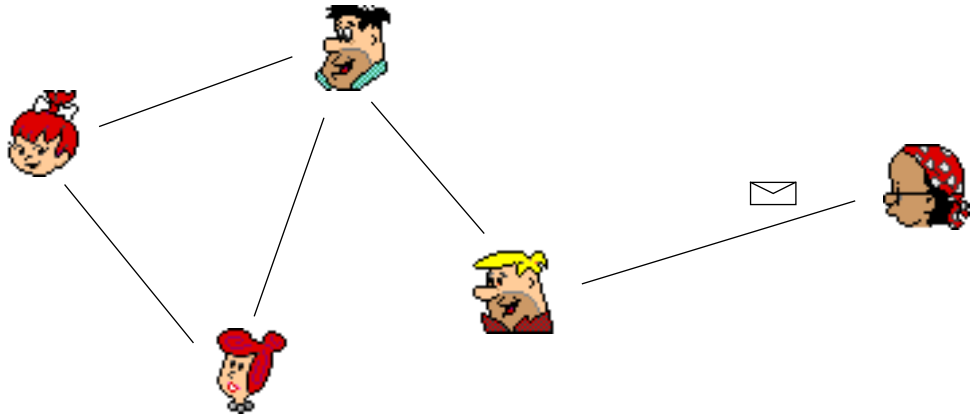
Disseminação Confiável



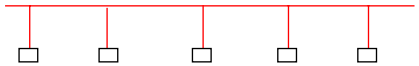
Disseminação Confiável

- mensagem é recebida por todos os participantes ou nenhum participante recebe a mensagem (disseminação atômica)
- todas as mensagens recebidas são recebidas na mesma ordem por todos os participantes (disseminação confiável)

Disseminação Confiável

Chang & Maxemchuk, “Reliable broadcast protocols”, Transaction on Computer Ssytems, ago84

- Solução “prática”
- Garante que todos os nós recebem “todas” as mensagens na mesma ordem
- Faz uso de uma rede de broadcast como sistema de comunicação
- Tem custo de menos de uma mensagem adicional por mensagem entregue



Algoritmo de Chang & Maxemchuk

Falhas

- processadores: falha e pára
- sistema de comunicação: omissão

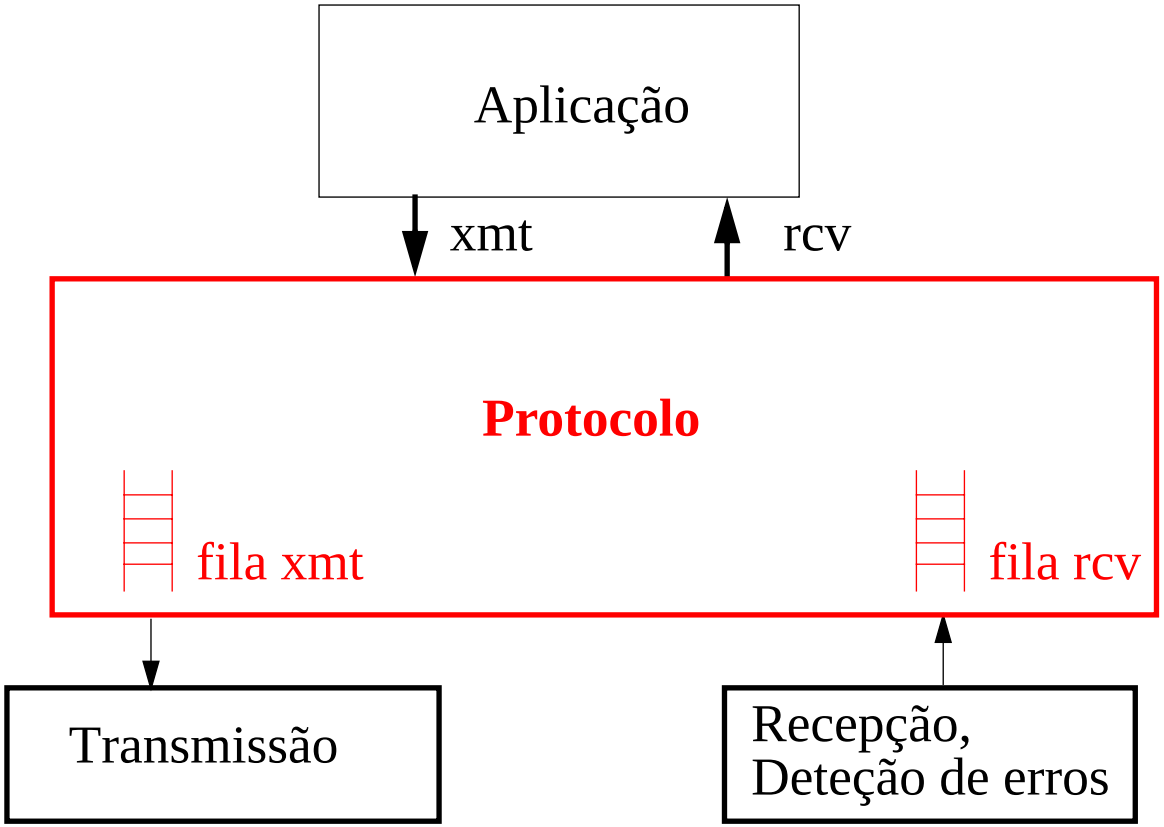
Falha ocorre quando nó não consegue comunicação após R tentativas

engano na deteção < R < tempo de deteçao "razoável"

Implementador deve achar um bom compromisso

Algoritmo de Chang & Maxemchuk

Descrição Geral do Sistema




Protocolo
fila xmt
fila rcv

Algoritmo de Chang & Maxemchuk

Algumas técnicas de “reconhecimento”(notícia de recebimento) de mensagens

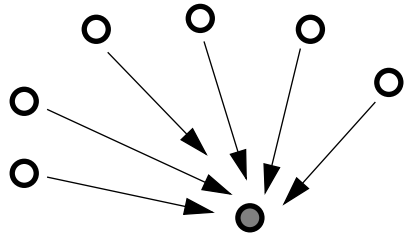
Remetente numera suas mensagens sequencialmente

- Reconhecimento positivo:
  - Remetente retransmite mensagem até receber reconhecimento (ACK) correspondente de todos os destinatários; remetente espera receber ACK antes de iniciar transmissão de nova mensagem
- Reconhecimento negativo:
  - Rementente não espera pelo reconhecimento, envia mensagens sequencialmente. Destinatário ao notar que perdeu uma mensagem envia mensagem de não reconhecimento (NAK) da mensagem extraviada. Remetente retransmite mensagens a partir da mensagem identificada no NAK.

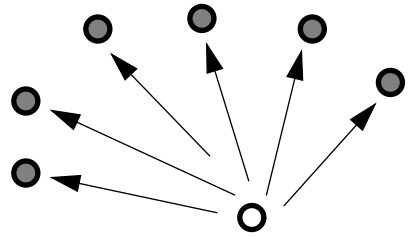
Algoritmo de Chang & Maxemchuk

Alguns casos que podem ser considerados:

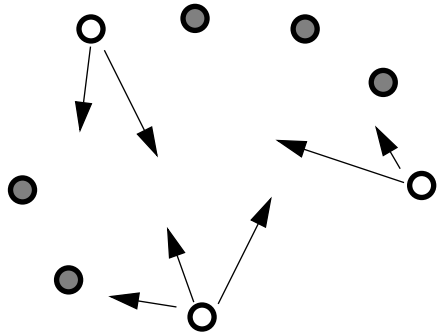
- receptor único:
  - recebimento e ordem triviais



- transmissor único:
  - reconhecimento negativo



- múltiplos transmissores e receptores
  - reconhecimento positivo
  - não garante sequência; por exemplo:
  - P_i: rec M1, rec M2
  - P_j: (perde M1), rec M2, rec M1

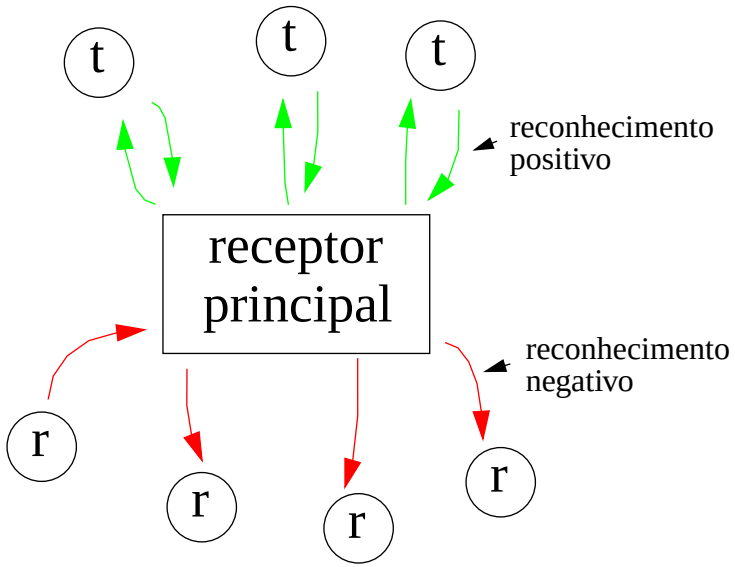


Algoritmo de Chang & Maxemchuk

Primeira tentativa

- Combinação de 2 sistemas simples

  transmissor único + receptor único



Note que o sistema de comunicação é do tipo broadcast.

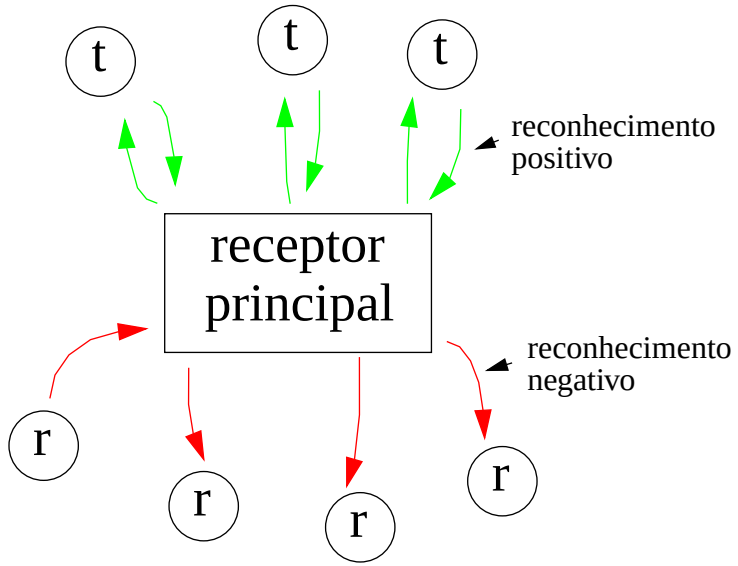
Receptor Principal

- transmite 1 ACK para cada mensagem; ACK contém timestamp
- receptores usam timestamp para detectar mensagens perdidas; ordenam mensagens pela timestamp

1 ACK por mensagem

Algoritmo de Chang & Maxemchuk

Deficiências da primeira tentativa

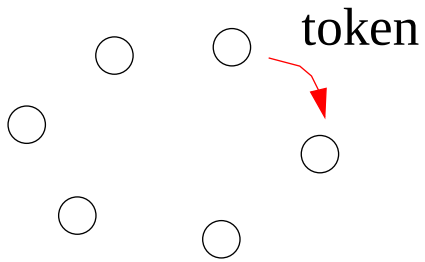


- Mensagens devem ser retidas indefinidamente
- Quando receptor primário falha, mensagens cujo ACK foi perdido por todos os receptores “desaparecem”

Algoritmo de Chang & Maxemchuk

Solução:

- responsabilidade como receptor deve ser rotativa: token
- receptores devem ter todas as mensagens carimbadas com ACK antes de aceitar responsabilidade
- pelo menos L outros receptores devem aceitar o token para que q mensagem seja “validada”



Bonus Grátis:

Mecanismo para deteção de falhas

Certas mensagens necessitam resposta:

- mensagem BRD transmitida, mesnagem ACK esperada
- retransmissão solicitada, MSG BRD esperada
- token transferido, confirmação esperada

O Protocolo

Cada nó mantém

- tl_i, versão da “lista de token”
- $M_i[s]$ n’mero do próximo BRD que nó i espera receber do nó s
- nts_i número do carimbo (timestamp) que i espera receber
- Qb, fila de broadcasts recebidos (mas não ainda carimbados)
- Qc fila de ACKS recebidos

Composto por duas fases

- normal
  - fase de operação; quando se inicia, TODOS os nós em tl_i têm o mesmo nts_i e $M_i[s]$
- reforma
  - quando falha é detectada, e tenta-se montar uma nova lista de token com os nós não falhos

Fase Normal

Cada transmissor

- Cransmite BRD, repete até receber ACK correspondente

(detecta falha do nó com token)

Nó com token

- Retransmite mensagem solicitada
- Carimba com ACK mensagem em Qb
- Transfere token em ACK, repete até obter confirmação

(detecta falha do proximo nó token)

Cada Receptor

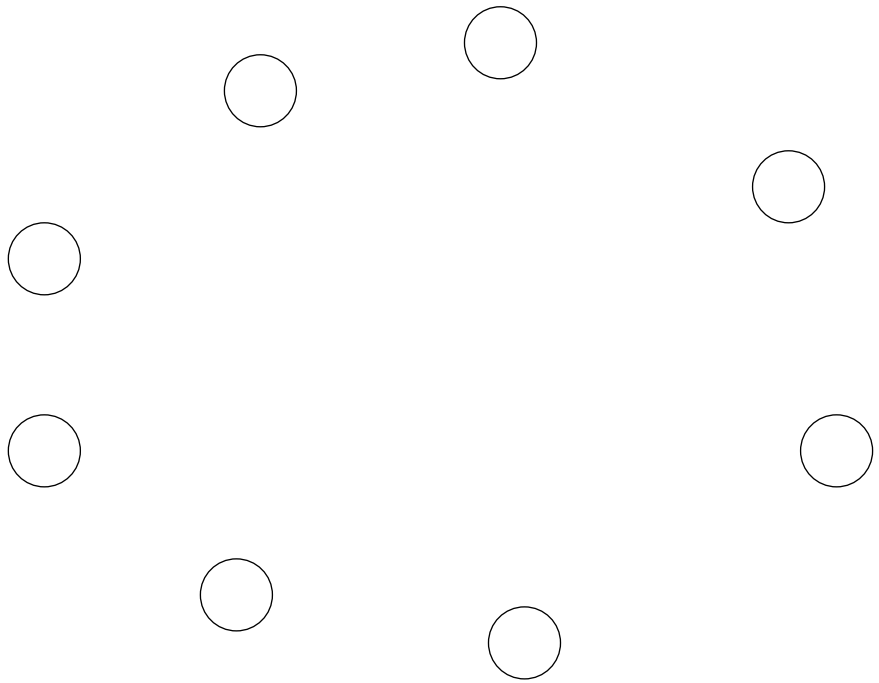
- Coloca BRD na fila Qb
- Coloca ACK na fila Qc
- Dependendo do carimbo ACK recebido
  - solicita ACK perdido
  - solicita BRD perdido
  - repete até obter resposta

(detecta falha do nó com token)

Próximo nó com token

- Adquire mensagens que foram carimbadas com ACK, se perdidas
- Carimba com ACK alguma mensagem em Qb ou transmite confirmção se Qb é vazia

Fase Normal



Mensagem BRD é validada quando o token tiver sido transferido L vezes após a mensagem ser carimbada (L+1 nós têm a mensagem)

Fase Reforma

Entra na fase de reforma quando falha é detectada

- redefine “lista de token”
- elege novo nó com token inicial
- gera novo token
- volta á operação normal

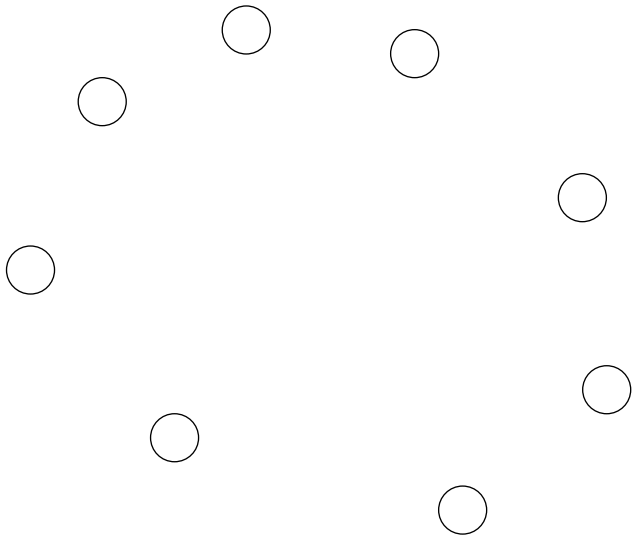
deve tratar de falhas, garantindo que

- existe uma única lista válida em qualquer momento
- nenhuma das mensagens VALIDADAS de listas anteriores sejam perdidas

Fase Reforma

Uma lista formada tentativamente na fase de reforma é válida se passa os testes de Maioria, Sequência e Robustez.

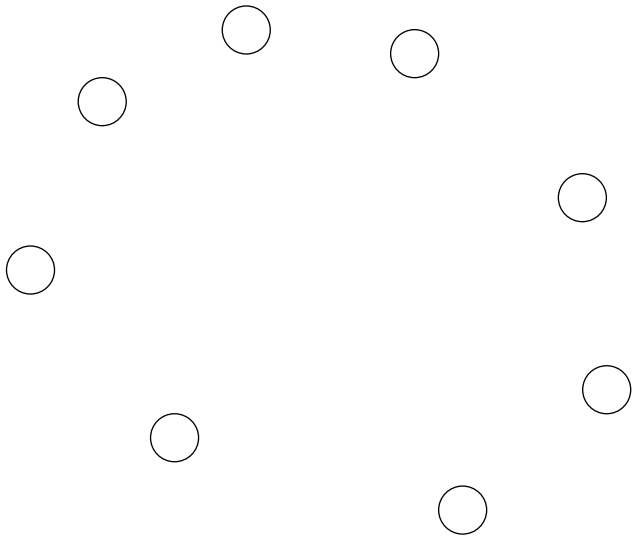
- MAIORIA: lista válida dever ter a maioria dos nós do sistema



Fase Reforma

Lista válida: maioria, sequência e robustez

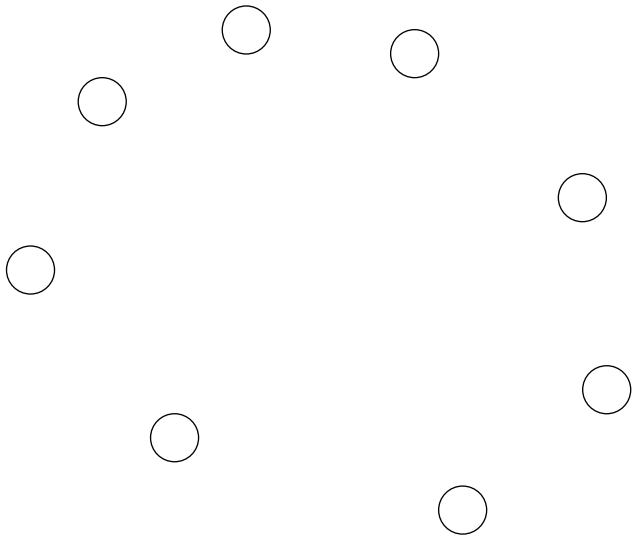
- SEQUÊNCIA: um nó só pode aderir a uma lista com número de versão maior do que qualquer lista conhecida. Listas tem carimbo tl = (versão, nó)



Fase Reforma

Lista válida: maioria, sequência e robustez

ROBUSTEZ: nenhuma das mensagens validadas pela lista anterior pode ser perdida. Nova lista deve conter pelo menos um dos L nós seguintes (na lista anterior) ao nó que validou a última mensagem na lista anterior



Fase Reforma

Protocolo para reforma: um nó é o iniciador, os outros são escravos

Iniciador

- Fase 1:

  quando falha ou recuperação é detectada, inicia reforma: faz broadcast de convite para todos os nós

- Fase 2:

  espera todas as respostas ou TIMEOUT
  if (todas as respostas == sim && passa testes Maioria, e Robustez)
  NovaLT = [todos os nós que responderam]
  Anuncia NovaLT a todos os nós que responderam
  else
  Anuncia ABORTO a todos os nós em NovaLT
  Modifica número de versão da LT, espera algum tempo e recomeça

- Fase 3:

  Espera por todas as respostas ou TIMEOUT
  if (todas as respostas de nós em NovaLT == sim)
  Gera novo token e passa a nó NovoToken
  valida NovaLT
  Retoma fase normal
  else
  Anunica ABORTO para nó NovoToken
  Espera e recomeça

Fase Reforma

Escravos

- Fase 1:

  Espera por convite
  if (passa teste sequência e não pertence a lista de reforma)
      vota sim
  else
      vota não

- Fase 2:

  Espera por NovaLT, ABORT ou TIMEOUT
  if (NovaLT recebida)
      if (ainda pertence à lista)
          Recupera todas as mensagens faltantes e vota sim
          Valida NovaLT
          Retoma Fase Normal (exceto nó NovoToken)
      else
          vota não
  if (ABORT recebido ou TIMEOUT)
      Deixa llista
      Espera e recomeça

Fase Reforma

Nó NovoToken apenas

- Fase 3:

    Espera por novo token, ABORT ou TIMEOUT

  if (novo token recebido && ainda pertence à lista)

    aceita token e começa a carimbar mensagens BRD com ACKs

    Final da reforma

  else

    Espera e recomeça

Algoritmo de Chang & Maxemchuk

Número de mensagens e espaço necessário

$P_{\overline{Q}}$ = probabilidade de que a fila de mensagens esperando por carimbo esteja vazia

Transferência do Token	Número de mensagens de controle	Espaço
1 por ACK	$1 + P_{\overline{Q}}$ (max 2)	N – 1
1 por K_w ACKs	$1 + P_{\overline{Q}}$ ($\rightarrow$ 1 se $K_w \rightarrow \infty$)	(N–1) × K_w ($\rightarrow \infty$)
K_r por ACK	$P_{\overline{Q}} + K_r$ ($\rightarrow$ N se $K_r \rightarrow$ N-1)	(N–1) / K_r ($\rightarrow$ 1)

Atraso Máximo (fase normal)

T× (L – 1), onde T = tempo para TIMEOUT

Maior Problema: atraso é ilimitado quando ocorrem falhas