New Load Balancing Algorithms for the HHC Interconnection Network

Esam Al-Nsour, Mohammad Asmaran

Mohammad Fasha

Feb, 2014

Outline

- **Introduction**
  - Hyper Hexa-Cell (HHC) interconnection network
  - Hyper Hexa-Cell (HHC) addressing scheme
  - Load balancing in interconnection networks
- **Proposed Load Balancing Algorithms**
  - Algorithm A.
  - Algorithm B.
  - Algorithm C.

Outline

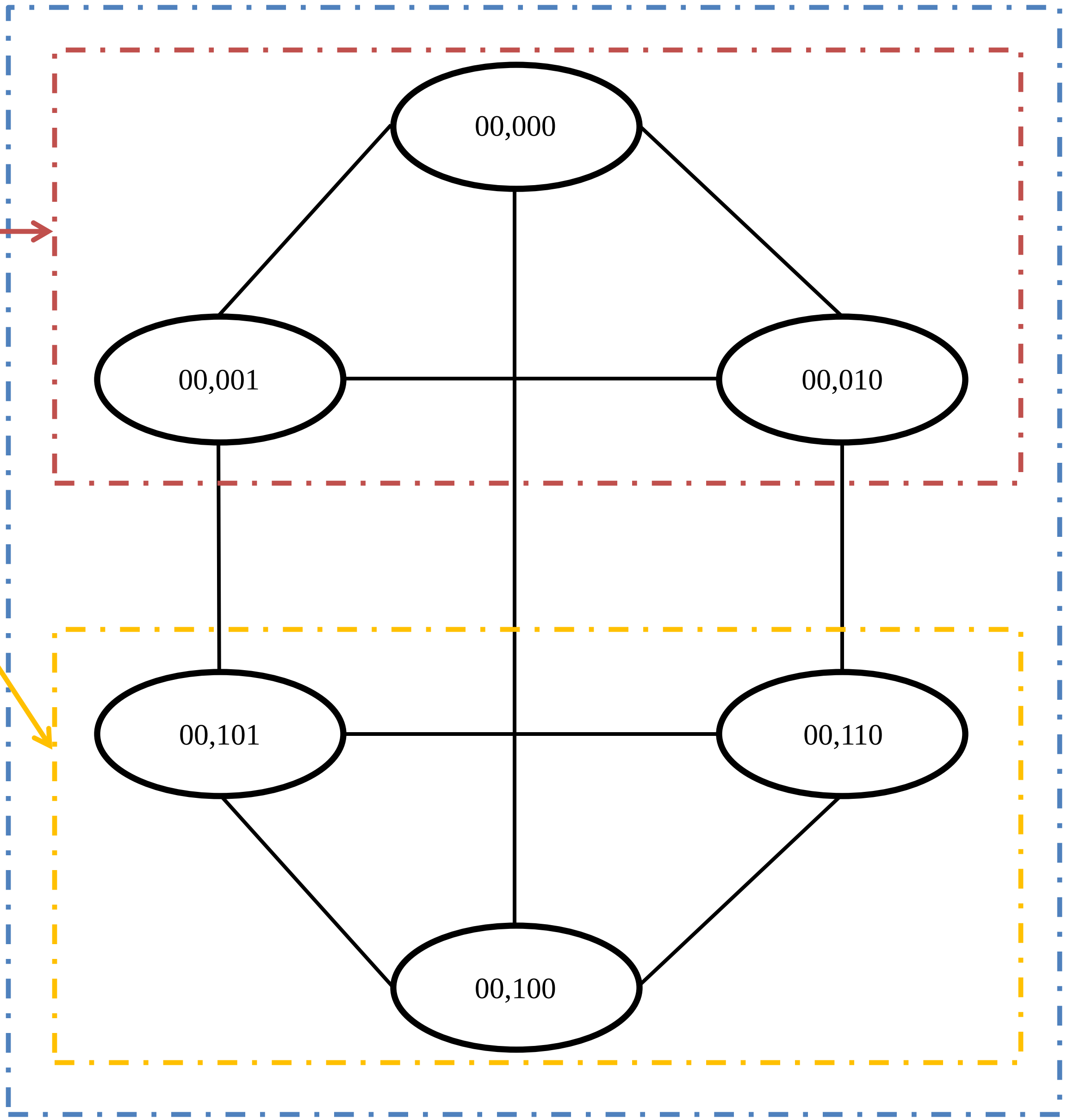
- **Algorithms A, B, C**
  - Description of algorithm work.
  - Algorithm analytical evaluation:
    - Execution time.
    - Load balancing accuracy.
    - Number of communication steps:
      - Maximum communication steps at any single node.
      - Total communication steps on the network.
    - speed
  - Experimental results.
- **Conclusion.**

Terminology

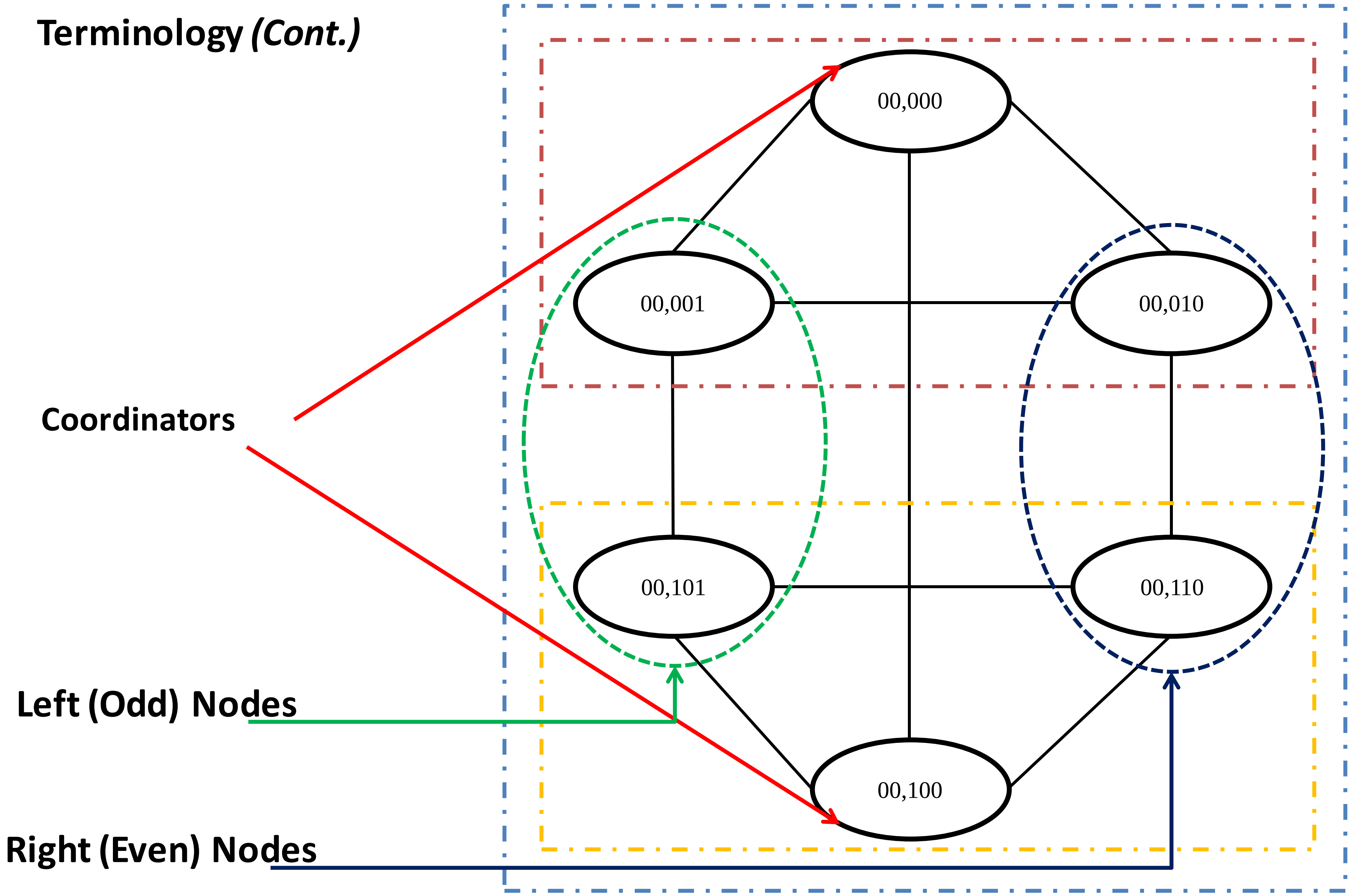
A: 1-dimensional HHC

B: Upper triangle

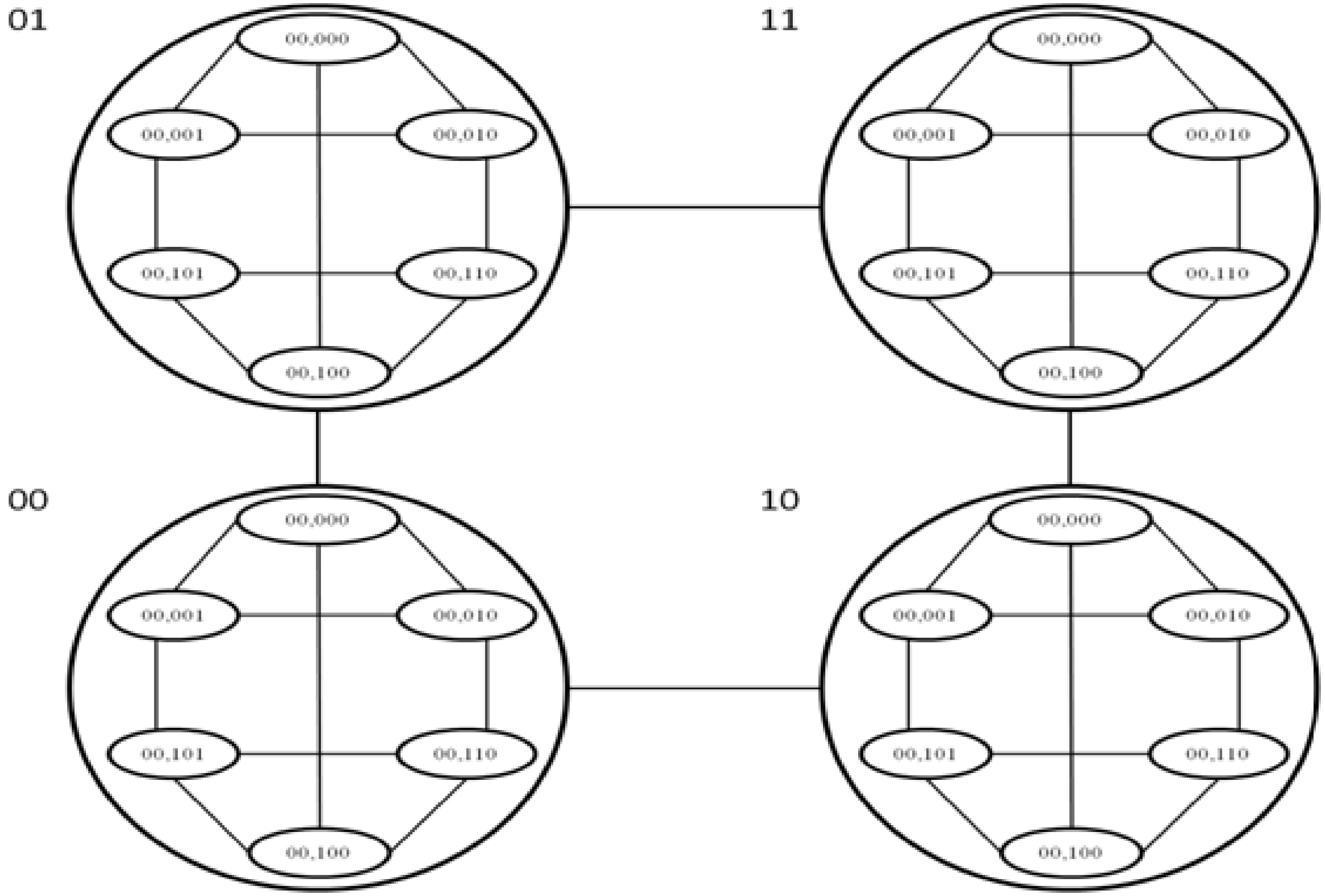
C: Lower triangle



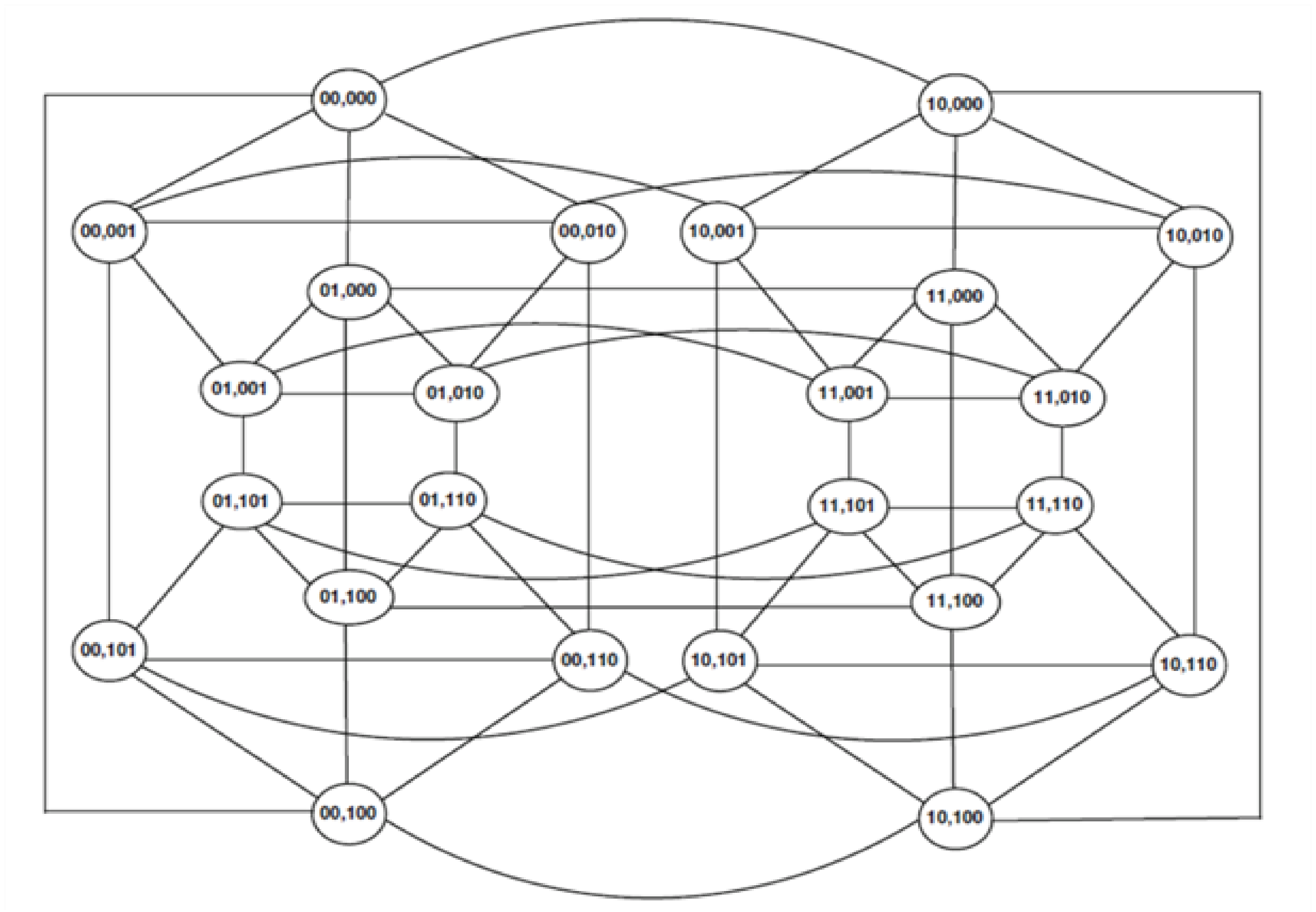
Terminology *(Cont.)*



How An HHC 1-dimensional replaces each single node of a Hyper-Cube inter-connection network



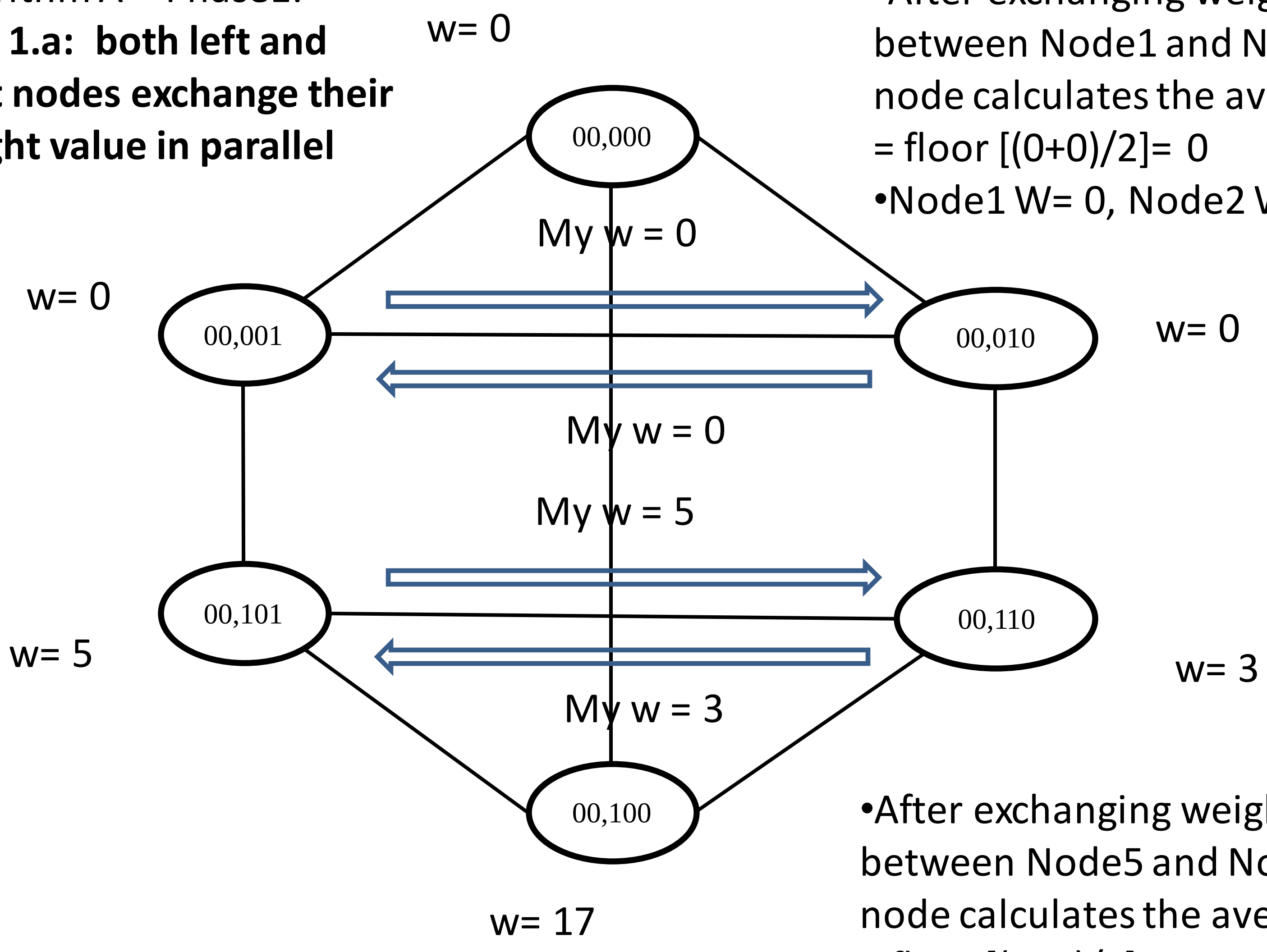
Example of A 3-dimensional HHC inter-connection network



Algorithm A

Example for tracing algorithm A – Phase1:

Step 1.a: both left and right nodes exchange their weight value in parallel

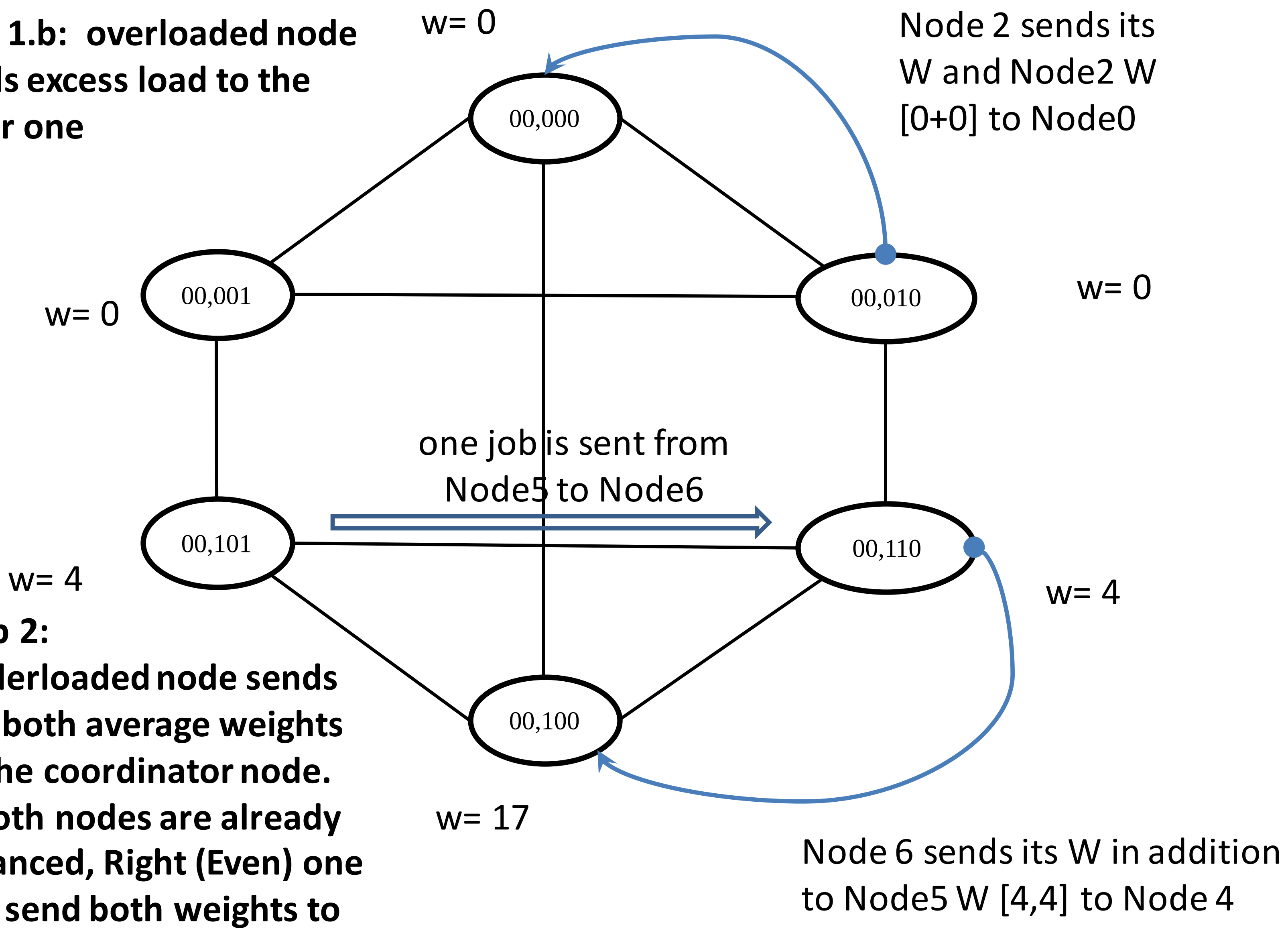


- After exchanging weights values between Node1 and Node2, each node calculates the average weight = floor [(0+0)/2]= 0
- Node1 W= 0, Node2 W= 0

- After exchanging weights values between Node5 and Node6, each node calculates the average weight = floor [(5+3)/2]= 4
- Node5 will sends 1 job to Node6
- Node5 W= 4, Node6 W= 4

Example for tracing algorithm A – Phase1:

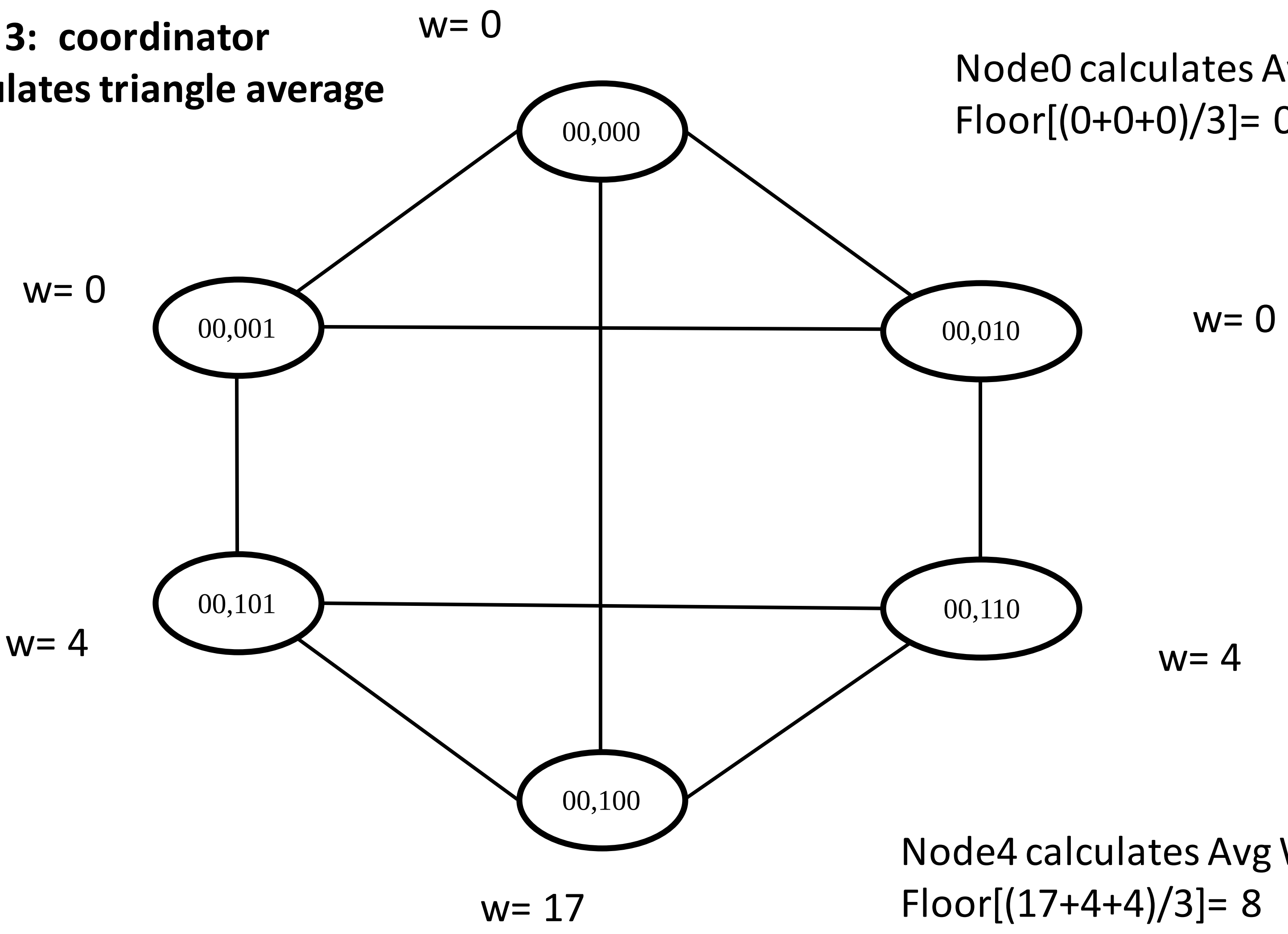
Step 1.b: overloaded node sends excess load to the other one


w= 0
00,000
Node 2 sends its W and Node2 W [0+0] to Node0
w= 0
00,001
00,010
w= 0
one job is sent from Node5 to Node6
w= 4
00,101
00,110
w= 4
00,100
w= 17
Node 6 sends its W in addition to Node5 W [4,4] to Node 4

Step 2:
Underloaded node sends the both average weights to the coordinator node. If both nodes are already balanced, Right (Even) one will send both weights to the coordinator.

Example for tracing algorithm A – Phase1:

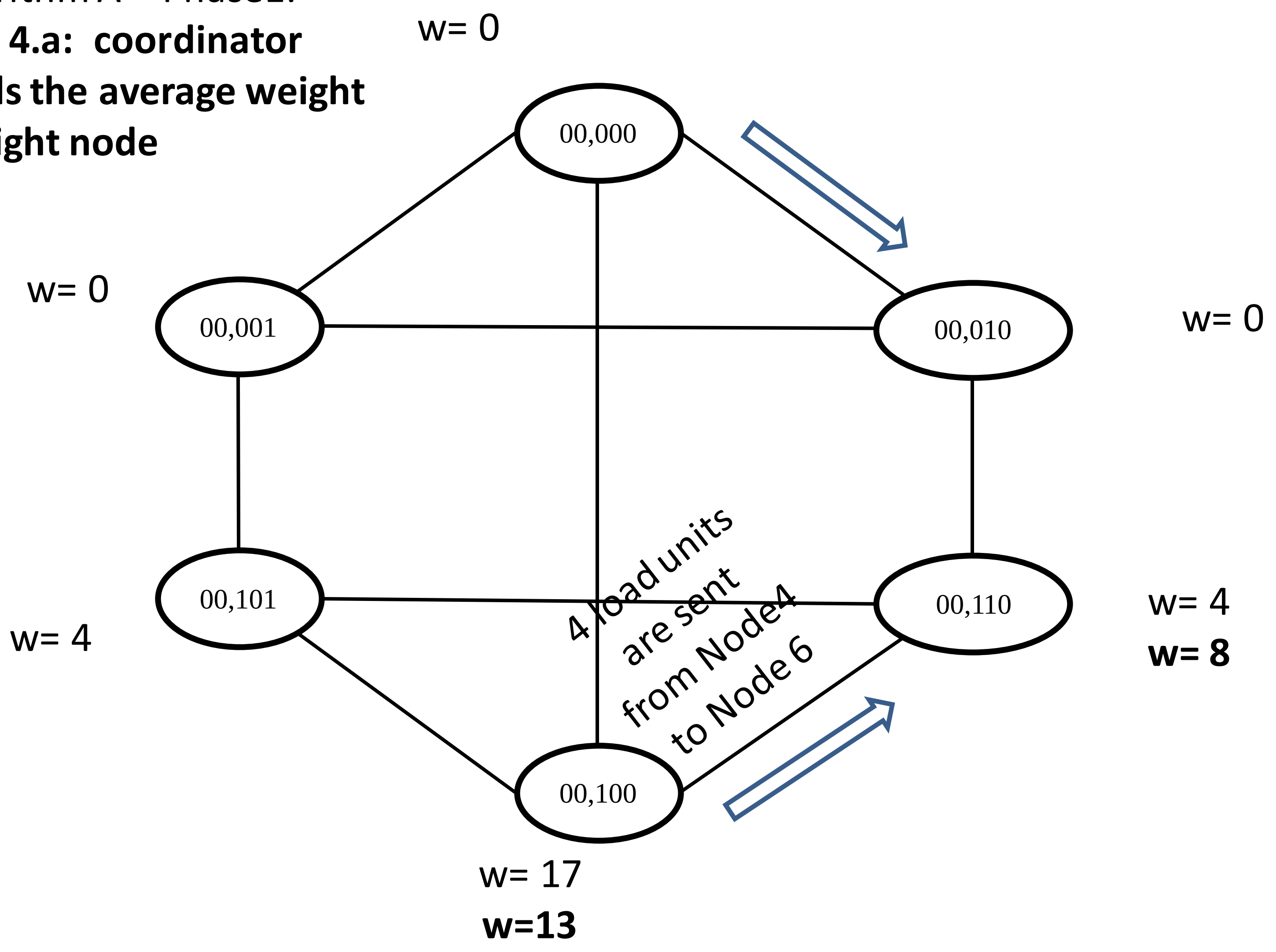
Step 3: coordinator calculates triangle average



Node0 calculates Avg W = Floor[(0+0+0)/3]= 0

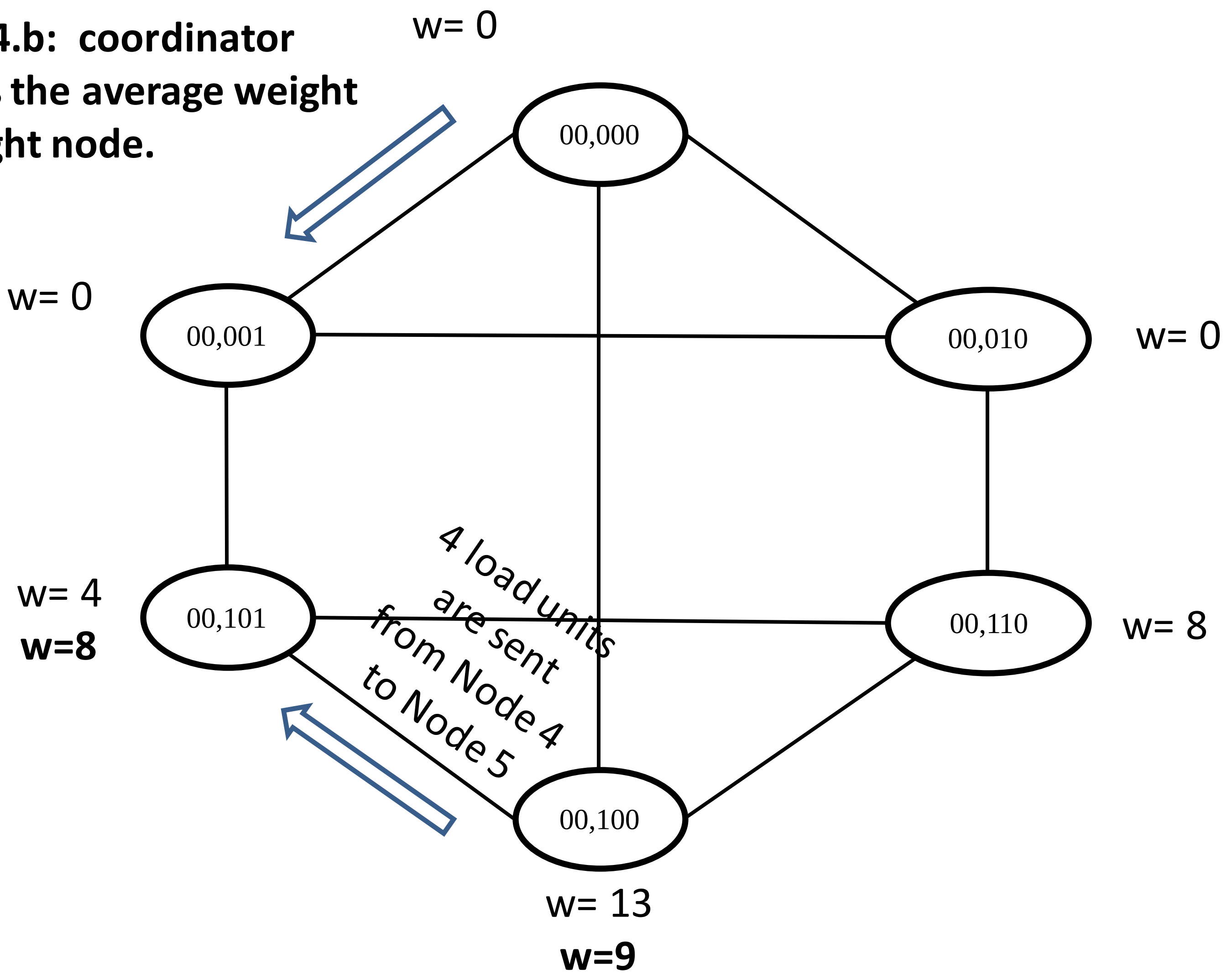
Node4 calculates Avg W = Floor[(17+4+4)/3]= 8

Example for tracing algorithm A – Phase1:
Step 4.a: coordinator sends the average weight to right node



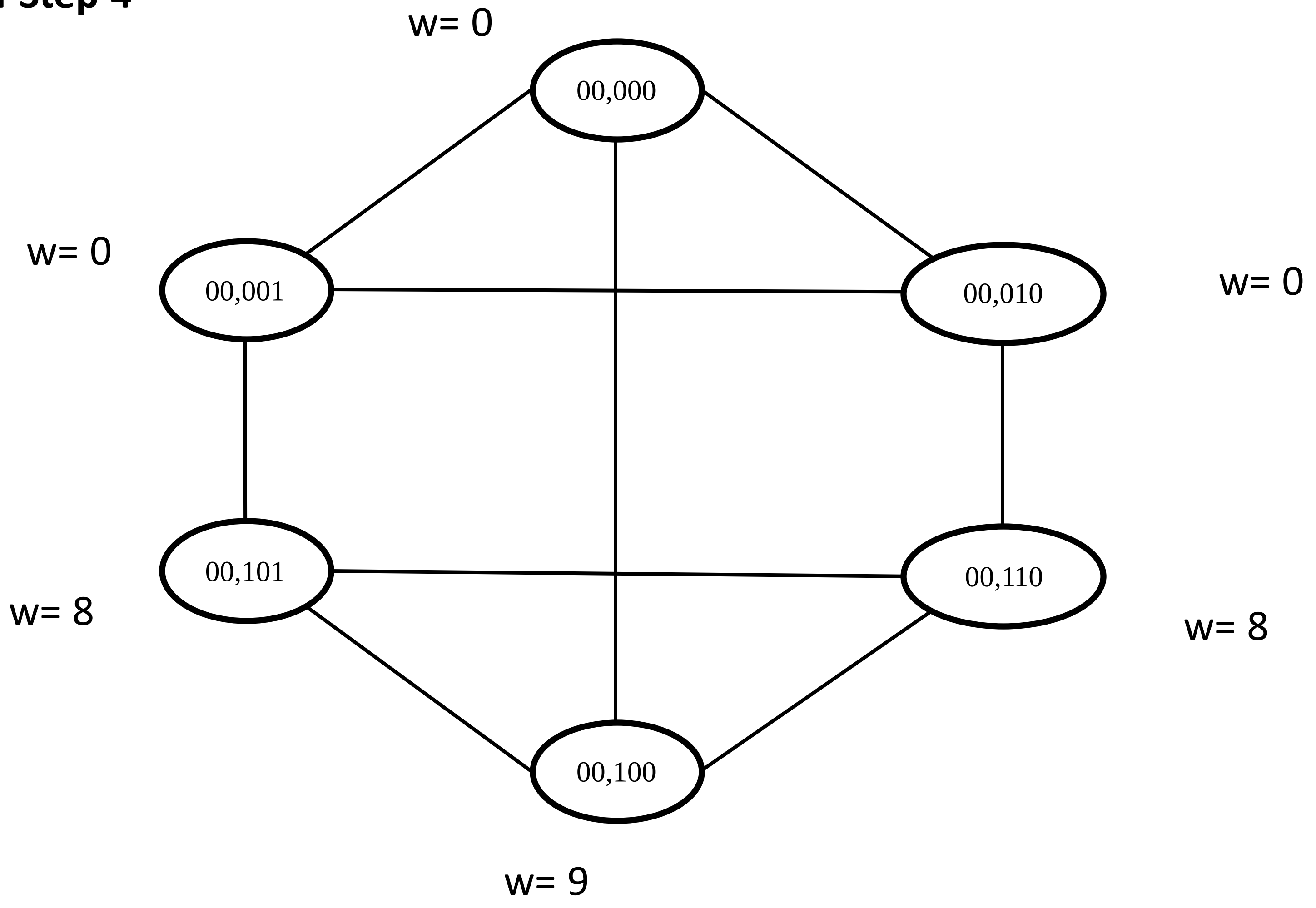
Coordinators send excess load to the right node

Example for tracing algorithm A – Phase1:
Step 4.b: coordinator sends the average weight to right node.



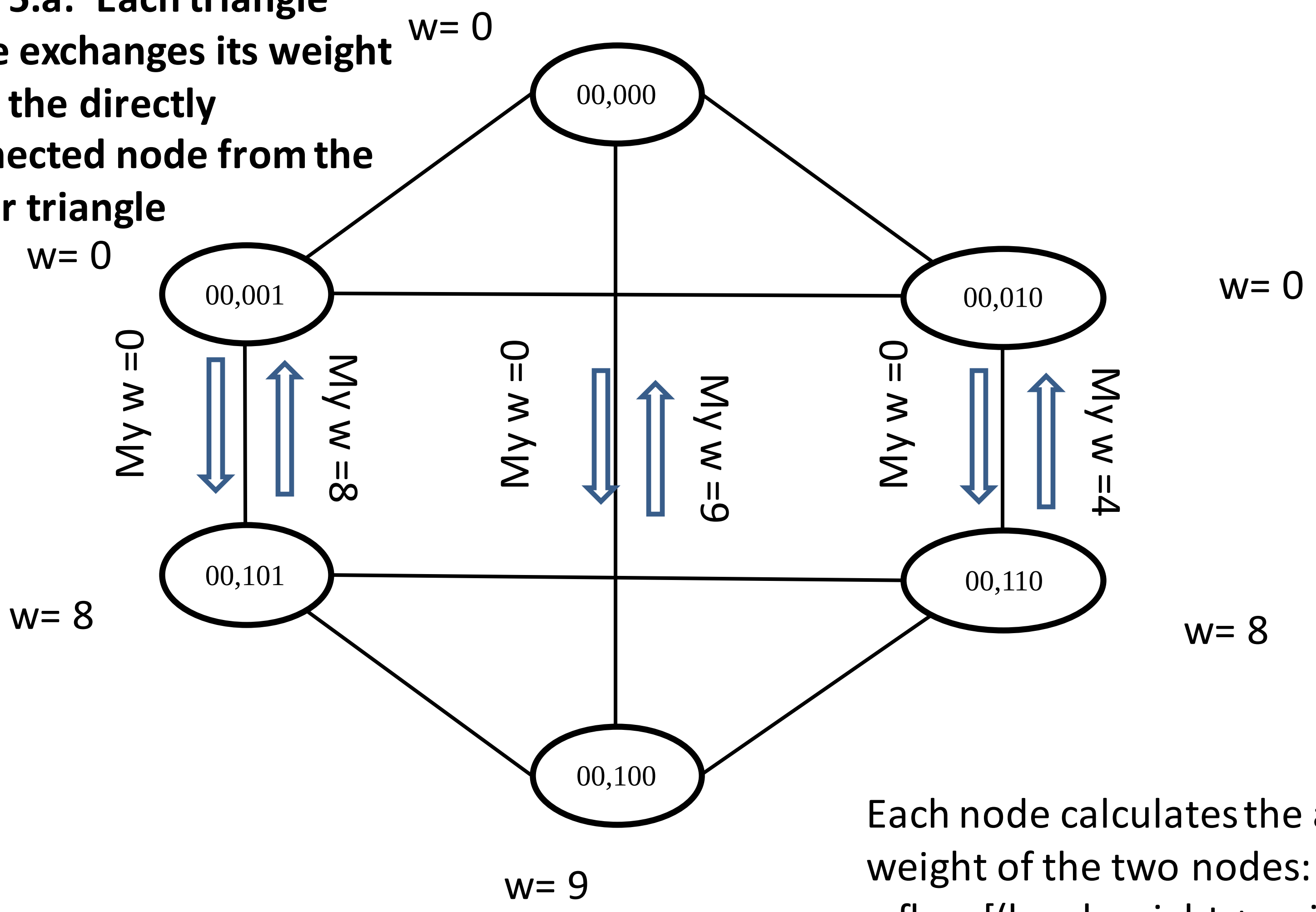
Coordinators send excess load to the left node

Example for tracing algorithm A – Phase1:
After Step 4



Example for tracing algorithm A – Phase1:

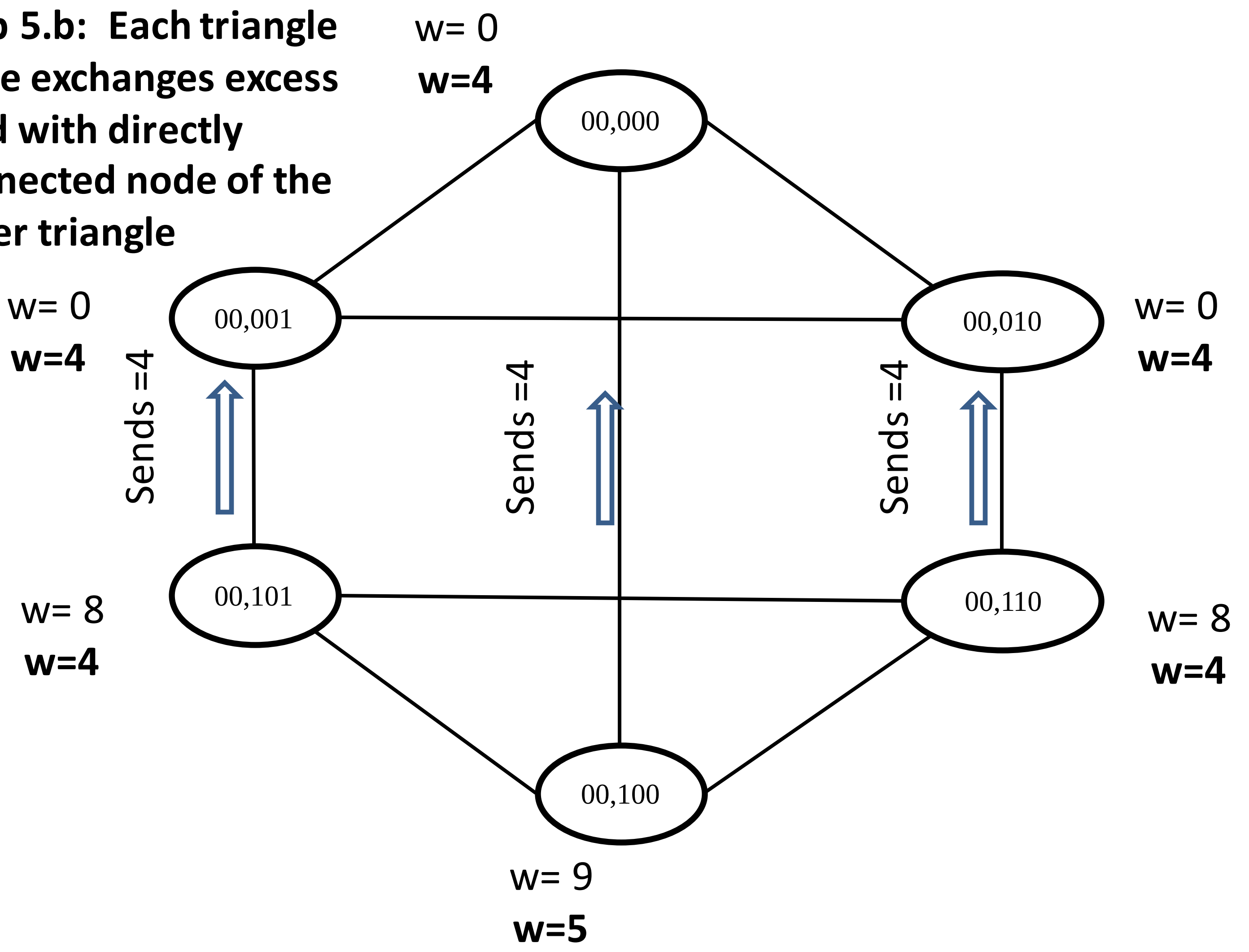
Step 5.a: Each triangle node exchanges its weight with the directly connected node from the other triangle



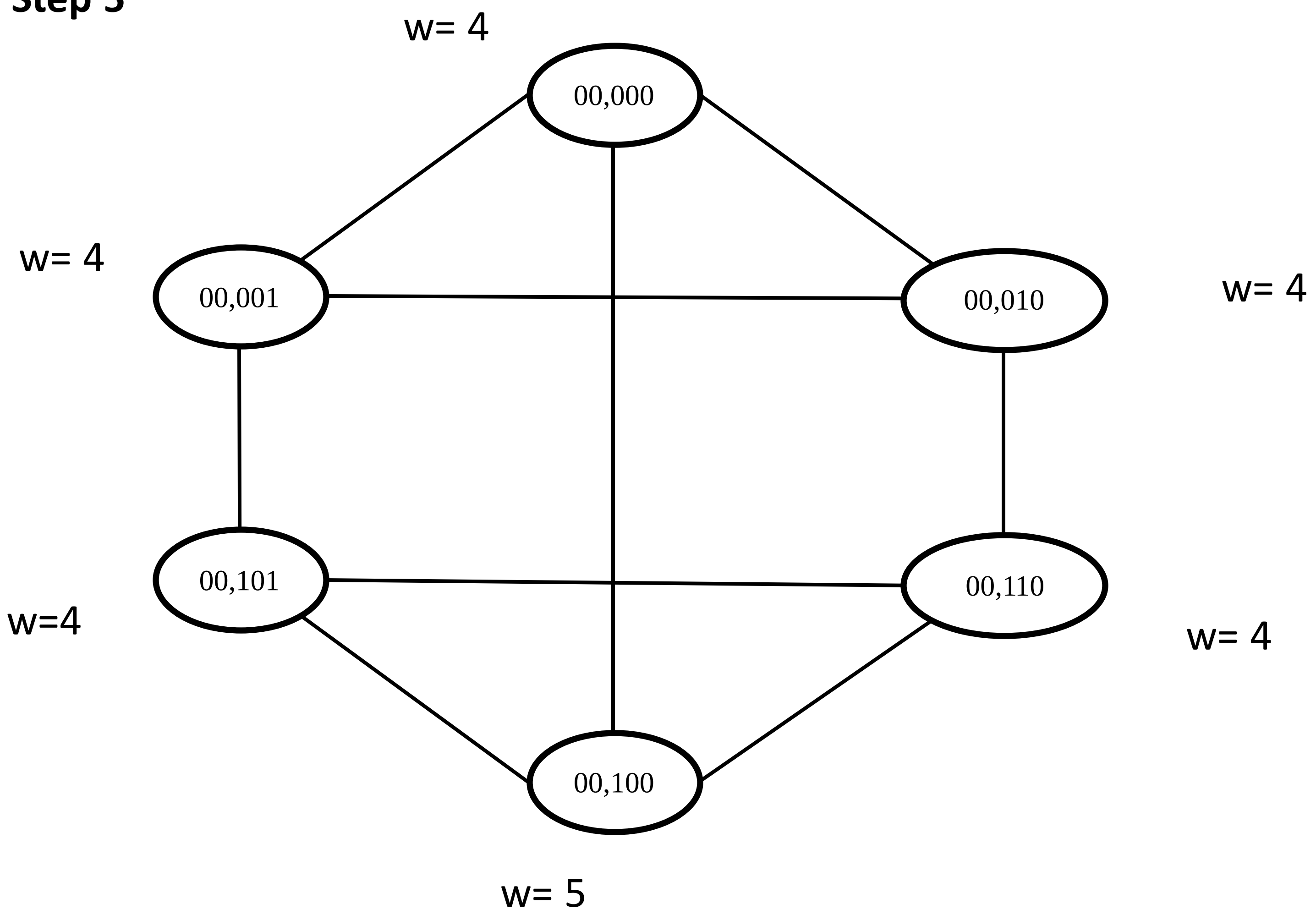
Each node calculates the average weight of the two nodes:
= floor[(local weight + neighbor weight)/2]

Example for tracing algorithm A – Phase1:

Step 5.b: Each triangle node exchanges excess load with directly connected node of the other triangle



Example for tracing algorithm A – Phase1: **After Step 5**



Example for tracing algorithm A – Phase2:

Applying the DEM algorithm to balance load between different dimension of the HHC.

• Each pair of nodes that differs only in the Jth bit position of its sub-group address exchanges its weights along the dimension J+1 and calculate average weight: Average = floor[((w_x + w_y)) / 2].

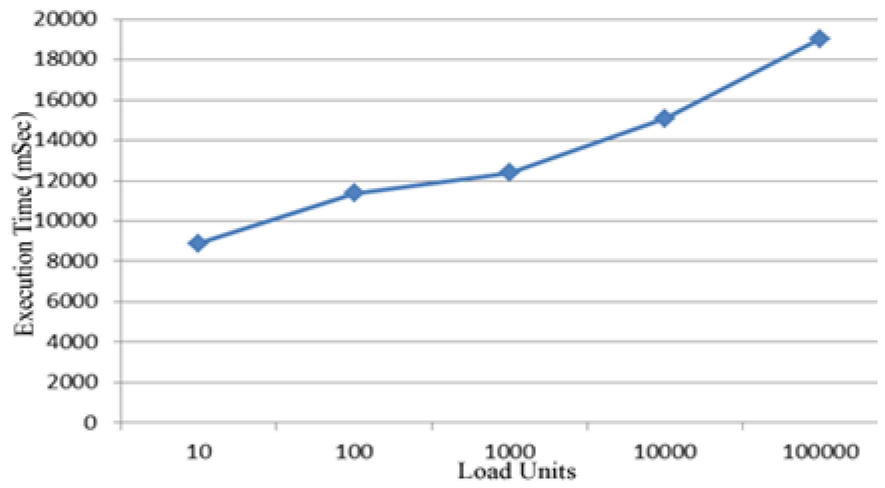
• The node with excess load would send excess load to its neighbor and the other node will receive the excess load.

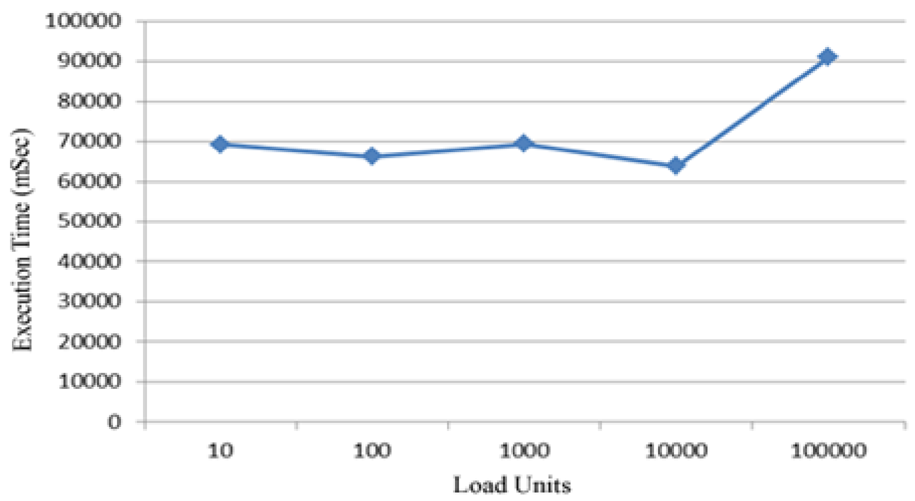
• The operation would look like as if six hyper-cubes are balancing at the same time.

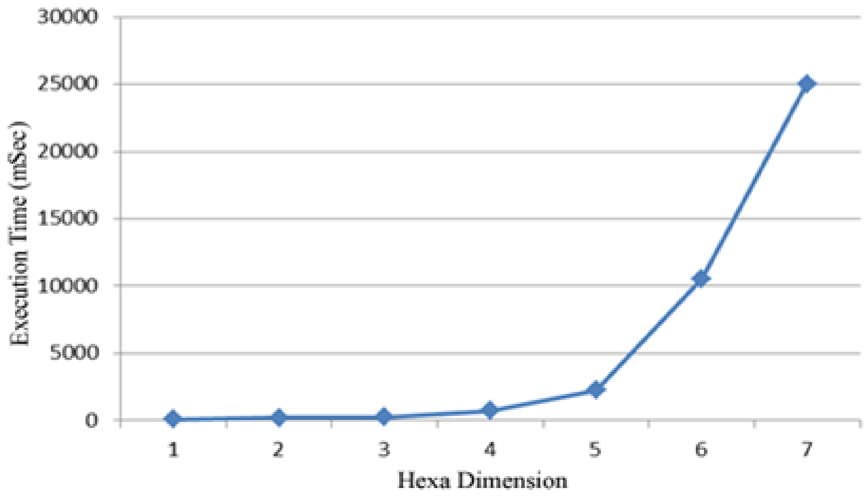
Analytical results

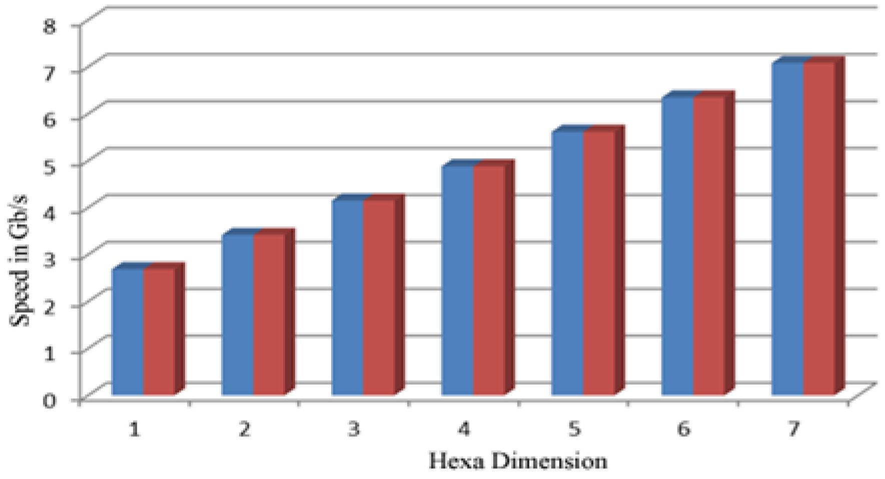
Metric (for Algorithm A)	Value
Execution time	$M + (M/6) * (1 - (1/2)^{dh-1}))$ $\approx O(M+M/6) = O(7M/6)$
Accuracy	$1 + d_h$
Communication cost (max of any node)	$3d_h + 8$
Total communication steps (whole network)	$(2^{d_h-1}) * (18d_h + 29)$
Speed	$(3d_h + 8)$ *250 Mb/s

Experimental Results





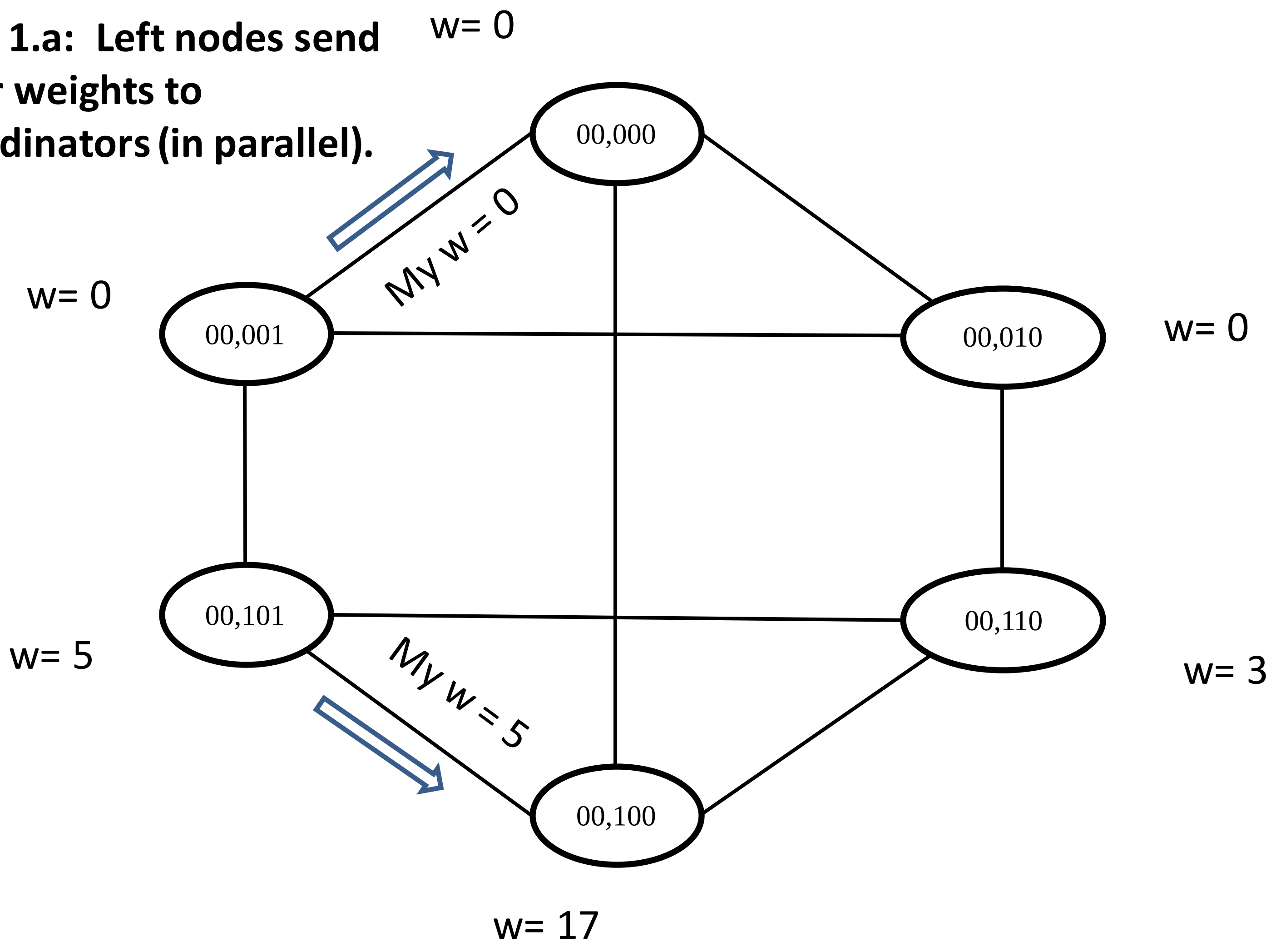




Algorithm B

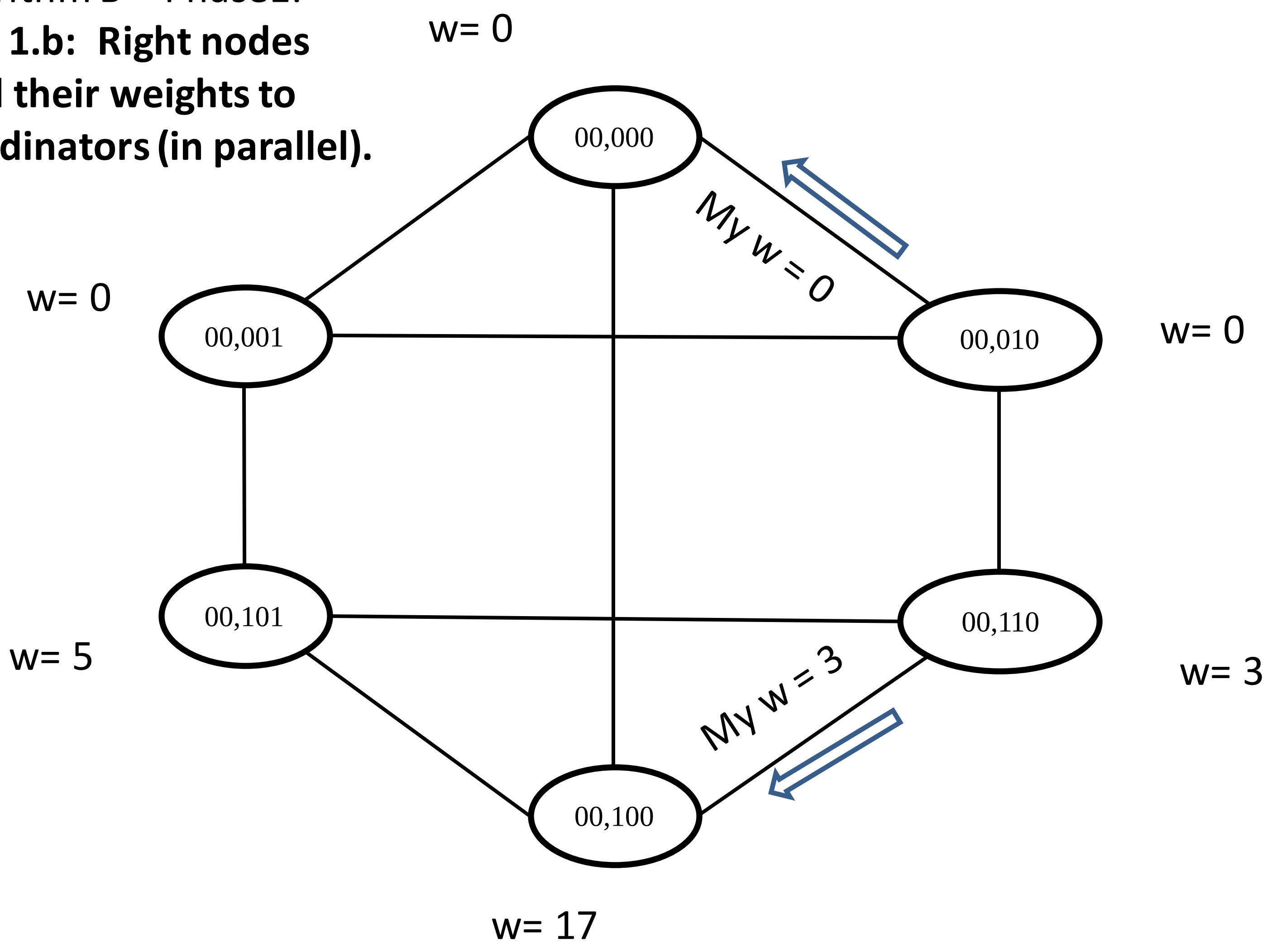
Example for tracing algorithm B – Phase1:

Step 1.a: Left nodes send their weights to coordinators (in parallel).



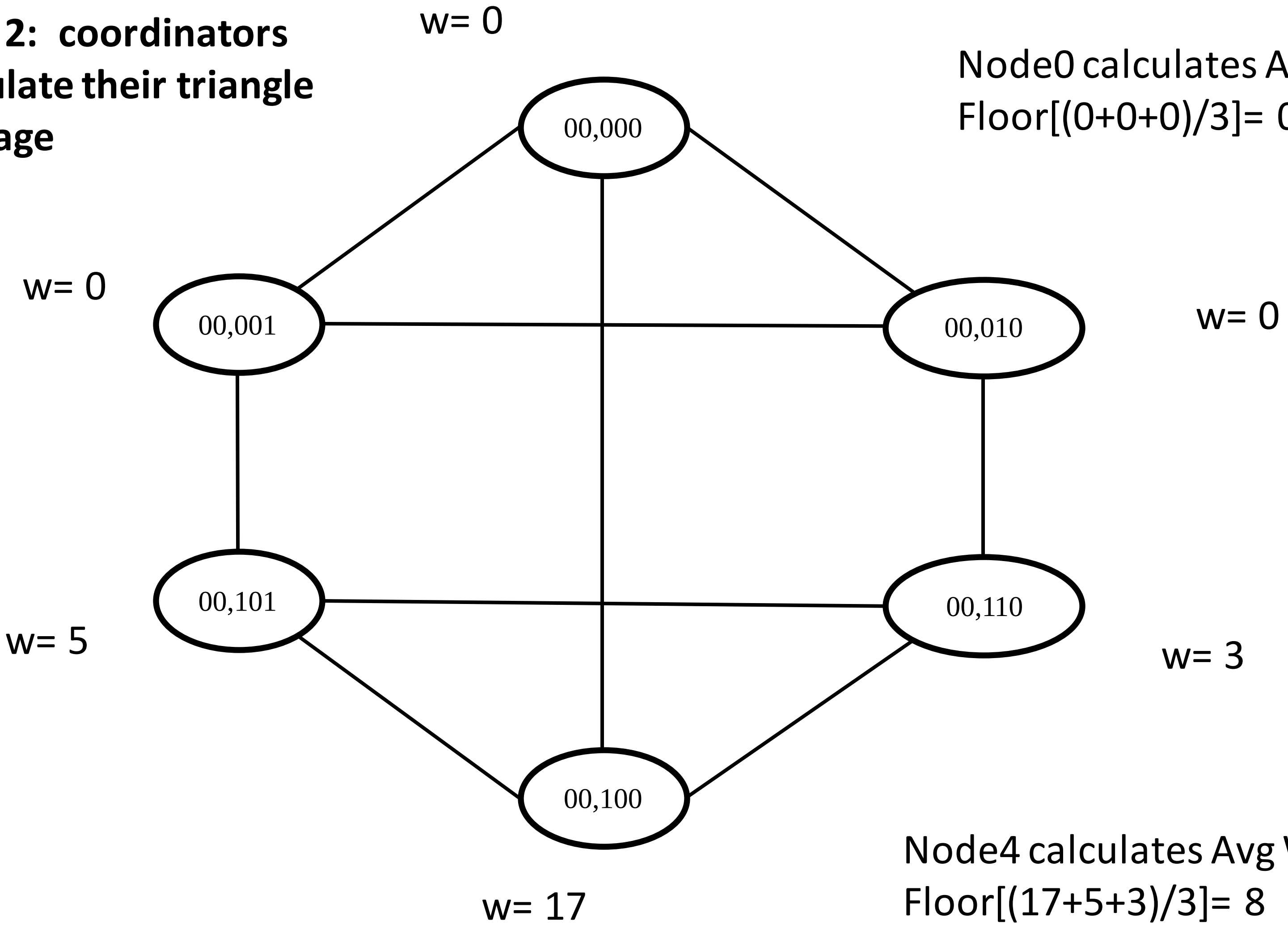
Example for tracing algorithm B – Phase1:

Step 1.b: Right nodes send their weights to coordinators (in parallel).



Example for tracing algorithm B – Phase1:

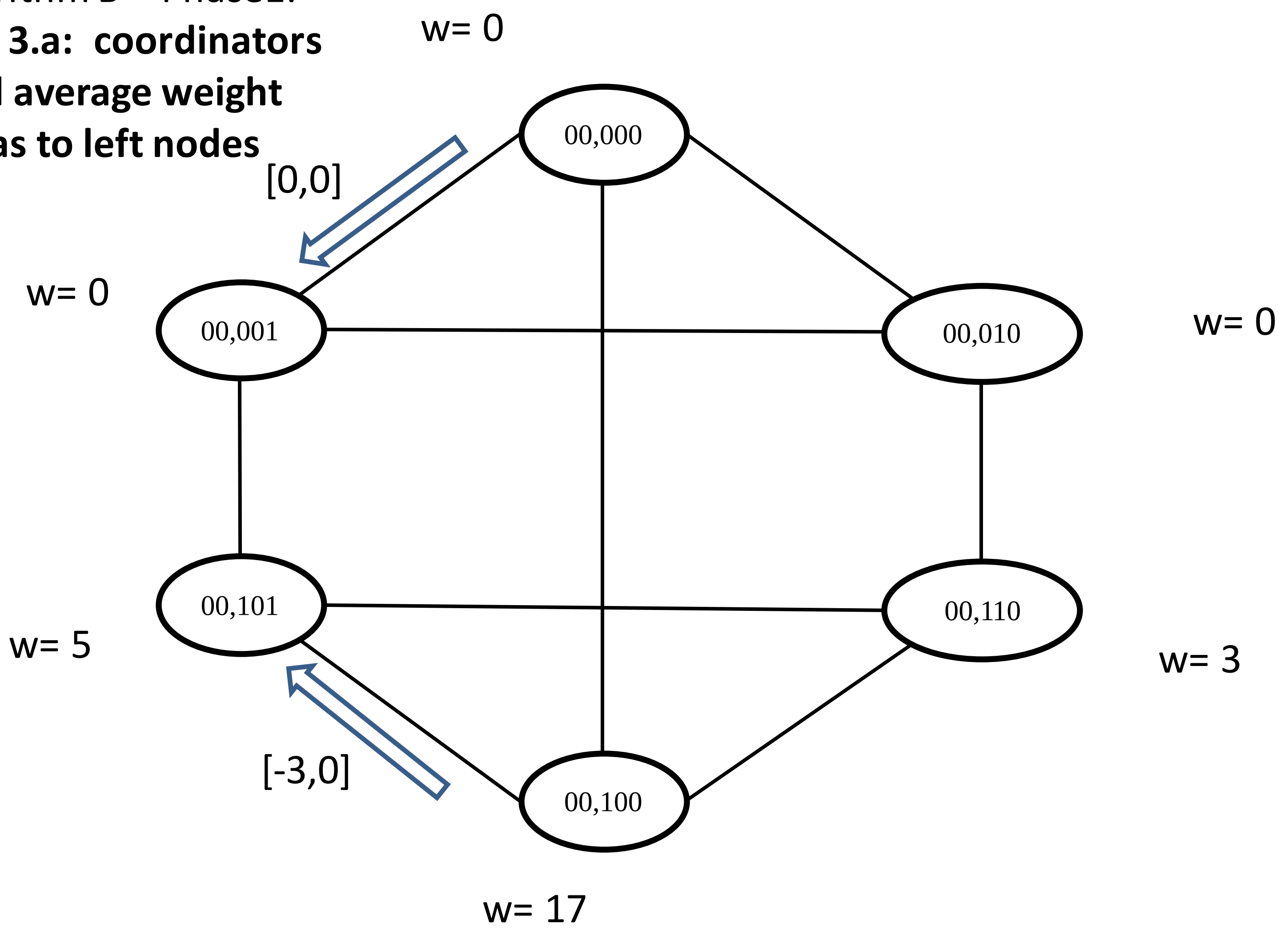
Step 2: coordinators calculate their triangle average



Node0 calculates Avg W = Floor[(0+0+0)/3]= 0

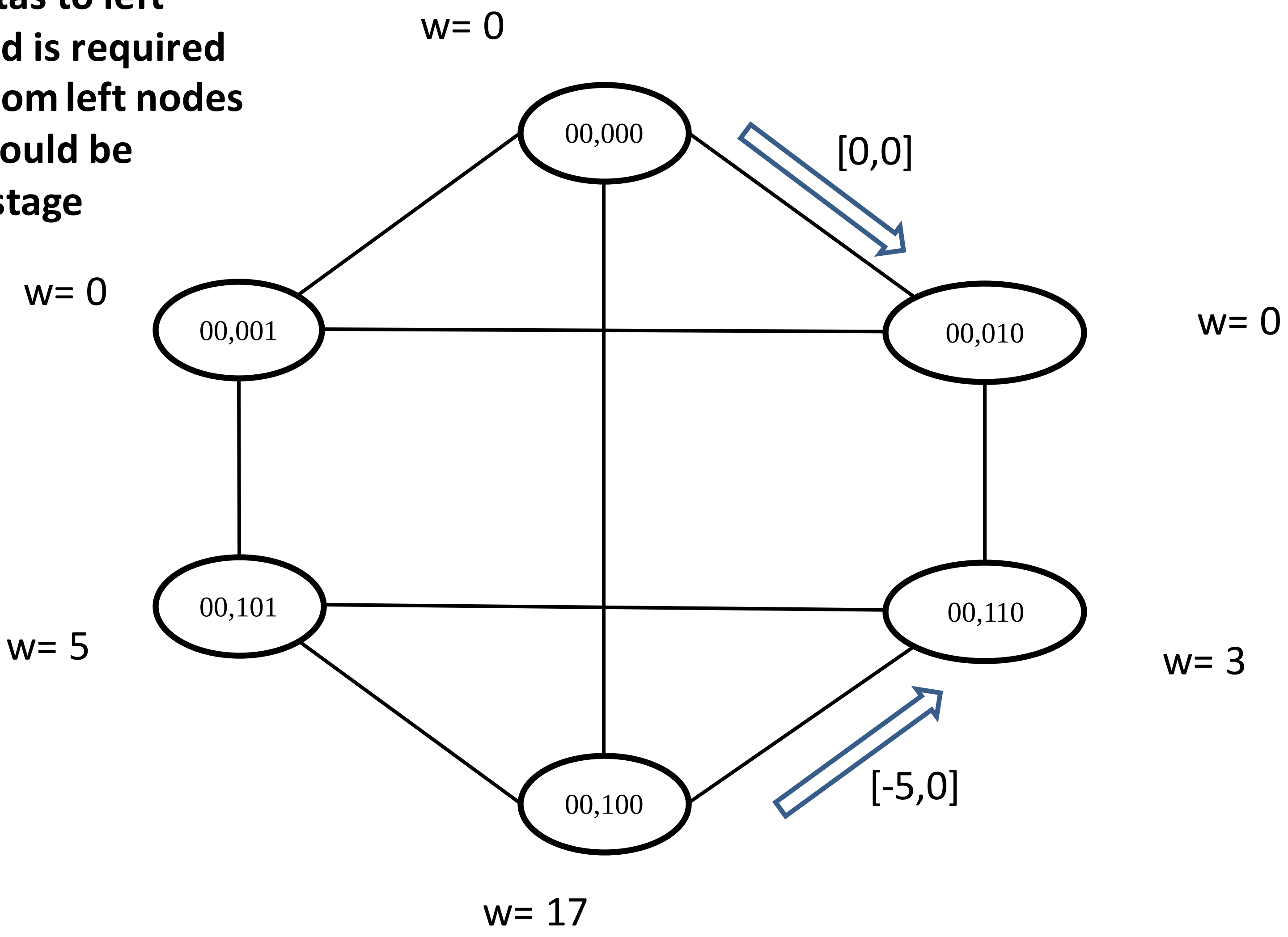
Node4 calculates Avg W = Floor[(17+5+3)/3]= 8

Example for tracing algorithm B – Phase1:
Step 3.a: coordinators send average weight deltas to left nodes



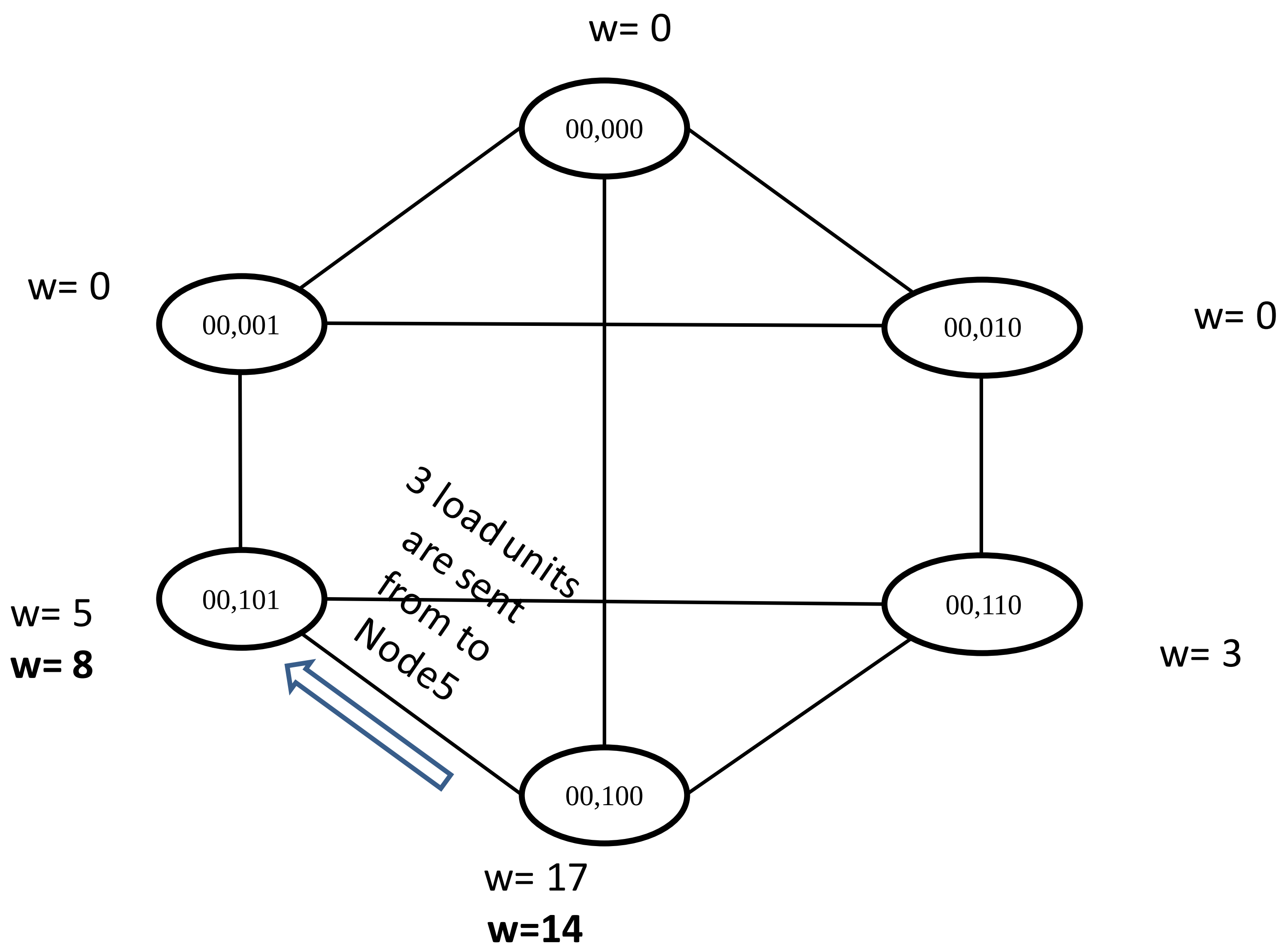
Example for tracing algorithm A – Phase1:

Step 3.b: coordinators send average weight deltas to left nodes. If excess load is required to be transferred from left nodes to coordinator, it would be transferred at this stage



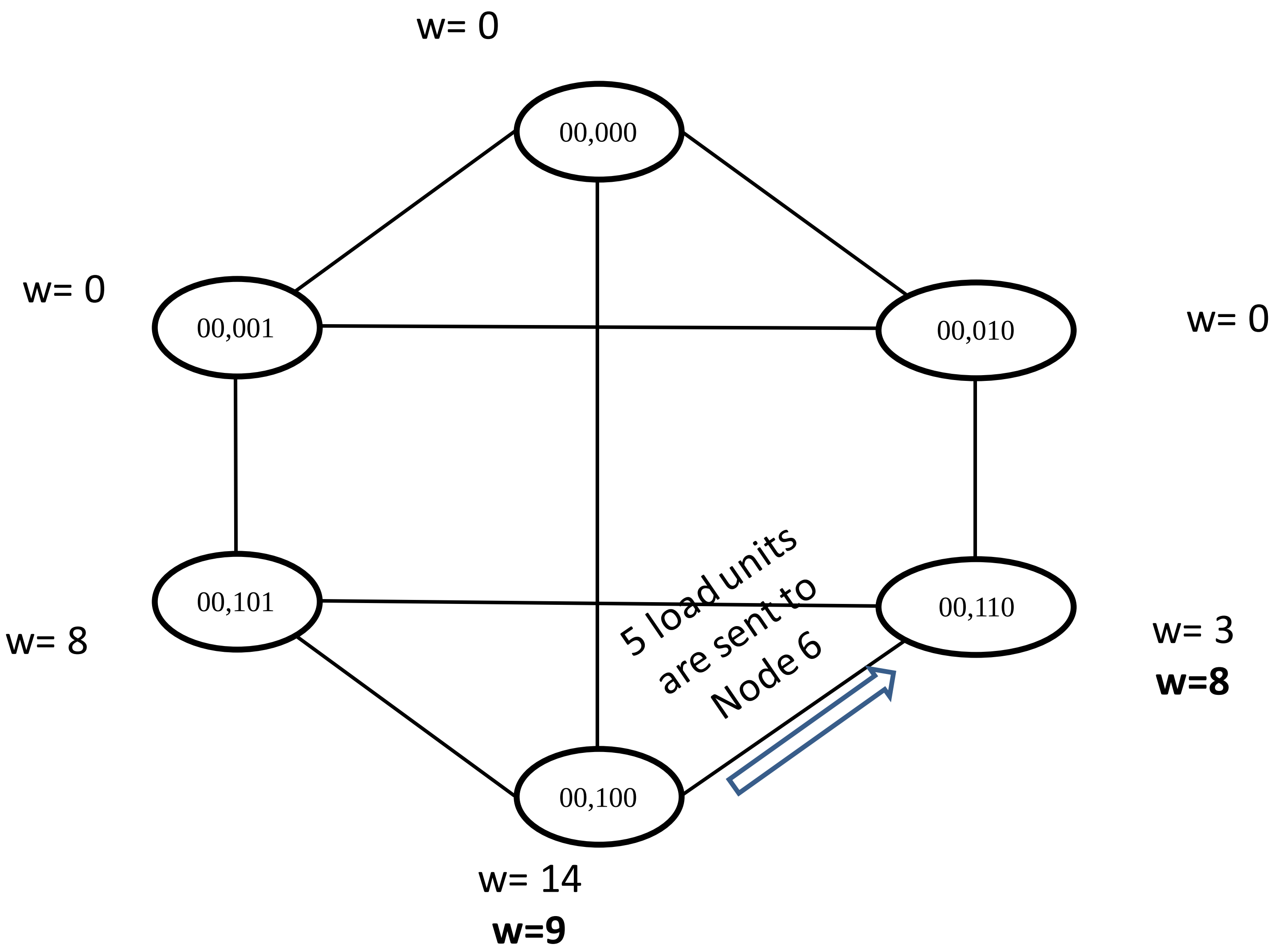
Example for tracing algorithm A – Phase1:

Step 4.a: coordinator sends excess load to left node. If excess load is required to be transferred from right nodes to coordinator, it would be transferred at this stage. If excess load is required to be transferred from left nodes to right nodes, it would be transferred at this stage

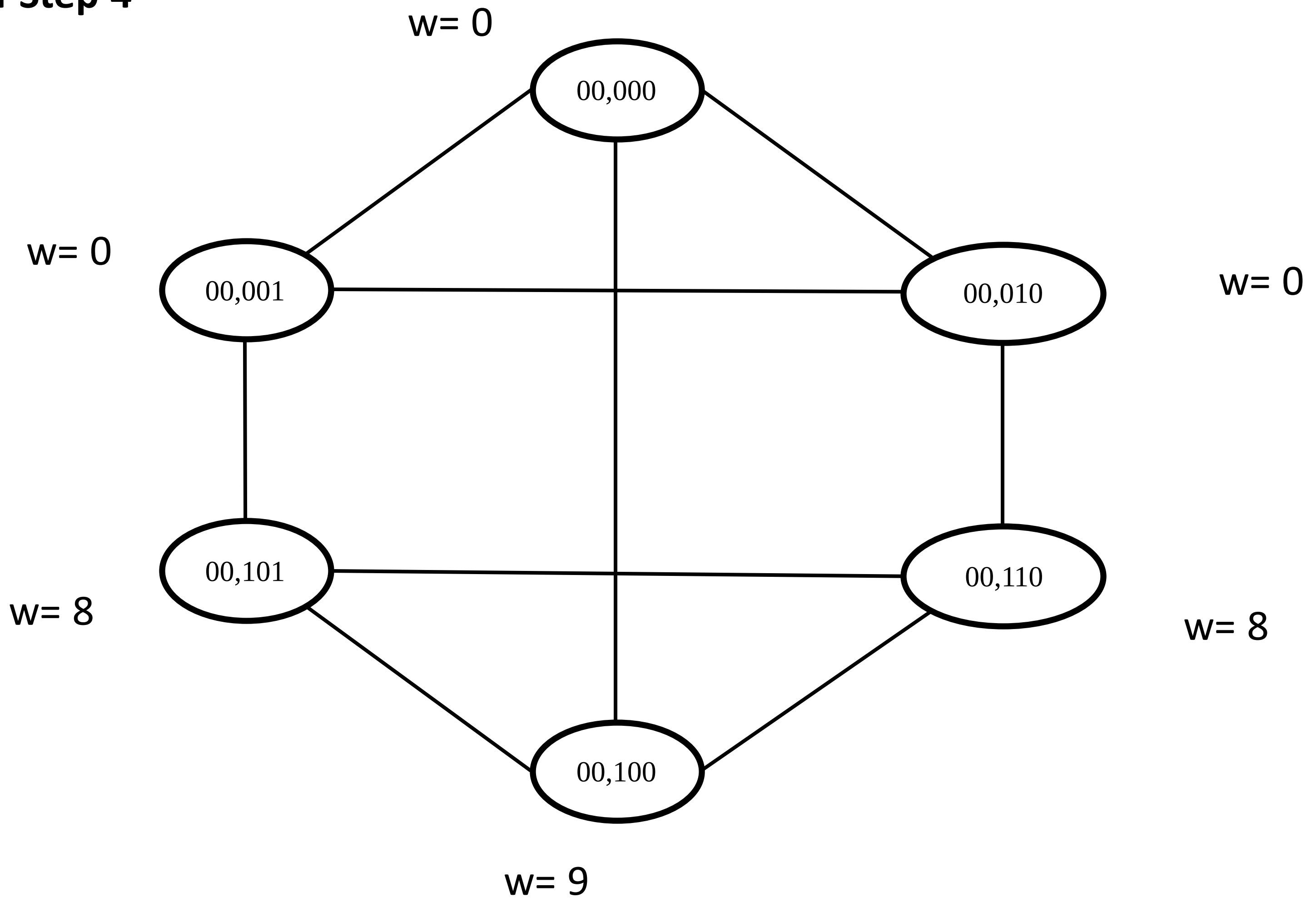


Example for tracing algorithm A – Phase1:

Step 4.b: coordinator sends excess load to right node. If excess load is required to be transferred from right nodes to left nodes, it would be transferred at this stage.

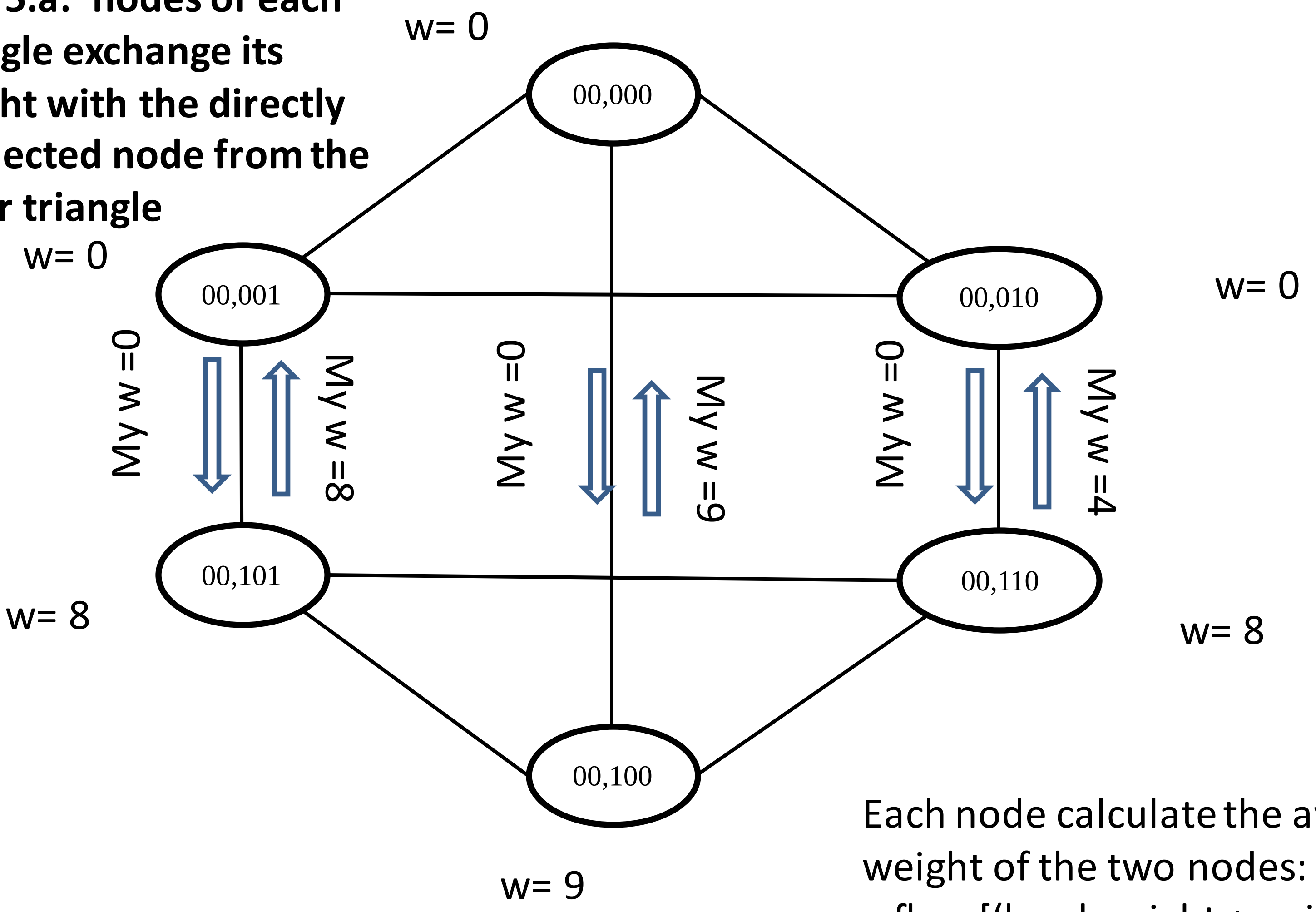


Example for tracing algorithm A – Phase1:
After Step 4



Example for tracing algorithm A – Phase1:

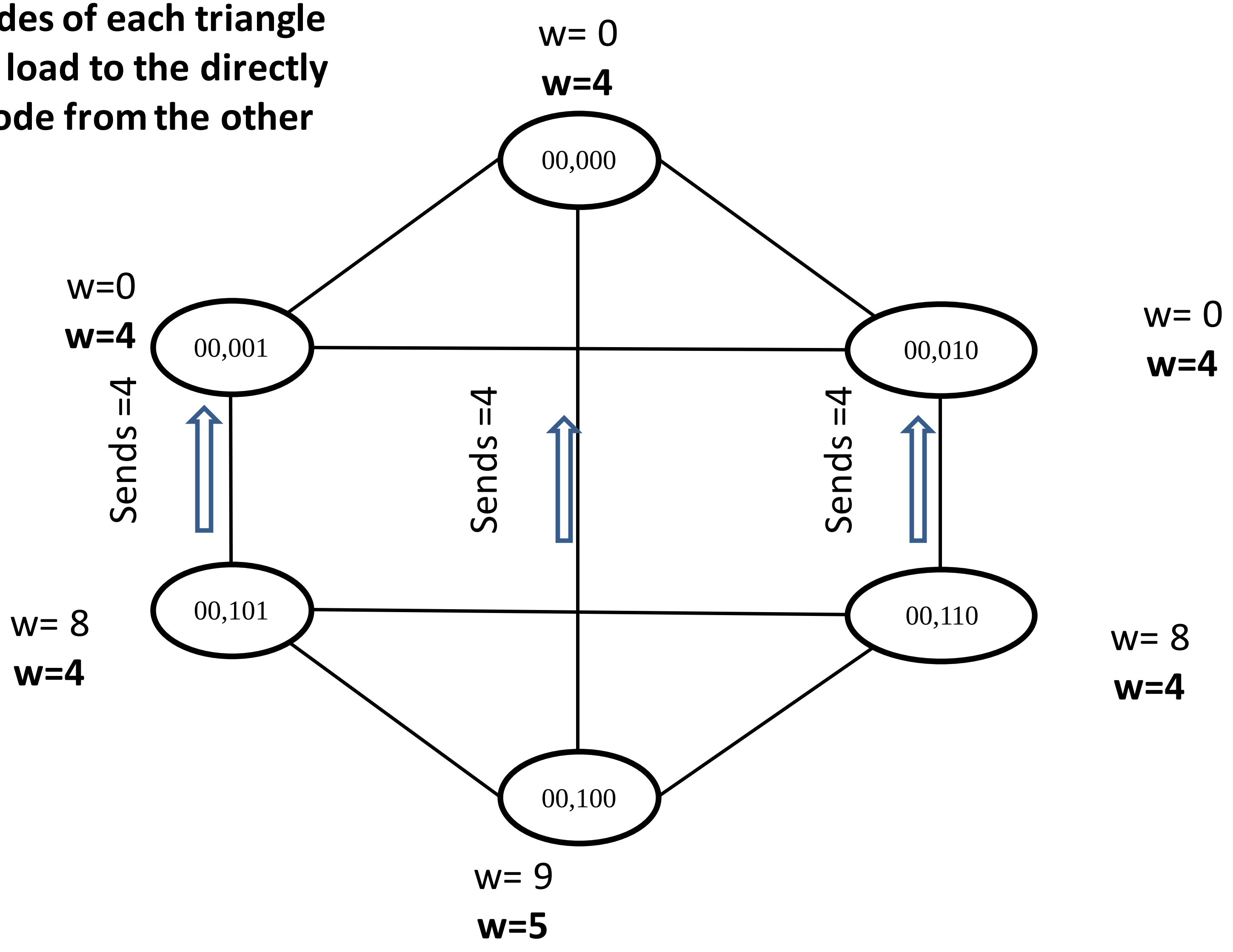
Step 5.a: nodes of each triangle exchange its weight with the directly connected node from the other triangle



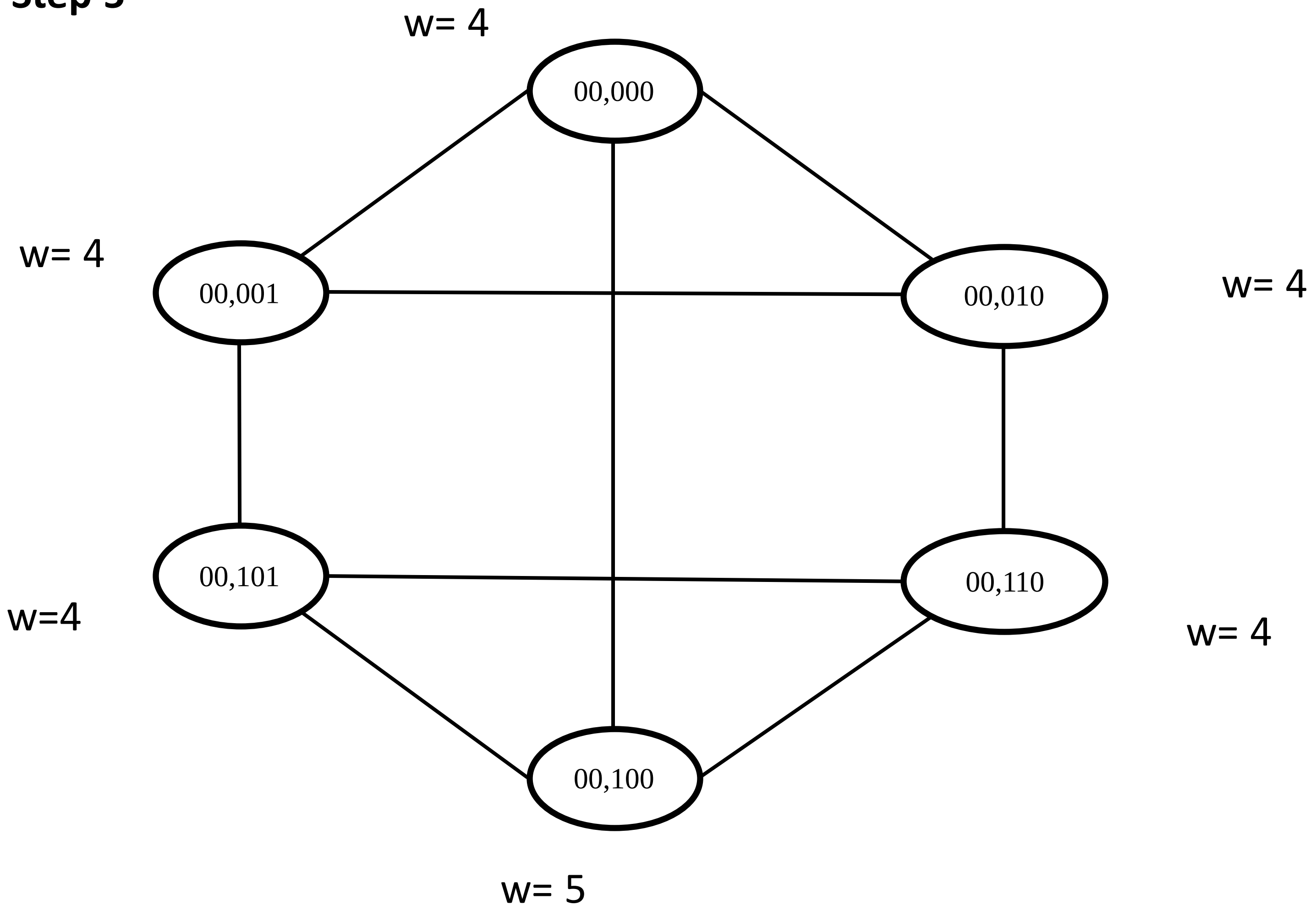
Each node calculate the average weight of the two nodes:
= floor[(local weight + neighbor weight)/2]

Example for tracing algorithm A – Phase1:

Step 5.b: nodes of each triangle sends excess load to the directly connected node from the other triangle



Example for tracing algorithm A – Phase1:
After Step 5



Example for tracing algorithm B – Phase2:
Applying the DEM algorithm to balance load between different dimension of the HHC.

• Each pair of nodes that differs only in the J^{th} bit position of its sub-group address exchanges its weights along the dimension J+1 and calculate average weight: Average = floor[((w_x + w_y)) / 2].

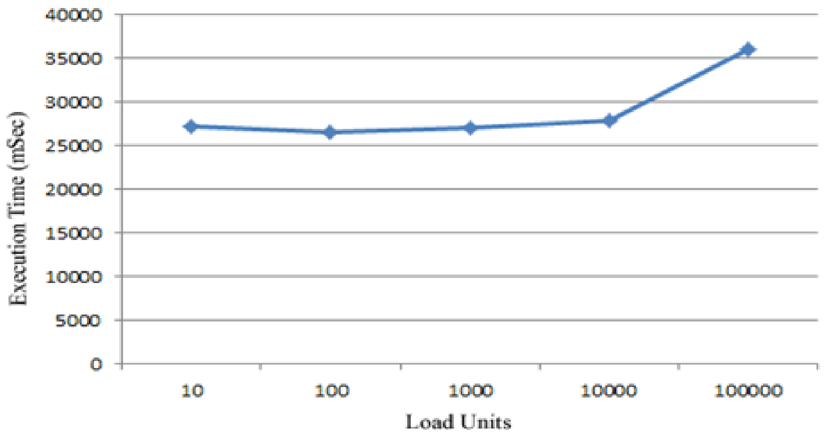
• The node with excess load would send excess load to its neighbor and the other node will receive the excess load.

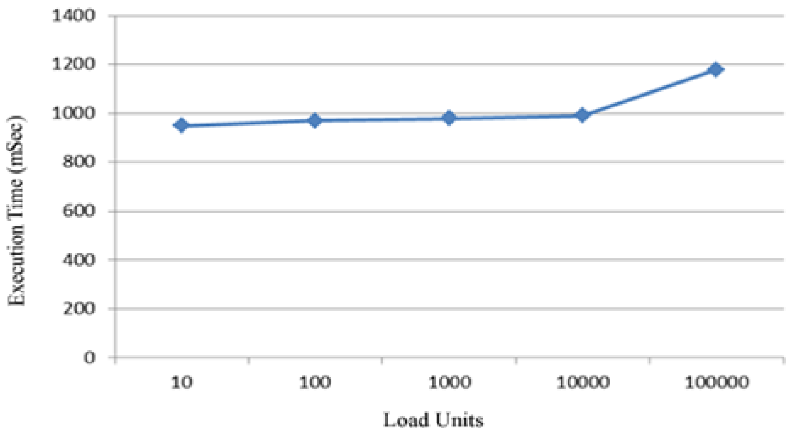
• The operation would look like as if six hyper-cubes are balancing at the same time.

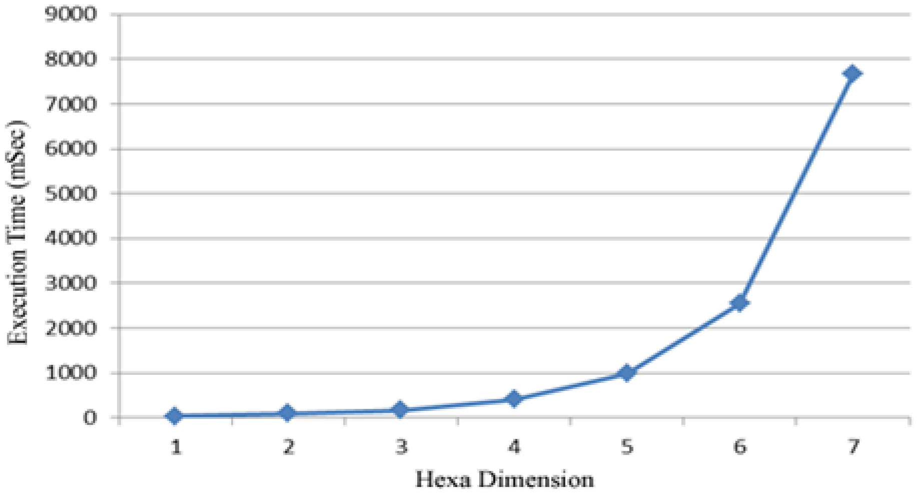
Analytical results

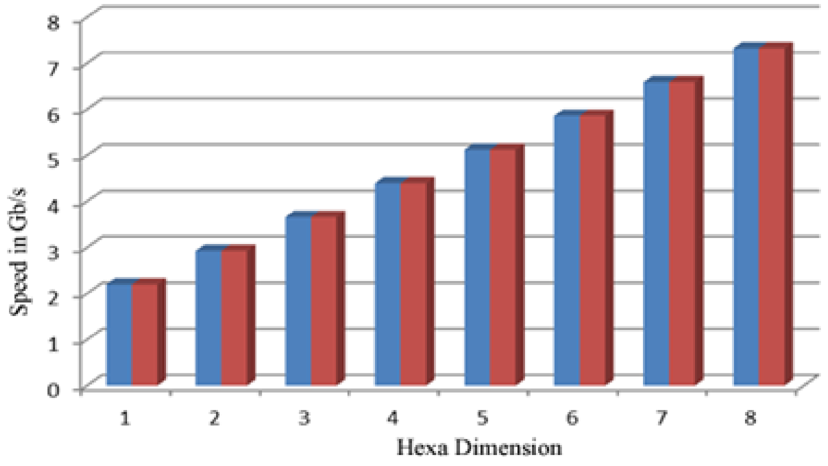
Metric (for Algorithm B)	Value
Execution time	$(5M/6) + (M/6) * (1 - (1/2)^{dh-1})) \approx O(5M/6 + M/6) = O(M)$
Accuracy	1 + dh
Communication cost (max of any node)	3dh + 6
Total communication steps (whole network)	(2dh-1) * (18dh + 24)
Speed	(3dh + 6) *250 Mb/s

Experimental Results

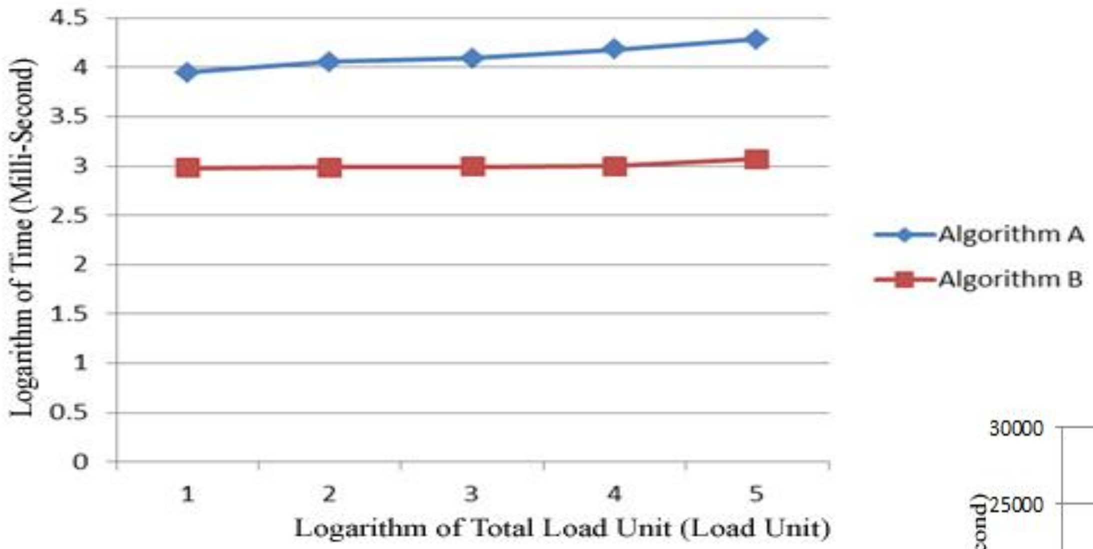


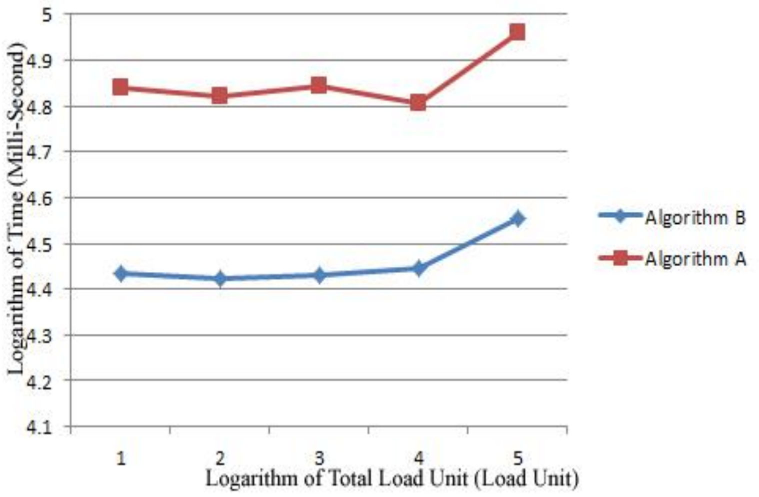


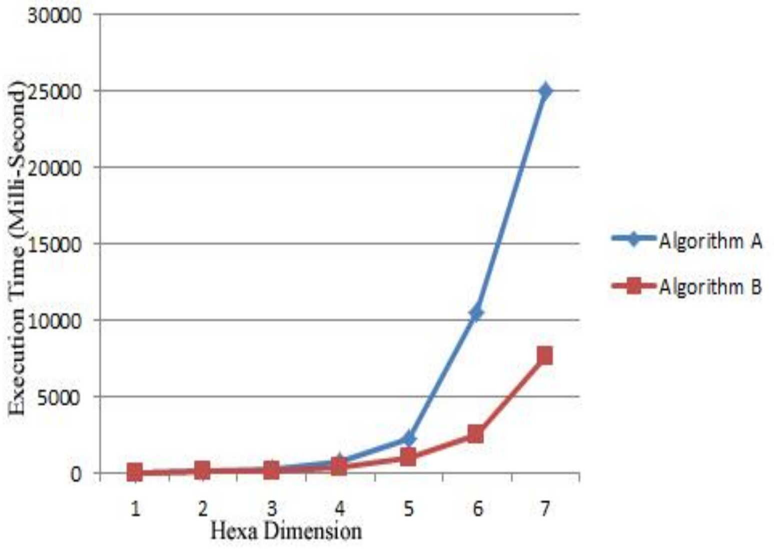




Comparison Between Algorithms A and B



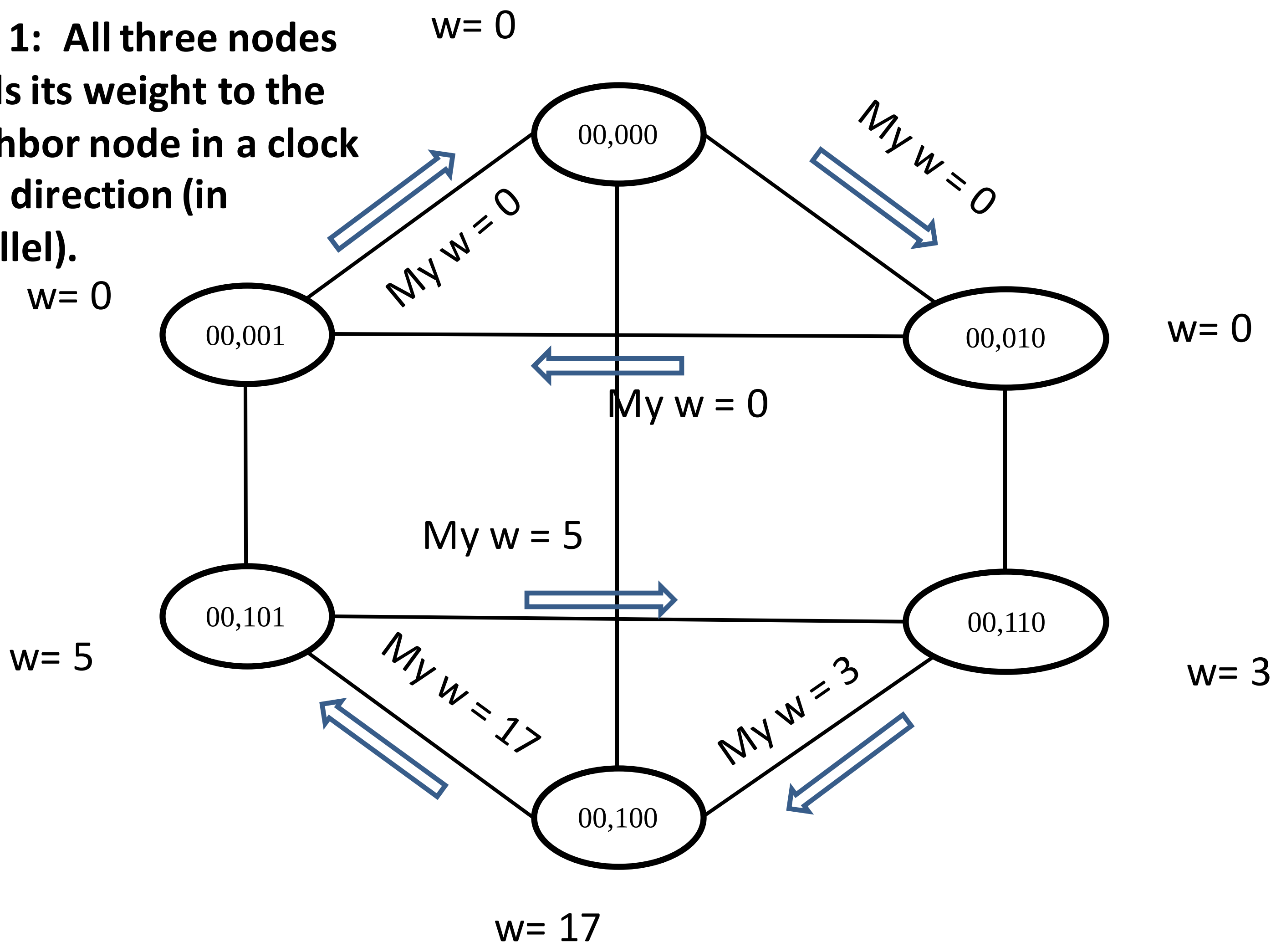




Algorithm C

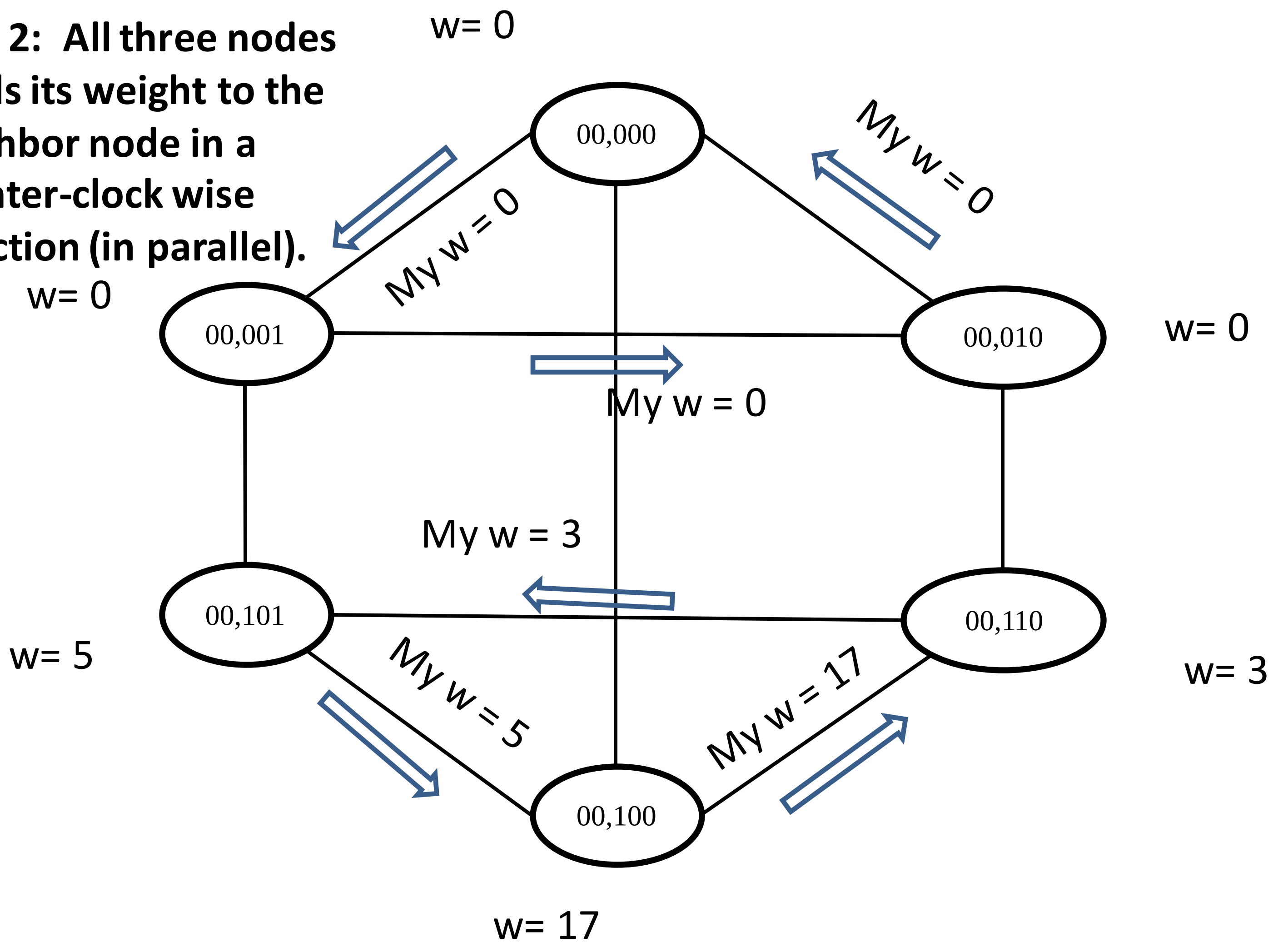
Example for tracing algorithm C – Phase1:

Step 1: All three nodes sends its weight to the neighbor node in a clock wise direction (in parallel).



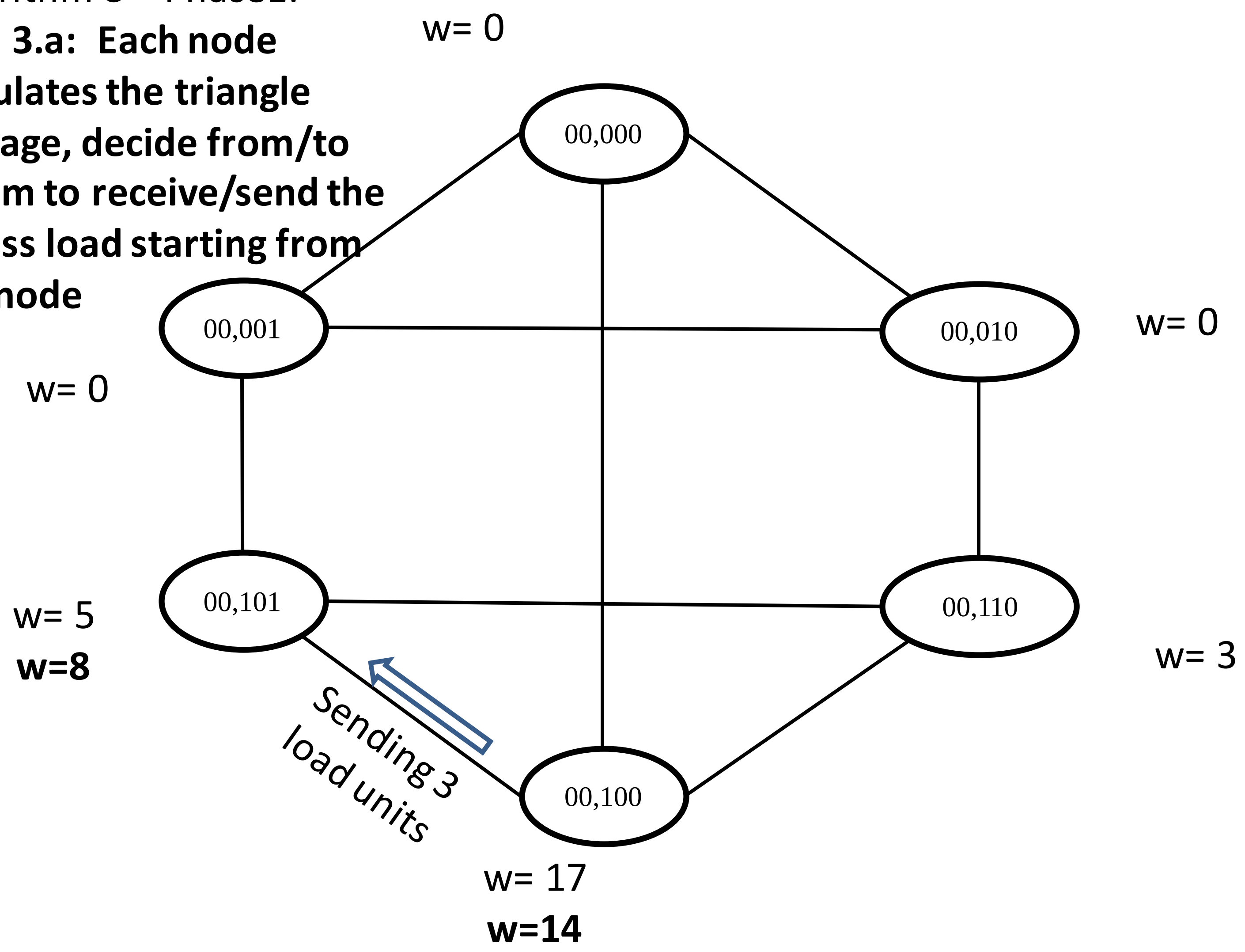
Example for tracing algorithm C – Phase1:

Step 2: All three nodes sends its weight to the neighbor node in a counter-clock wise direction (in parallel).

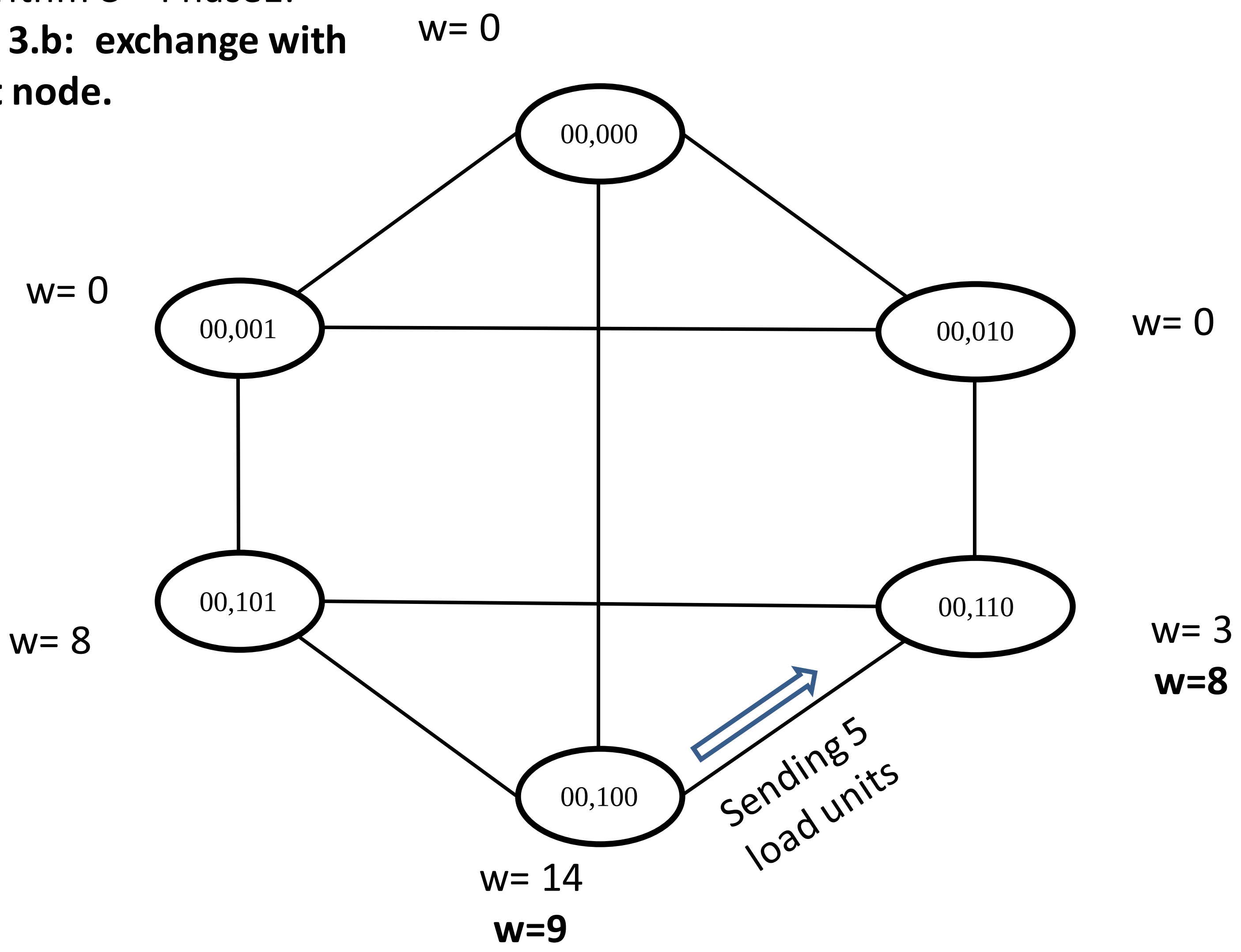


Example for tracing algorithm C – Phase1:

Step 3.a: Each node calculates the triangle average, decide from/to whom to receive/send the excess load starting from left node

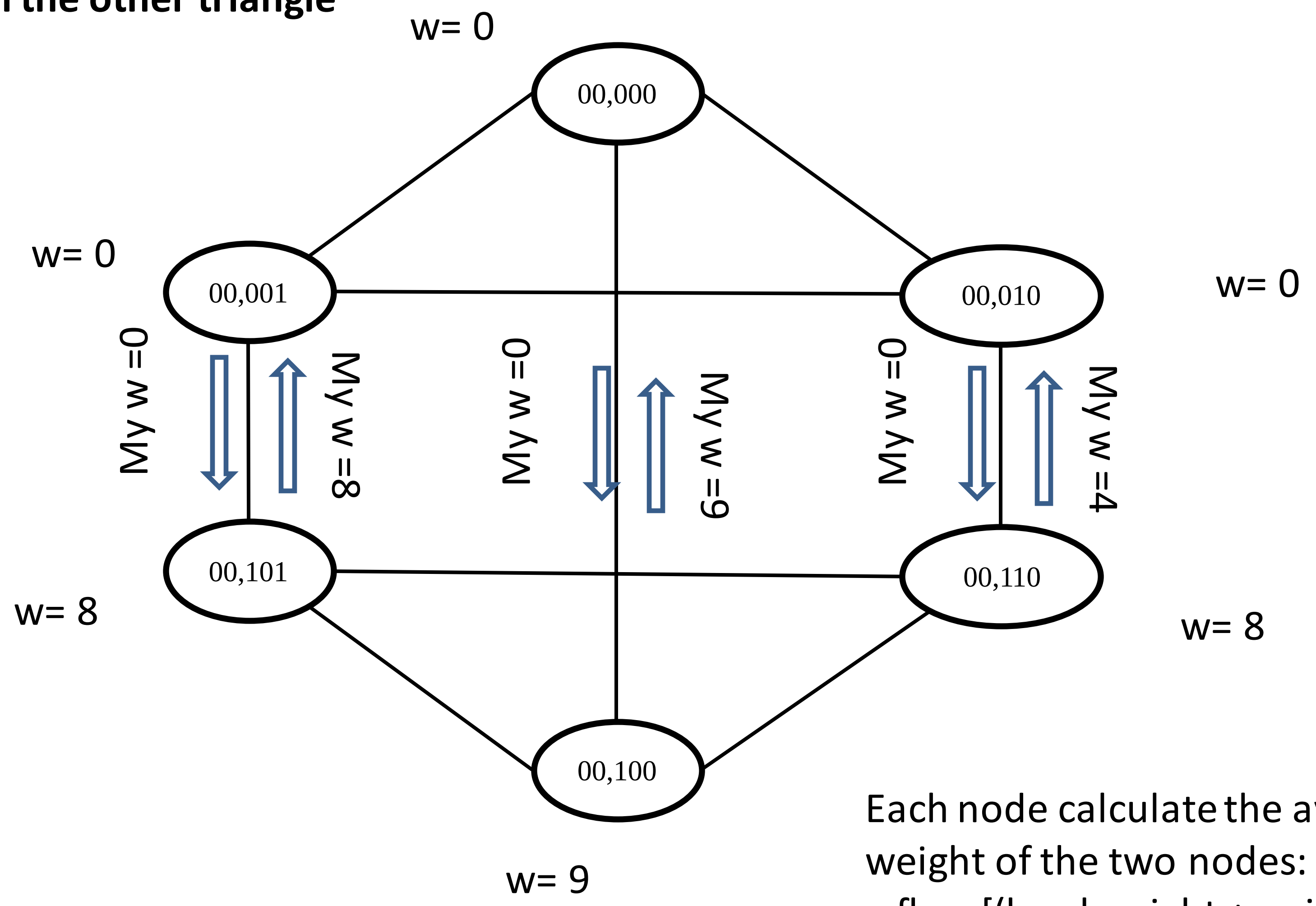


Example for tracing algorithm C – Phase1:
Step 3.b: exchange with right node.



Example for tracing algorithm A – Phase1:

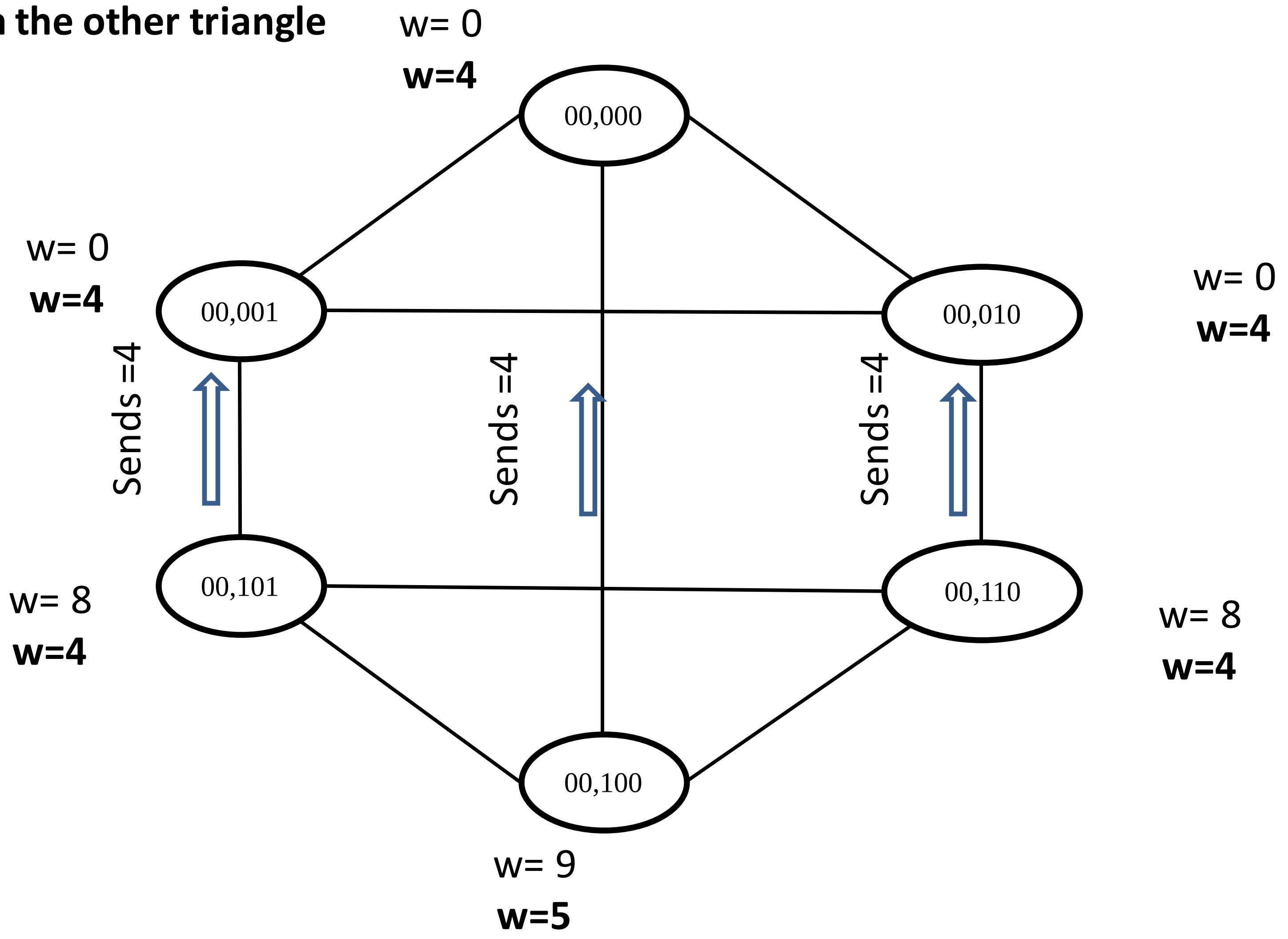
Step 4.a: each triangle node exchanges its weight with the directly connected node from the other triangle



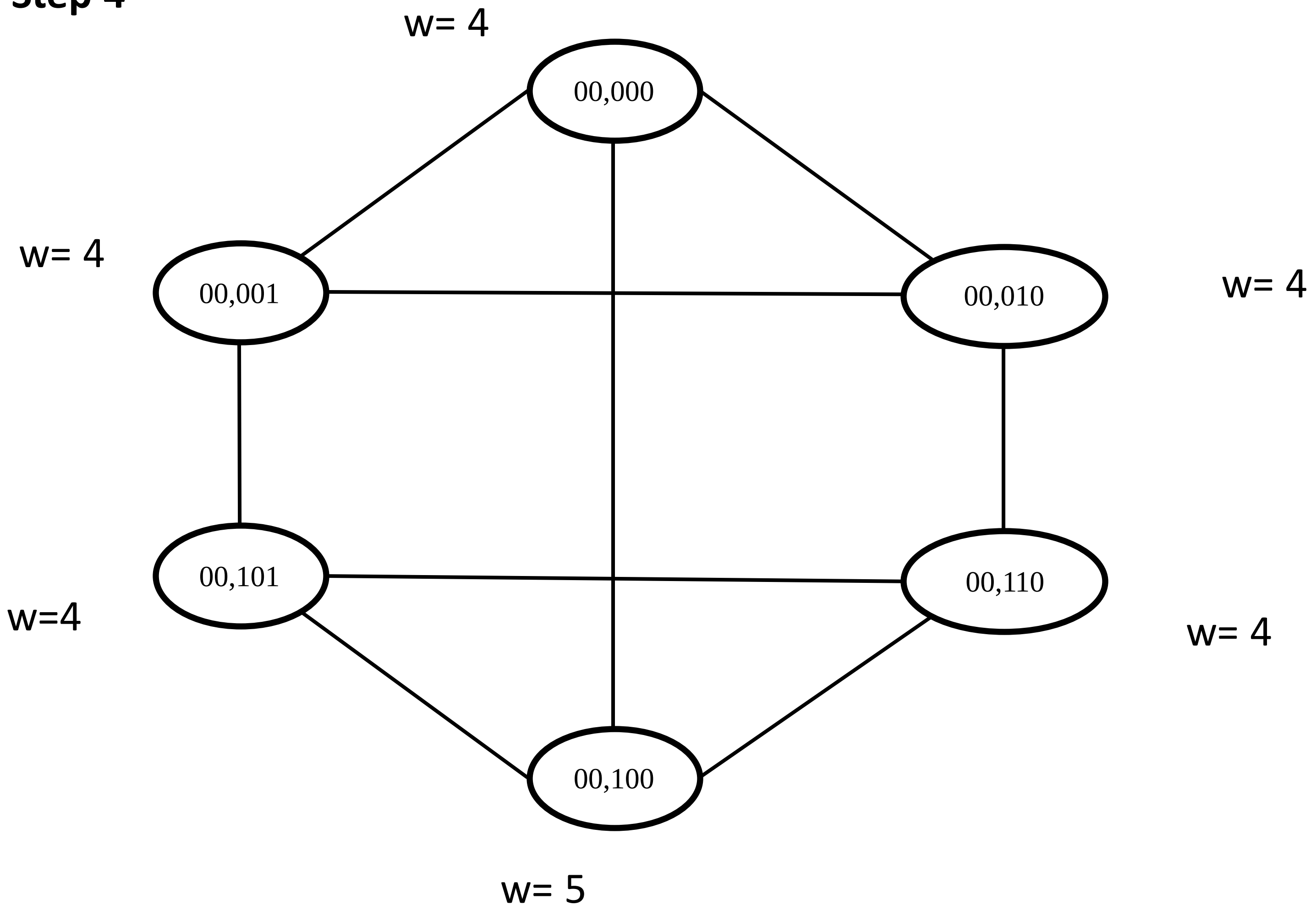
Each node calculate the average weight of the two nodes:
= floor[(local weight + neighbor weight)/2]

Example for tracing algorithm A – Phase1:

Step 4.b: Each triangle node sends its excess load with the directly connected node from the other triangle



Example for tracing algorithm A – Phase1:
After Step 4



Example for tracing algorithm C – Phase2:

Applying the DEM algorithm to balance load between different dimension of the HHC.

• Each pair of nodes that differs only in the Jth bit position of its sub-group address exchanges its weights along the dimension J+1 and calculate average weight: Average = floor[((w_x + w_y)) / 2].

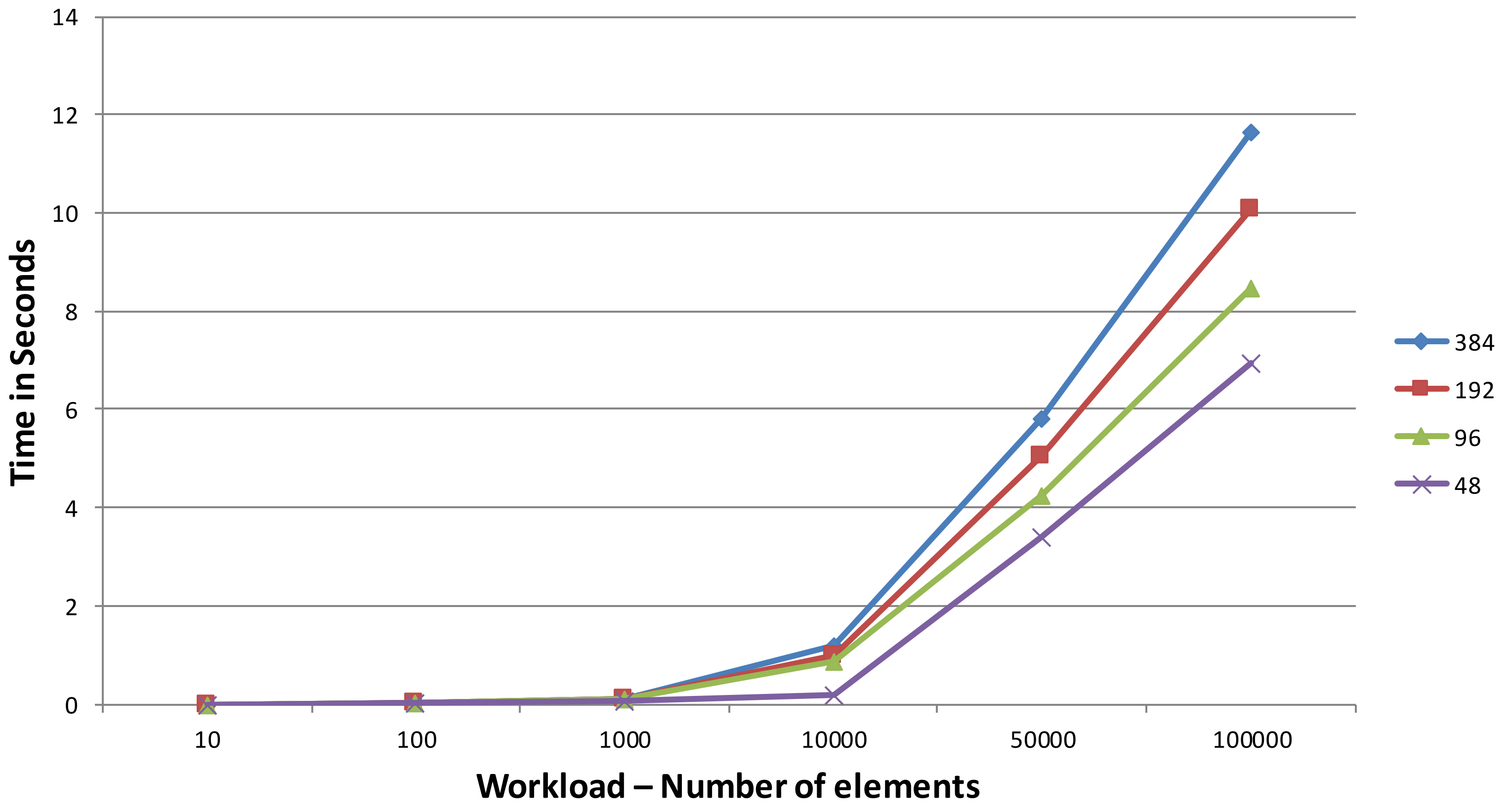
• The node with excess load would send excess load to its neighbor and the other node will receive the excess load.

• The operation would look like as if six hyper-cubes are balancing at the same time.

Analytical results

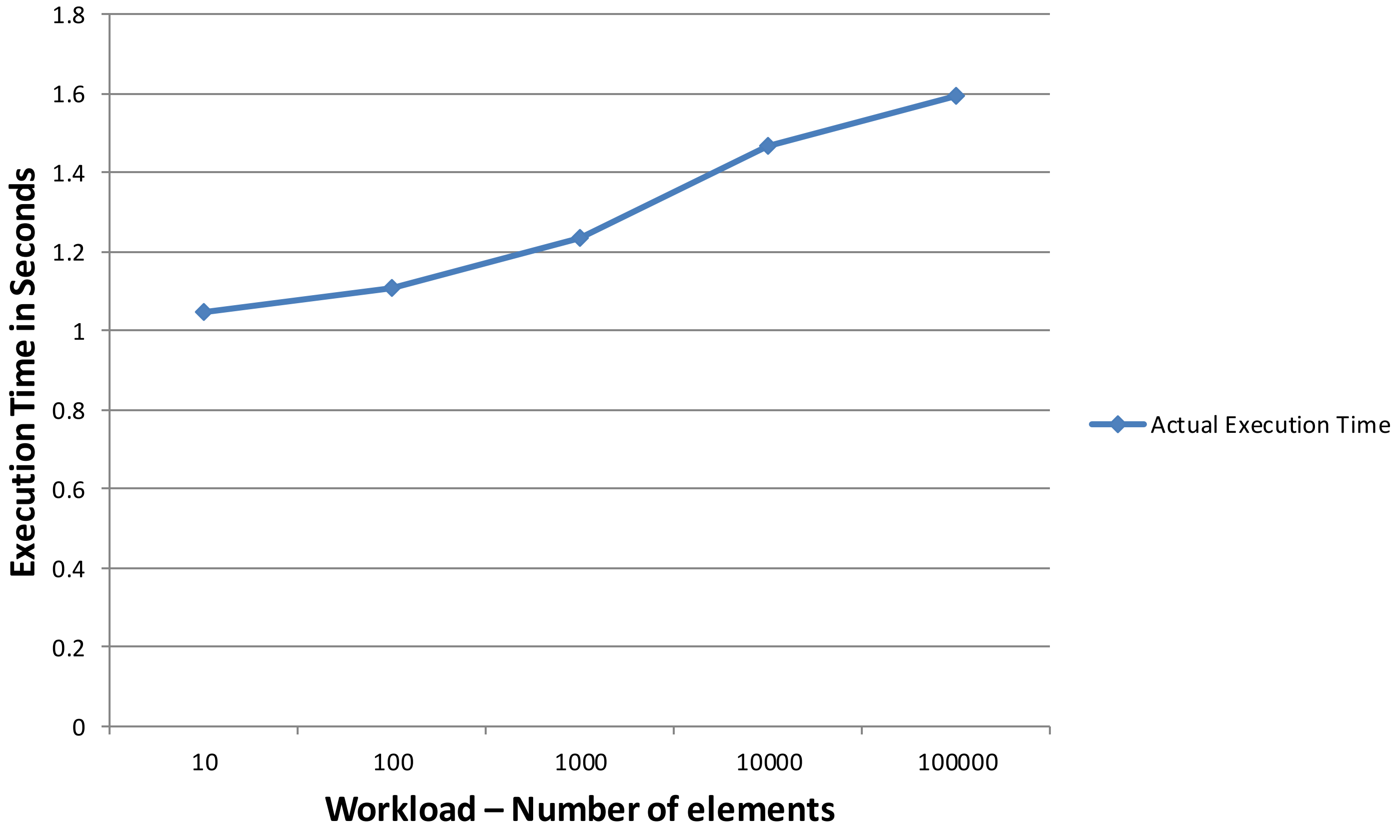
Metric (for Algorithm C)	Value
Execution time	*(5M/6) + (M/6) * (1 – (1/2)*$^{dh-1}$*)) ≈ O(5M/6 + M/6) = O(M)*
Accuracy	$1 + d_h$
Total communication steps (whole network)	*(29 * 2dh-1) + (12 * dh-1 * 2 ^ (dh-1))*
Speed	*((29 * 2dh-1) + (12 * dh-1 * 2 ^ (dh-1)))* *250 Mb/s

Execution Time



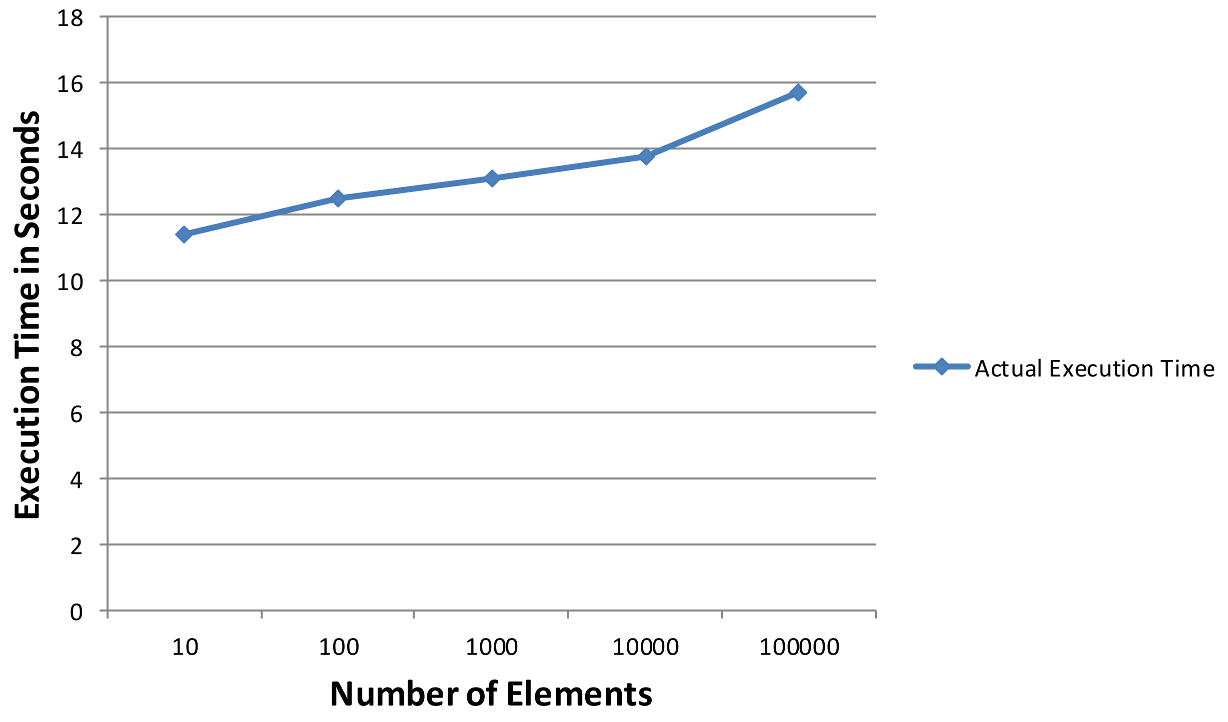
Execution time with different number of processors and different load for Algorithms (C).

Execution Time



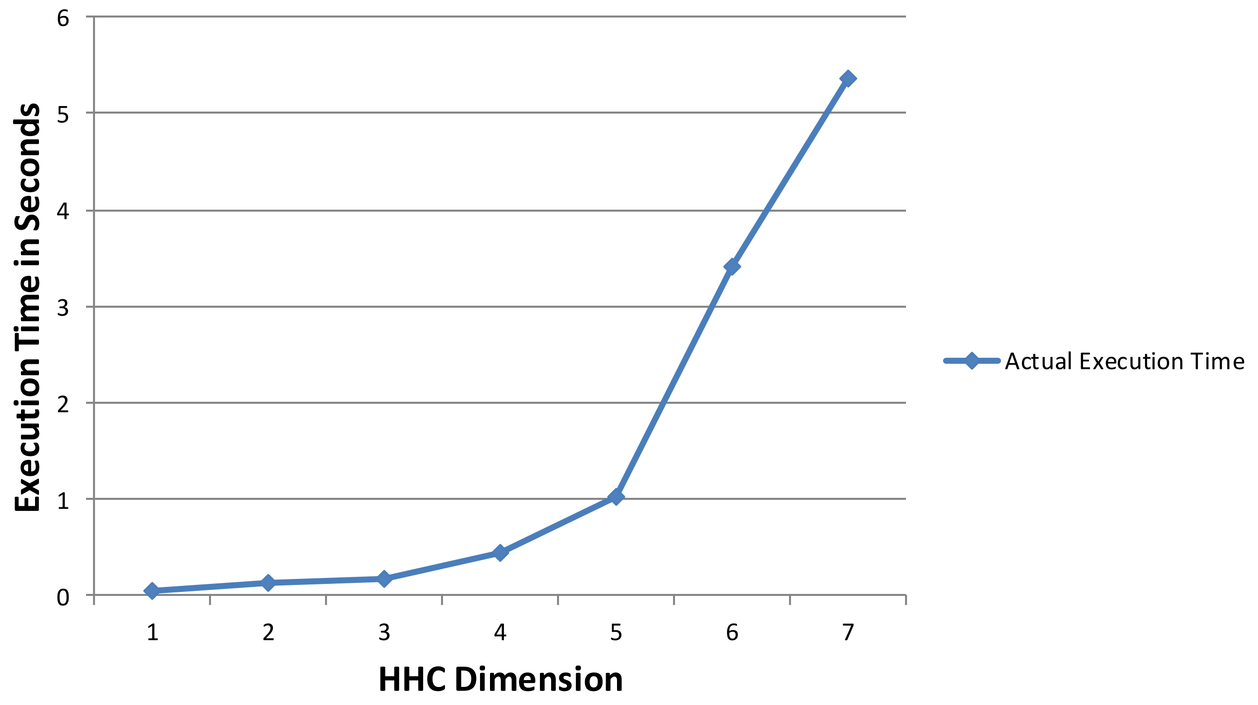
Execution time when the number of processors is 96, load sizes vary between 10 and 100000 for Algorithms (C).

Execution Time



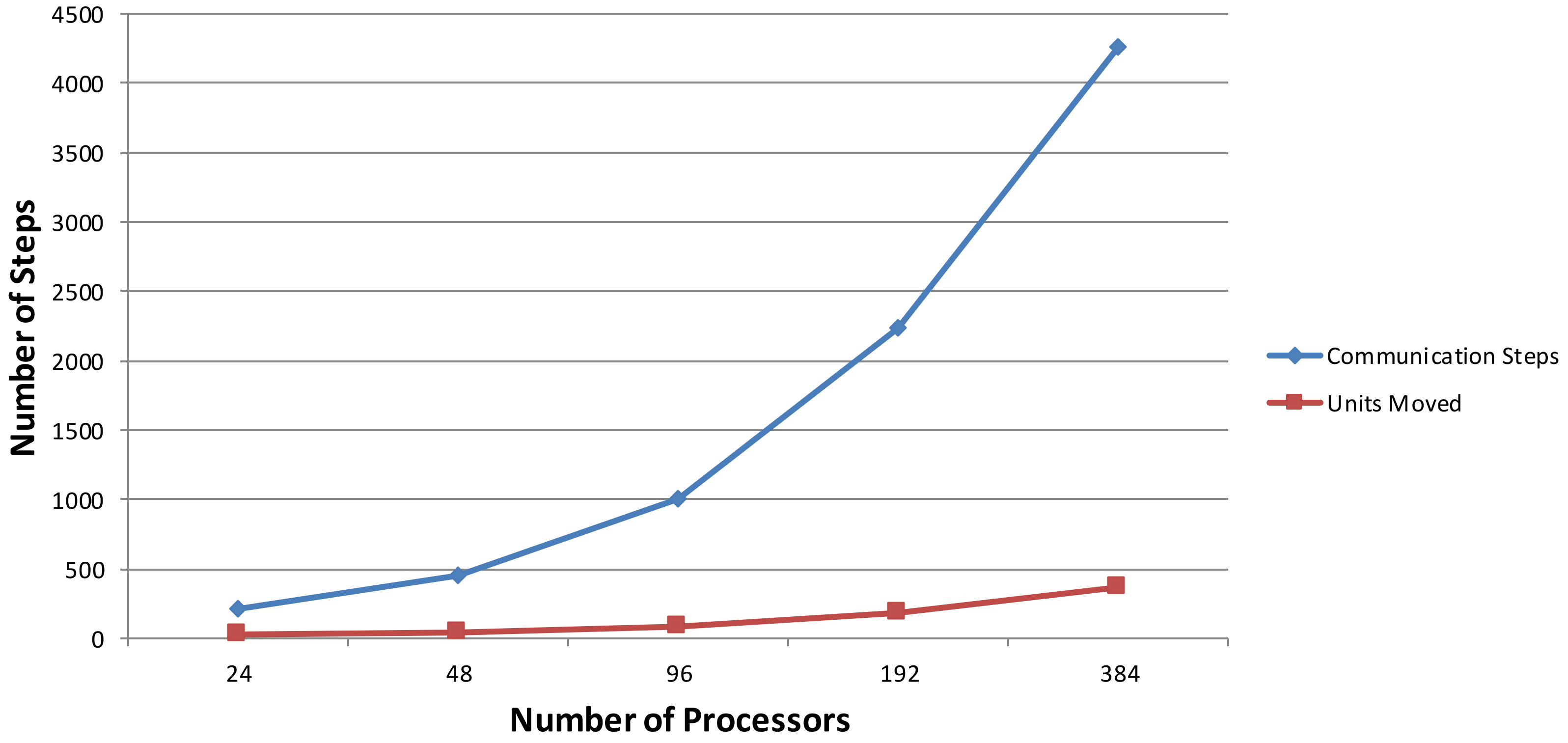
Execution time when the number of processors is 768, load sizes vary between 10 and 100000 for Algorithms (C).

Execution Time



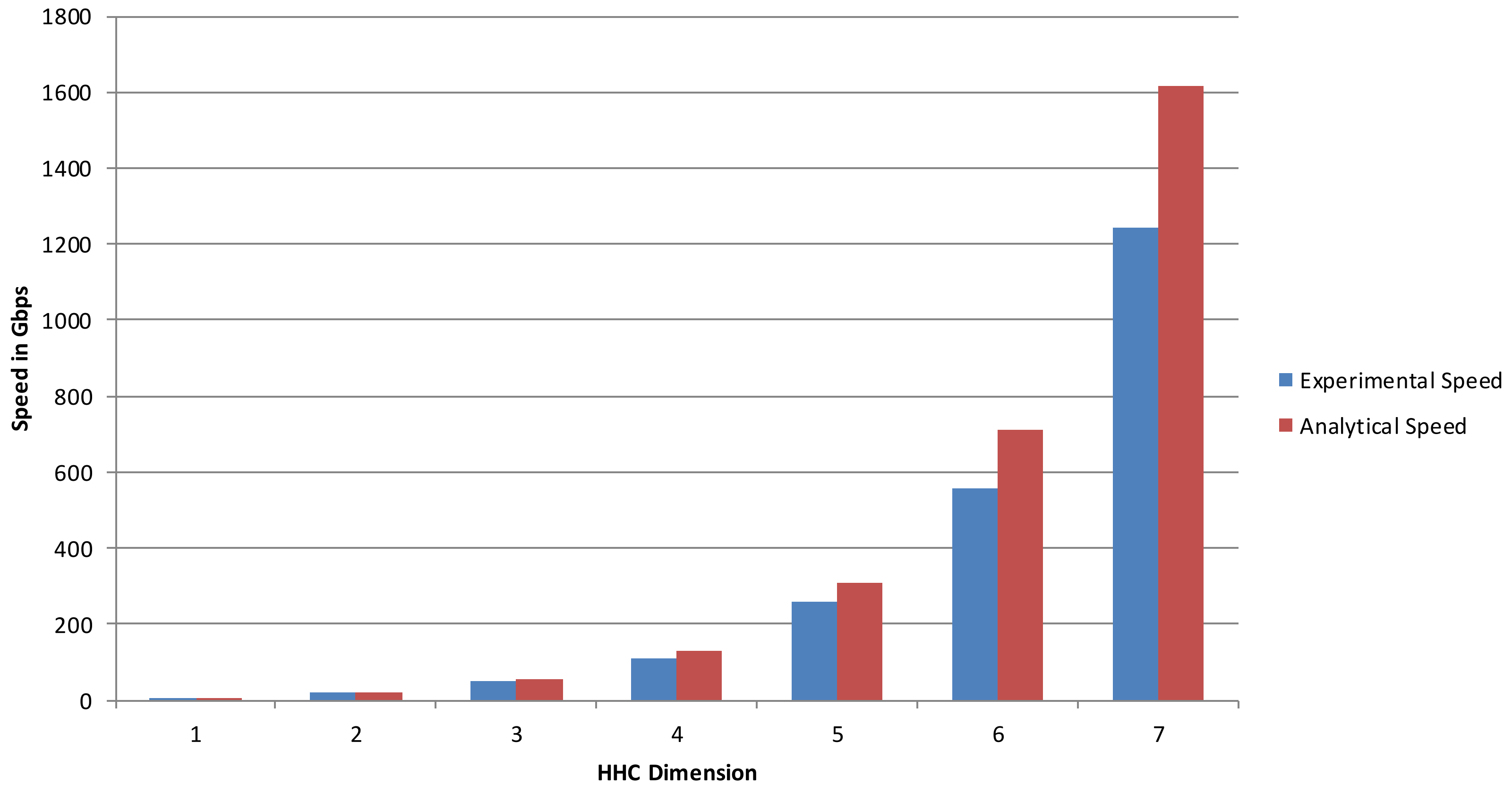
Execution Time for Max of (500) workload units over verity of dimensions for Algorithms (C).

Communications



Number of Communication Steps and Number of Data Moves for Algorithms (C) while varying the number of processors for a fixed workload.

Speed



Number of Communication Steps and Number of Data Moves for Algorithms (C) while varying the number of processors for a fixed workload.

Conclusion

- Algorithms B and C would performs faster than first algorithm.
- Busy waiting decreases the performance of the algorithm as in algorithm A.
- Increasing the number of processors or total number of load units increases the execution time.