Sistemas Distribuídos
Difusão Atômica

Prof. Elias P. Duarte Jr.
Universidade Federal do Paraná (UFPR)
Departamento de Informática
www.inf.ufpr.br/elias/sisdis

Sumário

- Definição de Difusão Atômica
- Revisando a definição de *Replicação Máquina de Estados – State Machine Replication*
- Revisando 1 instância do consenso
- Propriedades da Difusão Atômica
- Um Algoritmo de Difusão Atômica
- Fim desta edição de Sistemas Distribuídos!

Difusão Atômica

- *Atomic Broadcast* ou ainda *Total-Order Broadcast*
- Primitiva: mensagem é transmitida com *ABcast(msg)*
- Definição informal: a difusão atômica garante que as mensagens são entregues por todos os processos do sistema exatamente na mesma ordem

Difusão Atômica

- *Atomic Broadcast* ou ainda *Total-Order Broadcast*
- Primitiva: mensagem é transmitida com *ABcast(msg)*
- Definição informal: a difusão atômica garante que a mensagem é entregue por todos os processos do sistema exatamente na mesma ordem
- Qual ordem?

Difusão Atômica

- *Atomic Broadcast* ou ainda *Total-Order Broadcast*
- Primitiva: mensagem é transmitida com *ABcast(msg)*
- Definição informal: a difusão atômica garante que a mensagem é entregue por todos os processos do sistema exatamente na mesma ordem
- Qual ordem?

Difusão Atômica

- *Atomic Broadcast* ou ainda *Total-Order Broadcast*
- Primitiva: mensagem é transmitida com *ABcast(msg)*
- Definição informal: a difusão atômica garante que a mensagem é entregue por todos os processos do sistema exatamente na mesma ordem
- Qual ordem?
- Não importa, qualquer ordem desde que todos os processos entreguem todas as mensagens naquela ordem
  - evidentemente uma ordem é definida em cada caso de uso!

Um Contexto de Uso

- Toda vez que temos um sistema distribuído em que mensagens precisam ser entregues pelos processos na mesma ordem
- Várias aplicações, uma das mais relevantes é a
- Replicação Máquina de Estados (RME)
  - *State Machine Replication* (SMR)

Máquina de Estados

- Uma representação (modelo) de um processo
- Consiste de:
  - Um conjunto de estados que o processo pode assumir
  - Um conjunto de transições entre estados
  - Uma transição do “estado atual” para o próximo estado
  - Uma transição ocorre como consequência de um “evento”
    - um evento pode ser, por exemplo, a execução de uma operação

Máquina de Estado Determinística

- Uma máquina de estado determinística só permite a ocorrência de 1 evento de cada vez
- A transição de estados causada pelo evento também é única, produz uma saída bem definida
  - logicamente a saída produzida depende do estado atual e da transição que ocorreu

Em Sistemas Assíncronos com Falhas Crash

- A máquina de estados no modelo temporal assíncrono pode demorar um tempo arbitrário para realizar a transição entre estados
  - e para produzir a saída correspondente, se houver
- Considerando que processos (máquinas de estado) podem sofrer falhas crash
- O modelo assíncrono sofre daquele problema crucial: é impossível distinguir um processo falho de um processo lento

Replicação Máquina de Estados

- Aplicação típica: replicação de servidores
- Por exemplo: pense em um grupo de servidores Web replicados
- Na Replicação Máquina de Estados – *State Machine Replication:*

  → *cada réplica é uma máquina de estados*

  → *todas as réplicas iniciam no mesmo estado*

  → *executam a mesma sequência de operações*

  → *desta forma: todas as transições são idênticas!*

Réplicas Consistentes

- A Replicação Máquina de Estados garante que todas as réplicas estarão sempre no mesmo estado
- Desta forma: se uma réplica falha, sem problemas! O serviço continua disponível!
- Até N-1 réplicas podem falhar e o serviço continua disponível
- Além disso: aumenta o desempenho do serviço: ao invés de ter um servidor respondendo clientes, tem N servidores

Múltiplas Requisições Simultâneas – Vindas de Múltiplas Origens

- Múltiplos clientes geram requisições de operações que devem ser executadas pelo servidor
- As requisições devem carregar: o id do cliente (lógico), um *timestamp* e a operação a ser executada
- Considere que o servidor é replicado usando RME
- Note que isso fica totalmente transparente para os clientes!
- Múltiplos clientes podem enviar múltiplas requisições

Múltiplos Clientes & Múltiplas Requisições

- Múltiplos clientes podem enviar múltiplas requisições
- Não podemos deixar que sejam executadas pelas réplicas do servidor em ordens diferentes!
- Deve haver um **consenso** sobre exatamente qual requisição deve ser executada de cada vez
- É justamente isso que o Paxos garante: todas as réplicas executam exatamente a mesma operação

Instâncias do Consenso

- Uma instância do consenso determina qual é a operação a ser executada
- Define portanto a execução de 1 única operação
- Para definir a operação seguinte, outra instância do consenso é executada
- Levando em conta todas as instâncias executadas:

  → Como visto nesta aula, a Difusão Atômica vai garantir que a sequência de operações executadas por todos os processos é a mesma

Consenso: Primitivas

- Para executar o consenso os processos usam 2 primitivas:

  → propose(valor): um processo propõe um valor

  → decide(valor): notificação da decisão

Consenso Propriedades

- **Integridade**: **não-duplicação** & **não-espúrias**
- **Validade**: se um processo decide por um valor v, este valor foi proposto por um processo
- **Terminação**: se um processo propõe, todos os processos corretos decidem após um tempo
- **Acordo**: todos os processos decidem pelo mesmo valor → *no two processes decide differently*

Difusão Atômica: Propriedades

- **Integridade**: **no-creation**, **no-duplication**
- **Validade:** se um processo correto executa ABcast(msg), os processos corretos entregam msg após um tempo
- **Acordo:** se um processo correto entrega uma mensagem msg, então após um tempo todo processo correto entrega msg
- **Ordem Total:** sejam msg1 e msg2 duas mensagens entregues por dois processos corretos quaisquer x e z. Se x entrega msg1 antes de entregar msg2, então z entrega msg1 antes de entregar msg2

Descrevendo o Algoritmo

- O algoritmo de difusão atômica é organizado em rodadas
  - Rodada 1, Rodada 2, ...
  - Em cada rodada uma instância do consenso é executada
  - O consenso decide qual mensagem (ou sequência de mensagens) é/são entregues em uma rodada
  - Mensagens são entregues em uma ordem determinística, pré-definida
    - por exemplo: de acordo com o id da mensagem (orig, tmstp)

Descrevendo o Algoritmo

- Uma variável “wait” lógica é usada para garantir que o consenso não é chamado mais de uma vez seguida pelo mesmo processo
- “wait” é FALSE até o processo fazer chamar o consenso, isto é, fazer uma proposta
- No algoritmo, o consenso decide por uma mensagem/sequência de msgs
- Observe que enquanto wait é FALSE podem ir chegando mensagens

O Algoritmo da Difusão Atômica

Algoritmo Difusão Atômica

Init: Delivered ← Φ; // mensagens entregues
Received ← Φ; // mensagens recebidas ainda não entregues
rodada ← 1;
wait ← FALSE;

Upon AtomicBroadcast(msg): ReliableBroadcast(msg);

Upon DeliverReliableBroadcast(msg): if msg ∉ Delivered
then Received ← Received U {msg};

Upon (Received ≠ Φ) AND (wait == FALSE):
wait ← TRUE;
propose(rodada, Recebidas); // consenso!

O Algoritmo da Difusão Atômica

Algoritmo Difusão Atômica - cont.

// nesta última parte o consenso decidiu

Upon Decided(rodada, DecidedMsgs):

Delivered ← Delivered U DecidedMsgs;

Received ← Received - DecidedMsgs;

Deliver(DecidedMsgs ordenadas);

rodada ← rodada + 1;

wait ← FALSE;

Por que funciona?

- No creation: o sistema tem meios de evitar que propostas espúrias sejam realizadas
- No duplication: uma mensagem fica no no conjunto “Recebidas” até ser entregue uma vez, depois é movida para “Entregues” e não é entregue novamente
- Validade: garantida pelo consenso, que é usado para decidir cada mensagem a ser entregue
- Acordo: garantido pela validade do reliable broadcast e do consenso
- Ordem total: se um processo entrega m1 antes de m2,

  todos os processos corretos entregam m1 antes de m2

Referências:

A melhor referência para os diversos tipos de broadcast ainda é o Capítulo 5 do Sape Mullender:

V. Hadzilacos, S. Toueg, “Fault-Tolerant Broadcasts and Related Problems,” **in** Sape Mullender, *Distributed Systems*, Addison-Wesley, 1993.

O ABcast é definido nosso livro texto.

Conclusão

- Nesta aula primeiro definimos o que é Difusão Atômica
- Depois fizemos uma revisão da Replicação Máquina de Estados (para apresentar a Difusão Atômica neste contexto)
- Vimos as propriedades da Difusão Atômica
- Vimos algoritmo de Difusão Atômica

FIM!
Disciplina Sistemas Distribuídos
ERE4 2021

Meu muito obrigado aos alunos!

www.inf.ufpr.br/elias/sisdis