A New Dynamic Frame Slotted Aloha Anti-Collision Algorithm for the Internet of Things

Rafael Perazzo Barbosa Mota

IME - USP

05 de junho de 2013
Agenda

1. Motivation and relevance
2. Background
3. Related Work
4. Our Proposal
5. Results and discussion
6. Conclusions
7. References
Table of Contents

1 Motivation and relevance
2 Background
3 Related Work
4 Our Proposal
5 Results and discussion
6 Conclusions
7 References
Radio Frequency Identification (RFID) is a key technology of IoT since small passive RFID tags make it possible to link millions and billions of physical products with Internet [1].
Therefore, RFID tag anticollision algorithms will play an important role in IoT [1].
A New Dynamic Frame Slotted Aloha Anti-Collision Algorithm for the Internet of Things

Motivation and relevance

Background - DFSA

START

Set frame size L, slot=0

Send Query command and current slot

Tags replies? 0 1

idle++, slot++
collisions++, slot++

End of Frame?

Send ACK, slot++

Calculate new Frame Size, slot=0

Schoute: L=2.39*collisions
Lower Bound: L=2*collisions
L: Frame size

No

End of Frame?
# Table of Contents

1. Motivation and relevance
2. Background
3. Related Work
4. Our Proposal
5. Results and discussion
6. Conclusions
7. References
Dynamic Frame Slotted Aloha - DFSA

**Algorithm 1** DFSA algorithm

**Require:** $L$  ▷ $L$ is the initial frame size

1. $continue \leftarrow true$
2. $n \leftarrow L$
3. **repeat**  ▷ While collisions occurs
4. 
5. $i \leftarrow 0$  ▷ Initial slot time
6. $counter \leftarrow 0$  ▷ Number of received replies (=1, =0 or >1)
7. $collisions \leftarrow 0$  ▷ Collision counter
8. **for** $i \leq n$ **do**  ▷ Sends every slot time
9. 
10. Query$(n,i)$  ▷ Sends a Query Command with frame size $n$ and slot $i$
11. Wait for reply
12. 
13. **if** $(counter == 1)$ **then**
14. 
15. QueryRep()  ▷ Reader sends an ACK to identify the tag
16. **else if** $(counter > 1)$ **then**
17. 
18. $collisions \leftarrow collisions + 1$
19. **end if**
20. **end for**
21. 
22. **if** $(collisions == 0)$ **then**
23. 
24. $continue \leftarrow false$
25. **else**
26. 
27. $n \leftarrow Call a function to calculate the next frame size$
28. $L \leftarrow n$
29. **end if**
30. **until** $(continue==true)$
# Table of Contents

1. Motivation and relevance
2. Background
3. Related Work
4. Our Proposal
5. Results and discussion
6. Conclusions
7. References
Q Algorithm [2]

```
Qfp = 4 and 0.1 \leq c \leq 0.5

Q = \text{round}(Qfp)

Reader issues query command

Tags receive query command

Command type

Query

QueryReply

Update Q

Set Q

Re-select Slot

Slot = Slot - 1

Re-select Slot

Tags reply if Slot = 0

Reader waits (2* seconds) for replies from tags

Qfp = \text{min} (10, Qfp + C)

Collision

Any reply?

No reply

Qfp = \text{max} (0.0f - C)

Unique reply

Qfp = Qfp + 0

* \( Q \) = 0.001 (slot time)

\( \text{Lma} (\text{This slot time can be different according to the reader hardware}) \)
```
Algorithm 2 Schoute algorithm

1: function Schoute(collisions )
2:     return round(2.39 * collisions)
3: end function
Algorithm 3 Mota algorithm

1: function mota(collisions)
2: return round(2.62 * collisions)
3: end function
Eom-Lee [4]

**Algorithm 4** Eom-Lee Estimation Algorithm

1. **function** estimation_eomlee( $\epsilon$, collisions, success ) ▷

   collisions and success are the number of collision and success slots in last frame, respectively. $\epsilon$ is the stop criteria

2. $b_1 \leftarrow \infty$

3. $y_1 \leftarrow 2$

4. backlog $\leftarrow L$

5. repeat

6. $b_{prox} \leftarrow \frac{backlog}{y_1 \ast collisions + success}$

7. $y_{prox} \leftarrow \frac{1 - e^{-\frac{1}{b_{prox}}}}{b_{prox} \ast (1 - (1 + \frac{1}{b_{prox}}) \ast e^{-\frac{1}{b_{prox}}})}$

8. backlog $\leftarrow y_{prox} \ast collisions$

9. temp $\leftarrow y_1$

10. $y_1 \leftarrow y_{prox}$

11. $b_1 \leftarrow b_{prox}$

12. until $(|y_1 - temp| < \epsilon)$

13. **return** round(backlog)

14. **end function**
Dynamic, Adaptative and Splitting BTSA (Excellent System Efficiency. Many changes must be done on tags operation) [1]
Others

- Dynamic, Adaptative and Splitting BTSA (Excellent System Efficiency. Many changes must be done on tags operation) [1]
- Vogt (equivalent to Q Algorithm) [5]
Others

- Dynamic, Adaptative and Splitting BTSA (Excellent System Efficiency. Many changes must be done on tags operation) [1]
- Vogt (equivalent to Q Algorithm) [5]
- Chen (worse than Q Algorithm) [6]
Table of Contents

1 Motivation and relevance
2 Background
3 Related Work
4 Our Proposal
5 Results and discussion
6 Conclusions
7 References
Our Proposal

EDFSA-I

START

Estimate initial frame size $L$
slot=0

Send Query command and current slot

Tags replies?

idle++
slot++

0

>1

End of Frame?

End of Frame?

collisions++
slot++

Send ACK slot++

Calculate new Frame Size using Eom-Lee method slot=0

Yes

No

Yes

No
Our Proposal

EDFSA-II

1. **START**
   - Estimate the initial frame size
   - All tags select a number between 1 and the estimated number

2. **Next slot**
   - Send Query with slot 1
   - Wait for replies

3. **collisions++**
   - Number of replies?
     - >1
     - =1
       - QueryRep()
       - Next slot

4. **resolve_collisions()**
   - Next slot

5. **End of frame?**
   - No
     - End
   - Yes
     - End of frame

6. **END**
Our Proposal

EDFSA-II

Algorithm 5 Estimated Dynamic Framed Slotted Aloha - EDFSA

Require: \( \epsilon \) \( \triangleright \) \( \epsilon \) is the stop criteria

1. \( L \leftarrow estimation(1,3) \times 0.67 \)
2. \( i \leftarrow 1 \) \( \triangleright \) Initial slot time
3. \( counter \leftarrow 0 \) \( \triangleright \) Number of received replies (=1, =0 or >1)
4. \( collisions \leftarrow 0 \) \( \triangleright \) Collision counter
5. for \((i = 1; i \leq L; i \leftarrow i + 1)\) do \( \triangleright \) Sends every slot time
6. Query(L,i) \( \triangleright \) Sends a Query Command with frame size \( n \) and slot \( i \)
7. Wait for reply
8. if \((counter == 1)\) then
9. QueryRep() \( \triangleright \) Reader sends an ACK to identify the tag
10. \( success \leftarrow success + 1 \)
11. else if \((counter > 1)\) then
12. \( collisions \leftarrow collisions + 1 \)
13. \( resolve\_collisions() \quad \triangleright \) Collisions are resolved as soon as they occur
14. end if
15. end for
How to estimate the initial frame [7]?

1. **START**
   - collisions=0
   - idle=0
   - success=0

2. Send i times QueryEst command

3. If collisions=i or idle=i
   - Yes: Q++
   - No: Q--

4. If FinalQ=Q
   - Send i times QueryEst
   - Yes: FinalQ is the estimated number of tags
   - No: END

5. END

FinalQ is the estimated number of tags.
A New Dynamic Frame Slotted Aloha Anti-Collision Algorithm for the Internet of Things

Our Proposal

How to resolve local collisions?

START

L = 3

Next slot
Send Query with slot i
Wait for replies

> 1: collisions++

Number of replies?

= 1: QueryRep()
Next slot
success++

No: End of frame?

Yes: collisions = 0?

Yes: L = mota(collisions)

END
How to select best parameters for estimation?

![Graph showing best initial Q value vs number of tags for different parameter combinations](image-url)

- $c=1$ and $i=3$
- $c=1$ and $i=5$
- $c=0.3$ and $i=3$
- $c=0.2$ and $i=3$
- $c=0.2$ and $i=5$
How to select best parameters for estimation?

![Graph showing delay against number of tags for different parameter combinations]

- $c=1$ and $i=3$
- $c=1$ and $i=5$
- $c=0.3$ and $i=3$
- $c=0.2$ and $i=3$
- $c=0.2$ and $i=5$

Delay (slots) vs. Number of tags
How to select the best initial frame size for resolve local collisions?

- **Mean:** aprox. (2.616) 2.62 (2.633)
- **Sample size:** 88751 collisions slots
- **Confidence Interval (CI):** 99%

The graph shows the frequency distribution of the number of tags in collision for different frame sizes. The x-axis represents the number of tags in collision, while the y-axis represents the frequency. The data points are as follows:

- 2 tags: 60.2082%
- 3 tags: 25.0518%
- 4 tags: 9.5896%
- 5 tags: 3.3430%
- 6 tags: 1.1797%
- 7 tags: 0.4124%
- 8 tags: 0.1532%
- 9 tags: 0.0338%
- 10 tags: 0.0169%
- 11 tags: 0.0090%
- 12 tags: 0.0023%

The graph indicates that the most common collision scenario is with 2 tags, followed by 3 tags, and so on, down to the least common scenario with 12 tags in collision.
How to select the next frame estimation method?

-20% 0% 20% 40% 60% 80% 100% 120% 140% 160% 180% 200% 220% 240% 260% 280% 300% 320% 340% 360% 380% 400% 420% 440% 460%

Average Difference compared to Q Algorithm

Number of tags in collision

Initial Frame Size: 3

Lower Bound Schoute Eom–Lee Mota

PS. Negative number means a decrease on System Efficiency
PS. Positive number means a increase on System Efficiency
How to select the next frame estimation method?

![Graph showing system efficiency for different methods and number of tags in collision.](image-url)
Table of Contents

1 Motivation and relevance
2 Background
3 Related Work
4 Our Proposal
5 Results and discussion
6 Conclusions
7 References
System Efficiency

A New Dynamic Frame Slotted Aloha Anti-Collision Algorithm for the Internet of Things

Results and discussion

System Efficiency

Confidence Interval (CI) 95%

- Q Algorithm
- Schoute 128
- Eom–Lee 128
- EDFSA–I
- EDFSA–II

Number of tags vs. System efficiency
Results and discussion

Identification time (in timeslots)

![Graph showing identification time for different algorithms]
A New Dynamic Frame Slotted Aloha Anti-Collision Algorithm for the Internet of Things

Results and discussion

Comparison

![Comparison bar chart showing the average difference in number of slots (%) compared to Q Algorithm for different number of tags (100 to 1800) and varying number of slots (128, 256, 512, 1024, 2048). The chart compares Schoute, Eom-Lee, EDFSA-I, and EDFSA-II algorithms.]
Table of Contents

1 Motivation and relevance
2 Background
3 Related Work
4 Our Proposal
5 Results and discussion
6 Conclusions
7 References
Conclusions

- We presented two new anti-collision algorithms based on existing estimation methods;
Conclusions

- We presented two new anti-collision algorithms based on existing estimation methods;
- The EDFSA-I has a gain up to 12% related to Q Algorithm;
We presented two new anti-collision algorithms based on existing estimation methods;

- The EDFSA-I has a gain up to 12% related to Q Algorithm;
- The EDFSA-II has a gain up to 26% related to Q Algorithm while Splitting BTSA has a gain up to 23% and Adaptative, Dynamic has a gain up to 14%;
Conclusions

- We presented two new anti-collision algorithms based on existing estimation methods;
- The EDFSA-I has a gain up to 12% related to Q Algorithm;
- The EDFSA-II has a gain up to 26% related to Q Algorithm while Splitting BTSA has a gain up to 23% and Adaptative, Dynamic has a gain up to 14%;
- Both proposed methods make minor changes to default tag operations and do not require additional resources, except an optional memory;
Conclusions

- We presented two new anti-collision algorithms based on existing estimation methods;
- The EDFSA-I has a gain up to 12% related to Q Algorithm;
- The EDFSA-II has a gain up to 26% related to Q Algorithm while Splitting BTSA has a gain up to 23% and Adaptative, Dynamic has a gain up to 14%;
- Both proposed methods make minor changes to default tag operations and do not require additional resources, except an optional memory;
- An ns-2 module with all proposed algorithms was also developed and is available freely for students and researchers.
Table of Contents

1 Motivation and relevance
2 Background
3 Related Work
4 Our Proposal
5 Results and discussion
6 Conclusions
7 References
References


