Peer-to-Peer Systems
Exercise 6
Published: 15.01.2015
Presentation date: 22.01.2015

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems – Exercise
Winter Term 2014/2015

General Remarks

Welcome to the exercise for the lecture Peer-to-Peer Systems.
Please follow the general remarks regarding the organization of the exercise.

- The lecture's website is to be found here:
  `http://tsn.hhu.de/teaching/lectures/2014ws/p2p.html`
- For further inquiries, please contact the lecturer under the following email address: `graffi@cs.uni-duesseldorf.de`

Problem 6.1 - Sampling Information in Networks

In this exercise we test the effect of sampling in random network. Consider for this the following network with 21 nodes:

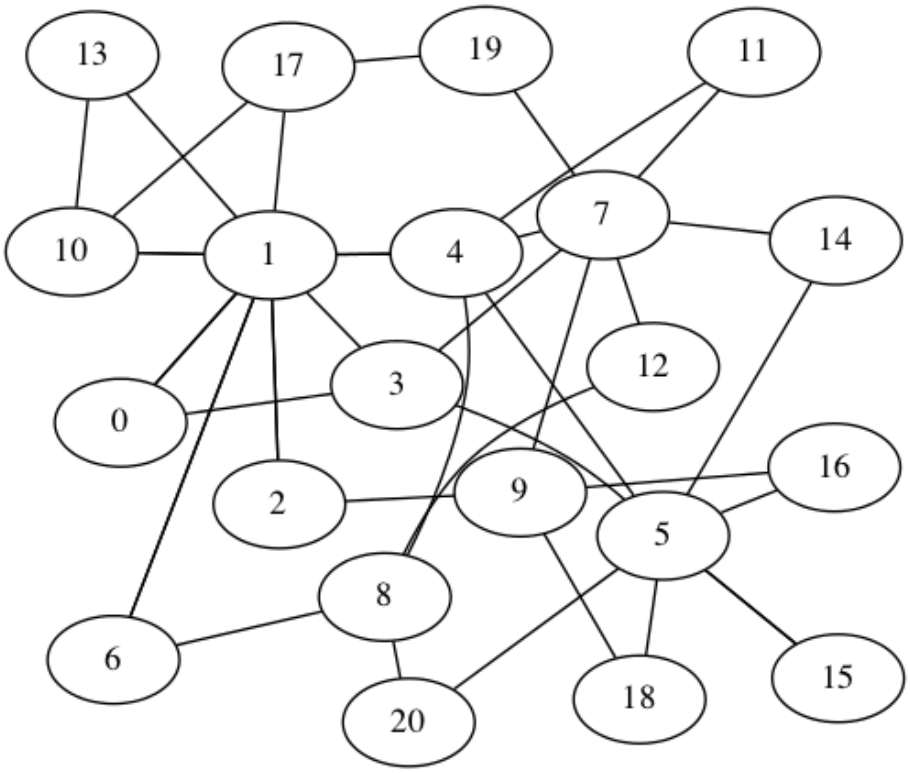


Abbildung 1: Random network with 21 nodes

a) Sample Sampling Run

Consider that node 20 and node 0 start a random walk with hop limit 10 trying to calculate the average node ID and degree in the network. The random walk passes 10 links and is replied by the node nr. 11 in the path to the querying node. Calculate the path of both random walks separately, the values gathered (11 samples) and the average node ID and degree obtained in each walk.
In the case that a node might choose from k neighbors as next hops, use following formula:

- Sort the neighbor node IDs: $ID_0 \ldots ID_{k-1}$
- Add the current node's ID and add (if possible) the previous node's ID, e.g. the sum is M
- Calculate the index: j = M mod k
- Pick as next hop: ID_j

Please note, that loops are allowed.

What are the obtained values? How close is the average node ID obtained by sampling to the actual average node ID? There are in total 34 links and 21 nodes. The average node ID is 10.5 and the average degree 3.4. Use following tables and discuss the obtained results. What weaknesses do you see?

Random walk from node 20:

Current node	List of k neigbors	Sum M	k	M mod k	Next hop

Random walk from node 0:

Current node	List of k neigbors	Sum M	k	M mod k	Next hop

Solution:

Random walk from node 20:

Current node	List of k neigbors	Sum M	k	M mod k	Next hop
20	5,8	20	2	0	5
5	3,4,14,15,16,18,20	25	7	4	16
16	5,9	21	2	1	9
9	2,7,16,18	25	4	1	7
7	3,4,9,11,12,14,19	16	7	2	9
9	2,7,16,18	16	4	0	2
2	1,9	11	2	1	9
9	2,7,16,18	11	4	3	18
18	5,9	27	2	1	9
9	2,7,16,18	27	4	3	18
18	5,9	27	2	1	9

Obtained list of node IDs: 20,5,16,9,7,9,2,9,18,9,18
Obtained node IDs: Sum = 122, average =11.09
Obtained list of node degrees: 2,7,2,4,7,4,2,4,2,4,2
Obtained degrees: sum = 40, average = 3.63

Random walk from node 0:

Current node	List of k neigbors	Sum M	k	M mod k	Next hop
0	1,3	0	2	0	1
1	0,2,3,4,6,10,13,17	1	8	1	2
2	1,9	3	2	1	9
9	2,7,16,18	11	4	3	18
18	5,9	27	2	1	9
9	2,7,16,18	27	4	3	18
18	5,9	27	2	1	9
9	2,7,16,18	27	4	3	18
18	5,9		2		
9	2,7,16,18	27	4	3	18

Obtained list of node IDs: 0,1,2,9,18,9,18,9,18,9,18
Obtained node IDs: Sum = 111, average =10.1
Obtained list of node degrees: 2,7,2,4,2,4,2,4,2,4,2
Obtained degrees: sum = 35, average = 3.6

Observations:
- Loops occur, catching the results of individual nodes several times
- High degree nodes are visited frequently
- Loops may manifest if random selection of neighbors is biased (see example random walk starting from node 0)

Problem 6.2 - Gossip based Monitoring

Consider in this exercise the small graph with 6 nodes. In this exercise we apply PushSum on this network.

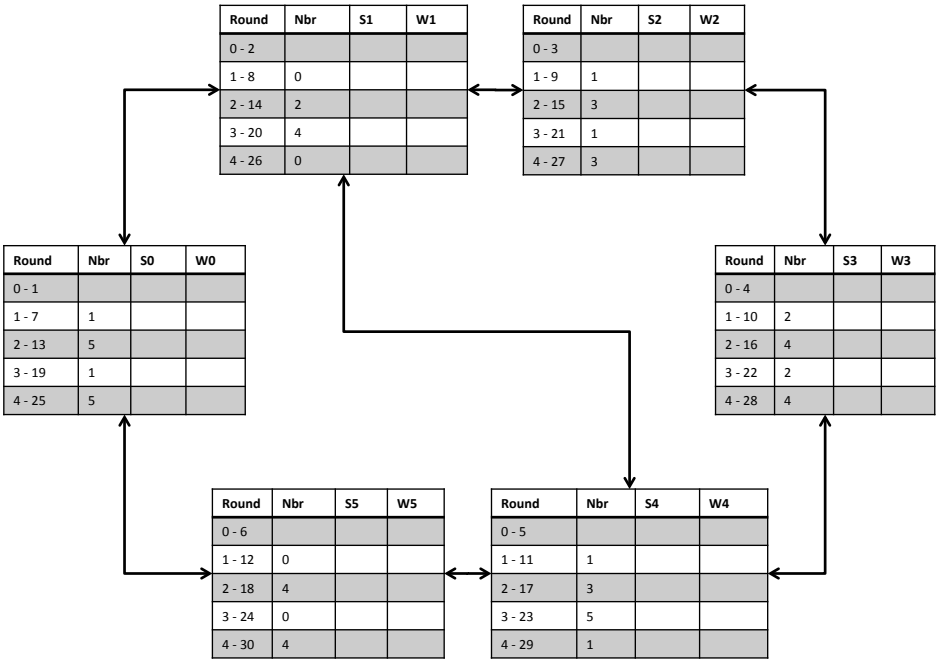
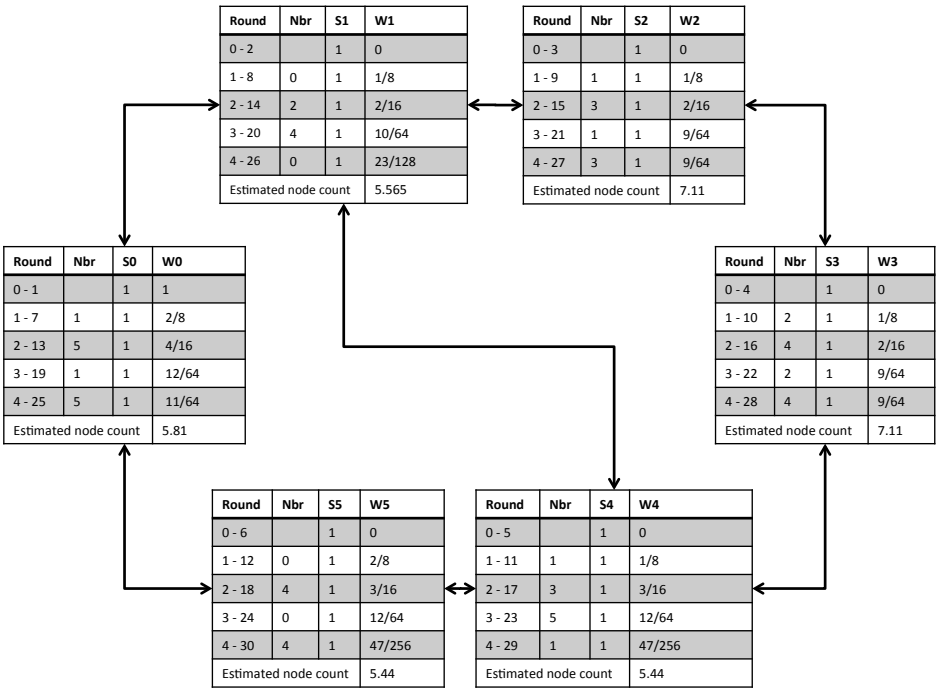


Abbildung 2: Network with 6 nodes

a) Sample Gossiping

Assume that the nodes in the small network want to obtain the number of nodes in the network. Therefore node 0 initiates a gossip epoch in round 0, all other nodes are prepared as well in round 0 and set their values s_i and w_i correctly. In each round the nodes act clockwise, i.e. starting with node 0 (action 0 - 1), then nodes 1,2,3,4 and finally node 5 is allowed to act. The round number also lists as a second parameter the action numer. For example the value (2 - 15) displays that in the second round, node 2 performs his gossip action, which is the 15th in the row. After him, node 3 performs his gossip action numbered (2 - 16). The gossip partner is listed in the column (*nbr*), a full gossip transaction is performed. This means, that the process of sending 1/2 of the own values, receiving the 1/2 of the other node values and aggregating it is fully completed. Thus, a single node might be involved in several gossip transactions per round. Note in the corresponding fields in the figure the final values s_i and w_i after the end of each round at each node. After round 4, what are the obtained values? What is the calculated node count in each node after round 4? Discuss the obtained results. What weaknesses do you see?

Solution:



Round	Nbr	S1	W1
0 - 2		1	0
1 - 8	0	1	1/8
2 - 14	2	1	2/16
3 - 20	4	1	10/64
4 - 26	0	1	23/128
Estimated node count			5.565

Round	Nbr	S2	W2
0 - 3		1	0
1 - 9	1	1	1/8
2 - 15	3	1	2/16
3 - 21	1	1	9/64
4 - 27	3	1	9/64
Estimated node count			7.11

Round	Nbr	S0	W0
0 - 1		1	1
1 - 7	1	1	2/8
2 - 13	5	1	4/16
3 - 19	1	1	12/64
4 - 25	5	1	11/64
Estimated node count			5.81

Round	Nbr	S3	W3
0 - 4		1	0
1 - 10	2	1	1/8
2 - 16	4	1	2/16
3 - 22	2	1	9/64
4 - 28	4	1	9/64
Estimated node count			7.11

Round	Nbr	S5	W5
0 - 6		1	0
1 - 12	0	1	2/8
2 - 18	4	1	3/16
3 - 24	0	1	12/64
4 - 30	4	1	47/256
Estimated node count			5.44

Round	Nbr	S4	W4
0 - 5		1	0
1 - 11	1	1	1/8
2 - 17	3	1	3/16
3 - 23	5	1	12/64
4 - 29	1	1	47/256
Estimated node count			5.44

Abbildung 3: Network with 6 nodes: Gossip applied

Observations:
- Redundant communication occurs, asking nodes with same values to compare
- High degree nodes are visited frequently, which is good in this case
- Convergence works, but is slow

Problem 6.3 - Monitoring P2P Systems - Joining the Tree

SkyEye.KOM is a monitoring approach that helps to gather and disseminate statistics on a p2p network.

a) Tree node positioning and parent node calculations

Consider Chord with an ID space ranging from 0 to $2^{160}-1$. Node **AEE9 6FC9 9741 FE3C EFD6 174A 014B 0556 07D4 5795** wants to participate in monitoring and has first to identify its position in the tree. The tree consists of approx. 100 nodes, thus the predecessor node's ID is approx. 1/100 th of the ID space away (i.e. 0.01 in the unified ID space). Please calculate the position of the node in the monitoring tree SkyEye.KOM with a branching factor **2 and 4**. Calculate also the IDs of the corresponding (father) contact nodes in the tree. Hint: Calculate the nodeID in the unified ID space, and the first i DomainIDs which might be relevant. Find through the DomainIDs the place of the node and also a way how its father node can be contacted.

Solution:

The formula to derive the Domain keys is for a Domain at level l, branching factor β, peer p with Peer ID ID_p, see Chapter 11, slide 8:

$$K_p^l := \frac{\frac{\lfloor ID_p \cdot \beta^l \rfloor + 1}{\beta^l} + \frac{\lfloor ID_p \cdot \beta^l \rfloor}{\beta^l}}{2}$$

We use Octave for presenting the results:
Hex: AEE9 6FC9 9741 FE3C EFD6 174A 014B 0556 07D4 5795

```
octave:> format long
octave:> nodeIDdec = hex2dec(¨AEE96FC99741FE3CEFD6174A014B055607D45795")
nodeIDdec = 9.98570205816156e+47
octave:> nodeID = nodeIDdec /(2 ^ 160)
nodeID = 0.683249460903660
```

Thus, the NodeID in the unified ID space: 0.683249460903660
ID space of the node: 0.674 to 0.683

Branching factor: 2
Using the formula given above, we obtain following Domain IDs: `Level 0 := 0,500000000000000`
`Level 1 := 0,750000000000000`
`Level 2 := 0,625000000000000`
`Level 3 := 0,687500000000000`
`Level 4 := 0,656250000000000`
`Level 5 := 0,671875000000000`

```
Level 6 := 0,679687500000000
Level 7 := 0,683593750000000
Level 8 := 0,681640625000000
Level 9 := 0,682617187500000
Level 10 := 0,683105468750000
```

Thus, the domainID 0.679687500000000 is the first one which is in the responsibility range of the node. Thus, the node is located at level 6 and its parent node is the one, which is responsible for the domainID 0.671875000000000 in the unified ID space.

Branching factor: 4

```
Level 0 := 0,500000000000000
Level 1 := 0,625000000000000
Level 2 := 0,656250000000000
Level 3 := 0,679687500000000
Level 4 := 0,681640625000000
Level 5 := 0,683105468750000
Level 6 := 0,683227539062500
Level 7 := 0,683258056640625...
Level 8 := 0,683250427246093...
Level 9 := 0,683248519897460...
Level 10 := 0,683248996734619...
```

Thus, the domainID 0.679687500000000 is the first one which is in the responsibility range of the node. Thus, the node is located at level 3 and its parent node is the one, which is responsible for the domainID 0,656250000000000 in the unified ID space.

ex11_monitoring_treejoin_DomainID-Berechnung

ID = ##### 0,6832494609

Beta = 2,00

Level =	Beta^L	(ID_p * Beta^L)	Floor (ID_p * Beta^L)	DomainID
0,00	1,00	0,683249461	0	0,500000000000000
1,00	2,00	1,366498922	1	0,750000000000000
2,00	4,00	2,732997844	2	0,625000000000000
3,00	8,00	5,465995687	5	0,687500000000000
4,00	16,00	10,93199137	10	0,656250000000000
5,00	32,00	21,86398275	21	0,671875000000000
6,00	64,00	43,7279655	43	0,679687500000000
7,00	128,00	87,455931	87	0,683593750000000
8,00	256,00	174,911862	174	0,681640625000000
9,00	512,00	349,823724	349	0,682617187500000
10,00	1024,00	699,647448	699	0,683105468750000

Beta = 4,00

Level =	Beta^L	(ID_p * Beta^L)	Floor (ID_p * Beta^L)	DomainID
0,00	1,00	0,683249461	0	0,500000000000000
1,00	4,00	2,732997844	2	0,625000000000000
2,00	16,00	10,93199137	10	0,656250000000000
3,00	64,00	43,7279655	43	0,679687500000000
4,00	256,00	174,911862	174	0,681640625000000
5,00	1024,00	699,647448	699	0,683105468750000
6,00	4096,00	2798,589792	2798	0,683227539062500
7,00	16384,00	11194,35917	11194	0,683258056640625
8,00	65536,00	44777,43667	44777	0,683250427246093
9,00	262144,00	179109,7467	179109	0,683248519897460
10,00	1048576,00	716438,9867	716438	0,683248996734619

Beta = 8,00

Level =	Beta^L	(ID_p * Beta^L)	Floor (ID_p * Beta^L)	DomainID
0,00	1,00	0,683249461	0	0,500000000000000
1,00	8,00	5,465995687	5	0,687500000000000
2,00	64,00	43,7279655	43	0,679687500000000
3,00	512,00	349,823724	349	0,682617187500000
4,00	4096,00	2798,589792	2798	0,683227539062500
5,00	32768,00	22388,71833	22388	0,683242797851562
6,00	262144,00	179109,7467	179109	0,683248519897460
7,00	2097152,00	1432877,973	1432877	0,683249235153198
8,00	16777216,00	11463023,79	11463023	0,683249443769454
9,00	134217728,00	91704190,3	91704190	0,683249462395906
10,00	1073741824,00	733633522,4	733633522	0,683249460998922

Abbildung 4: Calculation of DomainIDs

b) Issues in creating the tree

Having identified the position for one single node, we might identify some issues.

- Is the tree construction always deterministic?
- Can a parent node have more than β child nodes?
- What happens if nodes fail in the tree?
- Are there problems if the responsibility range of a node is increased / decreased (e.g. by a joining / leaving neighbor)
- Are there issues if the initial overlay is multi-dimensional?
- Which further issues do you see in creating the tree?

Solution:

Following observations can be made

- Is the tree construction always deterministic?
  - if a node is responsible for various domain IDs its position might depend on the implementation
- Can a parent node have more than β child nodes?
  - yes, for each domain ID a node is responsible for, it might have β child nodes, thus in total a parent node might have more than β child nodes.
- What happens if nodes fail in the tree?
  - only its child nodes are affected. However, as they send their messages based on domain IDs (converted to object IDs) the new responsible node for the domain ID is addressed.
- Are there problems if the responsibility range of a node is increased / decreased (e.g. by a joining / leaving neighbor)
  - with a modified responsibility range the node might get a new position in the tree. Problem: At its new positions it does not have a valid information to send to its parent node. Taking the example of the root node: If the root node looses it position it might re-inject its aggregated information at a leaf node position, thus creating peak / double result. The new node taking over the roots position has for a short time no clue on the total aggregated information, thus a monitoring gap occurs.
- Which further issues do you see in creating the tree?

Problem 6.4 - Example of using the Monitoring Tree

Consider in this exercise the small graph with 15 nodes. In this exercise we apply the SUM aggregation of SkyEye.KOM on this network.

a) Sample usage of the monitoring tree

Assume that the nodes in the small network want to obtain the sum of all values in the network. The nodes submit their aggregated values periodically in rounds, the acknowledgements on the other side are replied immediately. Thus, the aggregation updates are performed roundwise, starting with the leafs of the tree then with the levels towards the root. Write in the table the aggregation obtained in the corresponding round. Through the round based communication, the aggregated value in a node in round i is based on the values of its child nodes in round i-1. *AckR* stands for „received ACK". It denotes the global view obtained from the parent node in the ACK sent for the nodes aggregation update. Fill out the tables. How long does the information dissemination take to reach all nodes? Discuss the obtained results. What weaknesses do you see?

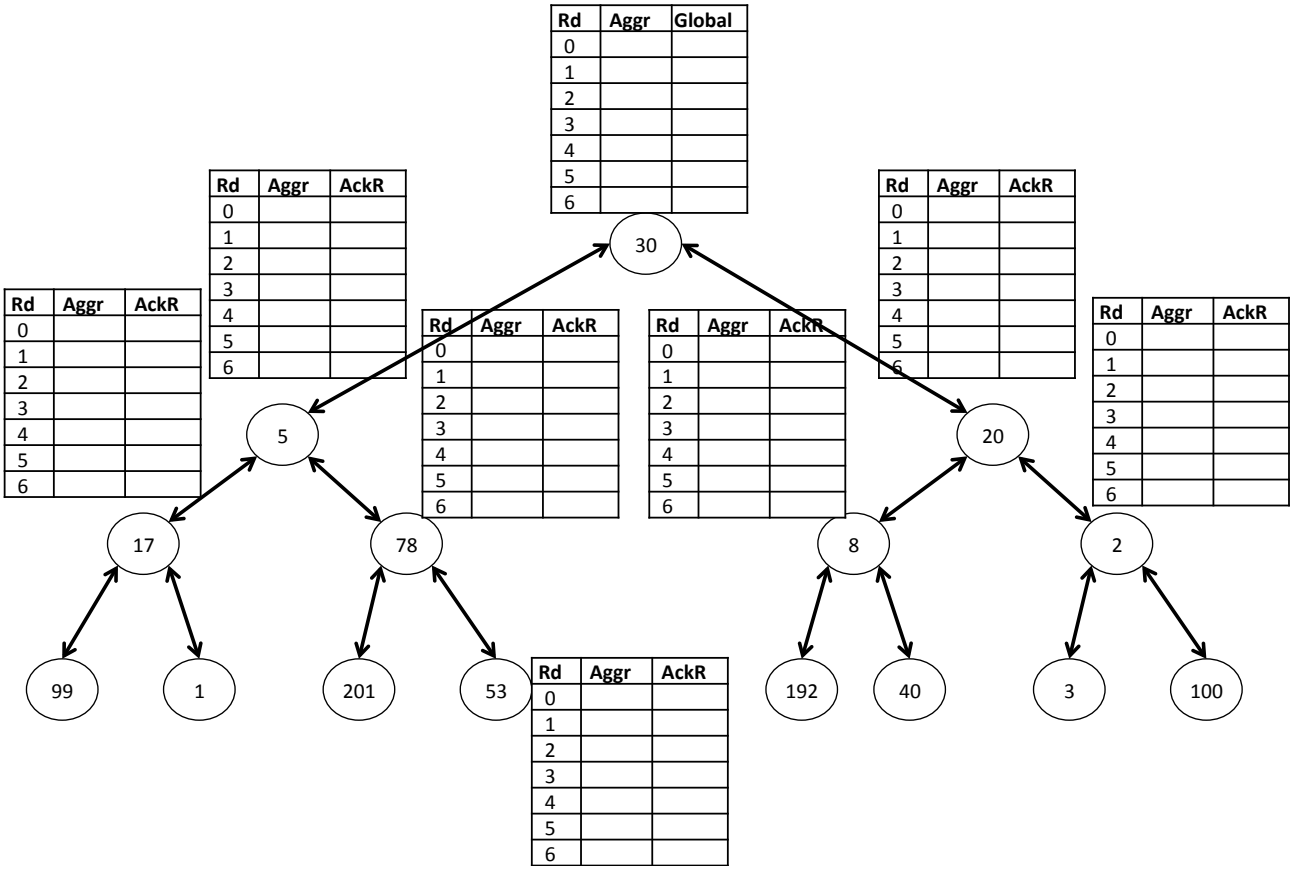


Abbildung 5: Network with 15 nodes

Solution:

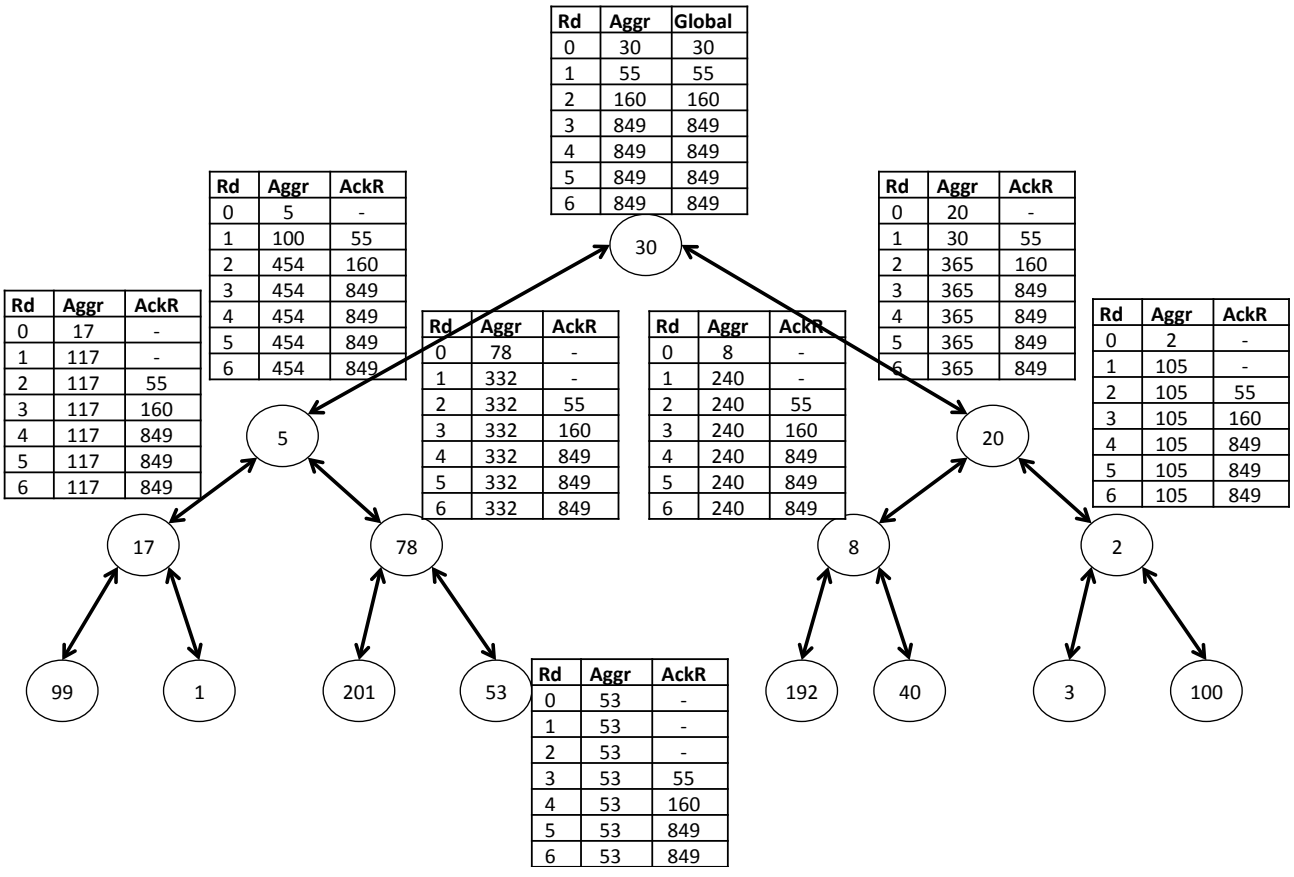


Abbildung 6: Network with 15 nodes: SkyEye.KOM applied

Observations:
- Pushing of global measurement from root to the leafs takes time through the usage of the ACKs: $O(logN)$ = tree height update rounds. One could also flush the results once they are complete and do not change.
- Optimization possible: start aggregation from leafs (how to identify if leaf?).

One extension of SkyEye.KOM from the lecture proposes to synchronize the aggregation updates to coordinate a quick gathering of the information towards the root. In addition, it also offers a quick flush of the obtained global aggregation values through the tree.
- The measurement is precise and quickly done