A distributed Task Scheduler Optimizing Data Transfer Time

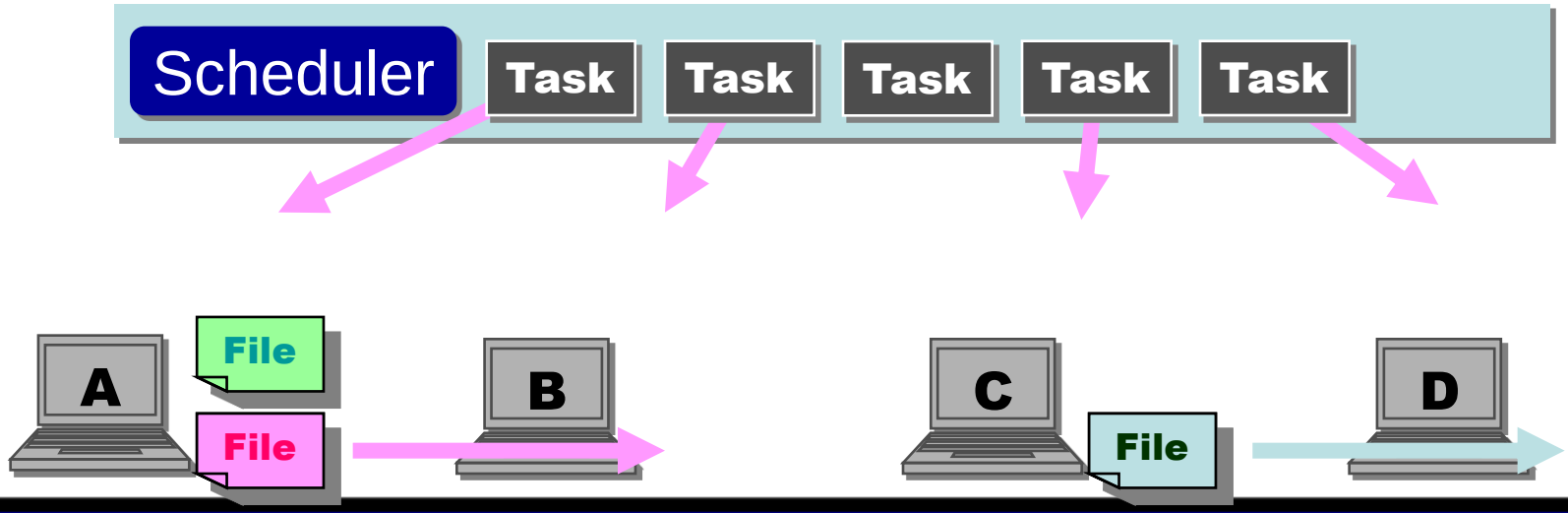
(データ転送時間を最適化する分散タスクスケジュラー)

Taura lab.

Kei Takahashi (56428)

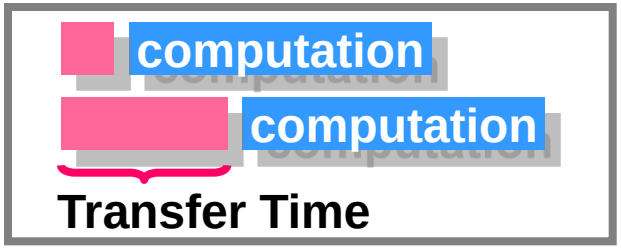
Task Schedulers

- A system which distributes many serial tasks onto available nodes
  - Task assignments
  - Data transfers
- Users can easily execute many serial tasks in parallel on distributed environment



Data Intensive Applications

- A computation using a large amount of data
  - Natural language processing, data mining, etc.
- Data transfer time takes considerable part of the total processing time
  - Example: parsing a set of data collected by crawlers located in some hosts
    - Typical network bandwidth: 1Gbps~50Mbps
    - Throughput of *Chasen* parser: 1.7MB/s = 14Mbps
    - In the worst case, the transfer takes 20% of the total processing time



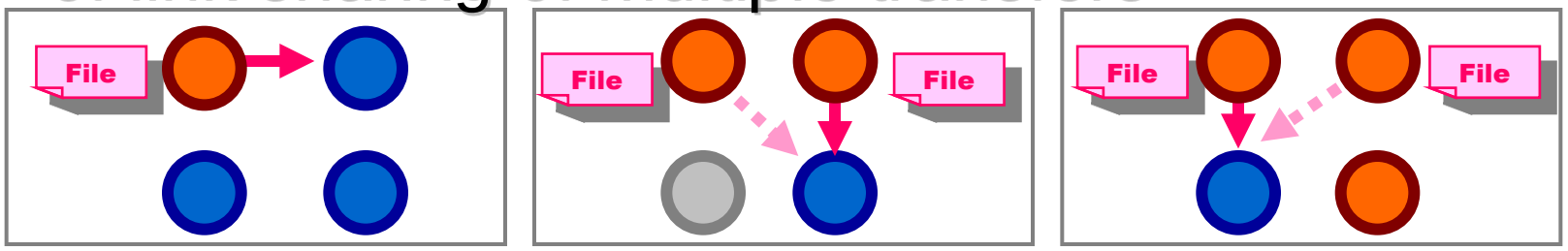
Considering Transfer Cost

- GrADS[1] considers transfer time when scheduling tasks
  - Measure bandwidth between any two hosts
  - Estimate transfer time by using the bandwidth
  - Assign task based on the time
- Since they only use static bandwidth values, their prediction can be far from the real network behavior when multiple data transfer share a link

[1] Mandal. et al. "Scheduling Strategies for Mapping Application Workflows onto the Grid“ in IEEEInternational Symposium on High Performance Distributed Computing (HPDC 2005)

Using Replicas

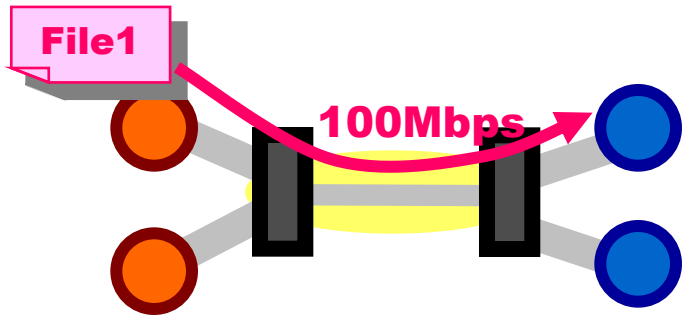
- Machida et. al[1] have developed a scheduler which utilizes multiple replicas
  - When data are copied from the source to another node, the copied node is registered as a replica
  - When another node requires the same data, the data is copied from the nearest replicas
- In this work, a task schedule itself is not optimized. Also, it does not care about the effect caused by of link sharing of multiple transfers

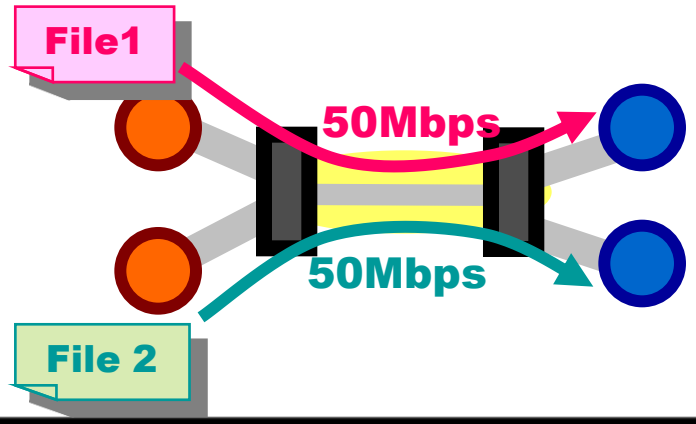


[1]町田悠哉 滝澤真一朗 中田秀基 松岡聡 ``レプリカ管理システムを利用したデータインテンシブアプリケーション向けスケジューリングシステム'' (HPCS2006)

Effects of Link Sharing

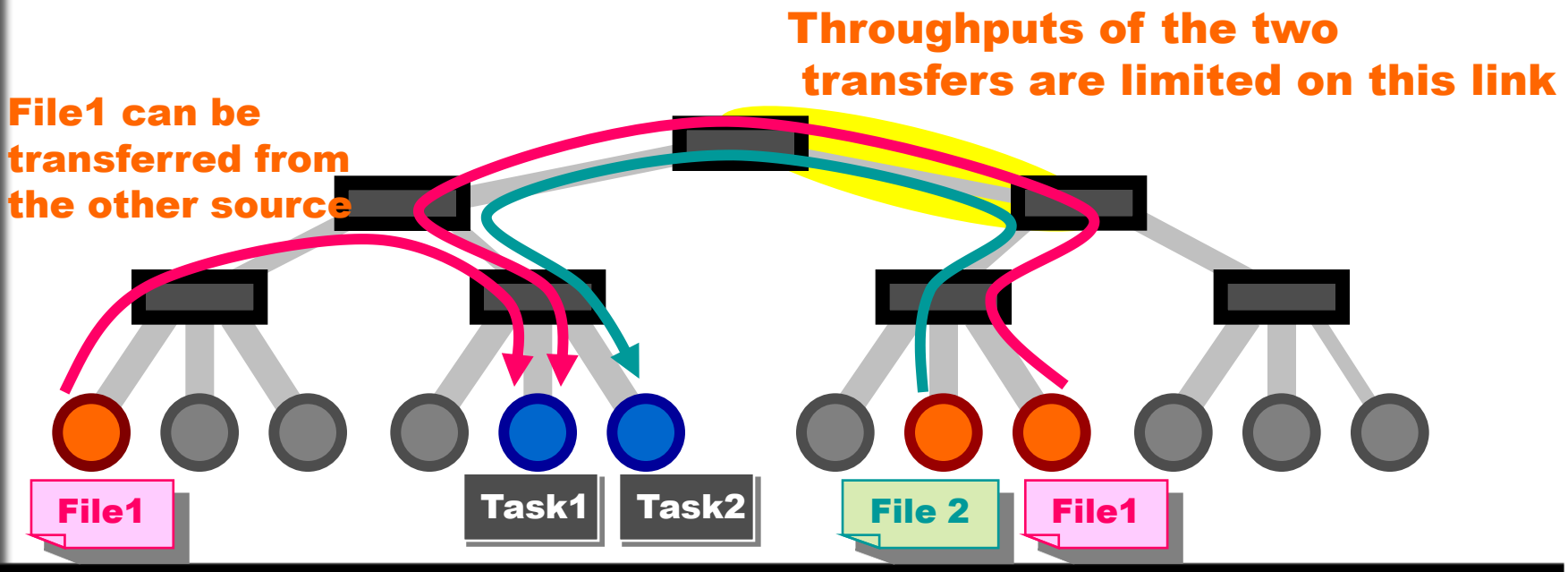
- If several transfers share a link, the sum of their throughputs cannot exceed the link bandwidth
  - throughput(File1) + throughput (File2) <= (Link bandwidth)
- Bandwidth between two hosts varies when multiple data are transferred
  - The value is reduced in 25% if 4 transfers share a link





Considering Topology

- The throughput is limited by the narrowest link in the transfer
- The throughput may become larger by altering source of transfers or changing task assignment



Research Purpose

- Design and implement an efficient distributed task scheduler for data intensive applications
  - Minimize data transfer time by using network topology and bandwidth information
    - Create a schedule which needs less transfers
    - Plan multicasts for data transfers
    - Maximize throughput by using linear programming

Agenda

- Background
- Purpose
- Our Approach
  - Task Scheduling Algorithm
  - Transfer Planning Algorithm
- Conclusion

Input and Output

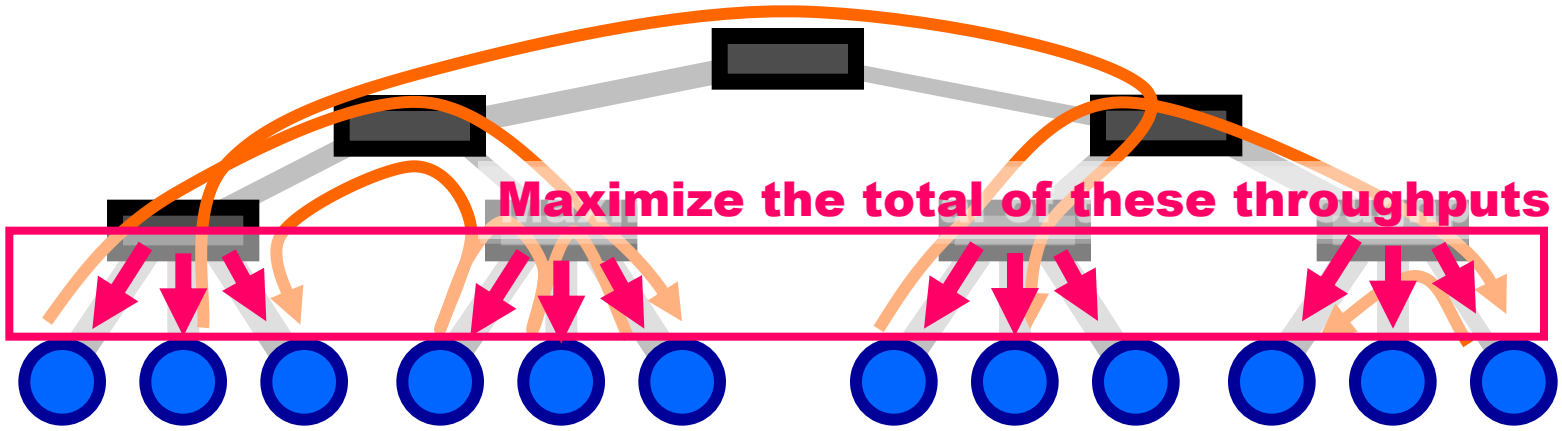
- Input:
  - Data locations
  - Network topology and bandwidth
  - Task information (required data)
- Output:
  - Task Scheduling:
    - Assignment of nodes to tasks
  - Transfer Scheduling:
    - From/to which host data are transferred
    - Limit bandwidth during the transfer if needed
- Final goal: Minimizing the total completion time

Immediate Goal

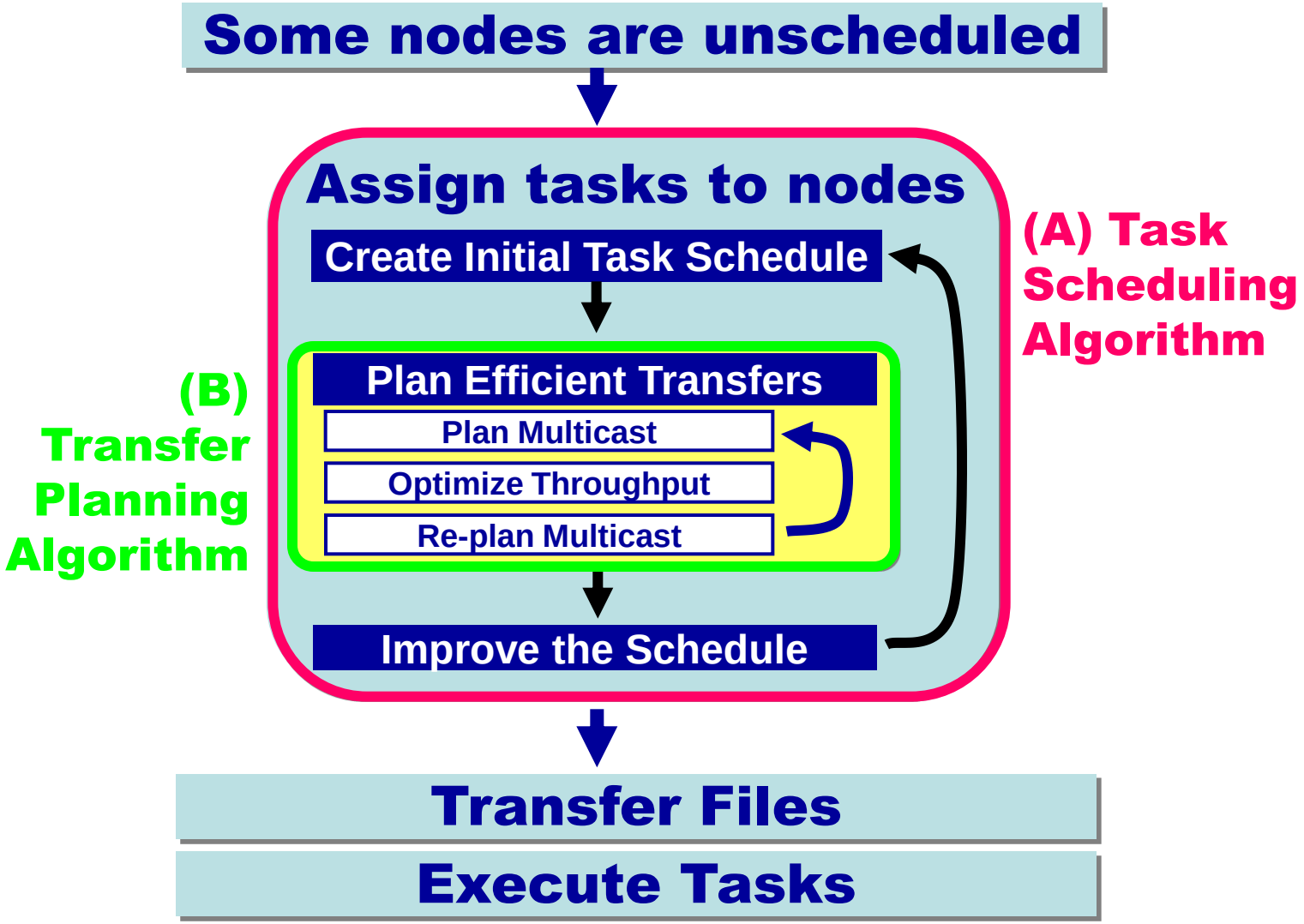
- Final goal: Minimizing the total completion time
- Our idea is to minimize data transfer time
- Immediate goal: Maximizing the total of data arrival throughput on each node

$$\sum_{i \in for\_every\_node} \left(\sum_{j \in for\_every\_file} bandwidth(file_{i,j}) \right)$$

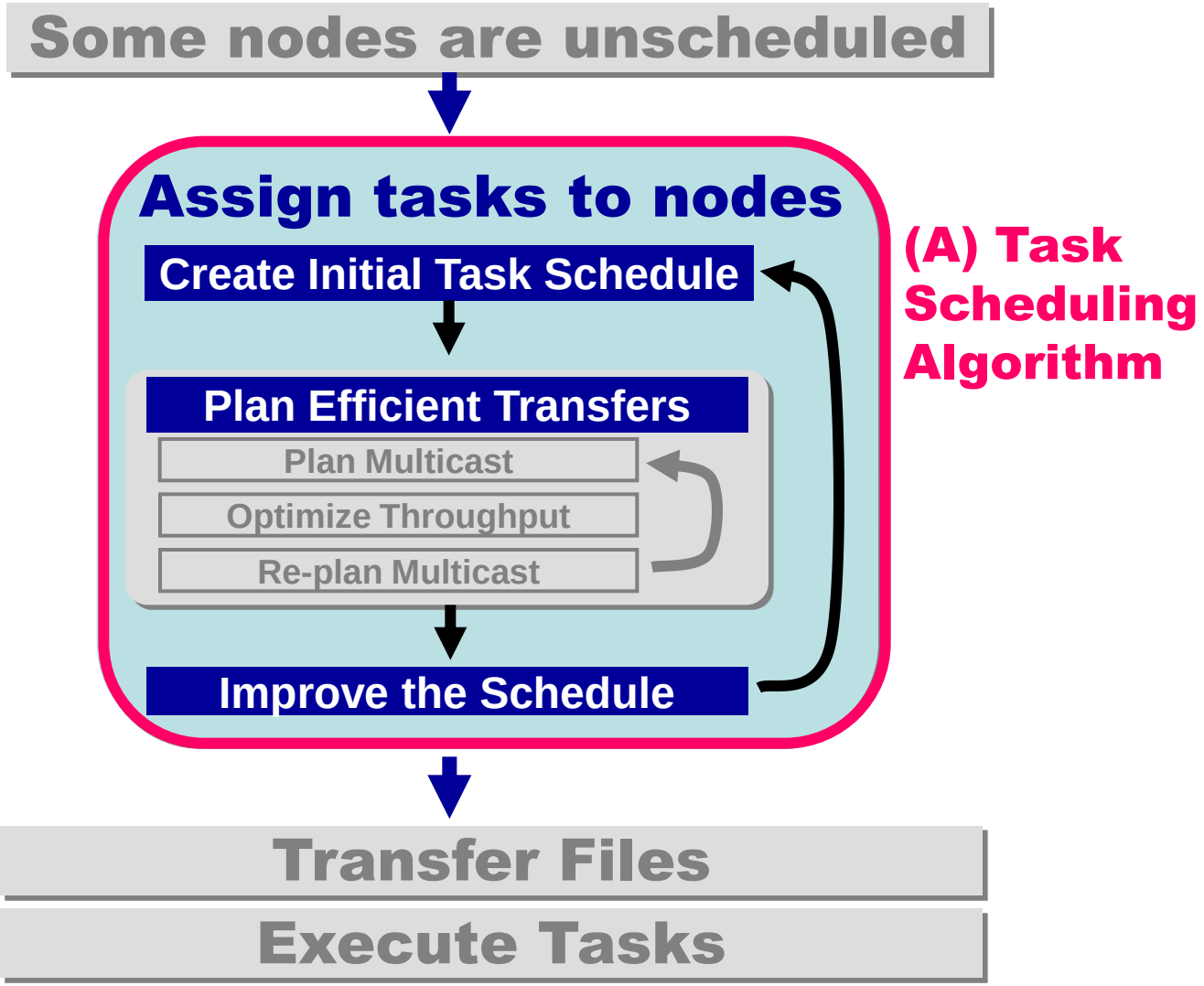
($file_{i,j}$: the j th file required by $task_i$)



The Whole Algorithm



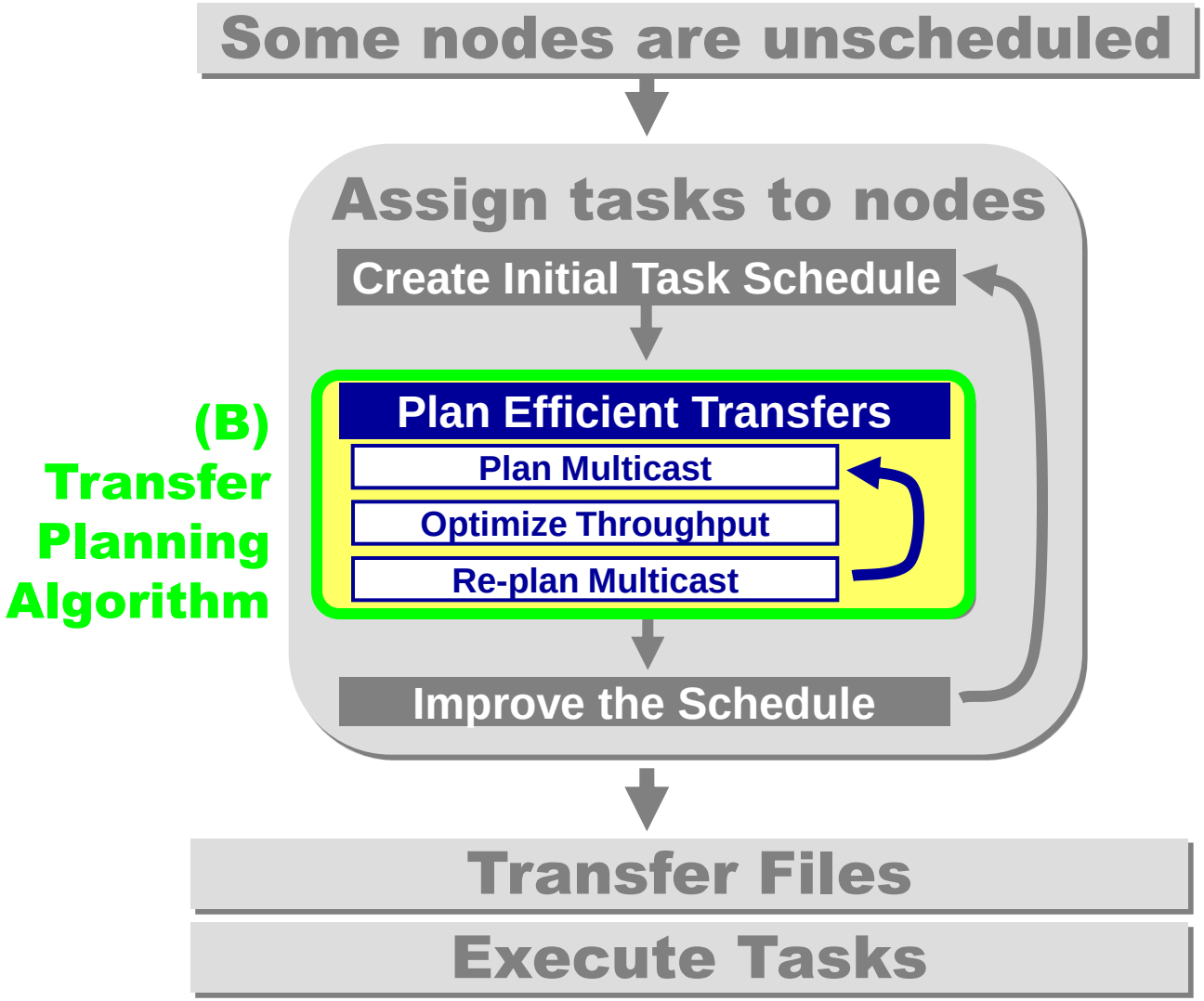
Task Scheduling Algorithm



Task Scheduling Algorithm

- When some nodes are unscheduled,
  - Create candidate task schedules
  - Plan efficient file transfers
    - From which node file is transfered
    - Bandwidth of each file transfer
  - Search for the best schedule which maximizes data transfer throughput (by using heuristics like GA, SA)
- Decide the task schedule, and start file transfers and task executions

Transfer Planning Algorithm

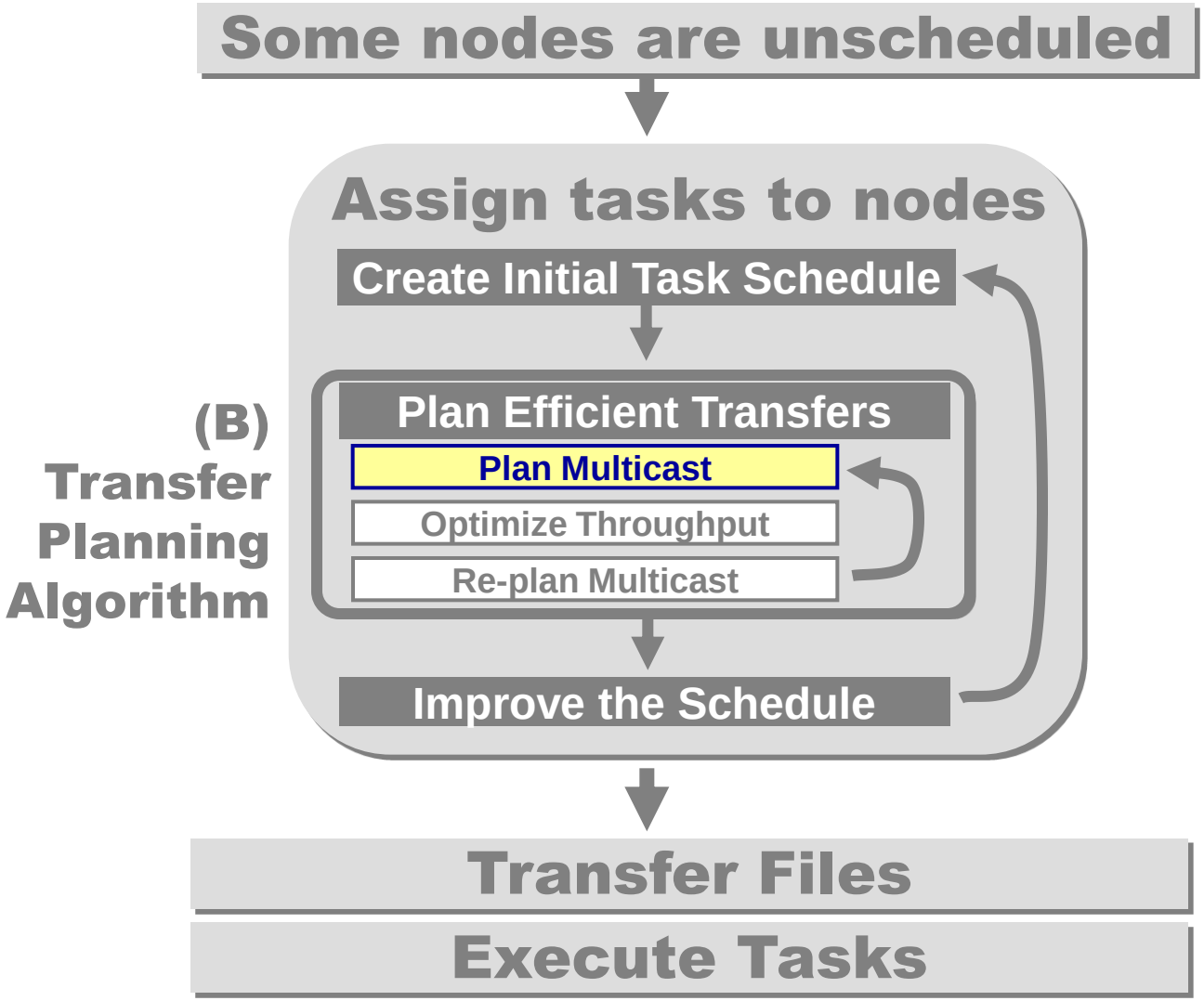


(B) Transfer Planning Algorithm

Transfer Planning Algorithm

- When source nodes and destination nodes for each file is given:
  - Decide source node for each destinations by using Kruskal's Algorithm
    - If multiple destinations uses one source, the data are multicasted
  - Calculate bandwidth value for each transfer to maximize throughput by using linear programming
  - If the originally chosen source is not optimal, modify the multicast tree by using a new bandwidth topology

Planing Multicast

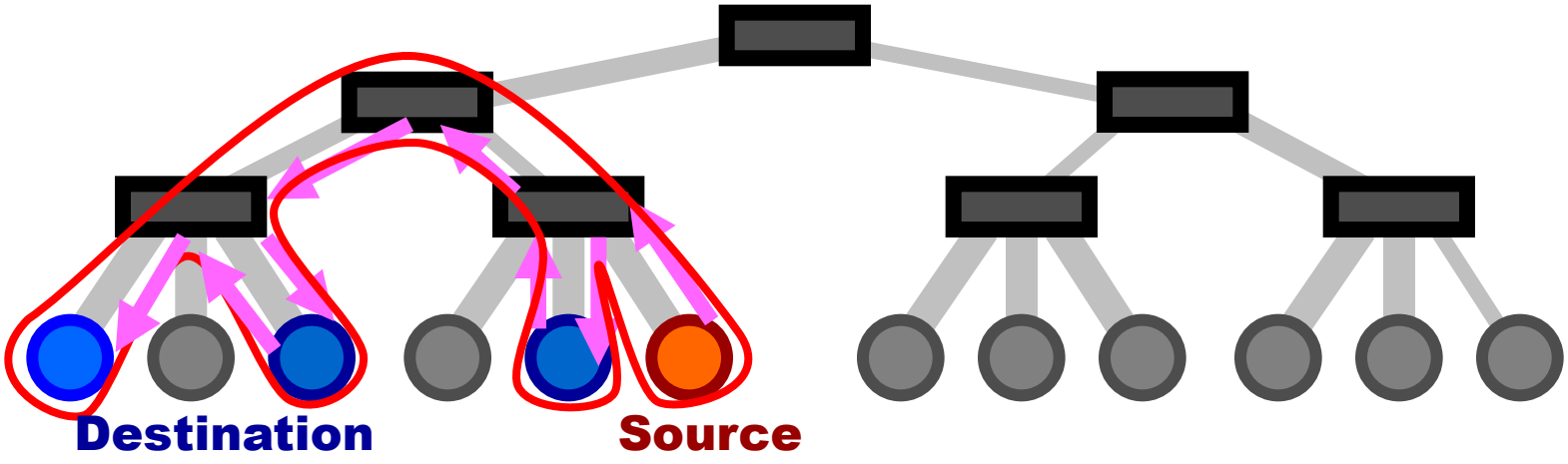


Pipeline Multicast (1)

- For a given schedule, it is known which nodes require which files
- When multiple nodes need a common file, pipeline multicast shortens transfer time (in the case of large files)
- The speed of a pipeline broadcast is limited by the narrowest link in the tree
- A broadcast can be sped up by efficiently using multiple sources

Pipeline Multicast (2)

- The tree is constructed in depth-first manner
- Every related link is only used twice (upward/downward)
- Since disk access is as slow as network, the disk access bandwidth should be also counted

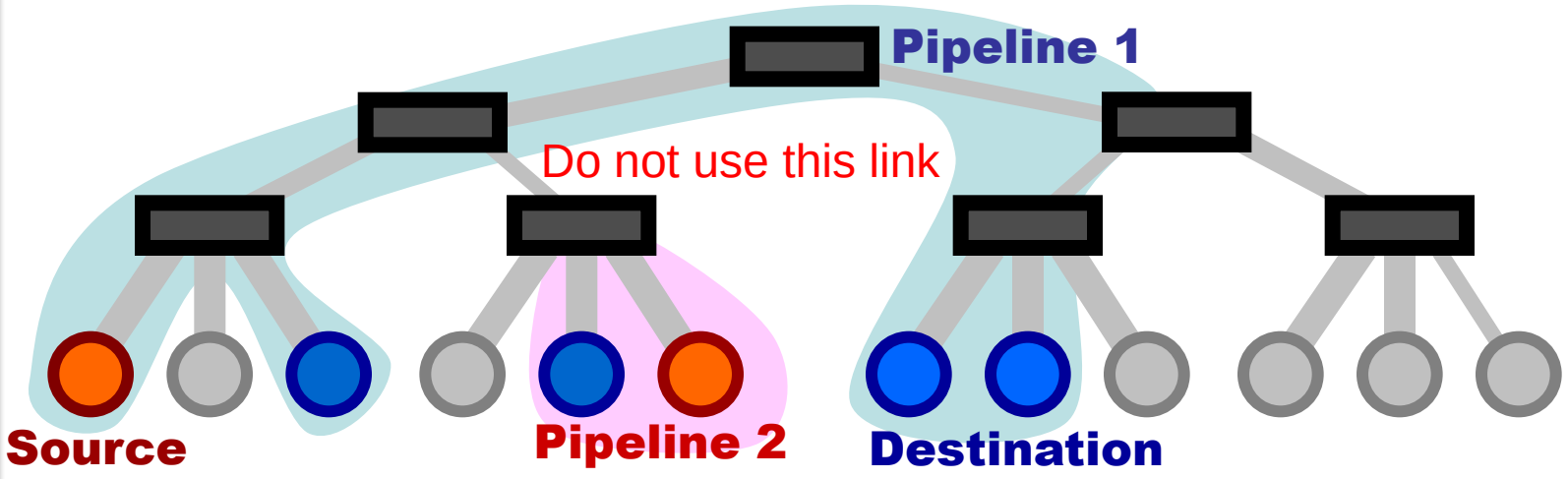


Multi-source Multicast

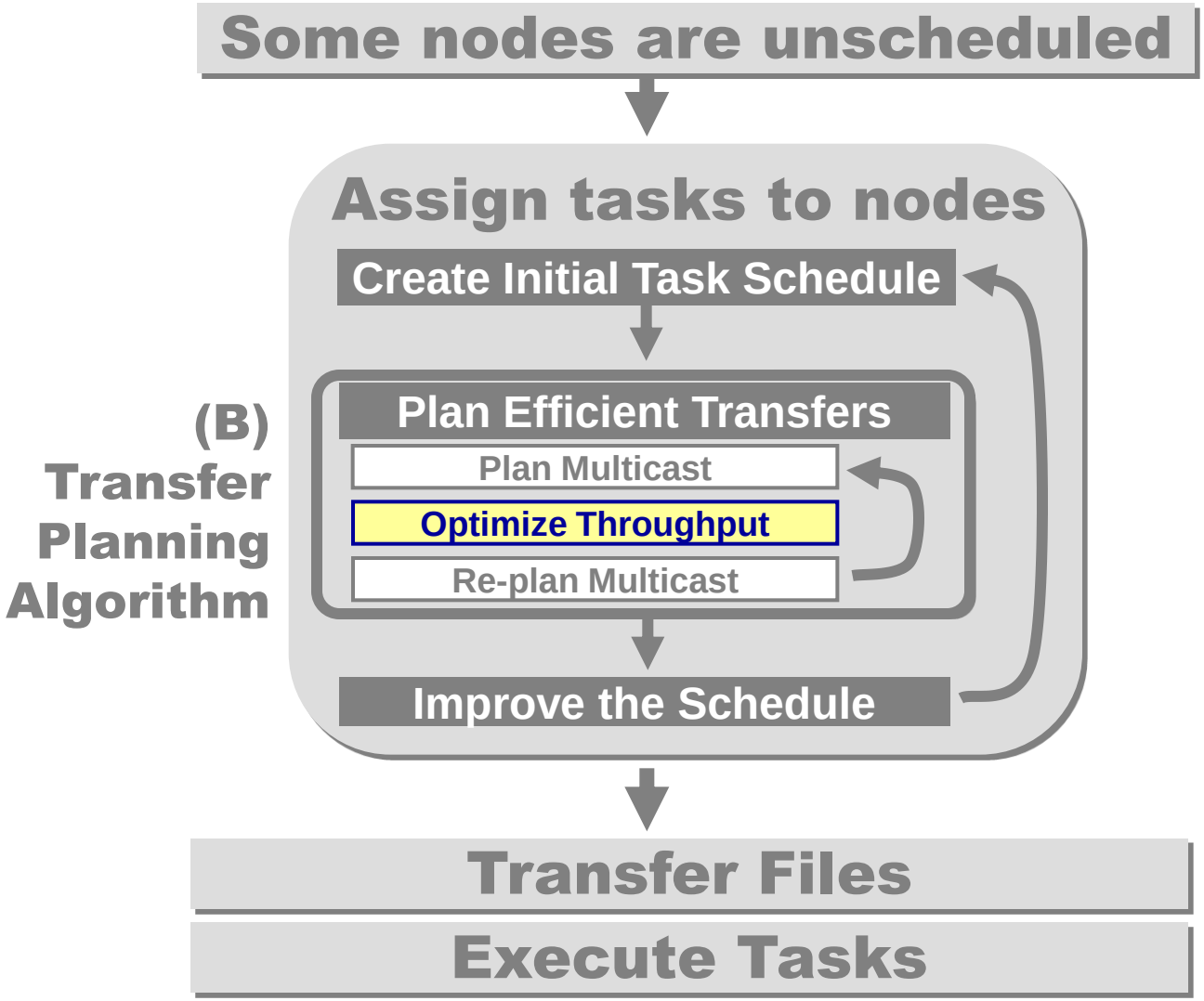
- M nodes have the same source data; N nodes need it
- For each link in the order of bandwidth:
  - If the link connects two nodes/switches which are already connected to the source node:

    → Discard the link
  - Otherwise: → Adopt the link

(Kruskal's Algorithm: it maximizes the narrowest link in the pipelines)

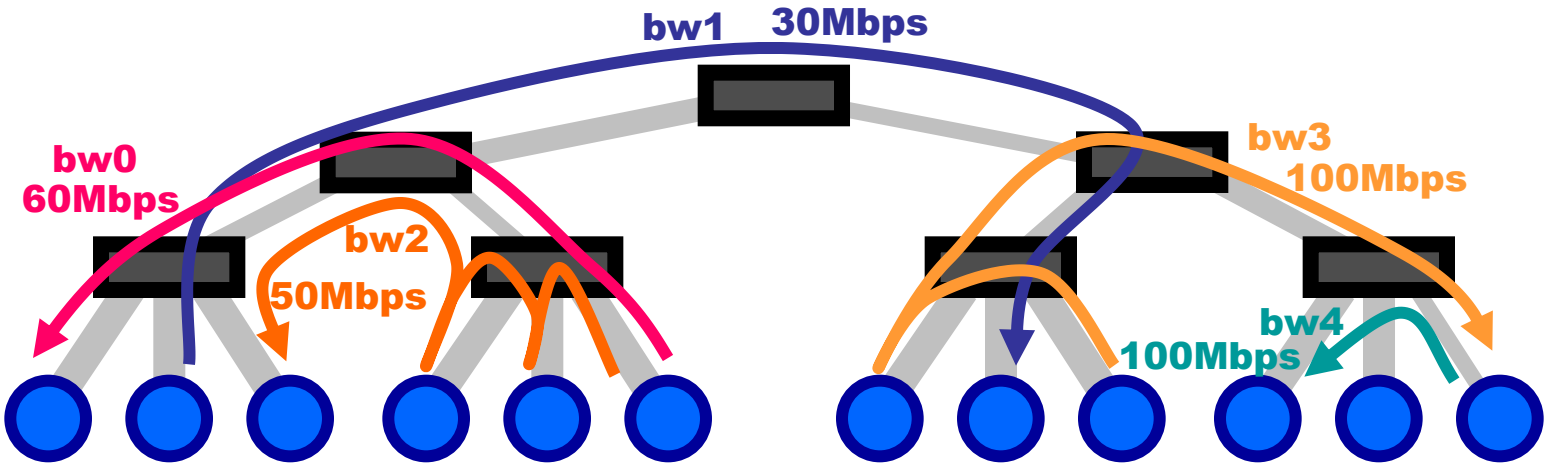


Maximizing Throughput

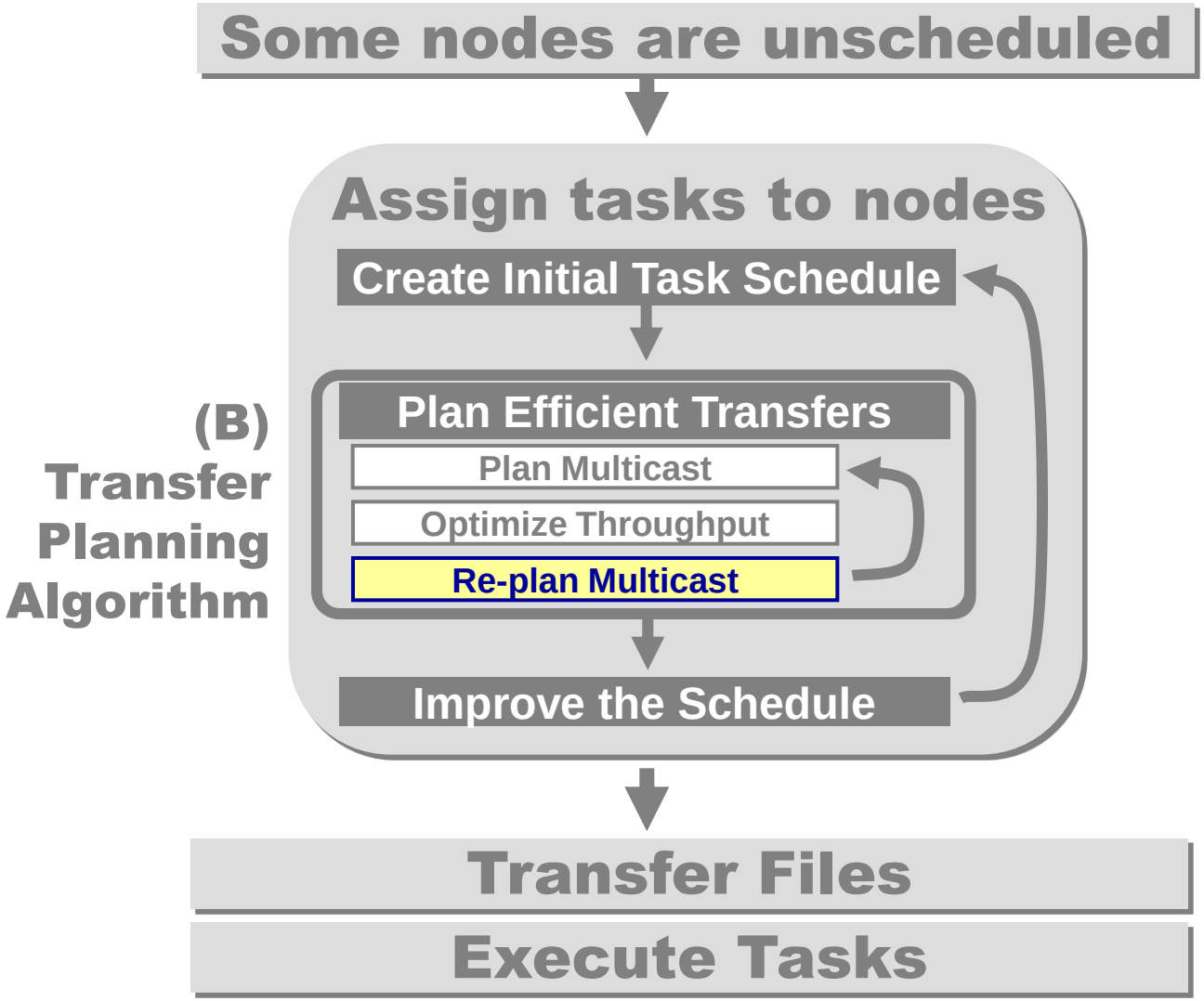


Maximizing Throughput

- After constructing multicast trees for every file, decide the bandwidth each transfer uses
  - By using linear programming
    - Maximize: (bw0 + bw1 + bw2 * 3 + bw3 * 2 + bw4)
    - Conditions: bw0 + bw1 ≤ (const)
    - bw1 + bw3 ≤ (const)
    - bw0 + bw2 + bw3 ≤ (const)
    - For local data, use disk access cost as the bandwidth

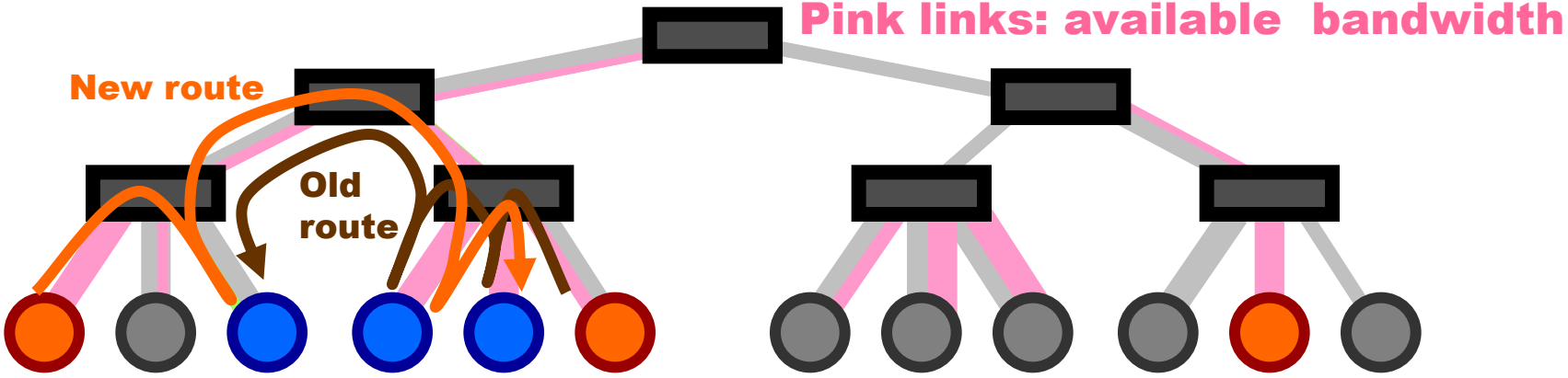


Re-planning Multicast

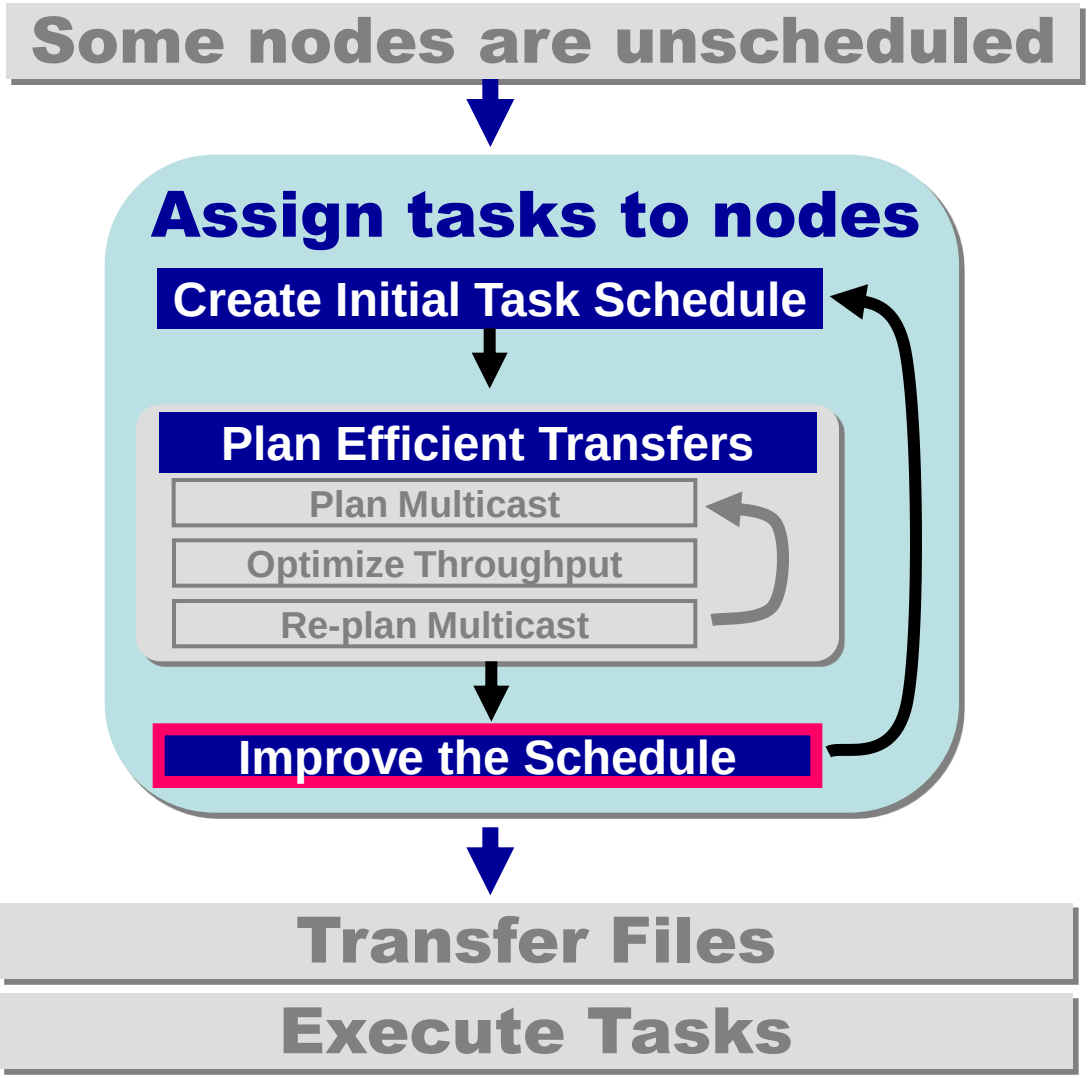


Re-planning Multicast

- Now every transfer schedule is decided, but it may not be optimal. Since multicast trees are planned independently, unnecessary conflictions may occur.
- By re-planning multicast by using current bandwidth information, the multicast trees are optimized
  - When re-optimizing a transfer, first create an available bandwidth map and construct multicast trees



Improve Task Schedule



Improve Task Schedule

- After efficient data transfer plan has obtained, the scheduler tries to reduce the transfer size by altering the task schedule
- We are thinking of using GA or Simulated Annealing. Since the most crowded link has found, we can try to reduce transfers on this link in the mutation phase.

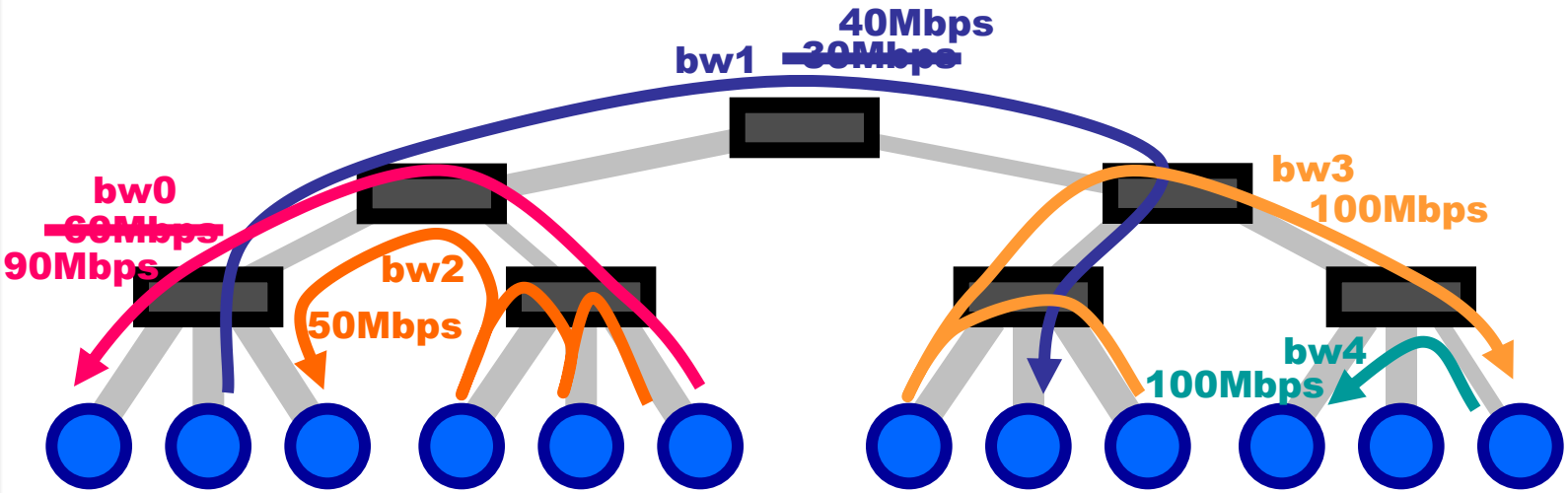
Actual Transfers

- After the transfer schedule is determined, the plan is performed as simulated
- The bandwidth of each transfer is limited to the previously calculated value
- When detecting a significant change in bandwidth, the schedule is reconstructed
  - The bandwidth is measured by using existing methods (eg. Nettimer[1])

[1] Kevin Lai et al. ``Measuring Link Bandwidths Using a Deterministic Model of Packet Delay'' SIGCOMM '00, Stockholm, Sweden.

Re-scheduling Transfers

- When one of the following events occurs, bandwidth assignments are recalculated
  - A transfer has finished
  - Bandwitdth has changed
  - New tasks are scheduled



Current Situation

- The algorithm has determined
- The implementation is ongoing
  - Plan data transfers when topology and a task schedule is given
  - Create a schedule with heuristics
  - Perform the real file transfer and task execution
- Evaluation will be done by comparing to existing schedulers

Conclusion

- Introduced a new scheduling algorithm
  - Predict transfer time by using network topology, and search for a better task schedule
  - Plan an efficient multicast
  - Maximize throughput by linear programming and by limiting bandwidth
  - Dynamically re-scheduling transfers

Publications

- 高橋慧, 田浦健次朗, 近山隆. マイグレーションを支援する分散集合オブジェクト. 並列／分散／協調処理に関するサマーワークショップ (ポスター発表)（SWoPP2005），武雄，2005年8月.

- 高橋慧, 田浦健次朗, 近山隆. マイグレーションを支援する分散集合オブジェクト. 先進的計算基盤シンポジウム（SACSIS 2005），筑波，2005年5月.