

Broadcast em MPI

Durante o broadcast (`MPI_Bcast`):

- ▶ Um processo chamado **raiz** envia os mesmos dados para todos os processos no comunicador.
- ▶ Ao final da operação, o buffer da raiz é copiado para todos os outros processos.
- ▶ Seja P o número de processos envolvidos.

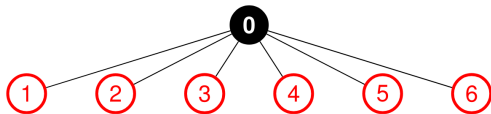
Considerações importantes:

- ▶ No broadcast tradicional sequencial (Flat Tree), a raiz envia a mensagem para cada processo individualmente. O tempo de comunicação cresce linearmente com P ($O(P)$).
- ▶ Árvores balanceadas (Binary Tree, Split Binary Tree, Binomial Tree) reduzem o número de passos de comunicação para $O(\log P)$, melhorando escalabilidade.
- ▶ Técnicas de **segmentação** permitem dividir grandes mensagens em partes menores, propagando-as em paralelo e reduzindo congestionamento de rede.
- ▶ Em clusters com muitos nós, escolher a topologia adequada impacta diretamente o tempo total de disseminação da mensagem.

Flat Tree [1]

- ▶ O nó raiz tem $P - 1$ filhos, todos recebem a mensagem diretamente.
- ▶ Mensagem transmitida sem segmentação.

Topologia Flat Tree



Chain Tree [2]

- ▶ Cada nó interno possui apenas um filho.
- ▶ Mensagem segmentada e transmitida sequencialmente até o último nó.

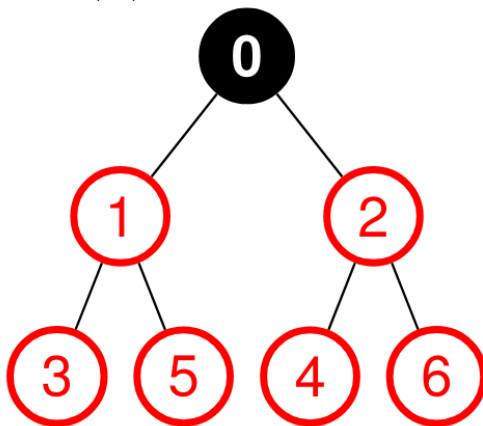
Topologia Chain Tree



Binary Tree [1]

- ▶ Cada nó interno possui dois filhos.
- ▶ Mensagem segmentada e distribuída da raiz para todas as folhas.

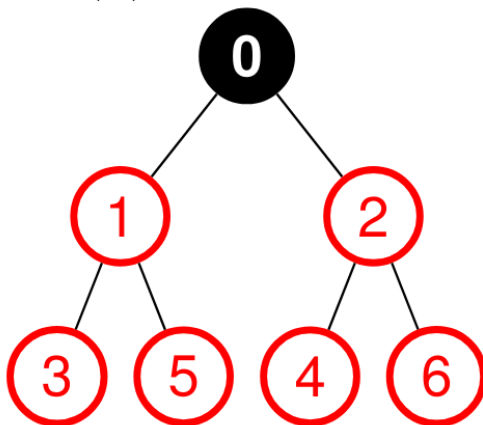
Topologia Binary Tree



Split Binary Tree [1]

- ▶ Utiliza a mesma topologia virtual da **Binary Tree**.
- ▶ Diferença principal: a mensagem é **dividida em duas metades** antes da transmissão.
- ▶ Cada metade da mensagem é enviada para uma subárvore diferente:
 - ▶ Metade esquerda → subárvore esquerda
 - ▶ Metade direita → subárvore direita
- ▶ Cada processo interno repassa sua metade para seus filhos correspondentes.
- ▶ Na fase final, as metades da mensagem são **trocadas entre os processos das subárvores** para que todos tenham a mensagem completa.
- ▶ Benefício: permite maior paralelismo na propagação e reduz o tempo de comunicação em comparação com a Binary Tree simples, especialmente para mensagens grandes.

Topologia Split Binary Tree



K-Chain Tree [1]

- ▶ Raiz possui K filhos; cada nó interno tem apenas um filho.
- ▶ Propagação paralela por níveis.
- ▶ Altura: $H_{k-chain} = \lfloor (P - 1)/K \rfloor$
- ▶ Mensagem segmentada.

K-Chain Tree: Topologia e Propagação [1]

- ▶ A raiz possui K filhos diretos; cada nó interno subsequente possui apenas um filho.
- ▶ A propagação ocorre **em níveis**:
 - ▶ Raiz envia a mensagem para seus K filhos em paralelo.
 - ▶ Cada filho propaga a mensagem ao seu único filho, continuando a cadeia.
- ▶ Permite que vários ramos da árvore avancem simultaneamente, aumentando o paralelismo.

K-Chain Tree: Altura e Segmentação [1]

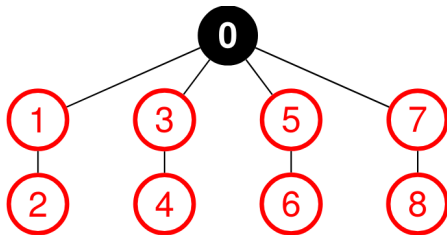
- ▶ A altura da árvore ($H_{k-chain}$) indica o número máximo de passos que a mensagem precisa percorrer até o último processo:

$$H_{k-chain} = \left\lfloor \frac{P-1}{K} \right\rfloor$$

onde:

- ▶ P = número total de processos
 - ▶ K = número de filhos diretos da raiz
 - ▶ $\lfloor \cdot \rfloor$ = função piso, arredondando para baixo
- ▶ O último processo precisa esperar $H_{k-chain}$ passos até receber a mensagem.
 - ▶ A segmentação da mensagem permite que partes diferentes sejam transmitidas em paralelo, acelerando a disseminação.

Topologia K-Chain Tree (K=4)



Binomial Tree [3, 1]

- ▶ Árvore binomial balanceada.
- ▶ Grau máximo dos nós diminui da raiz para as folhas.
- ▶ Altura da árvore: $H = \lfloor \log_2 P \rfloor$

Binomial Tree: Estrutura e Grau dos Nós [3, 1]

- ▶ A **Binomial Tree** é uma árvore balanceada construída segundo a definição matemática de árvores binomiais.
- ▶ Diferença principal em relação à Binary Tree:
 - ▶ O grau máximo dos nós (# de filhos) diminui da raiz para as folhas:

$$\lceil \log_2 P \rceil, \lceil \log_2 P \rceil - 1, \lceil \log_2 P \rceil - 2, \dots$$

- ▶ Isso cria uma estrutura mais balanceada e reduz congestionamento na raiz.

Binomial Tree: Altura, Nome e Benefícios [3, 1]

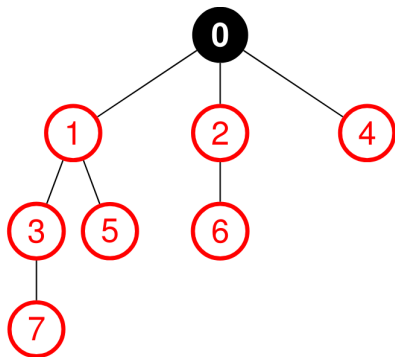
- ▶ A altura da árvore é:

$$H = \lfloor \log_2 P \rfloor$$




onde P é o número total de processos.

- ▶ O nome **Binomial** vem da relação com coeficientes binomiais:
 - ▶ O número de nós em cada nível segue a sequência dos coeficientes binomiais.
 - ▶ Garante distribuição equilibrada das mensagens entre os processos.
- ▶ Benefício: disseminação eficiente com menor número de passos de comunicação e maior paralelismo que a Binary Tree simples.

Topologia Binomial Tree



Referências I

-  J. Pjesivac-Grbovic, “Towards automatic and adaptive optimizations of MPI collective operations,” 2007.
-  P. Patarasuk, A. Faraj, and X. Yuan, “Pipelined broadcast on ethernet switched clusters,” in *Parallel and Distributed Processing Symposium*, 2006. IEEE, pp. 10–pp.
-  R. Thakur, R. Rabenseifner, and W. Gropp, “Optimization of collective communication operations in MPICH,” *The International Journal of High Performance Computing Applications*, vol. 19, no. 1, pp. 49–66, 2005.