Sistemas Distribuídos

Padrões de interação

abril de 2014

Padrões de programas distribuídos

- cliente/servidor como padrão principal
  - e o que mais temos...?
- padrões envolvendo processos *pares*: código idêntico

- G. Andrews. Paradigms for process interaction in distributed programs. ACM Computing Surveys, 23(1), march 1991, 49-90.

Padrões p2p

- simetria do código facilita entendimento do programa
- algoritmos em execução determinam padrões de comunicação

- relação entre discussão no artigo Andrews e aplicações mais recentemente rotuladas como p2p

Disseminação de informações

Motivação

- Disseminação de informações de um nó raiz para toda a rede
- Disseminação de informações de todos os nós para todos os nós da rede (ex., problema de descoberta de topologia da rede)
- Coleta de dados de todos os nós da rede

Probe/echo e heartbeat

Padrões de interação adequados para disseminação de informações quando os nós trocam mensagens apenas com seus vizinhos imediatos.

Algoritmos e Padrões

- algoritmos determinam/induzem/sugerem padrões
- padrões centralizados e distribuídos

Algoritmos e padrão da aplicação

possivel combinação de vários padrões

algoritmo de disseminação pode ser apenas pequena parte do que é computado na aplicação

Disseminação e roteamento

- redes físicas
- redes de overlay

Algoritmo *probe/echo*

- Um *probe* é uma mensagem enviada por um nó para o seu sucessor
- Um *echo* é uma resposta subsequente ao *probe* recebido

- Como os processos executam concorrentemente, *probes* são enviados em paralelo para todos os sucessores no grafo de conexões
- O padrão *probe/echo* implementa uma versão concorrente da busca em profundidade (para visitar todos os nós da rede/grafo)

problema exemplo: descoberta de topologia

- Um nó computa a topologia (e depois possivelmente encaminha a informação para toda a rede

A topologia é obtida em duas fases:

1. Cada nó envia um *probe* para seus vizinhos
2. Cada nó envia um *echo* com a informação da topologia obtida de volta para o nó do qual recebeu o primeiro *probe*

Quando o nó que iniciou a descoberta recebe todos os *echoes*, ele pode disseminar a topologia completa para os nós da rede

probe/echo em grafos acíclicos

probe/echo em grafos com ciclos

Algoritmo *probe/echo* para descoberta de topologia: grafos com ciclos

- Depois de receber um *probe*, o nó o repassa para todos os seus outros vizinhos e então espera o *echo* de cada um deles
- Dois nós vizinhos podem enviar *probes* ao mesmo tempo, um para o outro:
  - *probes* redundantes podem ser "ecoados" imediatamente
  - respostas ecoadas imediatamente devem conter "elementos neutros"
- Quando o nó receber um *echo* em resposta a cada *probe* que enviou, ele responde ao *probe* inicial "ecoando" a sua visão da topologia

Interface para componente de descoberta de topologia (pe)

```
Module:
    Name: Topologia-pe
Events:
  Request: <DescobreTopologia | src, vizinhos[N]>
  Indication: <TopologiaAtual | toplocal[N][N]>
```

Algoritmo P/E (parte 1)

```
Implements: Topologia-pe
Uses: PerfectP2PLink
event <Init> do
  global toplocal[1:n][1:n] := false;
  global num_vizinhos := 0; pai := NULL;
  global num_echoes := 0; inicializador := NULL;
```

```
Module:
   Name: PerfectP2PLink
Events:
  Request: <Send | dst, msg>
  Indication: <Deliver | src, msg>
```

Algoritmo P/E (parte 2)

```
event <DescobreTopologia | src, vizinhos[N]> do
  toplocal[p][1:n] = vizinhos[1:n];
  num_vizinhos := computa vizinhos;
  inicializador := src;

  forall (n in top[p]) do
    trigger <Send | n, [PROBE, NULL]>
```

Algoritmo P/E (parte 3)

```
event <Deliver | q, [tipo, newtop]> do
  if (tipo = PROBE and pai = NULL) then
    pai := q;
    forall (n in top[p] and n != q) do
      trigger <Send | n, [PROBE, NULL]>
    num_echoes := num_vizinhos - 1;

  else if (tipo = PROBE and pai != NULL) then
      trigger <Send | q, [ECHO, NULL]>

  else if (tipo = ECHO) then
     toplocal := toplocal or newtop;
     num_echoes := num_echoes - 1;
     call Topo();
  end
```

Algoritmo P/E (parte 4)

```
proc Topo() do
  if (num_echoes = 0) then
     if (p = inicializador) then
       trigger <TopologiaAtual | localtop>;
     else
       trigger <Send | pai, [ECHO, localtop]>
end
```

Observações sobre o algoritmo

- 2 mensagens são enviadas em cada enlace da AG construída pelos primeiros probes (uma para o *probe* e outra para o *echo*)
- nos demais enlaces trafegam 4 mensagens (um *probe* e um *echo* em cada direção)
- para disseminar a topologia final, são necessárias n mensagens (usando um algoritmo de broadcast com AG)

- relação com forma de programação: como programar com biblioteca C/S?