Saint-Petersburg State University of Aerospace Instrumentation

Load balancing routing algorithm for data gathering sensor network

Bakin E., Evseev G., Dorum D.

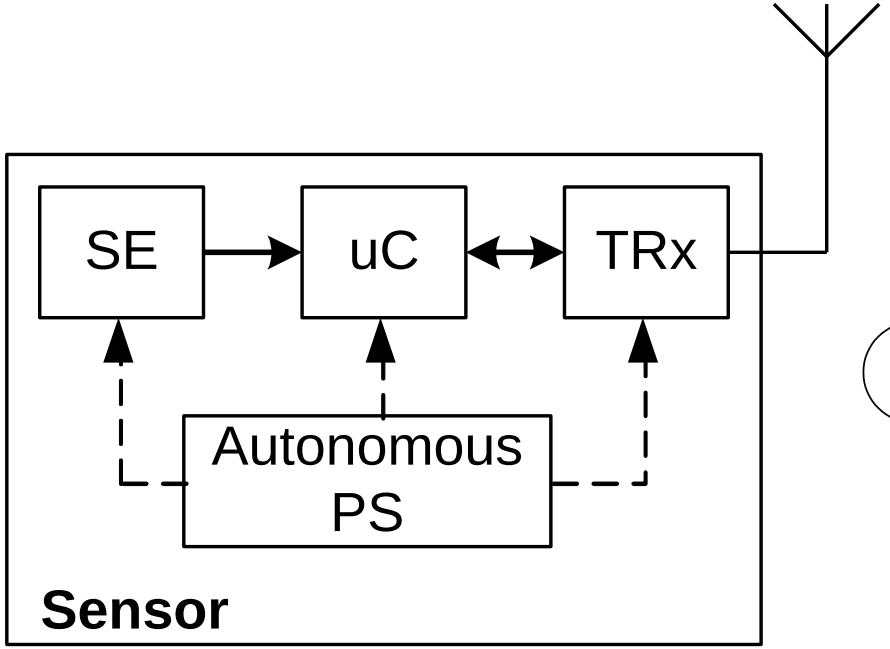
2012

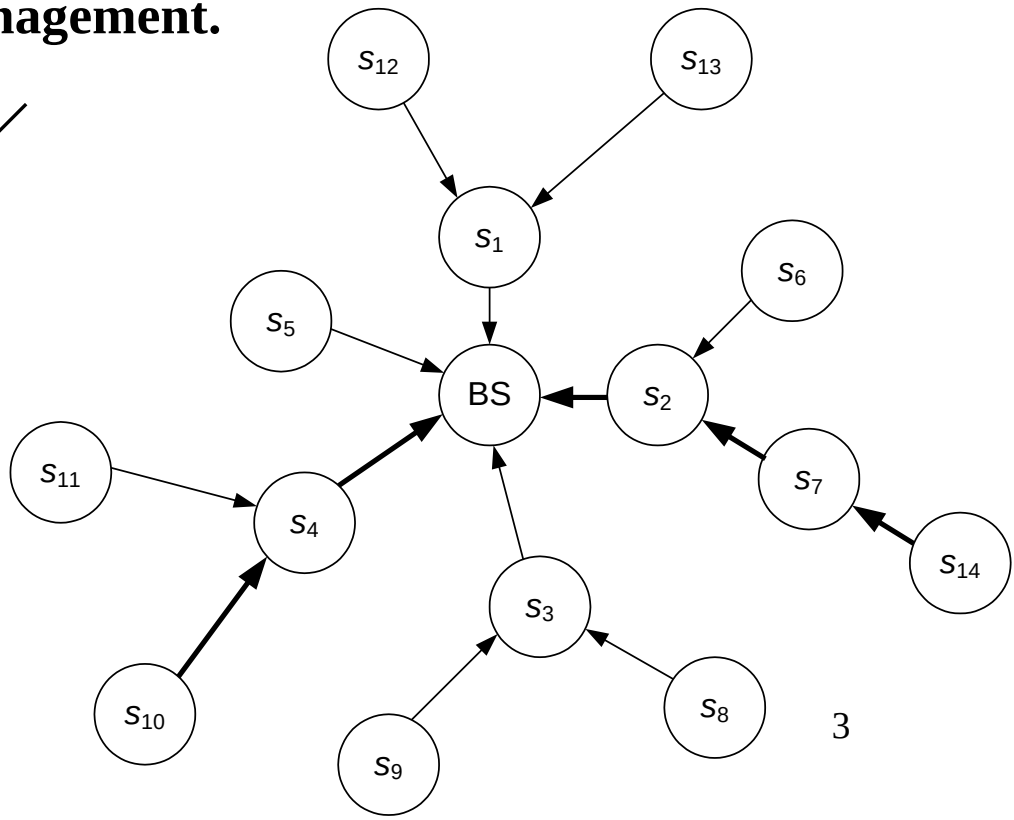
Agenda

- Data gathering sensor networks description
- Cyclic data gathering mechanism
- Problem of bottlenecks in routes
- Classic Dijkstra algorithm
- Modified Dijkstra algorithm
- Analysis of algorithm effectiveness
- Conclusion

Data gathering sensor network

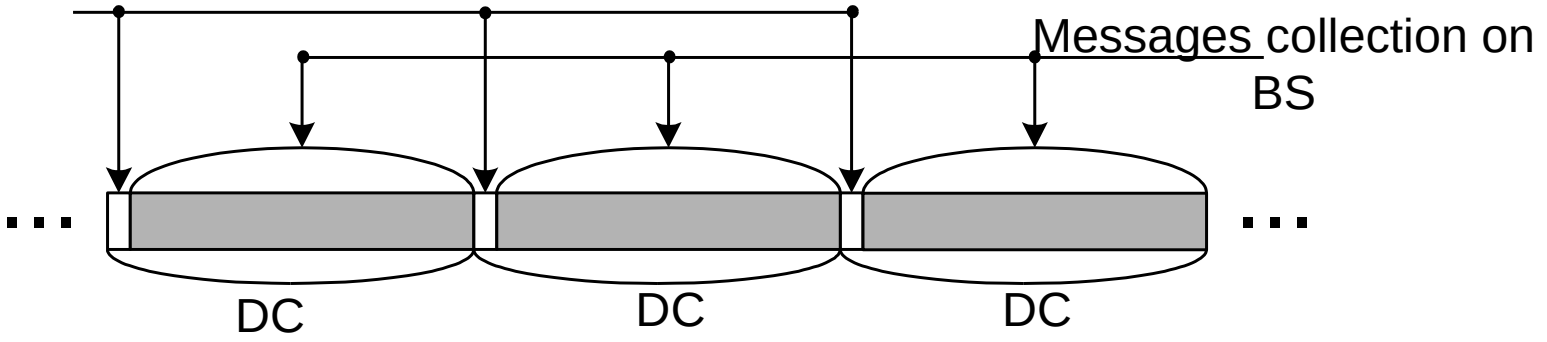
1. **Consists of a set of identical elements (sensors) and base station (BS).**
2. **Sensor purpose: environment parameters measurement, transmission of messages to base station, relaying of messages, going from other sensors.**
3. **BS purpose: gathering, processing and storage of information, coming from sensors, transmissions management.**





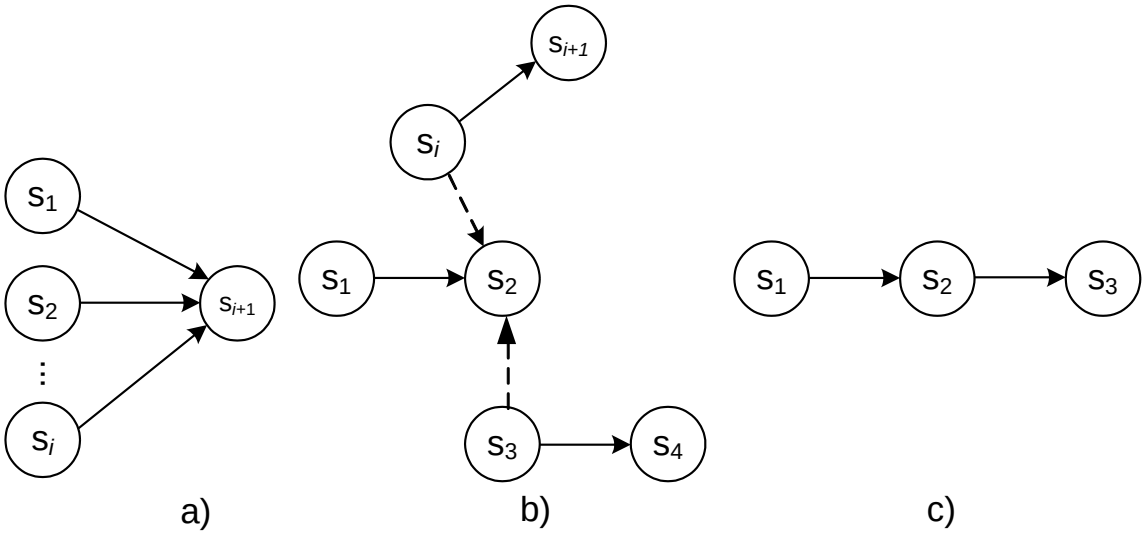
Cyclic data gathering

Each sensor forms a message

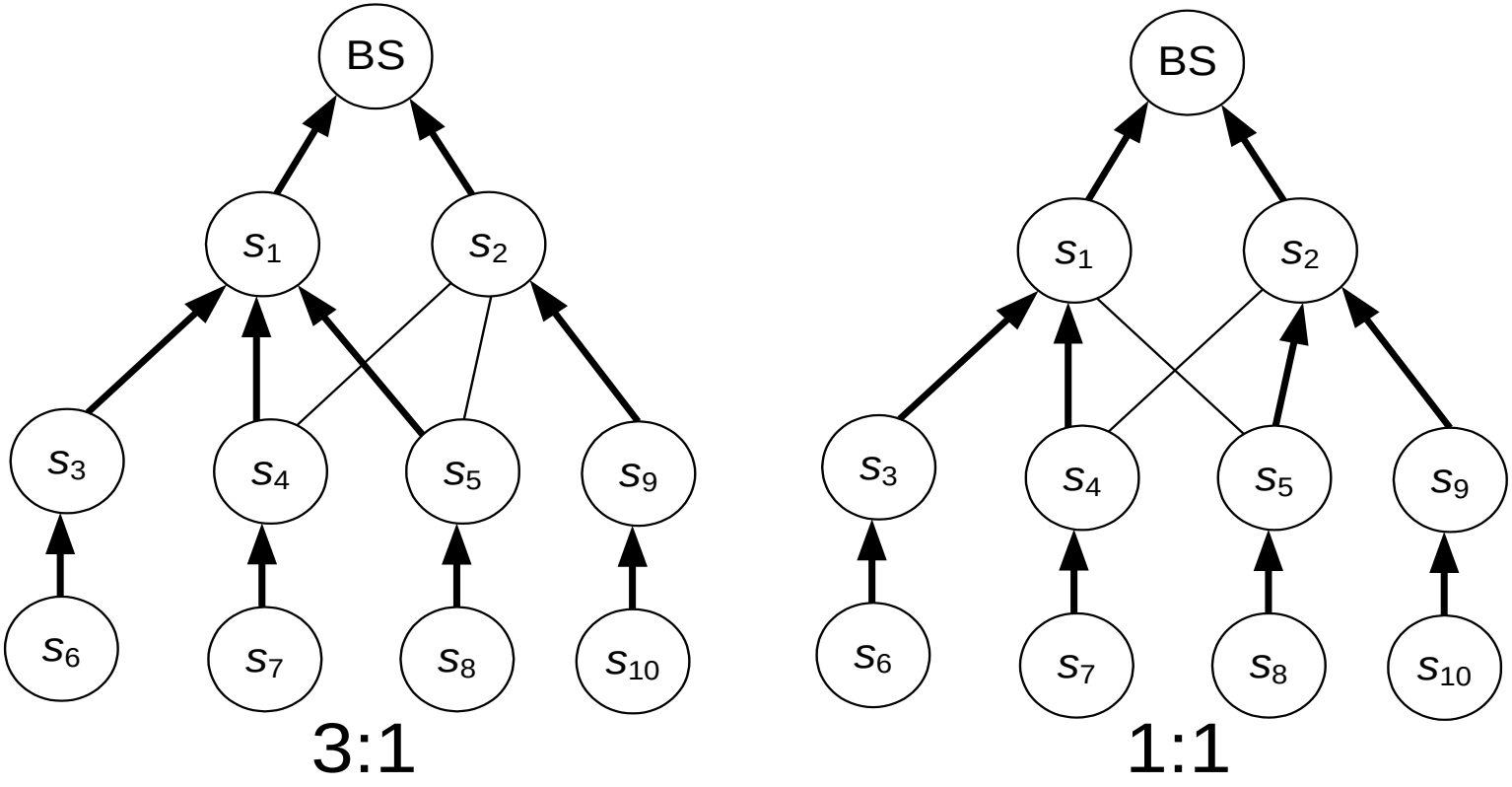


Duty cycle duration $\Rightarrow$ min

Collisions…



Bottlenecks in routes



Unbalance in routes lead to:

- **collision number increasing**
- **duty cycle duration increasing**
- **frequent discharge of particular sensors batteries**

Requirements to routing algorithm

- Balancing of traffic load between sensors (minimization of number of bottlenecks in routes).
- Minimization of quantity of transmissions during duty cycle (thus minimization of total energy consumption of the network). **This can be achieved with routing through the shortest path.**
- Low computation complexity.

Classical approach: Dijkstra algorithm

$\boldsymbol{S} = \{s_0, s_1, \ldots, s_N\}$ – set of sensors in the network

$\boldsymbol{L} = \{\ldots, l_{i,j}, \ldots\}$ – set of links in network. $l_{i,j} \in \boldsymbol{L}$ if reliable channel exists between sensors s_i and s_j

$w_{i,j}$ – weight of link $l_{i,j}$

When using classic Dijkstra algorithm for routing:

•Algorithm calculates the shortest path from sensor to BS minimizing sum of weights of links, used in a route.

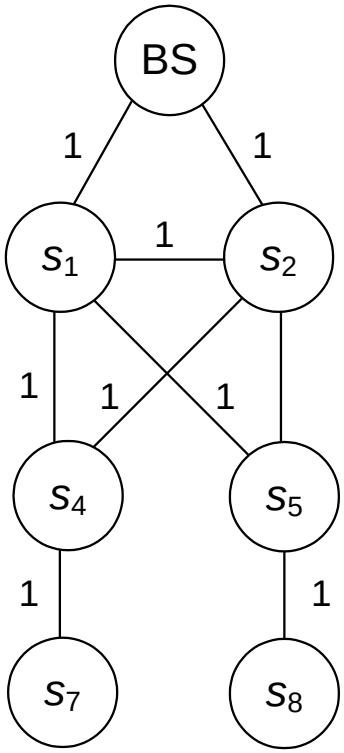
•Weights $w_{i,j}$ do not vary during algorithm work and usually equal to each other.

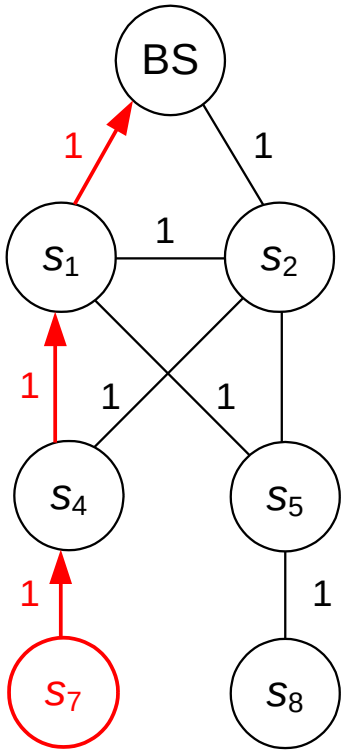
• Routes form a tree with a root in BS, thus loads are highly unbalanced.

Notation: $r_i = \text{Dijkstra}(s_i)$

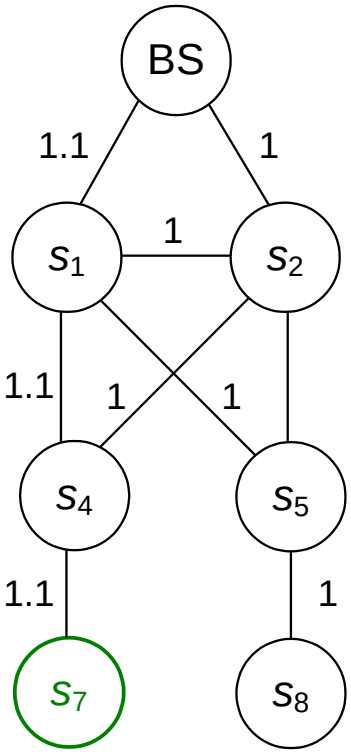
Modified Dijkstra algorithm: main idea

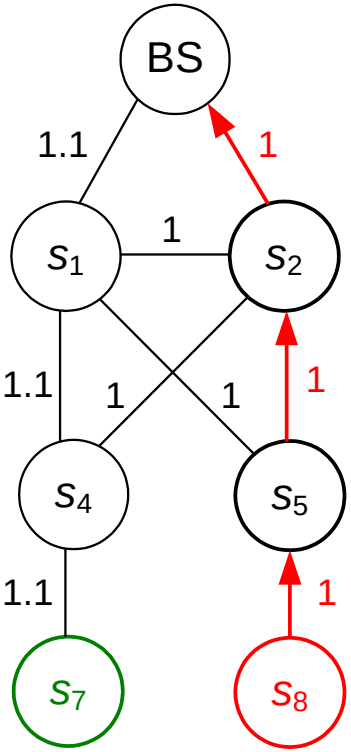
1. Initially all the weights $w_{i,j} = 1$.
2. Routes are calculated for all the sensors one-by-one.
3. After each iteration, weights of each link in calculated route are increased on Δw, thus increasing “fee” for these links usage in further routs.



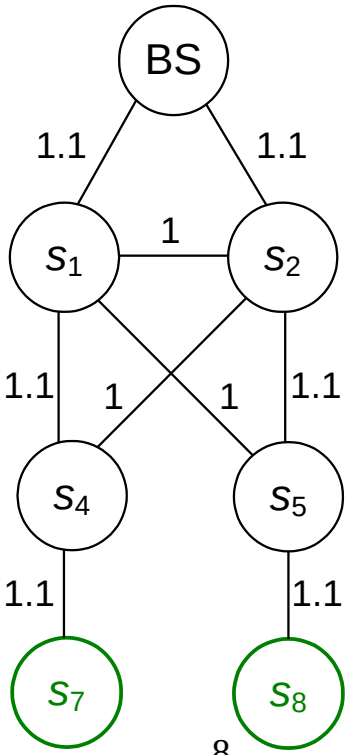


$S_7 > S_4 > S_1 > BS$



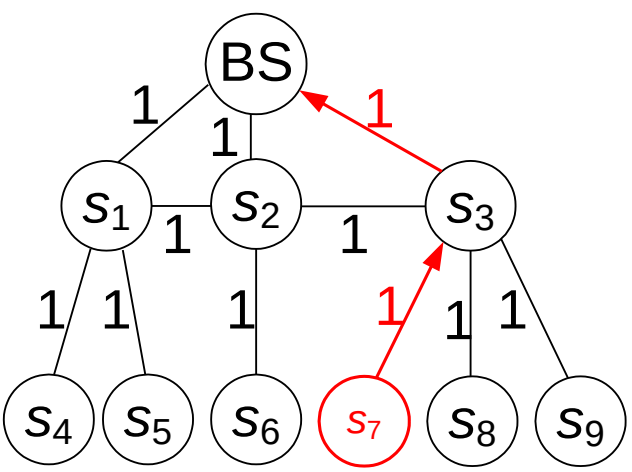


$S_8 > S_5 > S_2 > BS$

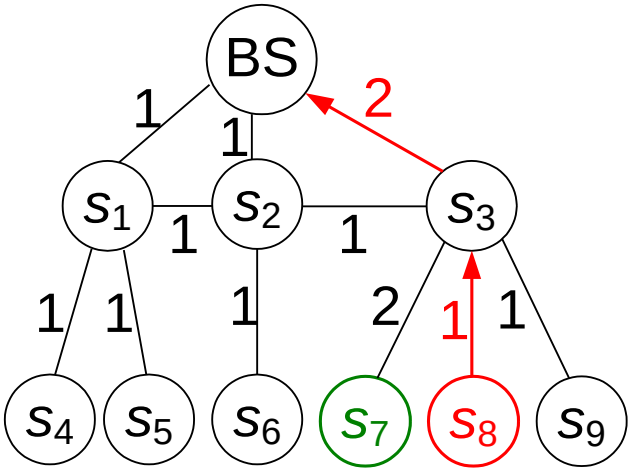


How to choose Δw correctly?

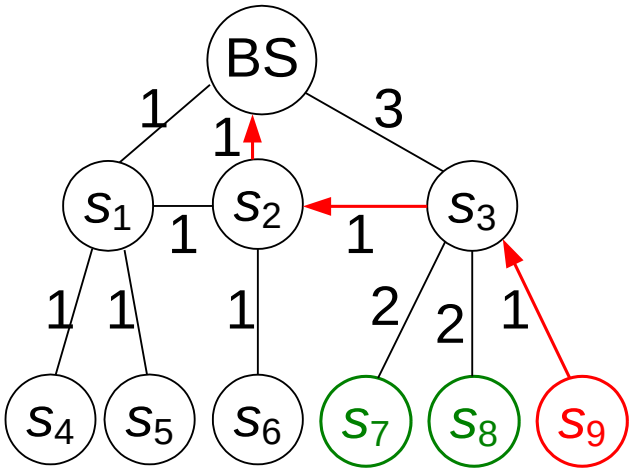
Sample. If $\Delta w = 1$:



$S_7 > S_3 > BS$



$S_8 > S_3 > BS$



$S_9 > S_3 > S_2 > BS$

One extra transition!

If Δw is chosen incorrectly total number of transitions can grow sufficiently.

Lemma…

If Δw is chosen less or equal to $1/N^2$ (here N is number of sensors in the network), than all the routes, calculated with proposed algorithm would go through the shortest path.

Proof…

After each iteration any link's weight is increased on value $1/N^2$. Thus after all N iterations any link's weight would increase no more than on $N/N^2 = 1/N$. Since the longest possible route contains N-1 links, total links weight wouldn't increase on more than $(N\text{-}1)/\text{N} < 1$. And this value is less than initial weight of the link.

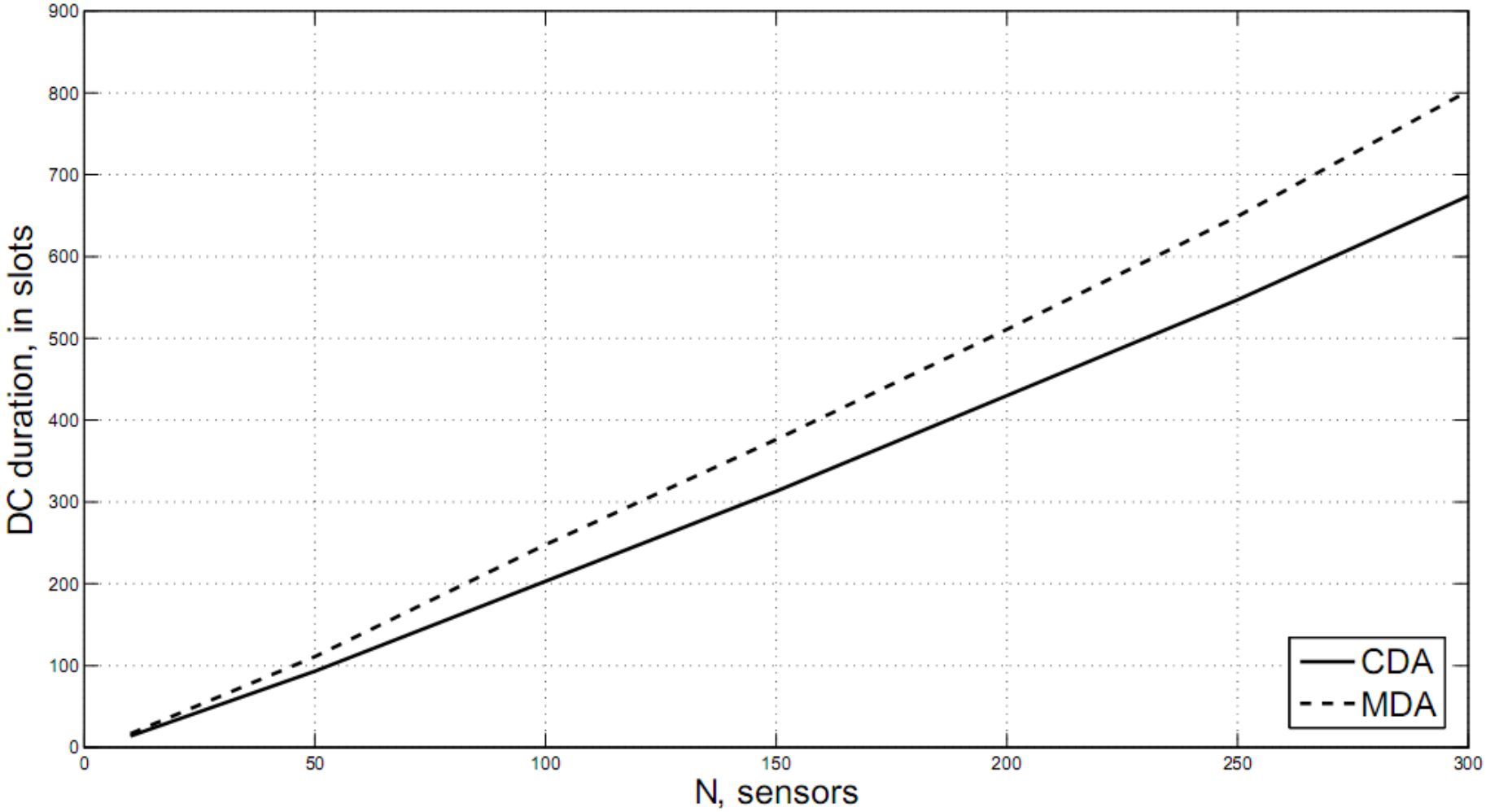
Algorithm pseudocode

$\forall w \leftarrow 1$
for $k = 1, 2, \cdots N$ **do**
  $r_k \leftarrow \text{Dijkstra}(s_k)$
  for $\forall l_{i,j} \in r_k$ **do**
    $w_{i,j} \leftarrow w_{i,j} + 1/N^2$
  end for
end for

Simulation (1/1)

1. Generate random graph of sensor network with given size N.
2. Calculate the routes by means of CDA and MDA.
3. For both cases calculate the collision free duty cycle schedule with any scheduling algorithm.
4. Repeat the procedure 1-3 for 10^4 times.

Simulation (1/2)



Conclusion

1. Proposed algorithm allows traffic load balancing of routes.
2. Traffic load balancing performed with proposed algorithm decreases duty-cycle duration approximately on up to 20%.
3. Proposed algorithm complexity is low and is polynomial of N.

Thank you!
Any questions?