overline{\mathbf{S}} = \cup \mathbf{S_i}$.

Objetivo

Particionar $\overline{\mathbf{S}}$ em M subespaços disjuntos $\overline{\mathbf{S}}_\mathbf{1}, \ldots, \overline{\mathbf{S}}_\mathbf{M}$ tal que

- Particionamento: $\forall k \neq m : \overline{\mathbf{S}}_\mathbf{k} \cap \overline{\mathbf{S}}_\mathbf{m} = \emptyset$ e $\cup_m \overline{\mathbf{S}}_\mathbf{m} = \overline{\mathbf{S}}$
- Cobertura de inscrição: $(\overline{\mathbf{S}}_\mathbf{m} \cap \mathbf{S_i} \neq \emptyset) \Rightarrow (\overline{\mathbf{S}}_\mathbf{m} \subseteq \mathbf{S_i})$

Gossiping: Sub-2-Sub



Considere um único atributo

- Os nós regularmente trocam suas inscrições através de gossip
- Uma interseção entre dois nós leva a uma referência mútua
- Se $\mathbf{S_{ijk}} = \mathbf{S_i} \cap \mathbf{S_j} \cap \mathbf{S_k} \neq \emptyset$ e $\mathbf{S_{ij}} - \mathbf{S_{ijk}} \neq \emptyset$, então:
  - nós i, j, k são agrupados em uma rede overlay única (para $\mathbf{S_{ijk}}$)
  - nós i, j são agrupados em uma rede overlay única (para $\mathbf{S_{ij}} - \mathbf{S_{ijk}}$)

Publish-subscribe seguro

Estamos enfrentando dilemas difíceis de resolver

- Desacoplamento referencial: as mensagens devem poder fluir de um publisher para subscribers enquanto garantem o anonimato mútuo $\Rightarrow$ não podemos configurar um canal seguro.
- Não saber de onde vêm as mensagens impõe problemas de integridade e confidencialidade.
- Supor um broker de confiança pode ser praticamente impossível, principalmente ao lidar com informações sensíveis.

Publish-subscribe seguro

Estamos enfrentando dilemas difíceis de resolver

- Desacoplamento referencial: as mensagens devem poder fluir de um publisher para subscribers enquanto garantem o anonimato mútuo $\Rightarrow$ não podemos configurar um canal seguro.
- Não saber de onde vêm as mensagens impõe problemas de integridade e confidencialidade.
- Supor um broker de confiança pode ser praticamente impossível, principalmente ao lidar com informações sensíveis.

Solução

- Permitir a pesquisa (e correspondência) em dados criptografados, sem a necessidade de decifração.
- Public Key Encryption with Keyword Search (PEKS): mensagens criptografadas acompanhadas de uma coleção de palavras-chave (também criptografadas) e pesquisar por correspondências nas palavras-chave.

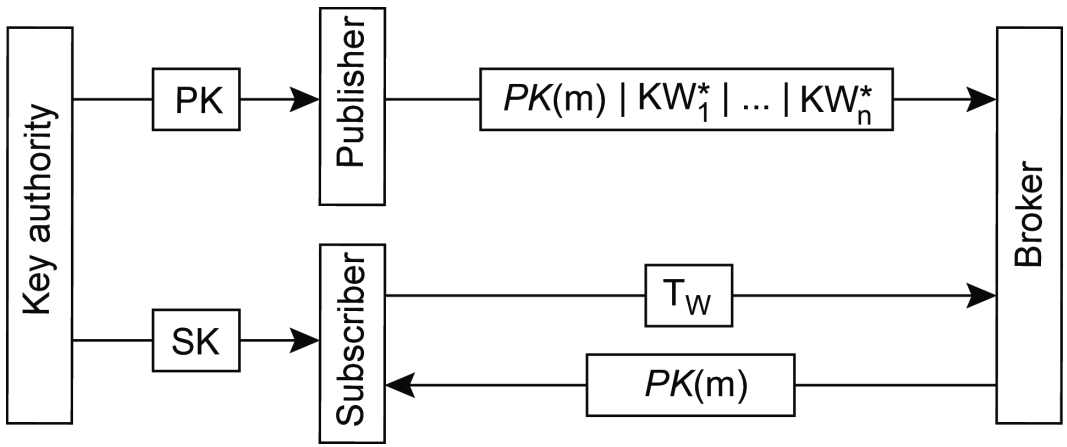
Public-Key Encryption with Keyword Search (PEKS)

Noções básicas

- Usar uma chave pública PK, uma mensagem m e suas n palavras-chave $KW_1, \ldots, KW_n$ armazenadas em um servidor como a mensagem m^*:

$$m^* = [PK(m)|PEKS(PK, KW_1)|PEKS(PK, KW_2)|\cdots|PEKS(PK, KW_n)]$$

- Um assinante recebe a chave secreta acompanhante.
- Para cada palavra-chave KW_i, é gerado um gatilho T_{KW_i}: $T_W(m^*)$ retornará *true* se $W \in \{KW_1, \ldots, KW_n\}$.



$$KW_i^* = PEKS(PK, KW_i)$$

Posicionamento de nós

Questão

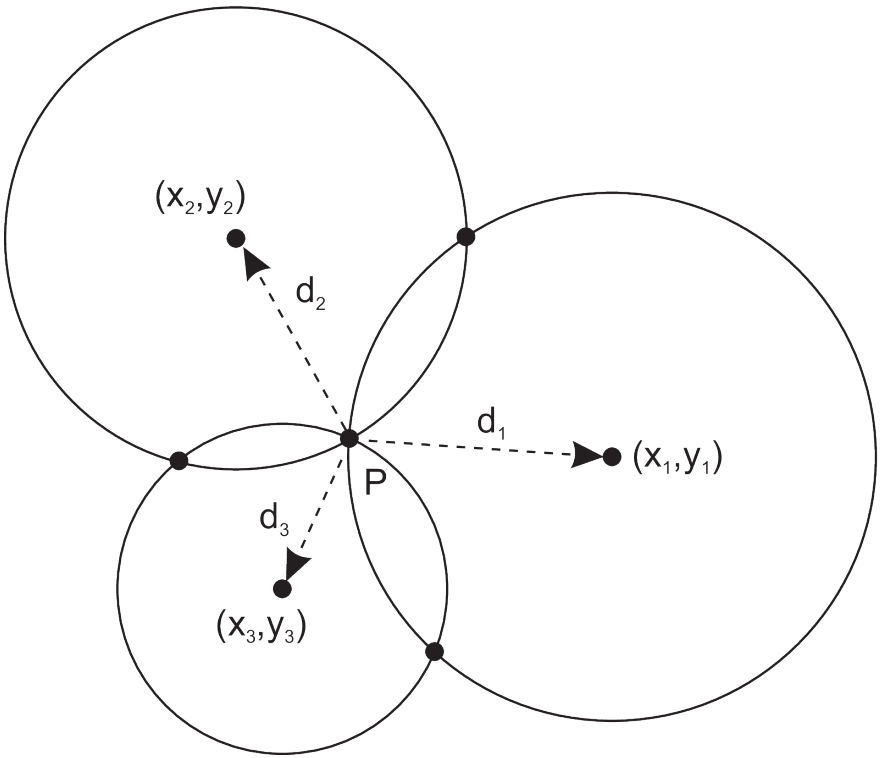
Em sistemas distribuídos em grande escala nos quais os nós estão dispersos em uma WAN, muitas vezes precisamos levar em consideração alguma noção de proximidade ou distância $\Rightarrow$ isso começa com a determinação da (relativa) localização de um nó.

Computando a posição

Observação

Um nó P precisa de $d+1$ marcos de referência para calcular sua própria posição em um espaço d-dimensional. Considere o caso bidimensional.

Computando uma posição em 2D



Solução

P precisa resolver três equações em duas incógnitas (x_P, y_P):

$$d_i = \sqrt{(x_i - x_P)^2 + (y_i - y_P)^2}$$

Global Positioning System (GPS)

Supondo que os relógios dos satélites sejam acurados e sincronizados

- Leva um tempo até que um sinal alcance o receptor
- O relógio do receptor está definitivamente fora de sincronia com o satélite

Noções básicas

- Δ_r: desvio desconhecido do relógio do receptor.
- x_r, y_r, z_r: coordenadas desconhecidas do receptor.
- T_i: timestamp em uma mensagem do satélite i
- $\Delta_i = (T_{now} - T_i) + \Delta_r$: atraso medido da mensagem enviada pelo satélite i.
- Distância medida ao satélite i: $c \times \Delta_i$ (c é a velocidade da luz)
- Distância real: $d_i = c\Delta_i - c\Delta_r = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$

Observação

4 satélites $\Rightarrow$ 4 equações em 4 incógnitas (com Δ_r como uma delas)

Serviços de localização baseados em WiFi

Ideia básica

- Suponha que temos um banco de dados de pontos de acesso (APs) conhecidos com coordenadas
- Suponha que podemos estimar a distância até um AP
- Então: com 3 pontos de acesso, podemos calcular uma posição.

Wardriving: localizando pontos de acesso

- Usar um dispositivo com WiFi junto com um receptor GPS e mova-se por uma área enquanto registra os pontos de acesso observados.
- Calcular o centróide: suponha que um ponto de acesso AP foi detectado em N locais diferentes $\{\vec{x_1}, \vec{x_2}, \ldots, \vec{x_N}\}$, com localização GPS conhecida.
- Calcular a localização de AP como $\vec{x}_{AP} = \frac{\sum_{i=1}^{N} \vec{x_i}}{N}$.

Problemas

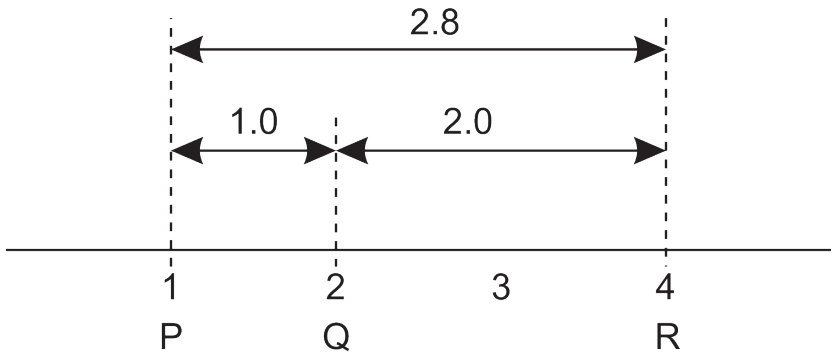
- Precisão limitada de cada ponto de detecção GPS $\vec{x_i}$
- Um ponto de acesso tem um alcance de transmissão não uniforme
- O número de pontos de detecção amostrados N pode ser muito baixo.

Posicionamento lógico de nós

Problemas

- As latências medidas para os marcos de referência flutuam
- As distâncias calculadas não serão consistentes

Distâncias inconsistentes no espaço 1D



Solução: minimizar erros

- Use N nós marcos de referência especiais $L_1, \ldots, L_N$.
- Os marcos de referência medem suas latências em pares $\tilde{d}(L_i, L_j)$
- Um nó central calcula as coordenadas para cada marco de referência, minimizando:

$$\sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(\frac{\tilde{d}(L_i, L_j) - \hat{d}(L_i, L_j)}{\tilde{d}(L_i, L_j)} \right)^2$$

onde $\hat{d}(L_i, L_j)$ é a distância após os nós L_i e L_j terem sido posicionados.

Computando a posição

Escolhendo a dimensão m

O parâmetro oculto é a dimensão m com $N > m$. Um nó P mede sua distância para cada um dos N marcos de referência e calcula suas coordenadas minimizando

$$\sum_{i=1}^{N} \left(\frac{\tilde{d}(L_i, P) - \hat{d}(L_i, P)}{\tilde{d}(L_i, P)} \right)^2$$

Observação

Na prática, m pode ser tão pequeno quanto 6 ou 7 para obter estimativas de latência dentro de um fator 2 do valor real.

Vivaldi

Princípio: Rede de molas exercendo forças

Considere uma coleção de N nós $P_1, \ldots, P_N$, cada P_i tendo coordenadas $\vec{x}_i$. Dois nós exercem uma força mútua:

$$\vec{F}_{ij} = \left(\tilde{d}(P_i, P_j) - \hat{d}(P_i, P_j)\right) \times u(\vec{x}_i - \vec{x}_j)$$

onde $u(\vec{x}_i - \vec{x}_j)$ é o vetor unitário na direção de $\vec{x}_i - \vec{x}_j$.

O nó P_i executa repetidamente as seguintes etapas

1. Medir a latência $\tilde{d}_{ij}$ até o nó P_j, e também receber as coordenadas de P_j $\vec{x}_j$.
2. Calcular o erro $e = \tilde{d}(P_i, P_j) - \hat{d}(P_i, P_j)$
3. Calcular a direção $\vec{u} = u(\vec{x}_i - \vec{x}_j)$.
4. Calcular o vetor de força $F_{ij} = e \cdot \vec{u}$
5. Ajustar a própria posição movendo-se ao longo do vetor de força: $\vec{x}_i \leftarrow \vec{x}_i + \delta \cdot \vec{u}$.