TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks

Samuel Madden, Michael Franklin, Joseph Hellerstein, and Wei Hong

Presented by Mo Dong

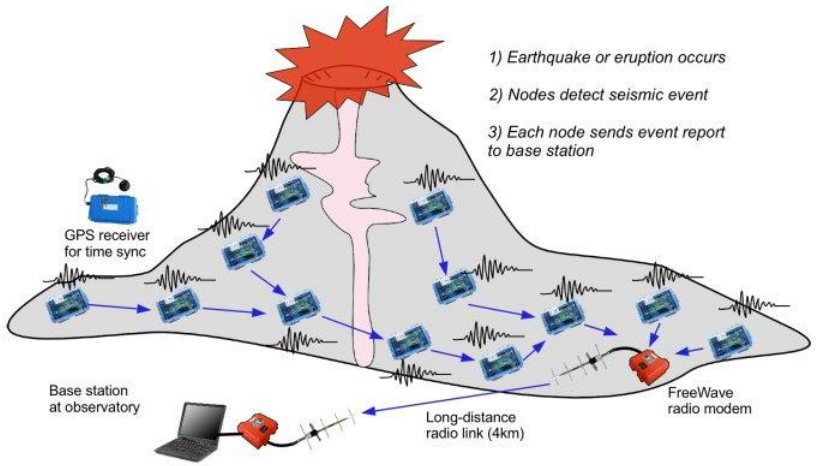
Outline

- Why TAG?
- Declarative Query
- In Network Aggregation
- Evaluation & Optimization
- Conclusion & Discussion

Sensor Networks

- Sensor Network is all about Data Querying

Environment Sensing :Earthquake detection



Habitat Monitoring: What are my cats doing when I am not at home.....

Why TAG?

- Challenges for Querying Data
  - Programming/Debugging is a nightmare

Why TAG?

- Challenges for Querying Data
  - Programming/Debugging is a nightmare
  - Have to take care of low level details

Application 1 have to take care of these

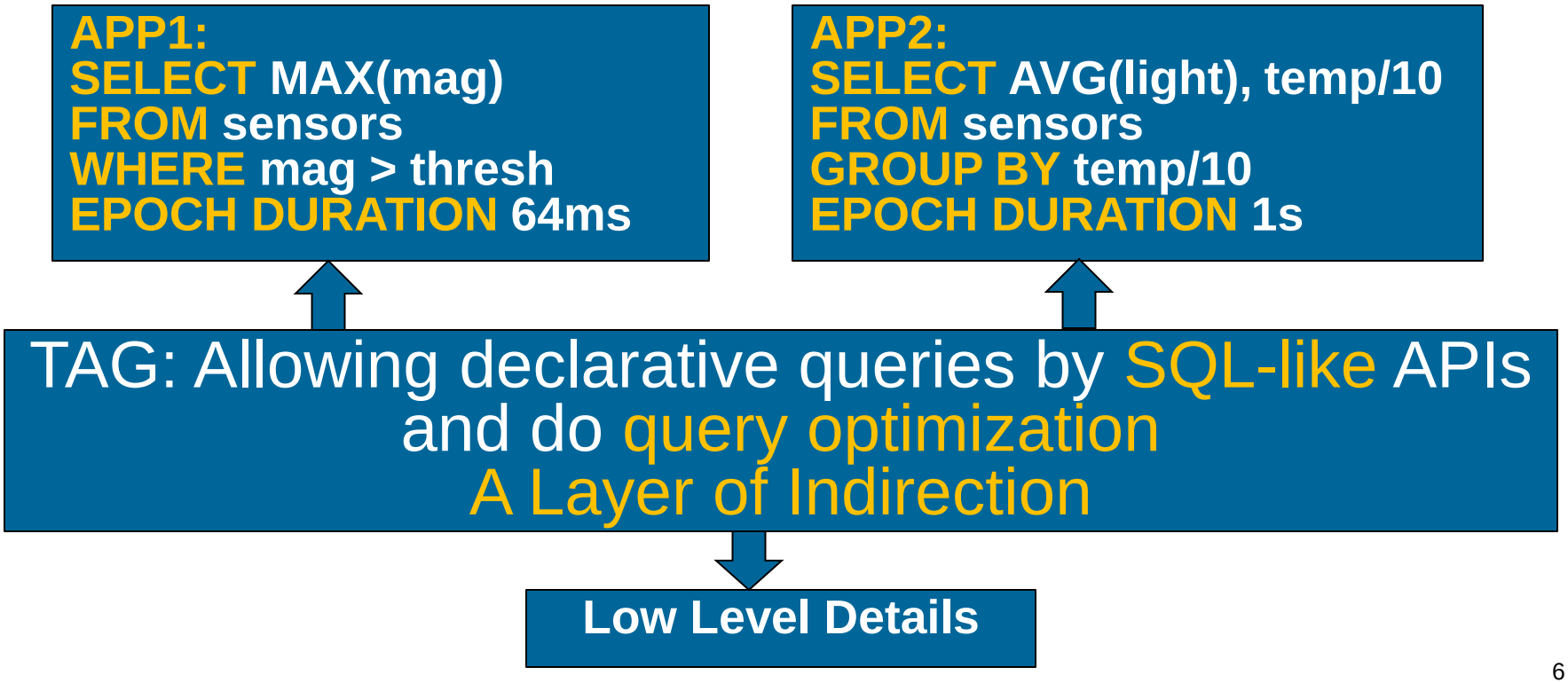
Application 2 have to take care of these AGAIN!

Routing, Radio Contention, data storage and fault tolerance

Routing, Radio Contention, data storage and fault tolerance

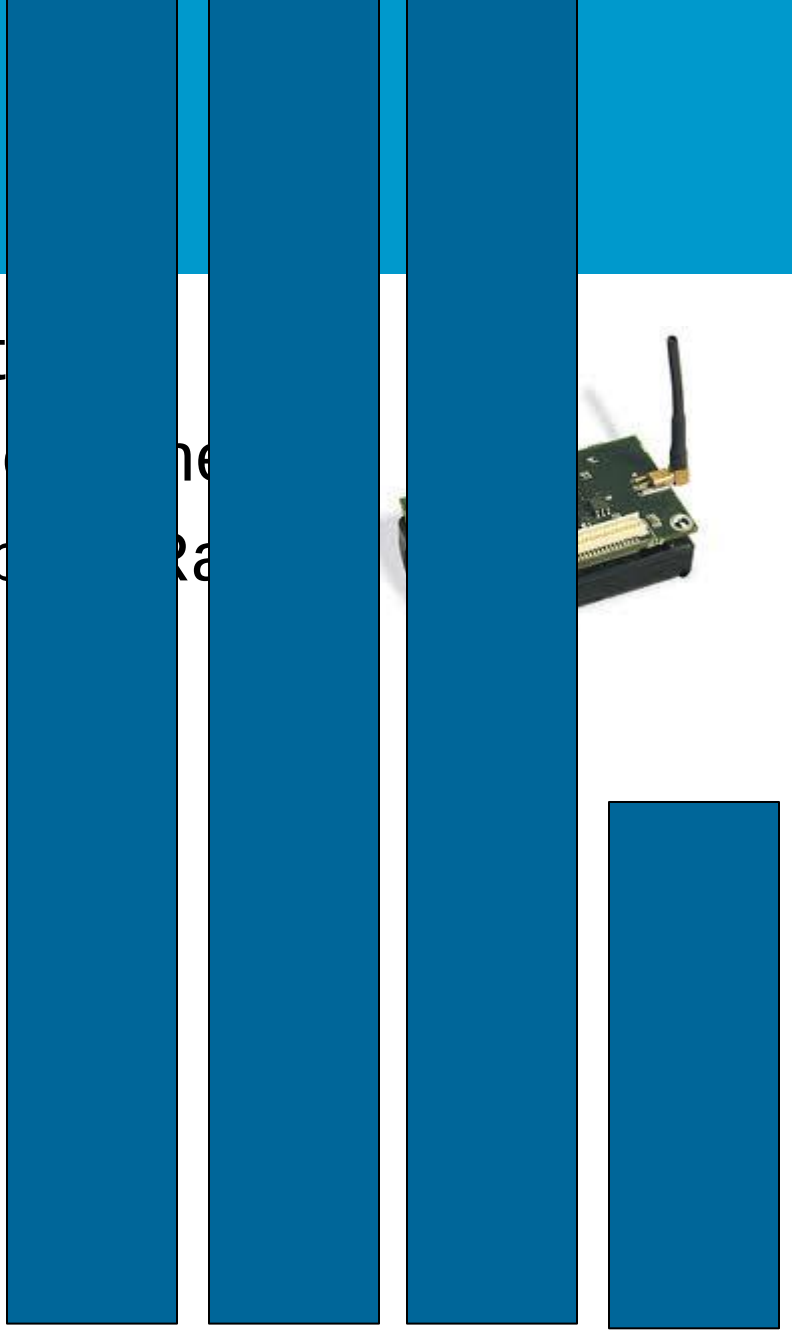
Why TAG is needed?

- Challenges for Collecting Data
  - Programming/Debugging is a nightmare
  - Have to take care of low level details



Why TAG?

- Challenges for Collecting Dat
  - Life time of a sensor is all ab
  - Energy Consumption is all ab



One Instruction **X 8000=** **One Bit of Transmission**

Why TAG?

- Challenges for Collecting Data
  - Life time of a sensor is all about energy
  - Energy Consumption is all about Radio
  - Reduce the amount of message transmission with In-network aggregation (Tiny Aggregation)
  - Exploit the semantic of SQL queries to reduce the amount of radio transmission

Outline

- Why TAG?
- Declarative Query
- In Network Aggregation
- Evaluation & Optimization
- Conclusion & Discussion

Declarative SQL-like Query

- Support Full Fledged SQL query

```
SELECT roomNo,  AVG(light)
FROM sensors
GROUP BY roomNo
HAVING AVG(light) > 200
EPOCH DURATION 5s
```

Epoch	roomNo	AVG(sound)
0	1	360
0	2	520
1	1	370
1	2	520

Declarative SQL-like Query

- Support Full Fledged SQL query and More

```
SELECT roomNo,  AVG(light)
FROM sensors
GROUP BY roomNo
HAVING AVG(light) > 200
EPOCH DURATION 5s
```

Epoch	roomNo	AVG(sound)
0	1	360
0	2	520
1	1	370
1	2	520

Declarative SQL-like Query

- Support Full Fledged SQL query and More

```
SELECT roomNo,  AVG(light)
FROM sensors
GROUP BY roomNo
HAVING AVG(light) > 200
EPOCH DURATION 5s
```

Epoch	roomNo	AVG(sound)
0	1	360
0	2	520
1	1	370
1	2	520

```
SELECT {agg(expr), attrs}
FROM sensors
WHERE {selPreds}
GROUP BY{attrs}
HAVING {having Preds}
EPOCH DURATION i
```

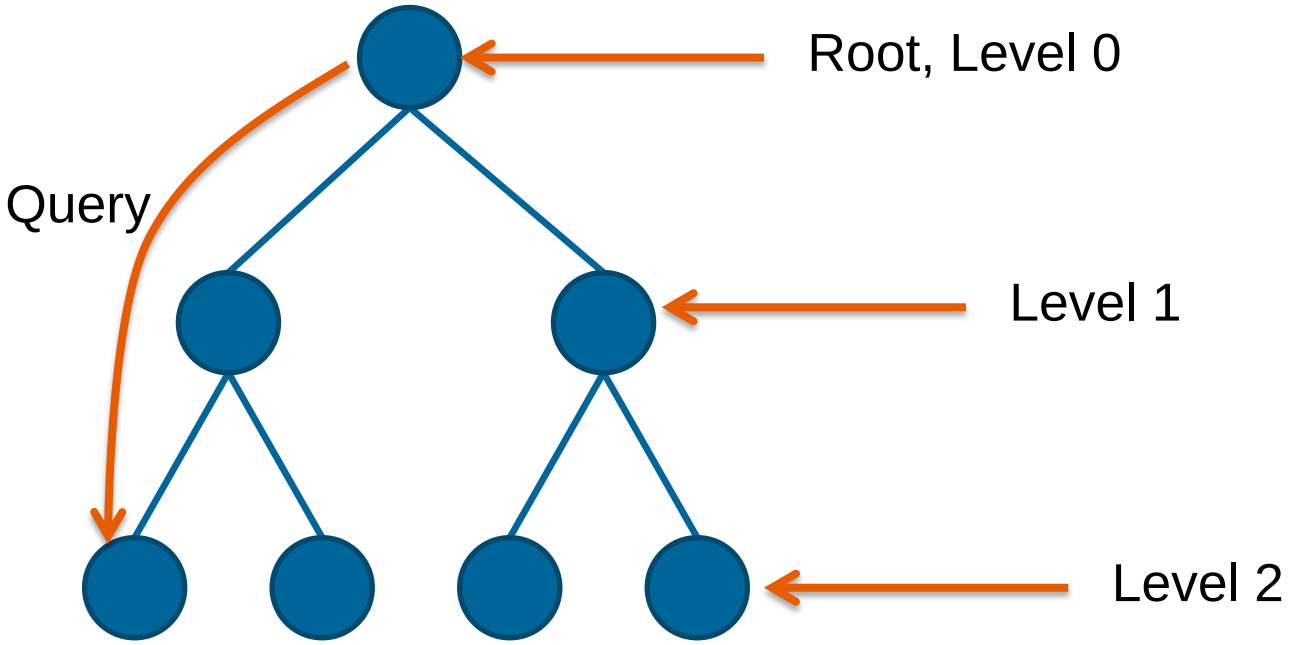
MAX,MIN,
COUNT,SUM,
AVG,MEDIAN,
COUNT DISTINCT,
HISTOGRAM

Outline

- Why TAG?
- Declarative Query
- In Network Aggregation
- Evaluation & Optimization
- Conclusion & Discussion

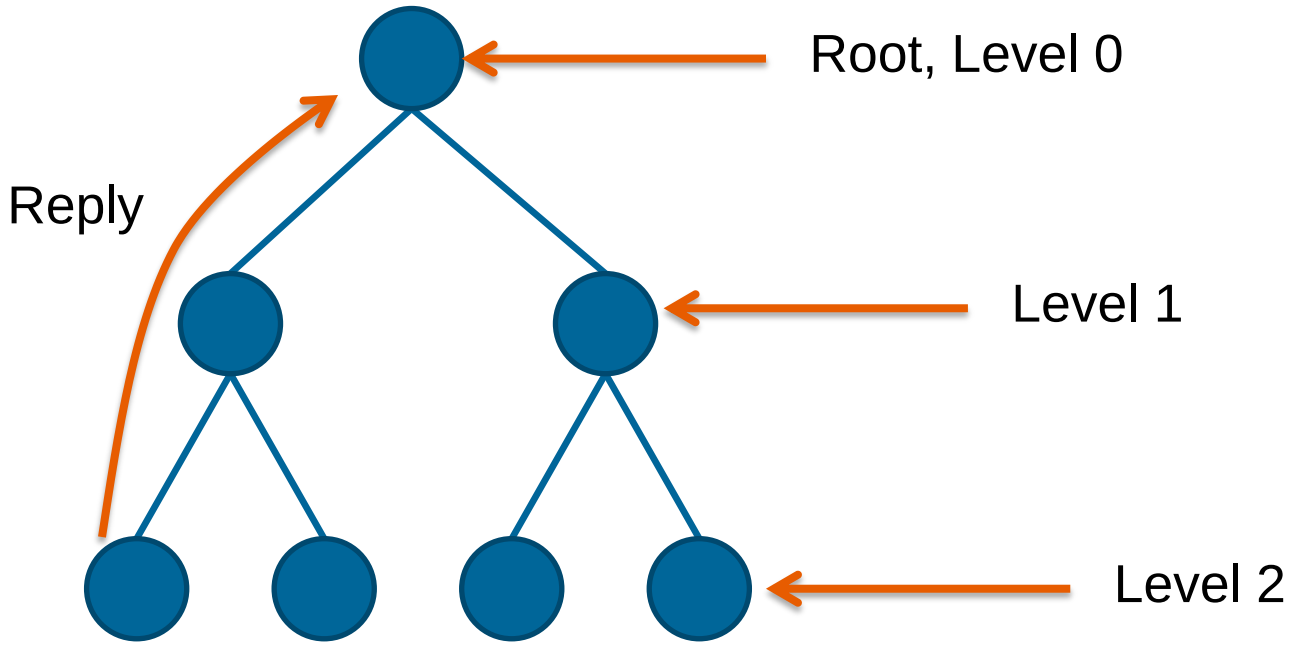
In Network Aggregation

- An Example:
  - Tree-like Topo & Level based Routing



In Network Aggregation

- An Example:
  - Tree-like Topo & Level based Routing



In Network Aggregation

- An Example:
  - ‘Global’ Synchronized Transmission

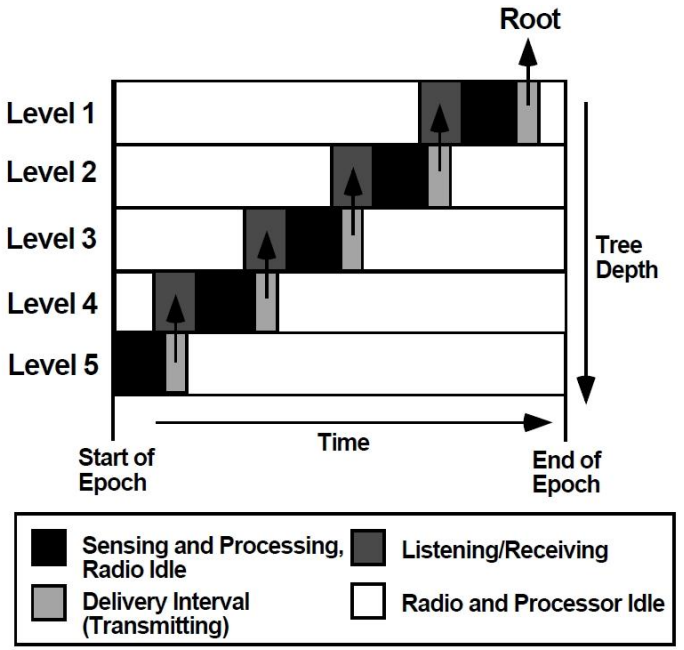
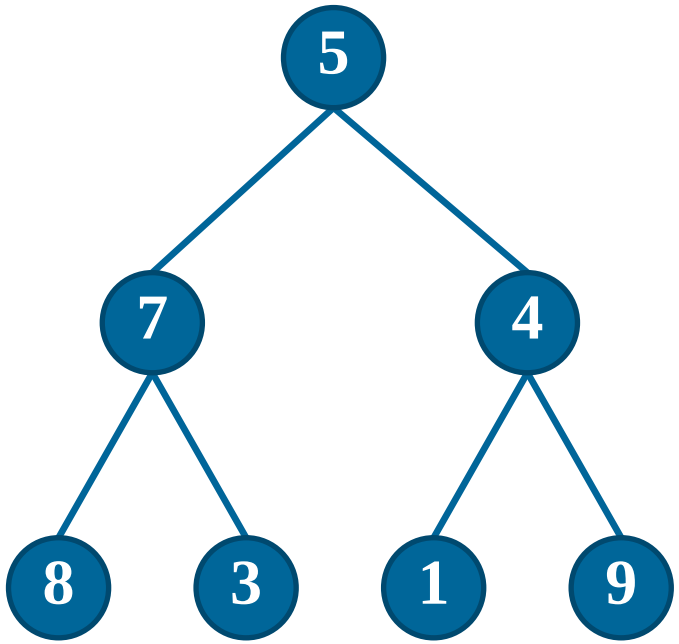


Figure 1: *Partial state records flowing up the tree during an epoch.*

In Network Aggregation

- An Example:
  - SELECT MAX(temp) FROM sensors (**without TAG)**

Total Messages: 0

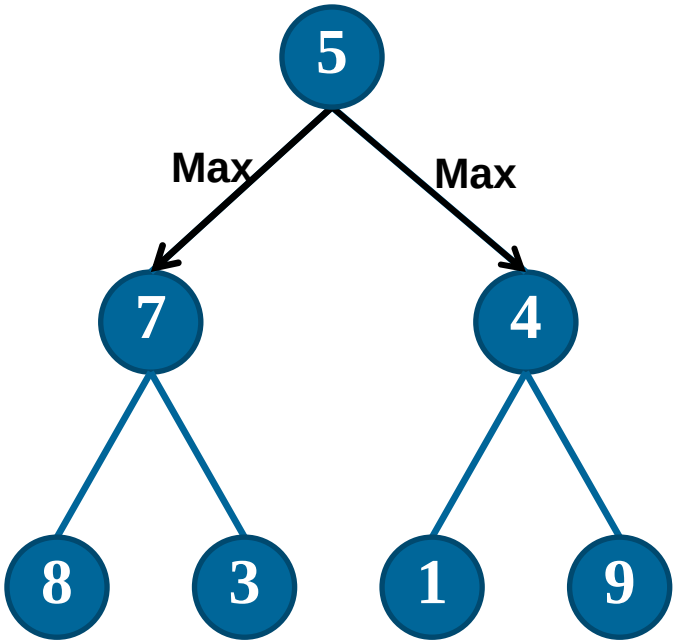


In Network Aggregation

- An Example:
  - SELECT MAX(temp) FROM sensors (**without TAG)**

Total Messages: 1

Numbers: [5]

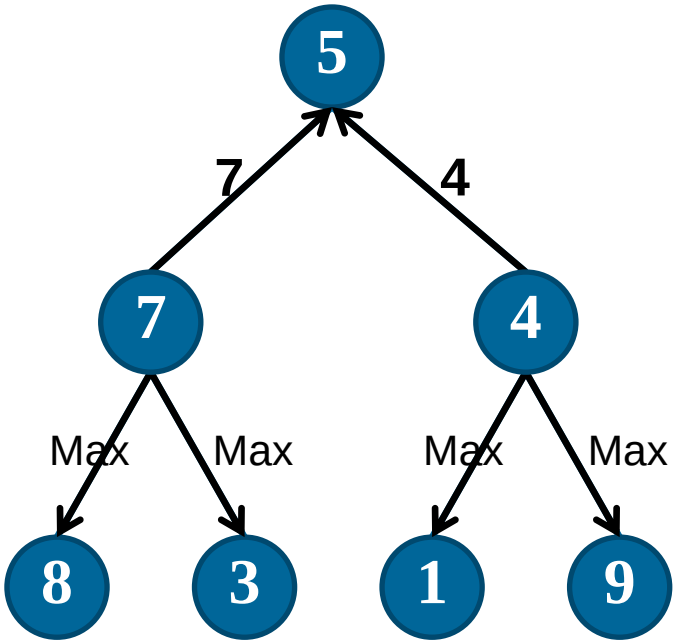


In Network Aggregation

- An Example:
  - SELECT MAX(temp) FROM sensors (**without TAG)**

Total Messages: 6

Numbers: [5,7,4]

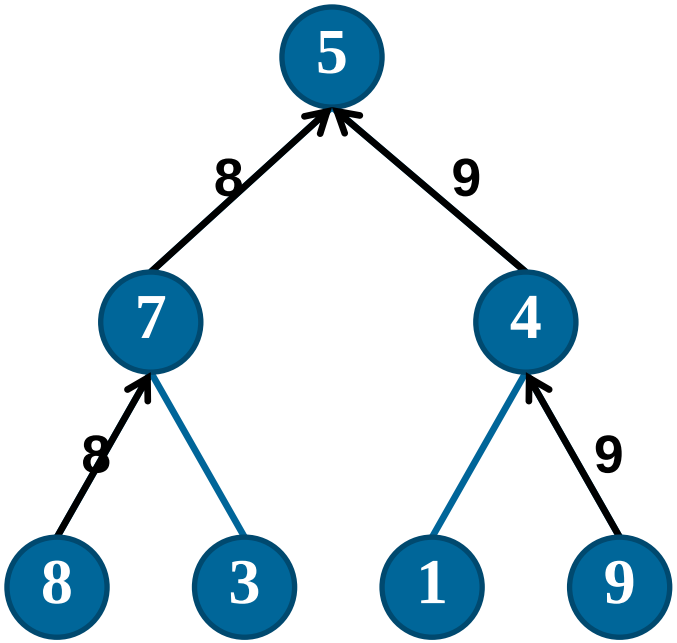


In Network Aggregation

- An Example:
  - SELECT MAX(temp) FROM sensors (**without TAG)**

Total Messages: 10

Numbers: [5,7,4,8,9]

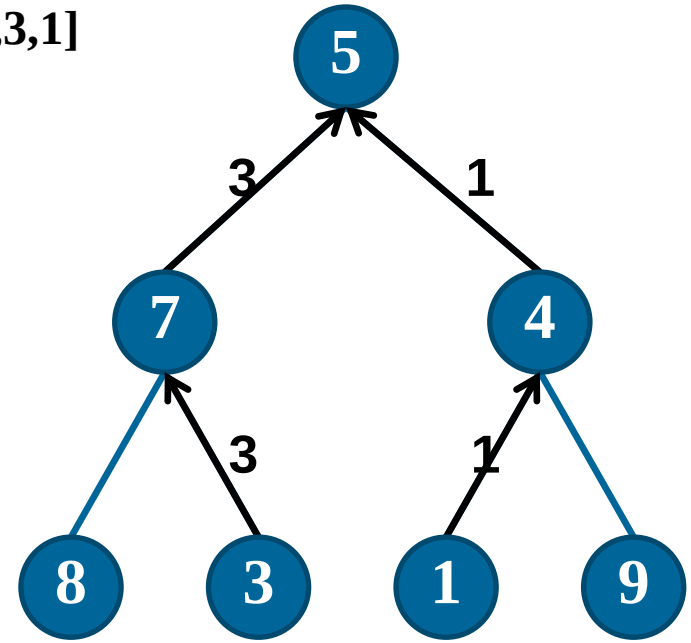


In Network Aggregation

- An Example:
  - SELECT MAX(temp) FROM sensors (**without TAG)**

Total Messages: 14

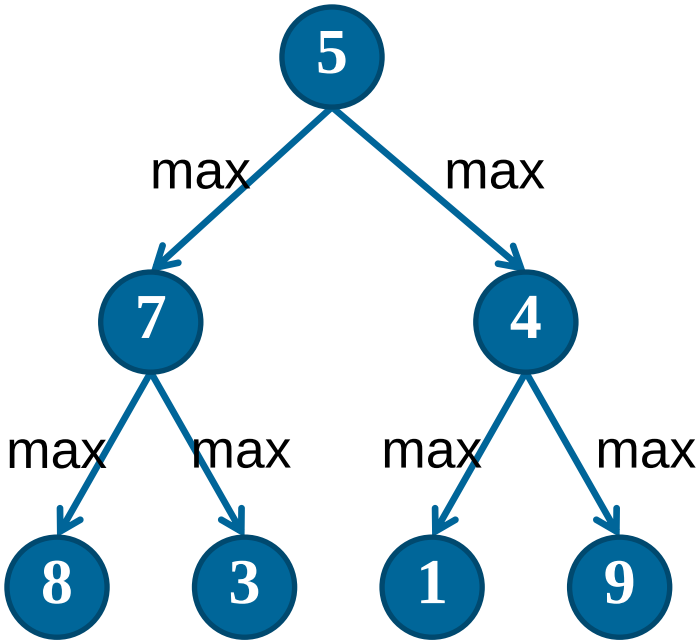
Numbers: [5,7,4,8,9,3,1]
MAX=9



In Network Aggregation

- An Example:
  - SELECT MAX(temp) FROM sensors **(with TAG)**

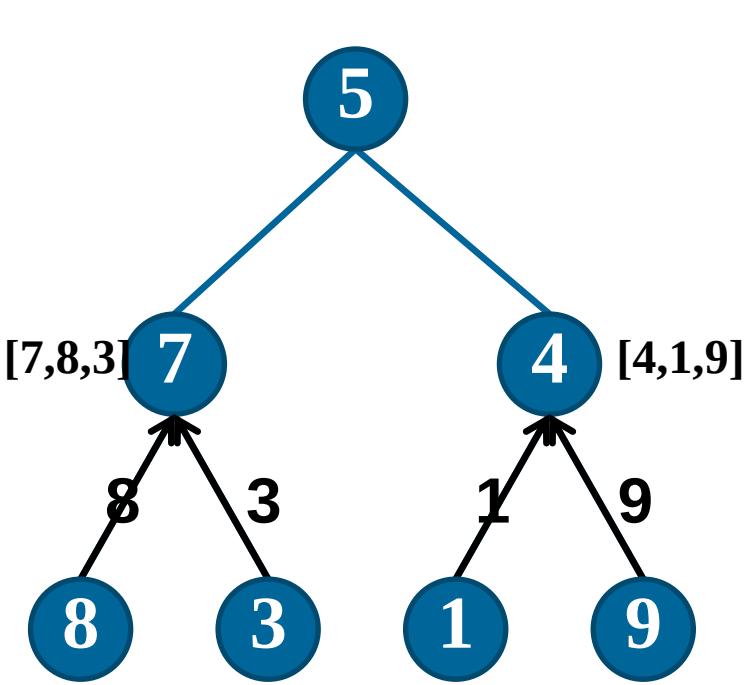
Total Messages: 3

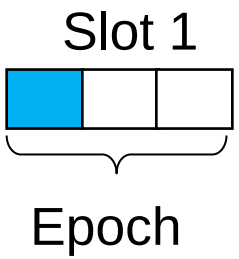


In Network Aggregation

- An Example:
  - SELECT MAX(temp) FROM sensors (**with TAG)**

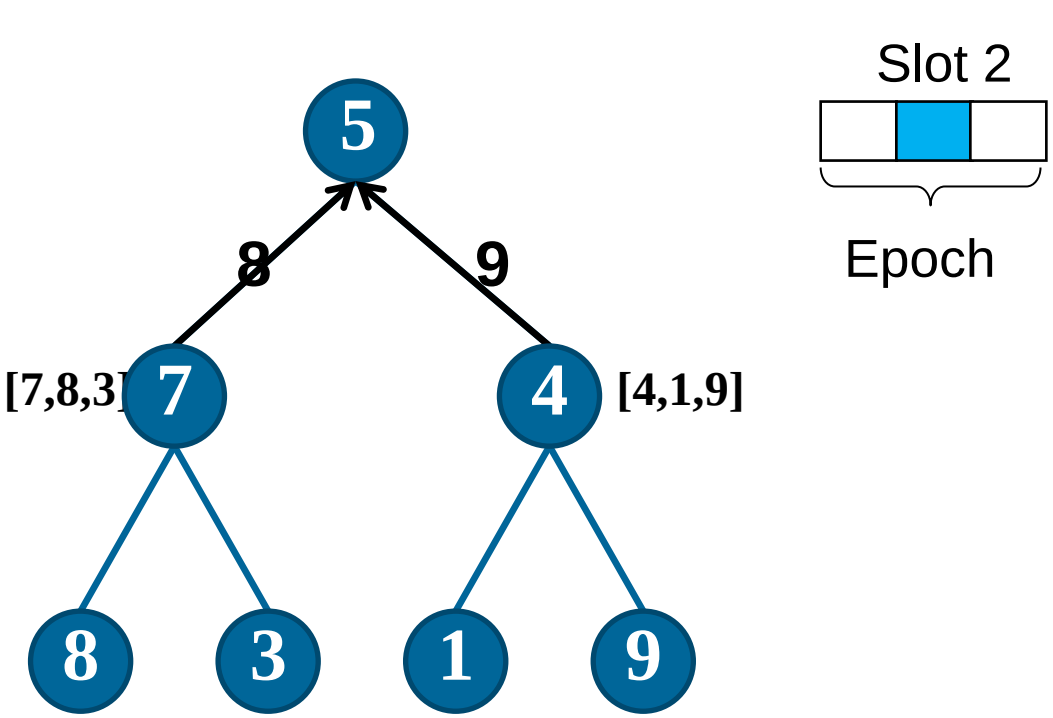
Total Messages: 7





In Network Aggregation

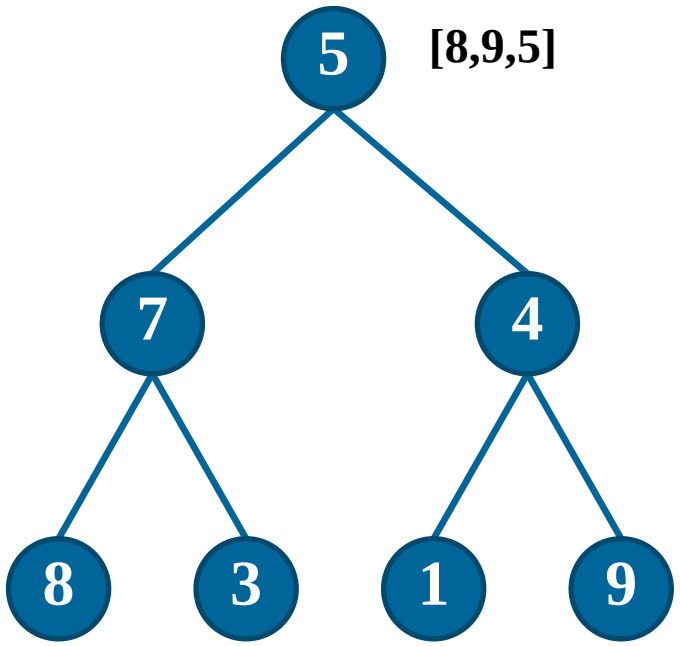
- An Example:
  - SELECT MAX(temp) FROM sensors (**with TAG)**

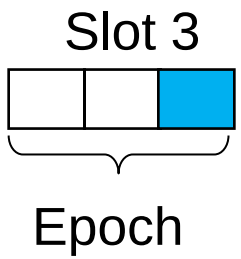


In Network Aggregation

- An Example:
  - SELECT MAX(temp) FROM sensors (**with TAG)**

Total Messages: 9 (vs 14)
MAX=9 delivered





In Network Aggregation

- Formal Definition

$Agg_n = \{f_{init}, f_{merge}, f_{evaluate}\}$

$f_{init}\{a_0\} \rightarrow \langle a_0 \rangle$

$F_{merge}\{\langle a_1 \rangle, \langle a_2 \rangle\} \rightarrow \langle a_{12} \rangle$

$F_{evaluate}\{\langle a_1 \rangle\} \rightarrow$ aggregate value

Partial State Record (PSR)

Example: Average

AVG_{init} $\{v\}$ $\rightarrow \langle v,1 \rangle$

AVG_{merge} $\{\langle S_1, C_1 \rangle, \langle S_2, C_2 \rangle\}$ $\rightarrow \langle S_1 + S_2 , C_1 + C_2 \rangle$

$AVG_{evaluate}\{\langle S, C \rangle\}$ $\rightarrow S/C$

In Network Aggregation

- Not All Operations can benefit from TAG
- Different Operations benefit from TAG differently
- Do further optimization based on different property

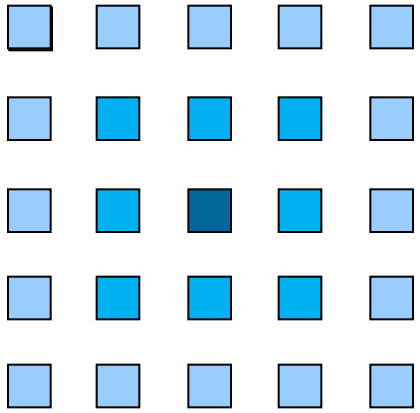
Property	Examples	Affects
Partial State	MEDIAN : unbounded, MAX : 1 record	Effectiveness of TAG
Duplicate Sensitivity	MIN : dup. insensitive, AVG : dup. sensitive	Routing Redundancy
Exemplary vs. Summary	MAX : exemplary COUNT: summary	Applicability of Sampling, Effect of Loss
Monotonic	COUNT : monotonic AVG : non-monotonic	Hypothesis Testing, Snooping

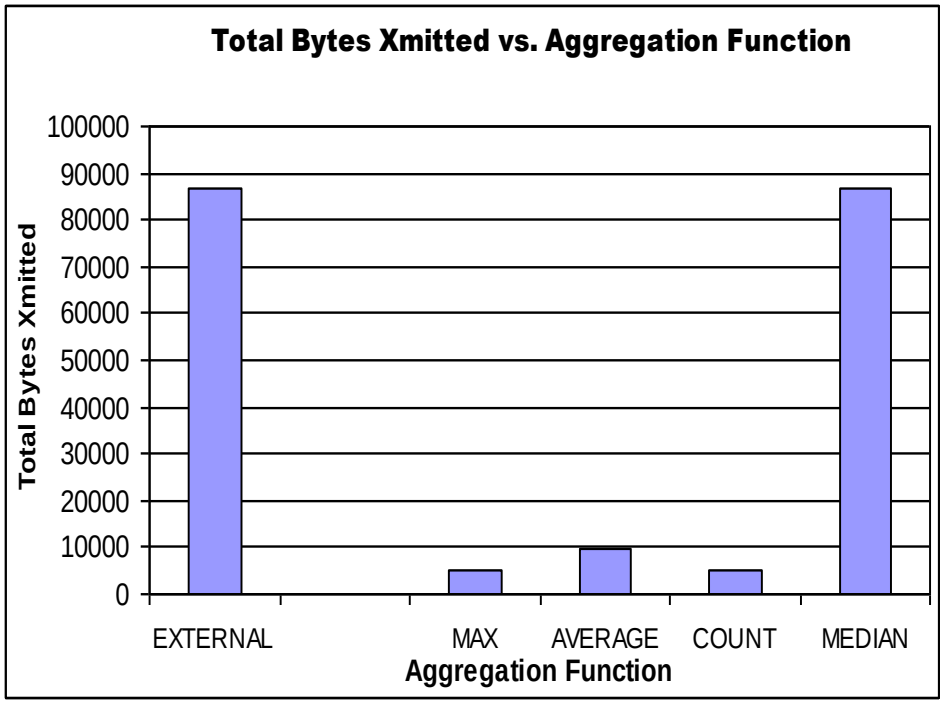
Outline

- Why TAG?
- Declarative Query
- In Network Aggregation
- Evaluation & Optimization
- Conclusion & Discussion

Evaluation & Optimization

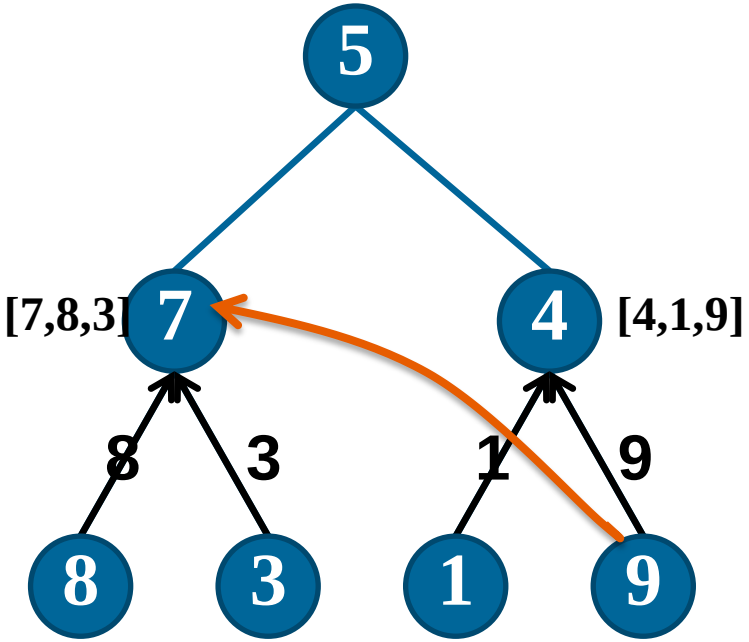
- Simulation Based Evaluation
  - Benefit From TAG varies
  - 50*50 Grid
  - No-loss link
  - Simple Topo





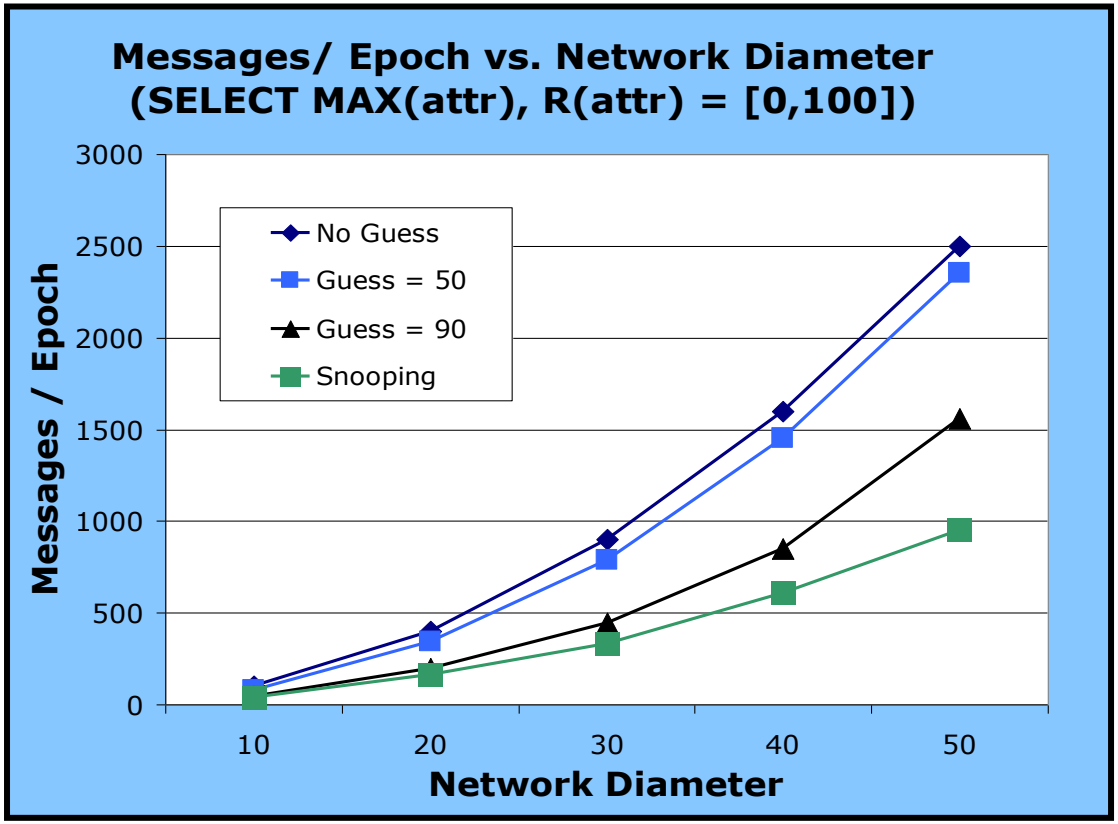
Evaluation & Optimization

- Optimization: Snooping
  - Utilize the Shared Channel to further reduce data transmission



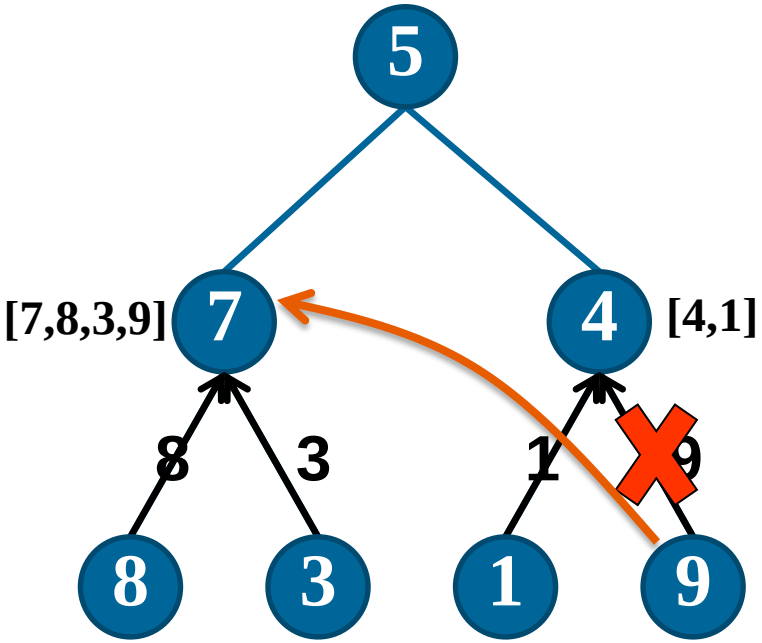
Evaluation & Optimization

- Optimization: Hypothesis & Test
  - Use some blind/statistical/subset guess to further reduce transmission



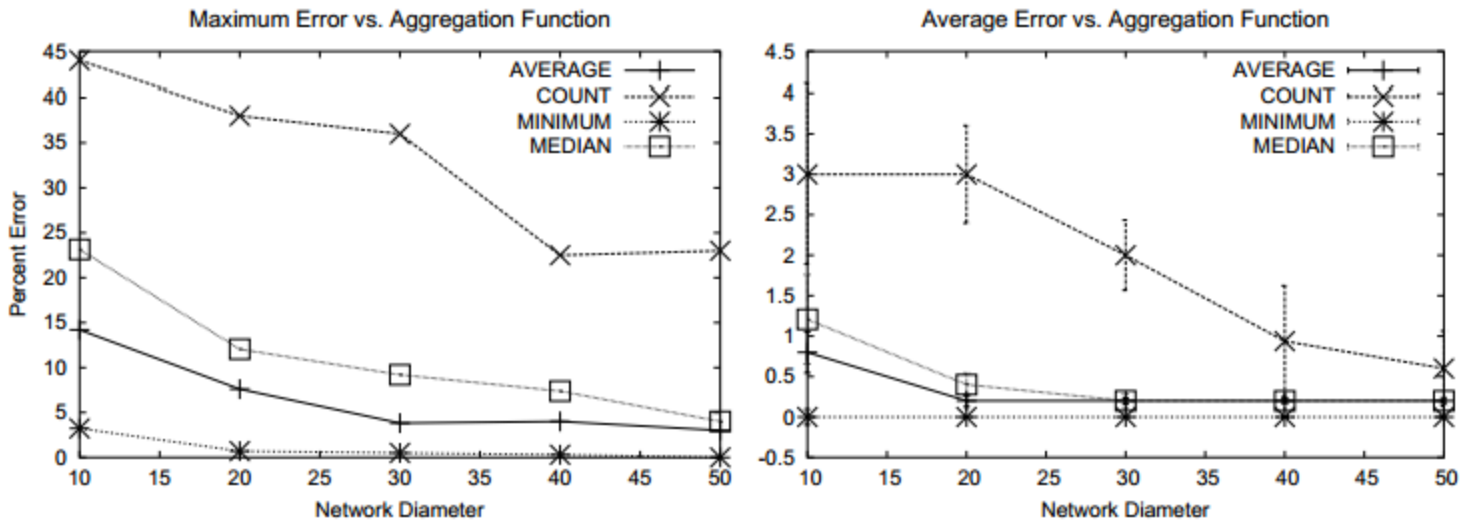
Evaluation & Optimization

- Optimization: Multiple Parents
  - Increased Reliability
    - Duplicate insensitive aggregates
    - Aggregates that can be expressed as a linear combination of parts



Evaluation & Optimization

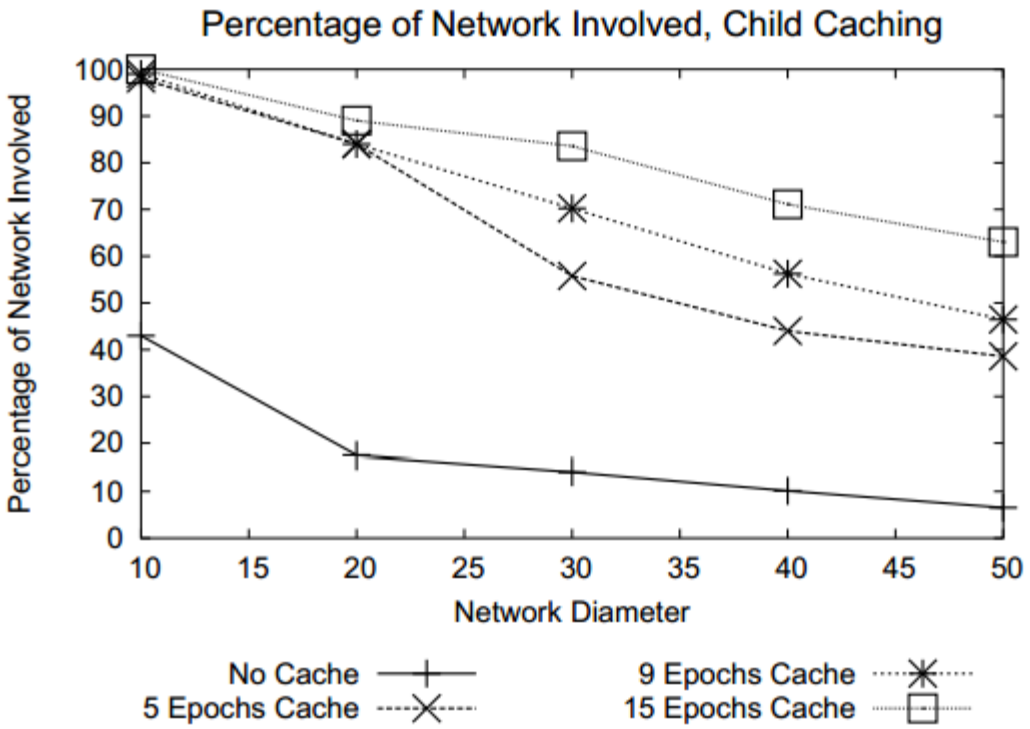
- Evaluation: Effect of Loss
  - Single loss



(a) Maximum Error (b) Average Error

Evaluation & Optimization

- Optimization: Child Cache
  - Increased Availability
    - Use old results when new results are not available



Evaluation & Optimization

- Optimization: Child Cache
  - Increased Availability
    - Use old results when new results are not available

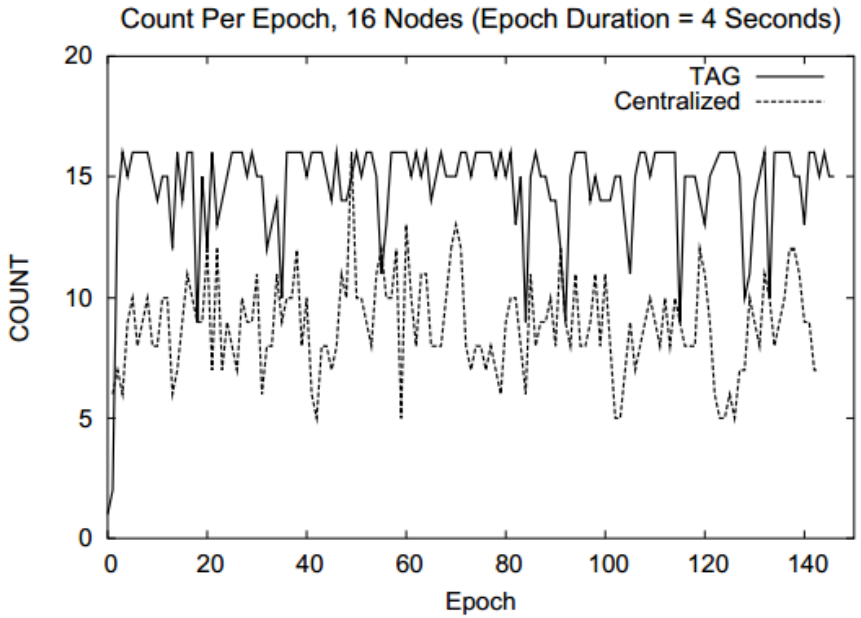


Figure 8: *Comparison of Centralized and TAG based Aggregation Approaches in Lossy, Prototype Environment Computing a COUNT over a 16 node network.*

Conclusion

- Contributions:
  - Declarative Query Framework for Sensor Networks
  - In Network Aggregation and Optimization based on Semantics
- Comments/Discussion Questions :
  - Time Synchronization in Epoch
  - Tree Based Topo is failure prone
  - Real Deployment
  - Malicious Nodes

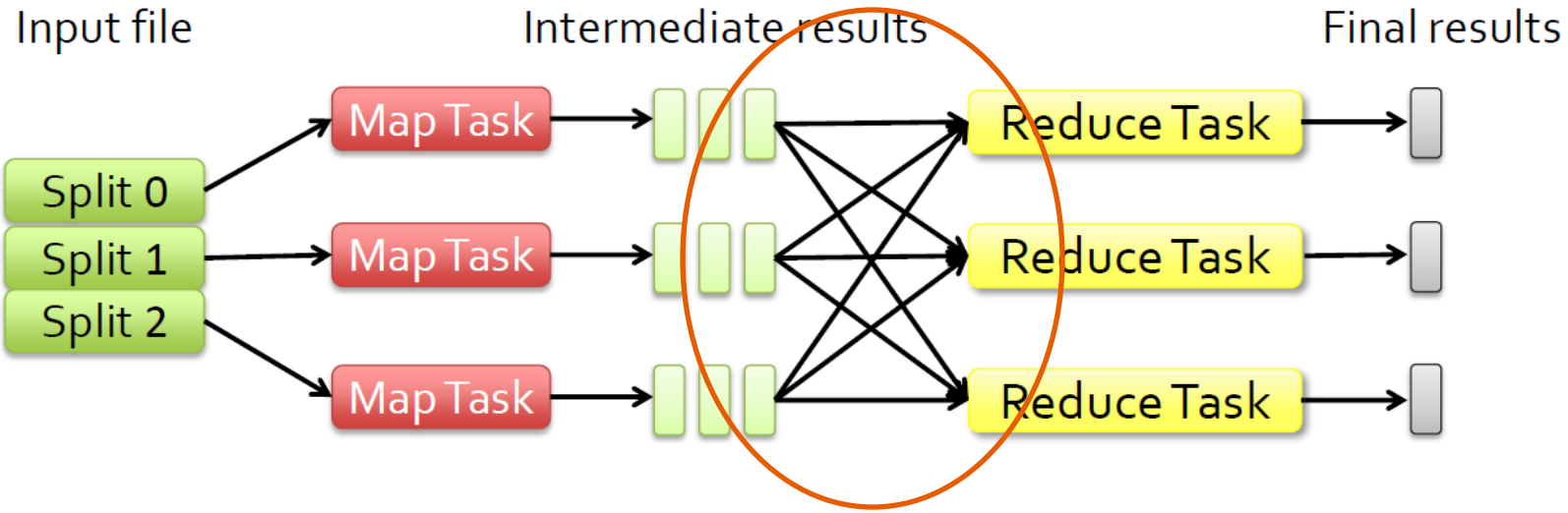
Camdoop : Exploiting In-network Aggregation for Big Data Applications

Austin Donnelly, Antony Rowstron, and Greg O'Shea

Presented by Mo Dong

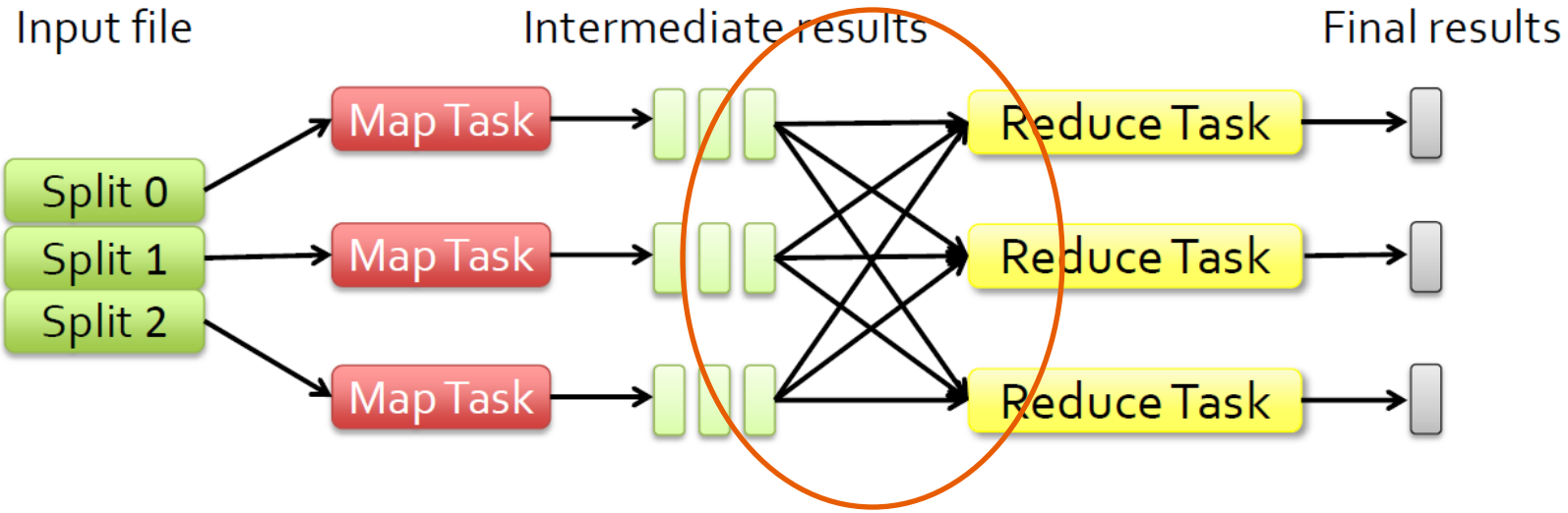
Why Camdoop

- Shuffle Phase Involved All to All traffic



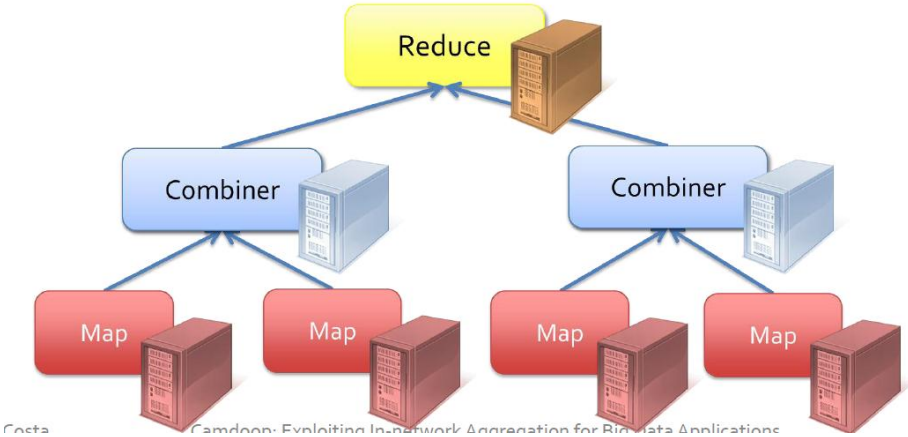
Why Camdoop

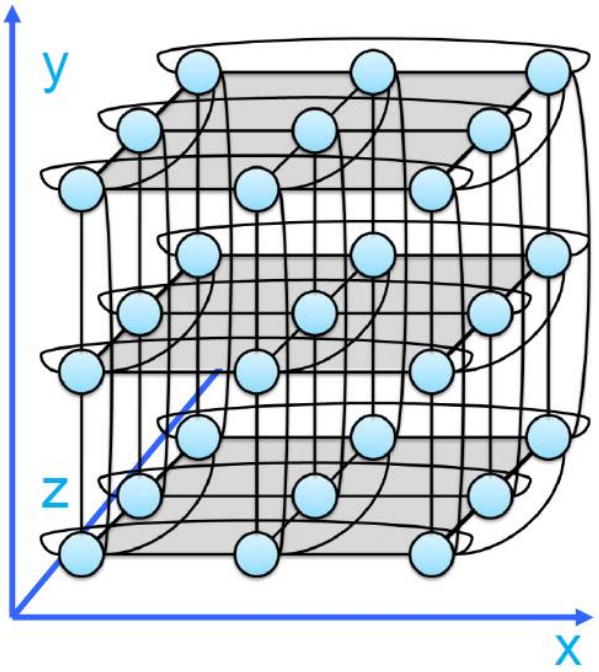
- Possible to Reduce Intermediate Data In Network



What is Camdoop

- Reduce Amount of Traffic using Camcube Topo



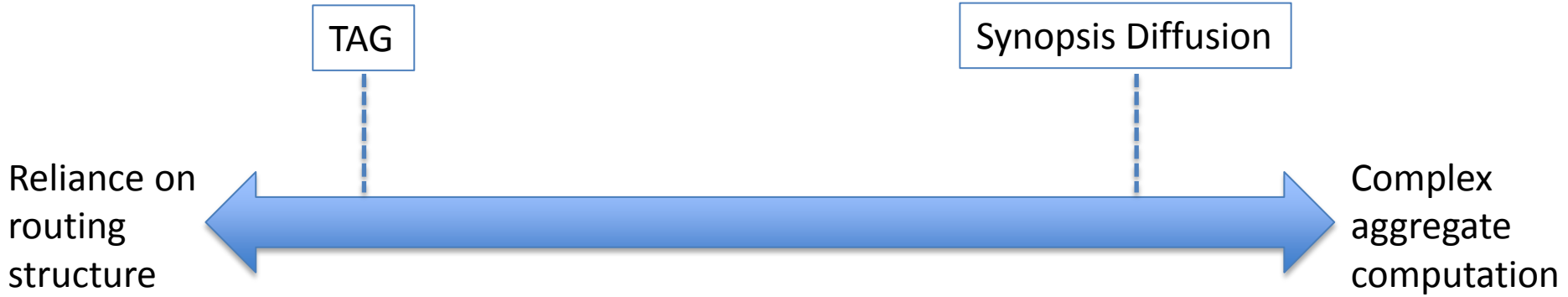


Synopsis Diffusion for Robust Aggregation in Sensor Networks

Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson

Presented by Hongyang Li

Motivation



- Problem with too much reliance on routing structure
  - Dynamic network condition due to node or link failure
  - High maintenance overhead
  - Difficult to find the best routing structure for all network conditions
- Synopsis Diffusion Approach
  - Less reliance on routing structure (though a good structure still helps)
  - Aggregate computation is more involving

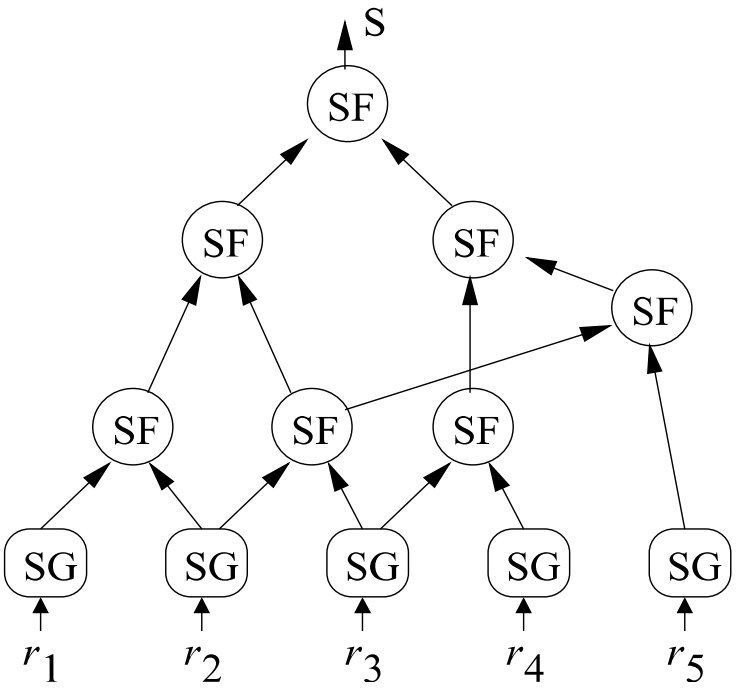
Synopsis Diffusion

- Less reliance on routing
  - Communication order doesn't matter: sensor A could report either before or after sensor B
  - Duplicate reading doesn't matter: sensor A can send 1 or 10 copy of its reading and the final answer remains the same
- Synopsis Diffusion is Order- and Duplicate-Insensitive (ODI)

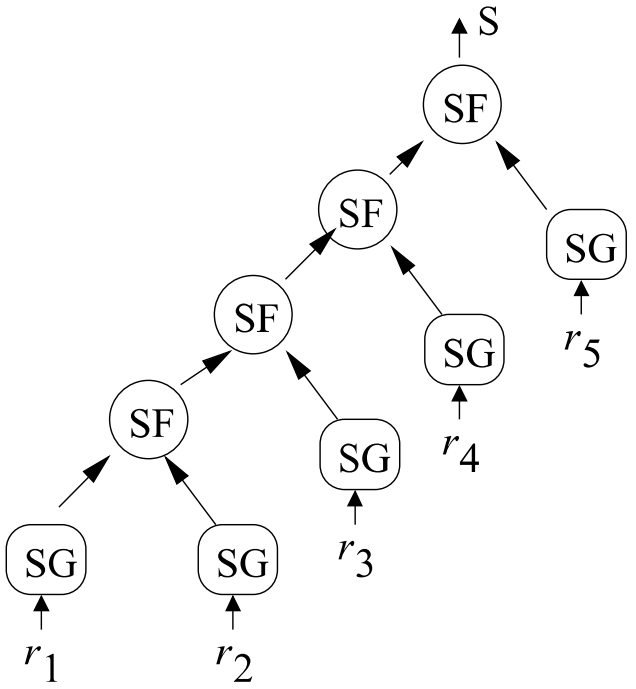
Synopsis Diffusion: 3 operations

- SG(r): generate a synopsis from sensor reading r
- SF(s1,s2): combine two synopses s1 and s2
- SE(s): evaluate synopsis s into final answer

Synopsis Diffusion: ODI



(a) Aggregation DAG



(b) Canonical left-deep tree

Check ODI

- SG(r1) = SG(r2) for duplicate readings r1, r2
- SF(s1,s2) = SF(s2,s1)
- SF(s1,SF(s2,s3)) = SF(SF(s1,s2),s3)
- SF(s,s) = s

Example: (Inefficient) Count

- Suppose there are N nodes, each voting either 0 or 1. We want to count how many nodes have voted 1.
- Synopsis s: N-bit array
- SG(r) of node i: set the i-th bit to 1 iff r > 0; set all other bits to 0.
- SF(s1,s2): s1 OR s2
- SE(s): count the number of 1's in s

Example: Probabilistic Counting

- Suppose each node voting 1 generate a random number.
  - The probability that the binary representation of random number ends with “0” is 1/2
  - The probability of ending with “00” is 1/4
- Think in the other direction
  - If some node outputs a binary random number ending with “0”, then with high probability there are about 2 nodes.
  - If some node outputs a binary random number ending with “00”, then with high probability there are about 4 nodes.
- Conclusion: the number of nodes is proportional to 2^i, where i is the length of the longest “0” tails

Example: Probabilistic Counting

- Uses k > log(N)-bit array as synopsis instead of N-bit array
  - CT(x): toss a fair coin for maximally x times, output the first time that heads came up, or x if no heads came up.
  - SG: Output a bit vector of length k with only CT(k) bit set
    - Set bit 1: I see a trail of "0"
    - Set bit 2: I see a trail of "00"
    - Set bit 3: I see a trail of "000"
  - SF(s1, s2): s1 OR s2
  - SE(s): For the smallest i such that s[i]=0, output 2^(i-1)/0.77351

Example: Uniform Sampling

- Synopsis: K tuples <value, rand, id>
- SG: each node i generate a random number and construct <value_i, rand_i, i>
- SF(s1, s2): keep the K tuples <value, rand, id> with the highest rand
- SE(s): output all the value fields in <value, rand, id>

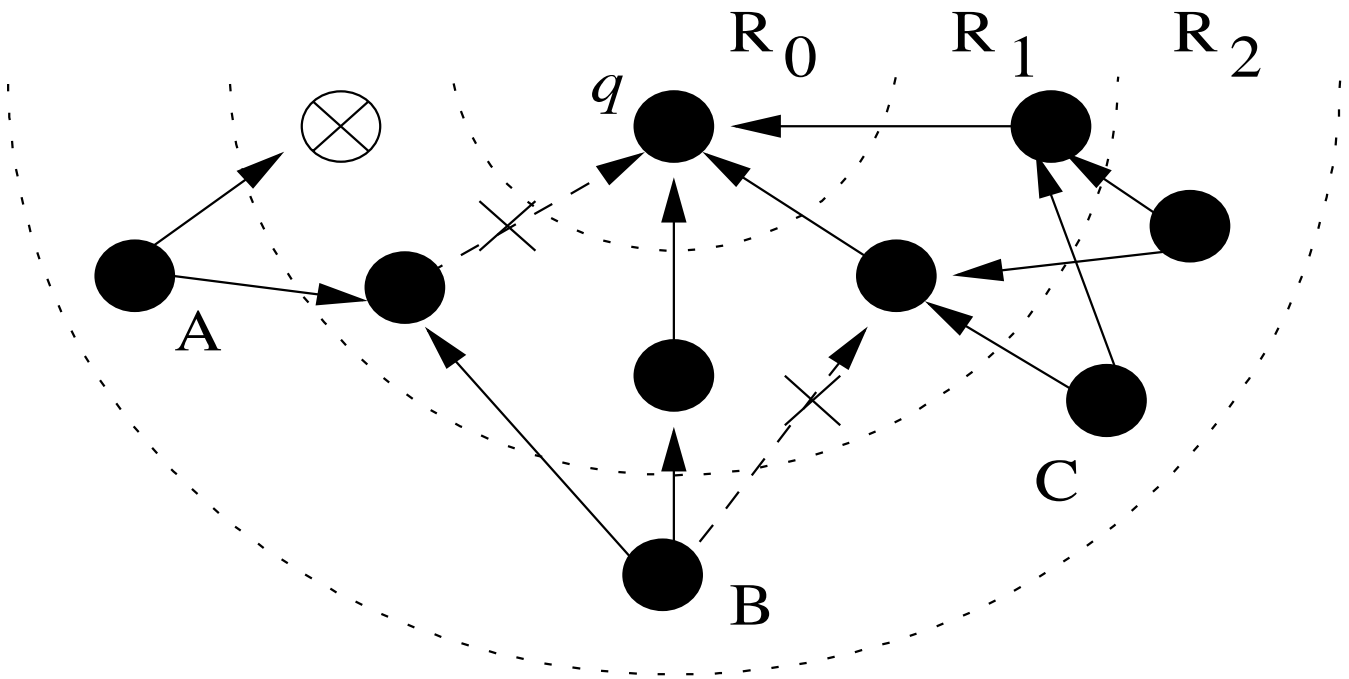
Uniform Sampling is ODI

- SG(r1) = SG(r2) for duplicate r1,r2: trivial
- SF(s1,s2) = SF(s2,s1)
  - the K tuples with highest rand in s1 and s2
- SF(s1,SF(s2,s3)) = SF(SF(s1,s2),s3)
  - the K tuples with highest rand among s1,s2, and s3
- SF(s,s) = s
  - the K tuples with highest rand in s

Implicit Acknowledgement

- Suppose A sends its reading to B for aggregation. How does A know that B has received its reading?
- Explicit ack: waste energy/bandwidth
- Implicit ack:
  - A sends x
  - B aggregates x with y and obtains z = SF(x,y)
  - We must have x <= z (<= as defined in the context)

Routing Structure: Ring

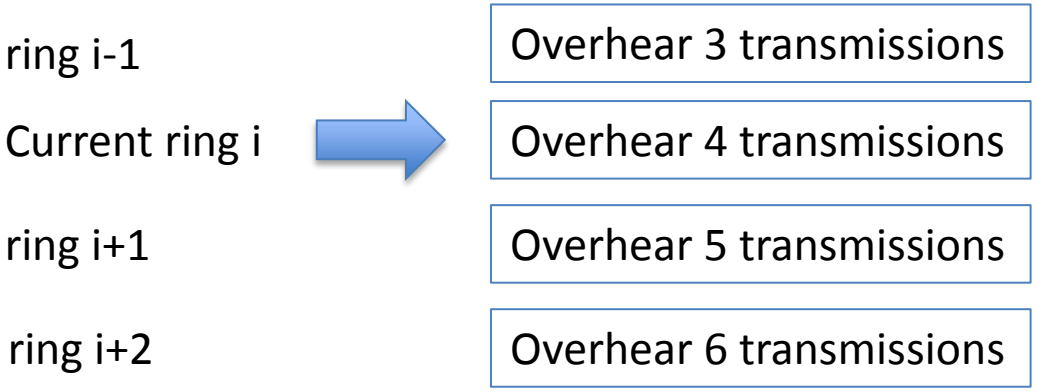


Routing Structure: Adaptive Ring

- Node can use implicit ack to check whether transmission is successful
- In case of unsuccessful transmission
  - Retransmit: waste energy/bandwidth/time
  - Adaptive Ring: choose another set of parents with better connectivity

Routing Structure: Adaptive Ring

- When to switch to a new ring?
  - If the number of times that any node in my parent ring retransmits my synopsis falls below a threshold
- How to switch to a new ring?
  - Wake up at the corresponding time slot
- Which ring to switch to?



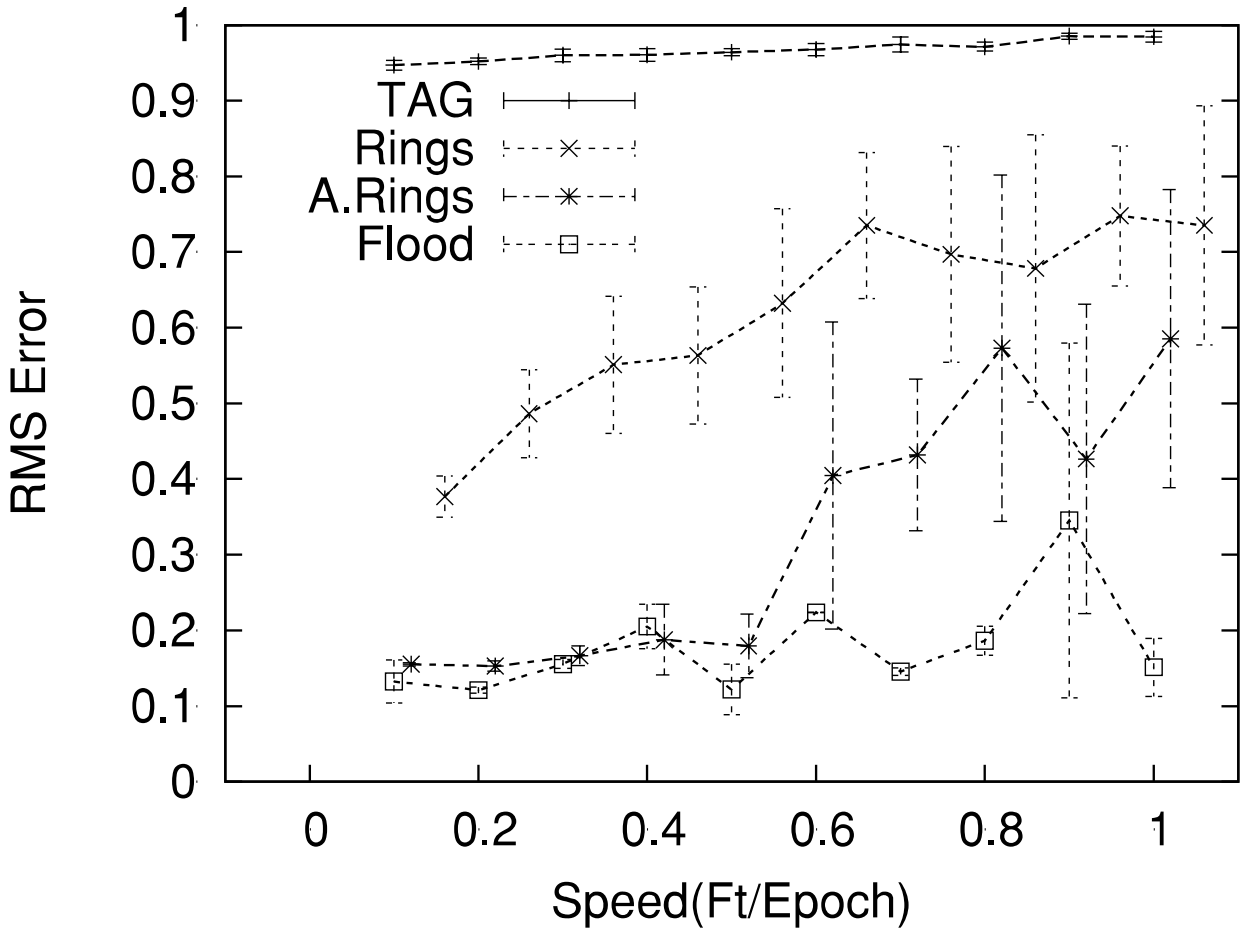
- More transmissions overheard as ring depth i increases: moving down is probably a good choice
- After moving to ring i+1, the node has better connectivity with its parents

Evaluation

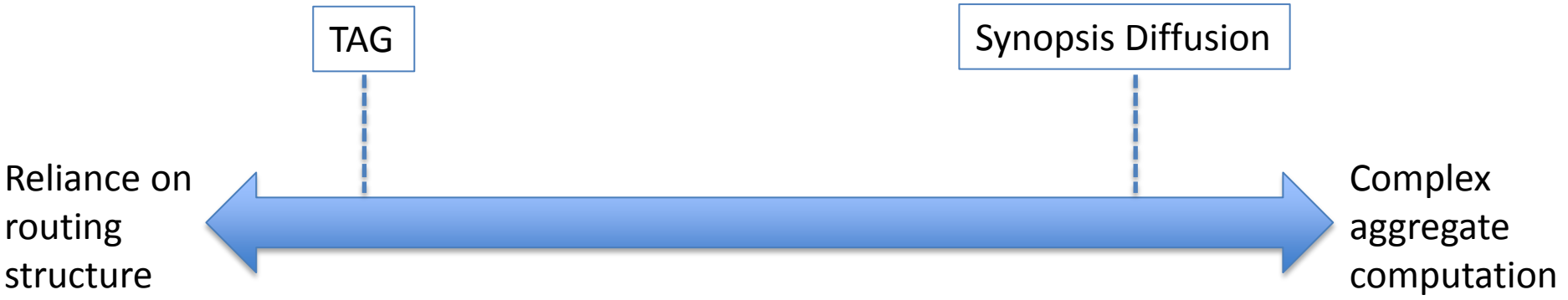
Table III. Comparison of Aggregation Schemes

Scheme	% Nodes	Error (uniform)	Error (skewed)	Error (Gaussian)
TAG	$< 15\%$	0.87	0.99	0.94
TAG2	N/A	0.85	0.98	0.92
Gossip	N/A	0.91	0.99	0.93
Rings	65%	0.33	0.19	0.21
Adapt. Rings	95%	0.15	0.16	0.15
Flood	$\approx 100\%$	0.13	0.13	0.13

Evaluation


RMS Error
Speed(Ft/Epoch)
TAG
Rings
A.Rings
Flood
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1

Conclusion


TAG
Synopsis Diffusion
Reliance on routing structure
Complex aggregate computation

Discussion Points

- Can every aggregation task be written in an ODI style?
- Is aggregation error always a bad thing?