Protocols for Sensor Augmented RFID Tags

Mauro Piva

Sapienza
Università di Roma

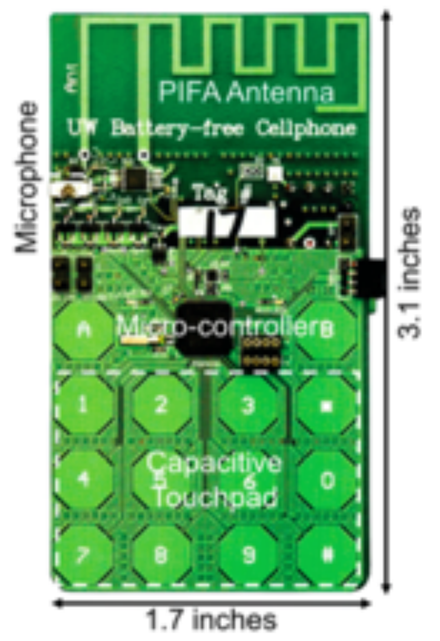
Plan of Attack

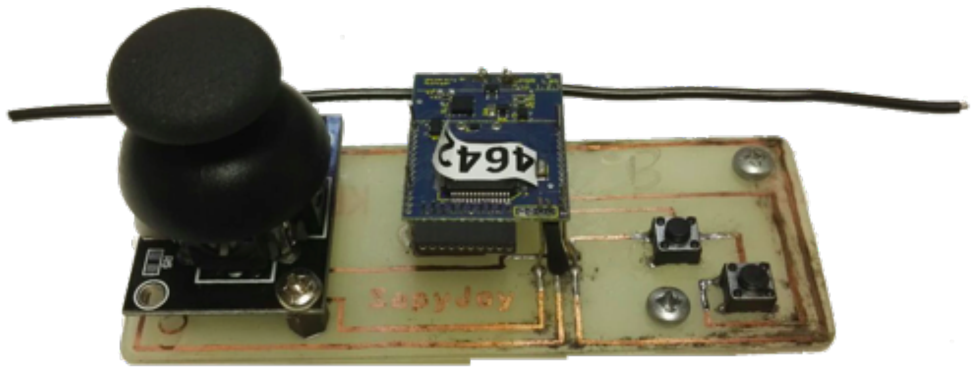
- RFID Sensors: features, protocols and problems
- Hash Function based Protocols
- Reinforcement Learning
- K-Arm Bandit
- Q Learning
- Simulating an environment for Reinforcement Learning

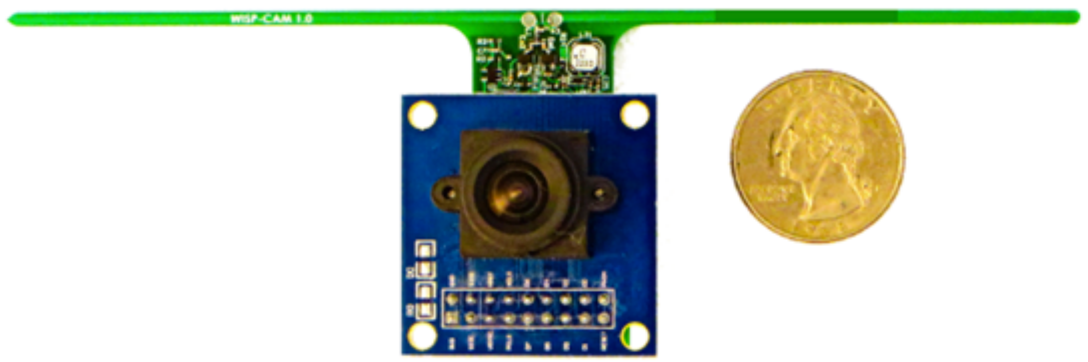
Plan of Attack

- **RFID Sensors: features, protocols and problems**
- Hash Function based Protocols
- Reinforcement Learning
- K-Arm Bandit
- Q Learning
- Simulating an environment for Reinforcement Learning

Sensor Augmented RFID Tags

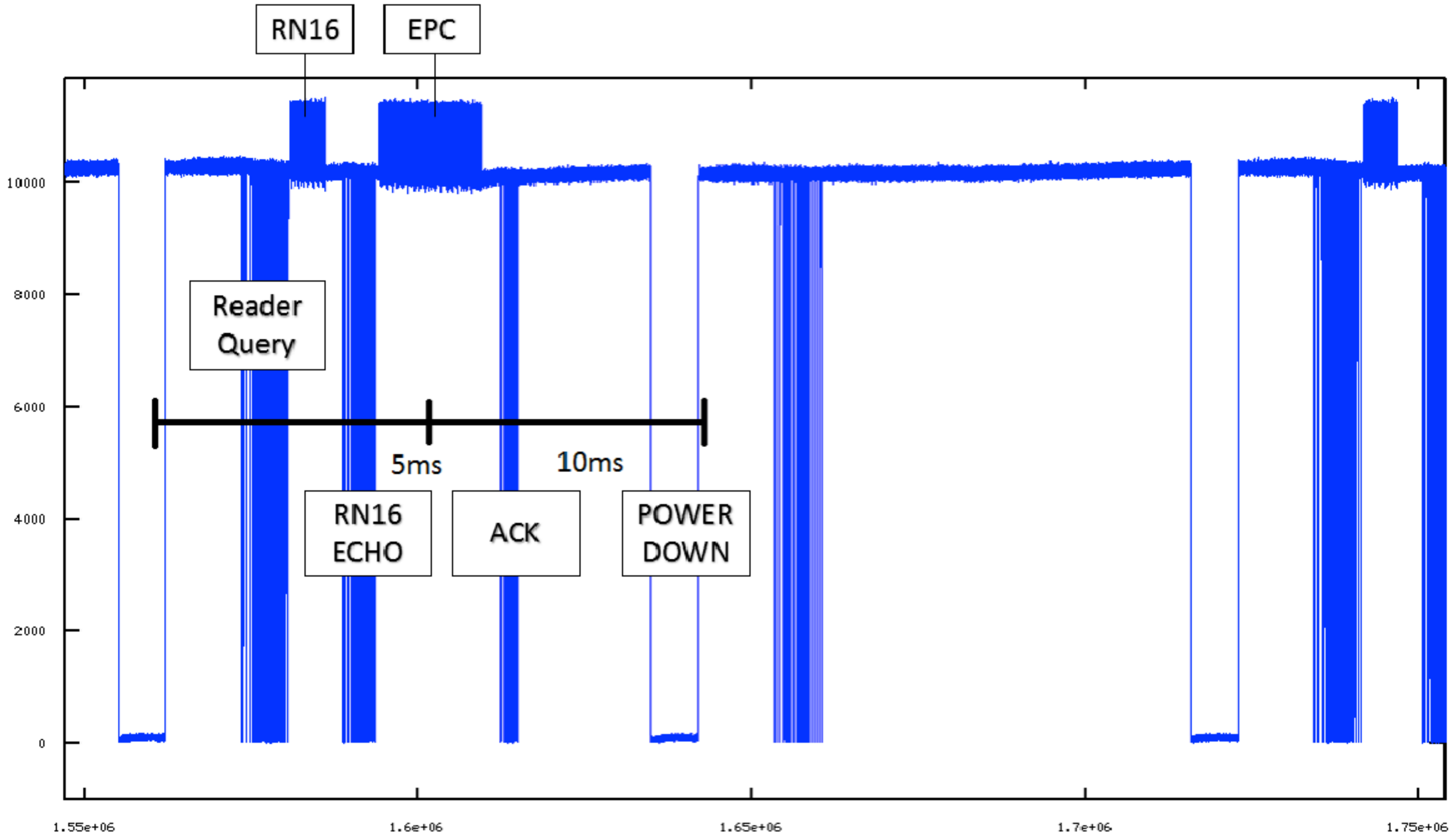




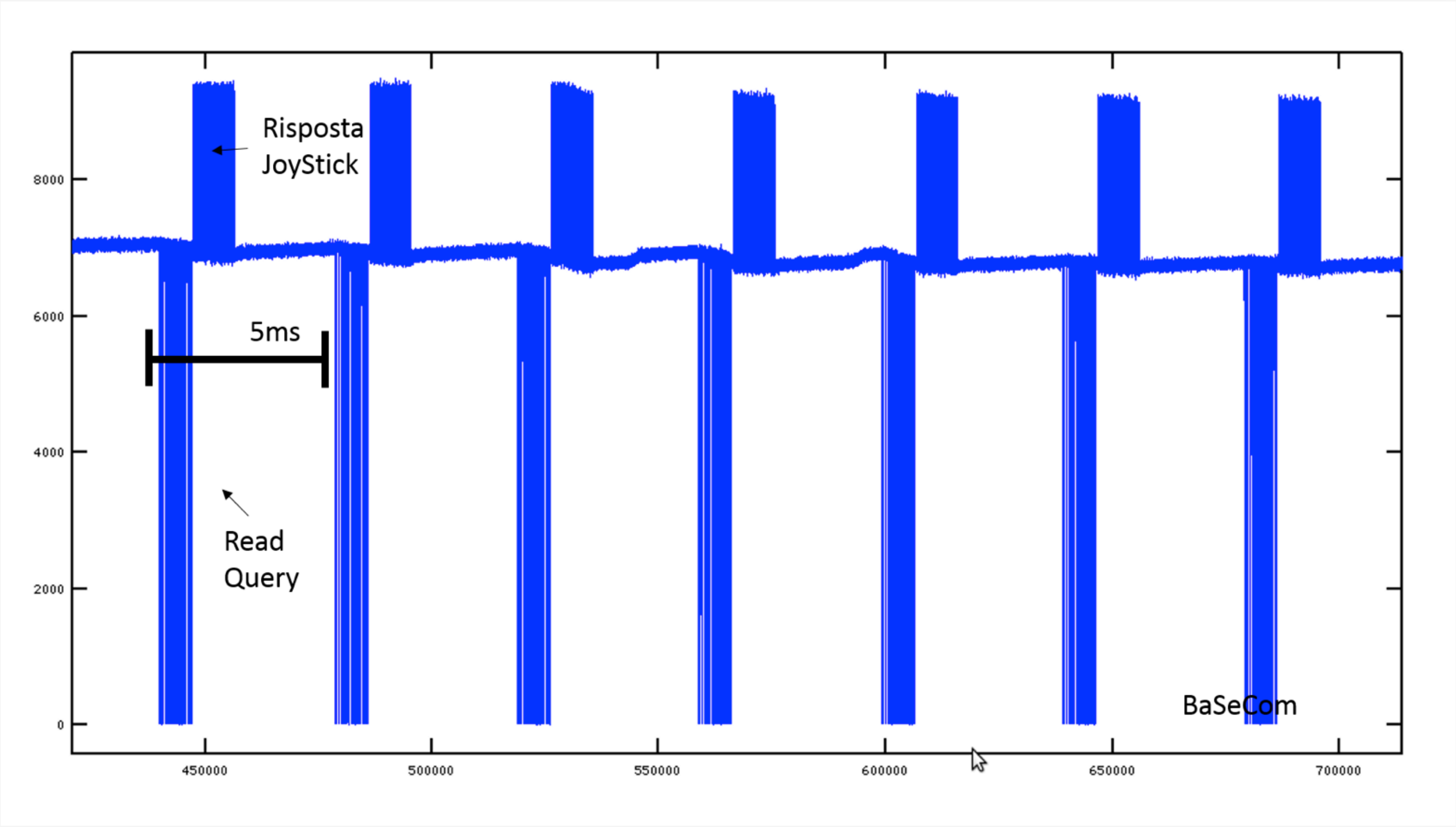


- RFID Tag with sensors embedded: PIR, Camera, Accelerometer…
- No Battery
- Low Power
- Short Distances

Standard Approach: EPC



Adapting EPC



Evolved EPC

Sensors	s1	s2	s3	s4	s5	s6	s7	s8

Channel												
	Query (12 Bit)	Answer(48 Bit)	Query (12 Bit)	Answer(48 Bit)	Query (12 Bit)	Answer(48 Bit)	Query (12 Bit)	Answer(48 Bit)	Query (12 Bit)	Answer(48 Bit)	Query (12 Bit)	Answer(48 Bit)

If we already know the sensors IDS,
in order to query *n* sensors we need *n**(12+48)bit

$$\text{Execution Time} = n*(t_{id}+t_{inf})$$

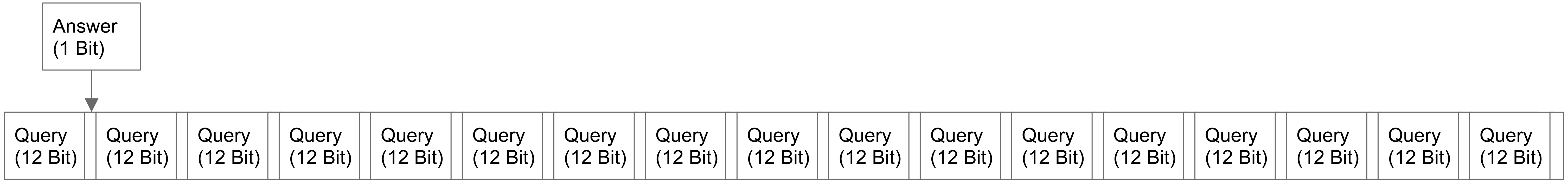
Execution Lowerbound

Execution Lowerboud: $n*(t_{inf})$

Sensors may have less than 48bits of data!

Presence sensor: 1bit

Channel

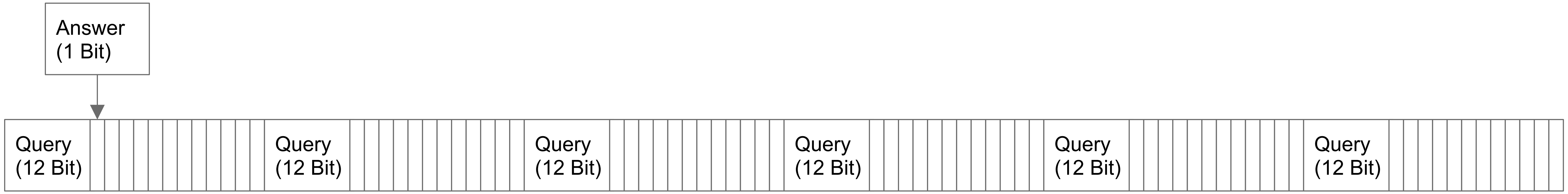


In our implementation, t_{id} could be 12 times t_{inf}!

Our Goal

Execution Lowerbound: $n*(t_{inf})$

Channel



We have one reader query for many devices

Execution Time = $t_{id} + n*t_{inf}$

But, how devices knows in which slot they should transmit?
How to avoid collisions?
We may send all the ordered ids in the reader query, but it would be useless.

Plan of Attack

- RFID Sensors: features, protocols and problems
- **Hash Function based Protocols**
- Reinforcement Learning
- K-Arm Bandit
- Q Learning
- Simulating an environment for Reinforcement Learning

Hash Function based Protocols

- SIC - MIC

Chen, Shigang, Ming Zhang, and Bin Xiao. "Efficient information collection protocols for sensor-augmented RFID networks." INFOCOM, 2011 Proceedings IEEE. IEEE, 2011.

Hash Functions based Protocols: SIC

Recall: Hash Function H(ID,r)-> PseudoRandom Number

- Single-Hash Information Collection Protocol steps:

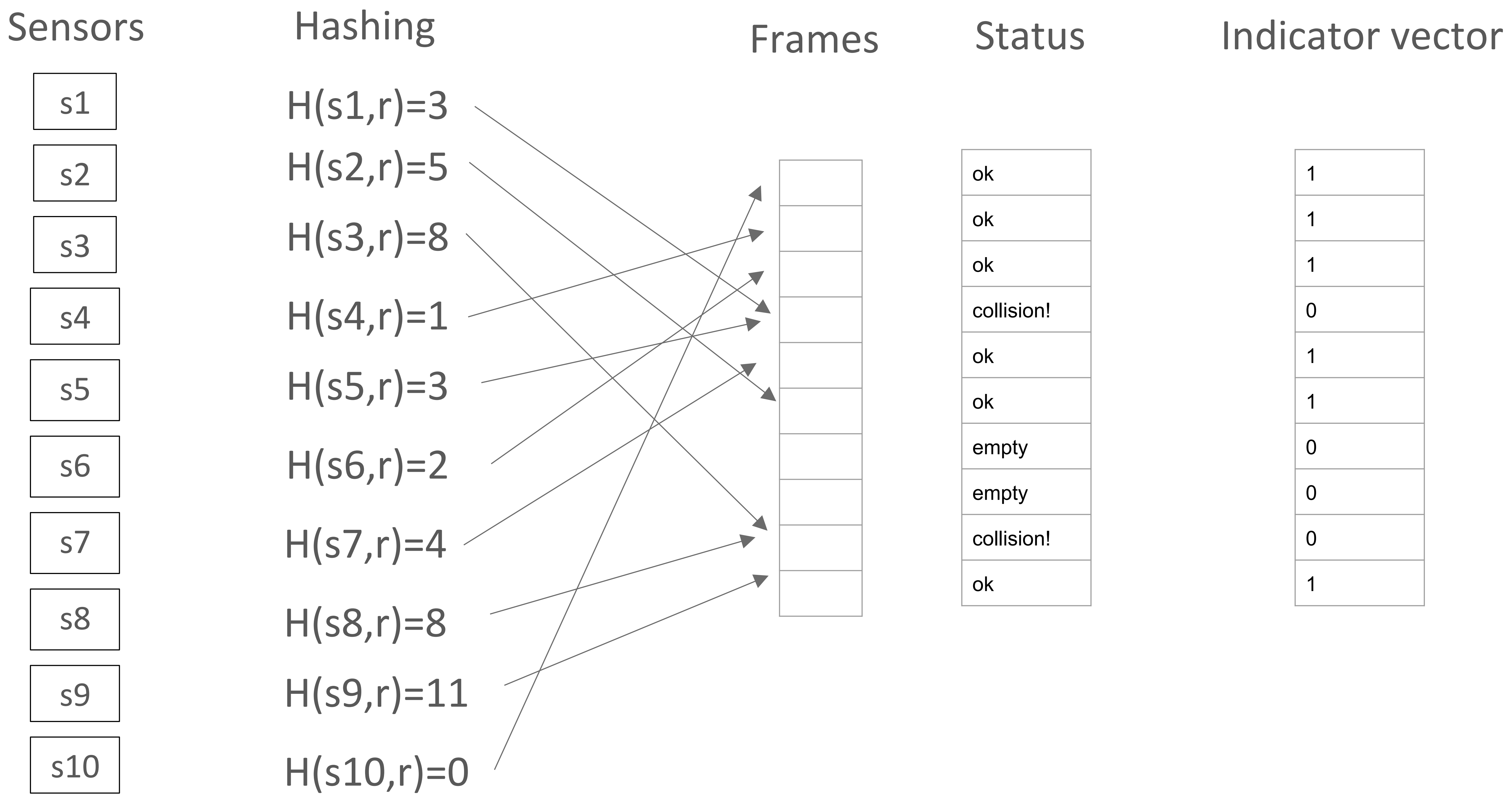
  On the reader

  - Given that the reader already knows the sensors IDs, it calculates a random r and the hash of each sensor. The hash of each sensor is used in order to map slots to IDs.
  - The reader calculates which IDs obtains the same slots.
  - The reader elaborated the *indicator vector,* in order to avoid collisions
  - The reader sends a query with *r* and the *indicator vector*

  Each tag

  - Receive r and calculates its slot.
  - Check in the indicator vector if it has to send or not.

SIC, Reader Side



SIC, Sensor Side

Sensors

s1
s2
s3
s4
s5
s6
s7
s8
s9
s10

Receive Query

(with r and IndVc)

Hashing

H(s1,r)=3
H(s2,r)=5
H(s3,r)=8
H(s4,r)=1
H(s5,r)=3
H(s6,r)=2
H(s7,r)=4
H(s8,r)=8
H(s9,r)=9
H(s10,r)=0

Indicator vector

1
1
1
0
1
1
0
0
0
0

Actions

s1: collision, sleep
s2: send in slot 5
s3: collision, sleep
s4: send in slot 1
s5: collision, sleep
s6: send in slot 2
s7: send in slot 4
s8: collision, sleep
s9: send in slot 9
s10: send in slot 0

Hash Functions based Protocols: SIC

- Results:

  No collisions

  In the first phase, we expect to have the 63.2% of wasted slots.

  Execution time is 2.72 times the lower bound

Much room for improvements

Hash Functions based Protocols: MIC

Recall: Hash Function H1(ID,r)-> PseudoRandom Number n1
Hash Function H2(ID,r)-> PseudoRandom Number n2
We expect $n1 \neq n2$, with the same ID and r

- Multiple-Hash Information Collection Protocol

j hash functions

n' is the number of devices, we set *n*(number of slots) equal to *n'*

We have *k* rounds, each one involving a different hash function H[*k*]. In first round:

Apply H[1] to map sensors to slots, like in SIC

Remove assigned tag from being considered and mark used slots

In each successive round we apply SIC with H[k], considering only unassigned sensors and unmarked slots

Inside the indicator vector, we have 0 if no tags have been assigned, *k* otherwise

Indicator vector size became $\lceil \log_2(k+1) \rceil$

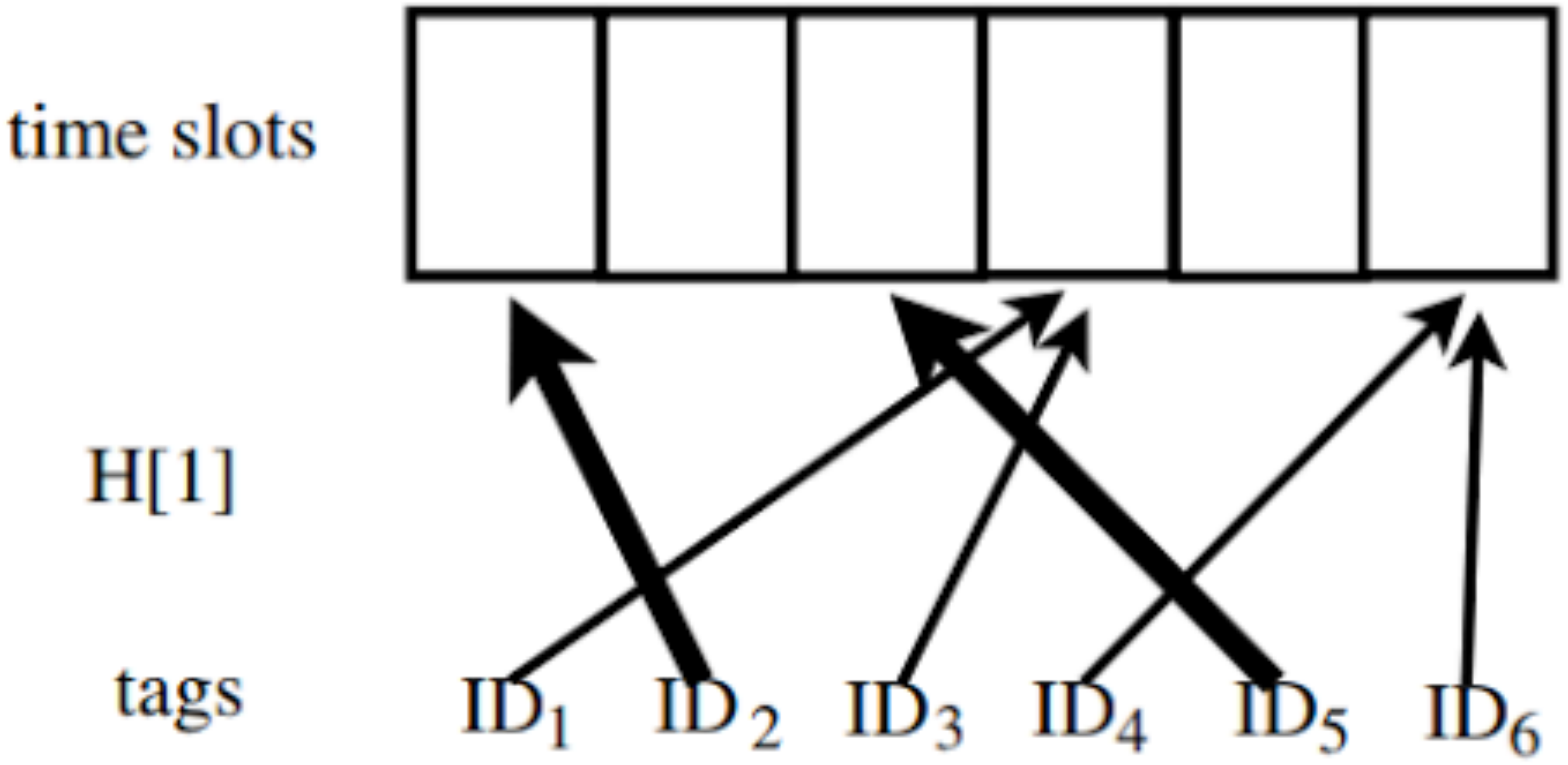
If a tag finds that:

a) it is mapped to a slot s using the *j*th function

b) the corresponding element in the hash-selection vector is also *j*

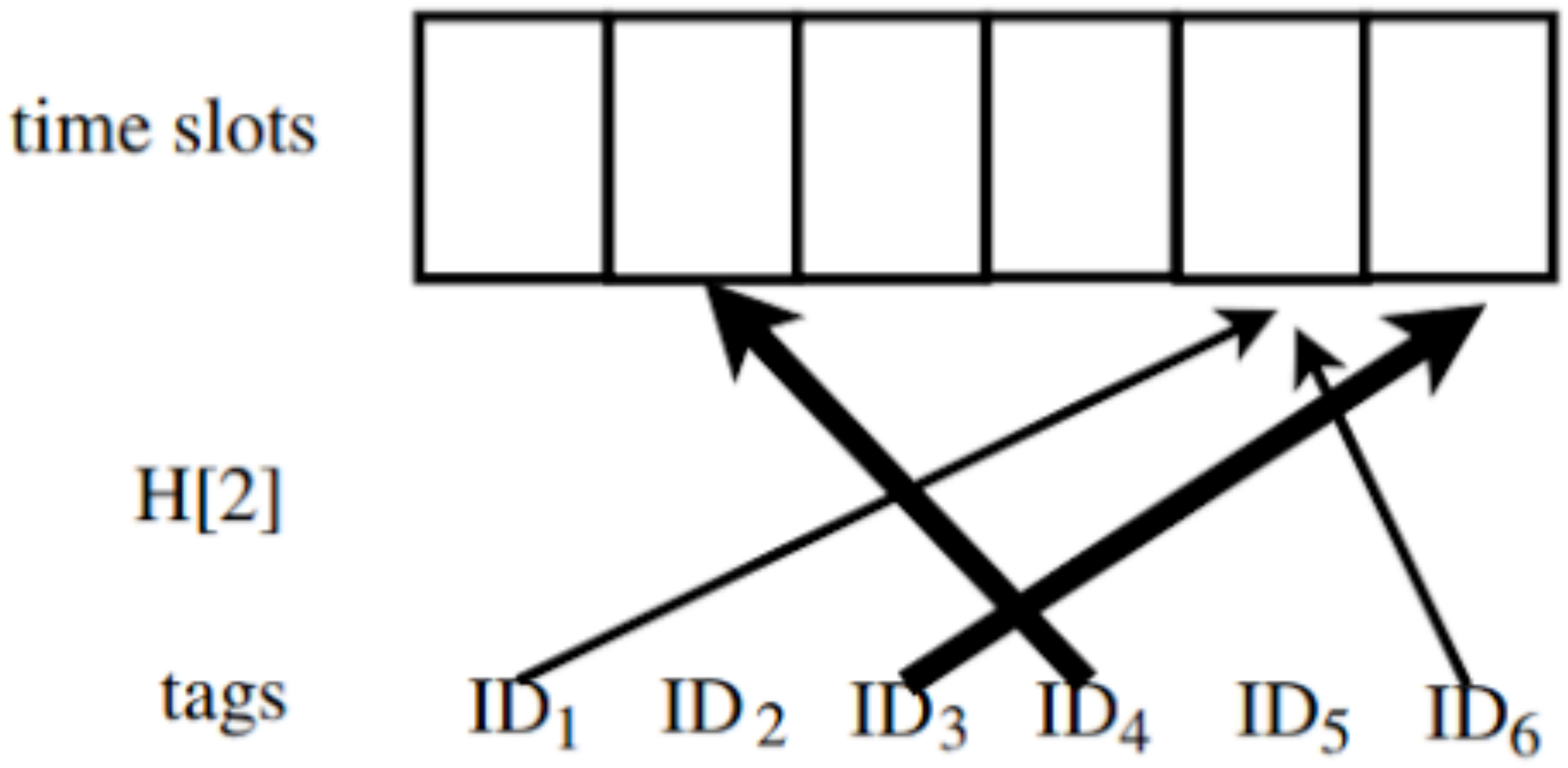
than it can conclude that it must have been assigned to slot s by the reader. If multiple hash functions satisfy the conditions, the tag only uses the one that has the smallest value of *j.*

Hash Functions based Protocols: MIC



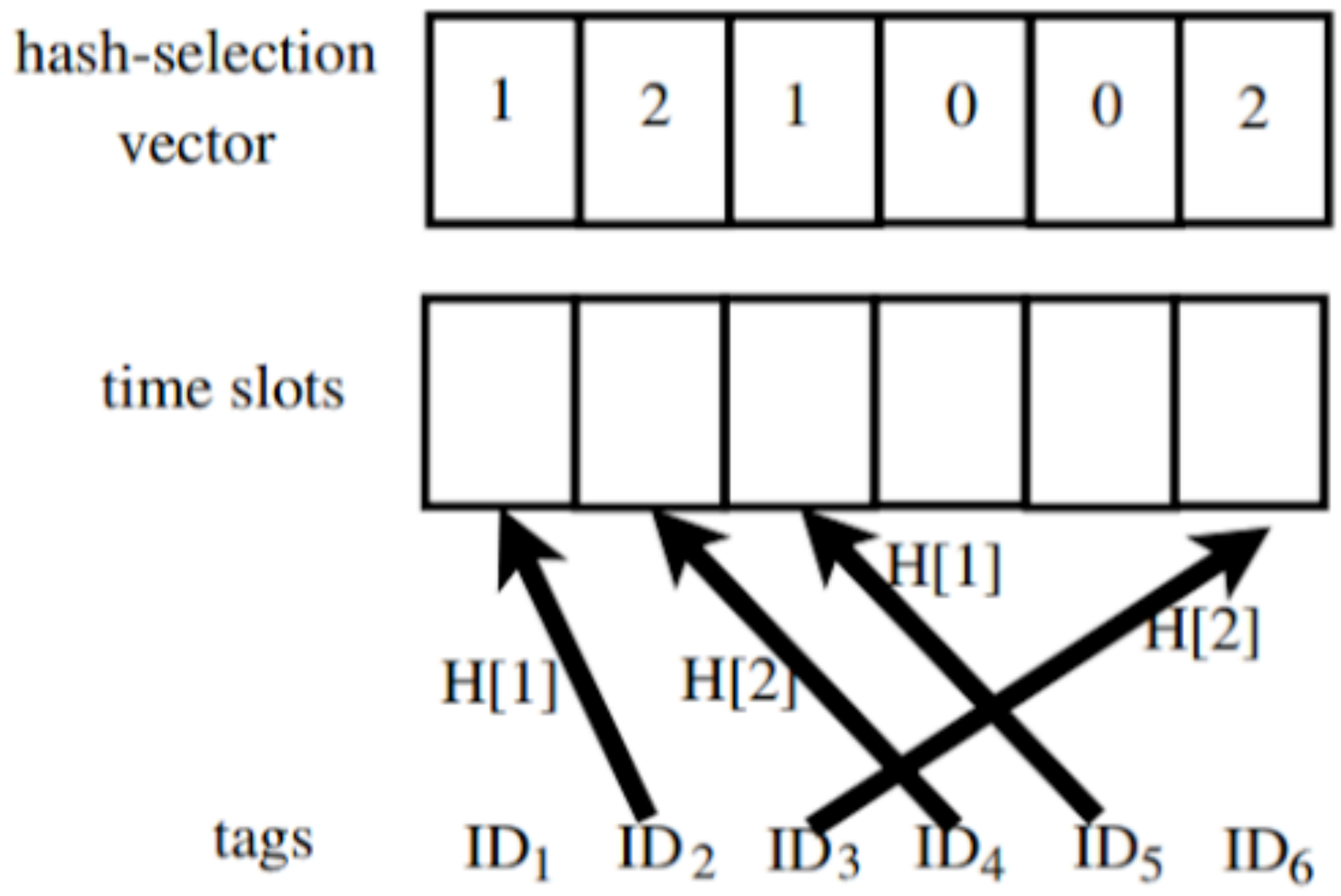
(a) First Roud of Slot Assignment

Hash Functions based Protocols: MIC



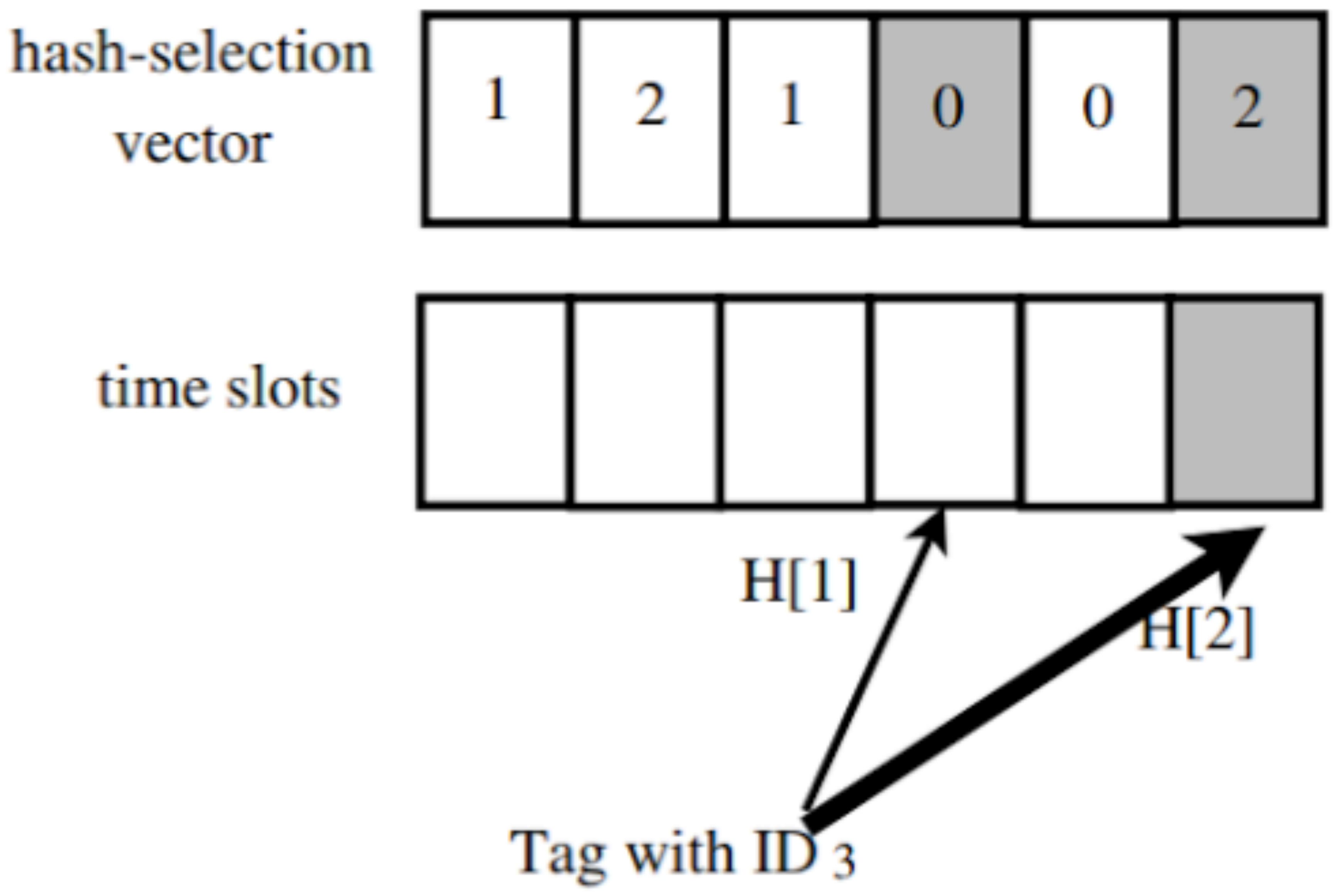
(b) Second Roud of Slot Assignment

Hash Functions based Protocols: MIC



(c) Construction of Hash-selection Vector

Hash Functions based Protocols: MIC



(d) Tag with ID3 finds its assigned slot.

Hash Functions based Protocols: MIC

NUMERICAL VALUES OF P_i

P_1	P_2	P_3	P_4	P_5	P_6	P_7
36.8%	58.0%	69.6%	76.4%	80.8%	83.9%	86.1%

Hash Function based Protocols

- TOP

Qiao, Y., Chen, S., & Li, T. (2013). Tag-ordering polling protocols in RFID systems. In RFID as an Infrastructure (pp. 59-82). Springer New York.

Coded Polling Protocol

With Coded Polling Protocol is possible to reduce the amount of data each tag has to receive by half.

We need a protocol where each tag ID carries: IdentificationNumber+CRC

This is the case of EPCgen2 standard, where each 96 bit tag ID contains a CRC

Evolved EPC

Sensors

s1	s2	s3	s4	s5	s6	s7	s8

Channel

Query (12 Bit)	Answer(48 Bit)	Query (12 Bit)	Answer(48 Bit)	Query (12 Bit)	Answer(48 Bit)	Query (12 Bit)	Answer(48 Bit)	Query (12 Bit)	Answer(48 Bit)	Query (12 Bit)	Answer(48 Bit)

If we already know the sensors IDS,
in order to query *n* sensors we need *n**(12+48)bit

$$\text{Execution Time} = n*(t_{id}+t_{inf})$$

Coded Polling Protocol

Sensors

s1	s2	s3	s4	s5	s6	s7	s8

Channel

Query (12 Bit)	Answer 1(48 Bit)	Answer 2(48 Bit)	Query (12 Bit)	Answer 1(48 Bit)	Answer 2(48 Bit)	Query (12 Bit)	Answer 1(48 Bit)	Answer 2(48 Bit)	Query (12 Bit)	Answer 1(48 Bit)	Answer 2(4

Reduced Execution Time

Reduce Sensor Rx Time

Reduced Consumption

Coded Polling Protocols, CRC

CRC property:

if x and y are two tags IDs with valid CRC

$x \oplus y$ also has a valid CRC

given $\hat{y} = y$ in the reverse order

$x \oplus \hat{y}$ does NOT have a valid CRC

Coded Polling Protocols, reader steps

1) arrange tag IDs in randomly chosen pairs ={x,y}.
2) For each pair, define *polling code* as c = $x \oplus \hat{y}$
3) Instead of sending tags IDs, send *polling codes*
4) After each polling code, wait for two slots, in which x and y will answer

Coded Polling Protocols, tag *z* steps

1) Tag *z* receives the polling code *c*

2) Compute the reversal of $z \oplus c$.
If it has a valid CRC, tag will transmit

1) Otherwise, compute $\hat{z} \oplus c$.
If it has a valid CRC, tag will transmit

Otherwise, tag will not transmit.

Coded Polling Protocols, demonstration

If z = x:

tag computes $z \oplus c = x \oplus x \oplus \hat{y} = \hat{y}$. Its reverse has a valid CRC, so z will transmit.

Given that now z knows also y, if x>y, it will transmit in the first slot.

If z = y:

$y \oplus c = y \oplus x \oplus \hat{y}$. It reversal will have an invalid CRC with an high probability.

$\hat{z} \oplus c = \hat{y} \oplus x \oplus \hat{y} = x$. It will have a valid CRC.

If z ≠ x and z ≠ y:

$y \oplus c = y \oplus x \oplus \hat{y}$ and $\hat{z} \oplus c = \hat{z} \oplus x \oplus \hat{y}$ will have an invalid CRC with an high probability.

What if we have a valid CRC? We can not, as pairs are computed by the reader.

Tag-Ordering Polling protocol

In Coded Polling, tags must always listen the channel

Like Single Hash Protocol, but resolves the problem of collisions

We want to query only a subset M of the entire population of tags.

Three phases:

- Ordering phase
- Polling phase
- Reporting phase

Tag-Ordering Polling protocol

Ordering Phase:

Reader sends the “*reporting-order vector*”, built like the indicator vector in SIC.

If a tag finds that it representative bit is 0, it will not communicate and will go sleep.

If otherwise it finds a 1, it may be member of M or that slot may be occupied by another tag, provocating a false positive. The reader can put the false positive tags inside a set F.

We may also have a 1 with a collision. In such case the reader will put all the colliding tags except one inside a set C.

Tag-Ordering Polling protocol

Polling Phase:

For each tag in F, the reader sends the tag ID with a negative polling request(a bit=0 at the end of the id). These tags will immediately go in sleep mode.

For each tag in C, the reader sends the tag ID with a positive polling request, and immediately waits for the tag answer. Tags in C will not participate in Reporting Phase.

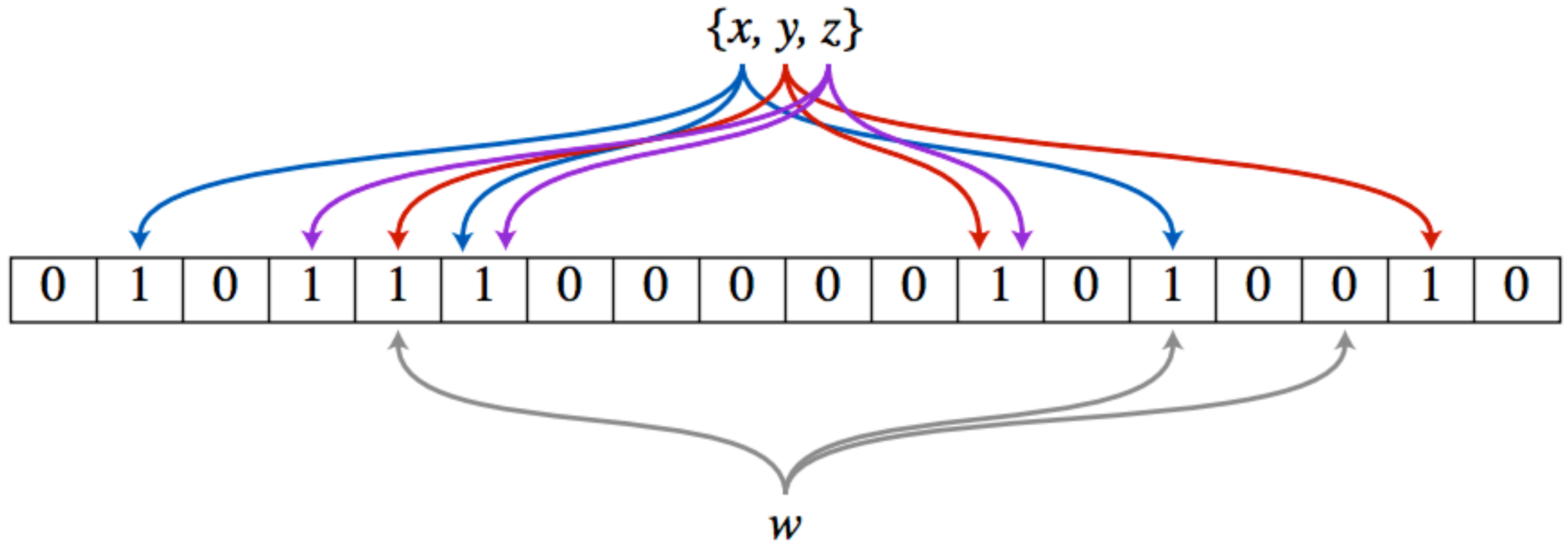
Reporting Phase:

Each tag in M-C, following the reporting order vector and the logic of the single hash communication protocol, will transmit.

E-Top: reducing collisions with Bloom Filter

Bloom filter

Hashed representation of a set of n items
Store array of 2nq bits, initially all zero
Map each item x to q bits hash(x,1), hash(x,2), ..., hash(x,k)
For each item in the set, store a one in all of its mapped bits



If an item belongs to the set, all of its bits will be one
If an item does not belong to the set, w/prob $\geq 1 - 2^{-q}$, some bit will be zero

Algorithms for media

D. Eppstein, UC Irvine, WADS 2007

Limits of these protocols:

Can not define device priority

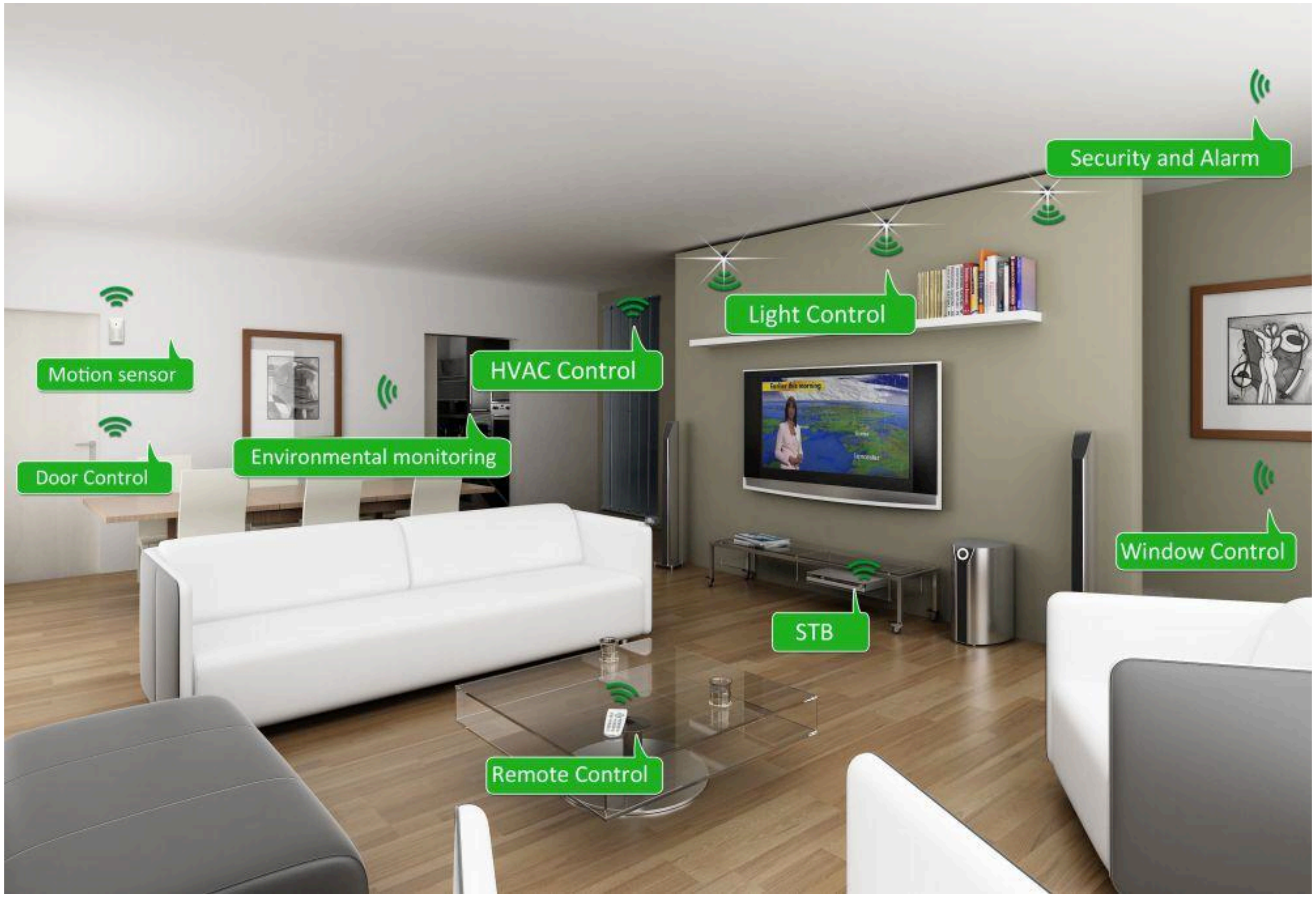
Can not follow burst of data

No dynamic adapting

Plan of Attack

- RFID Sensors: features, protocols and problems
- Hash Function based Protocols
- **Reinforcement Learning**
- K-Arm Bandit
- Q Learning
- Simulating an environment for Reinforcement Learning

The Battery-less Smart Home



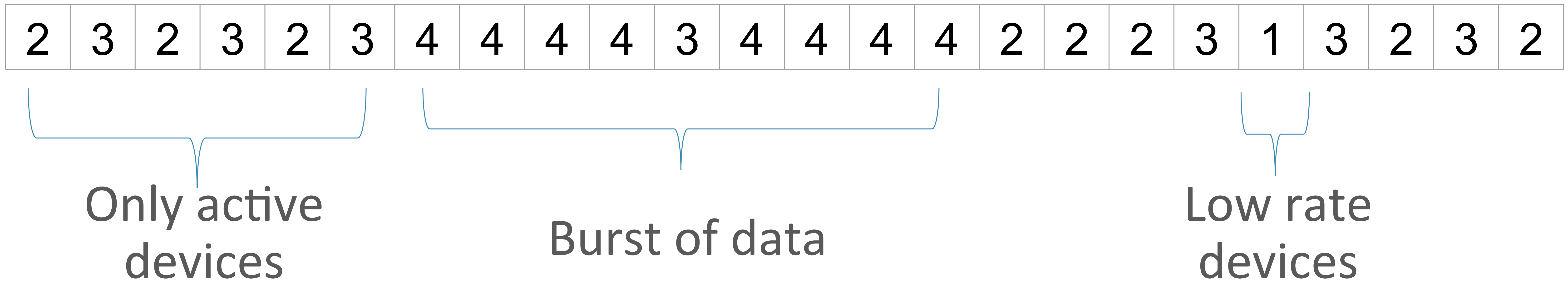
A new protocol

1) Tv Remote

2) Presence Sensor

3) Temperature Sensor

4) Videogame Controller



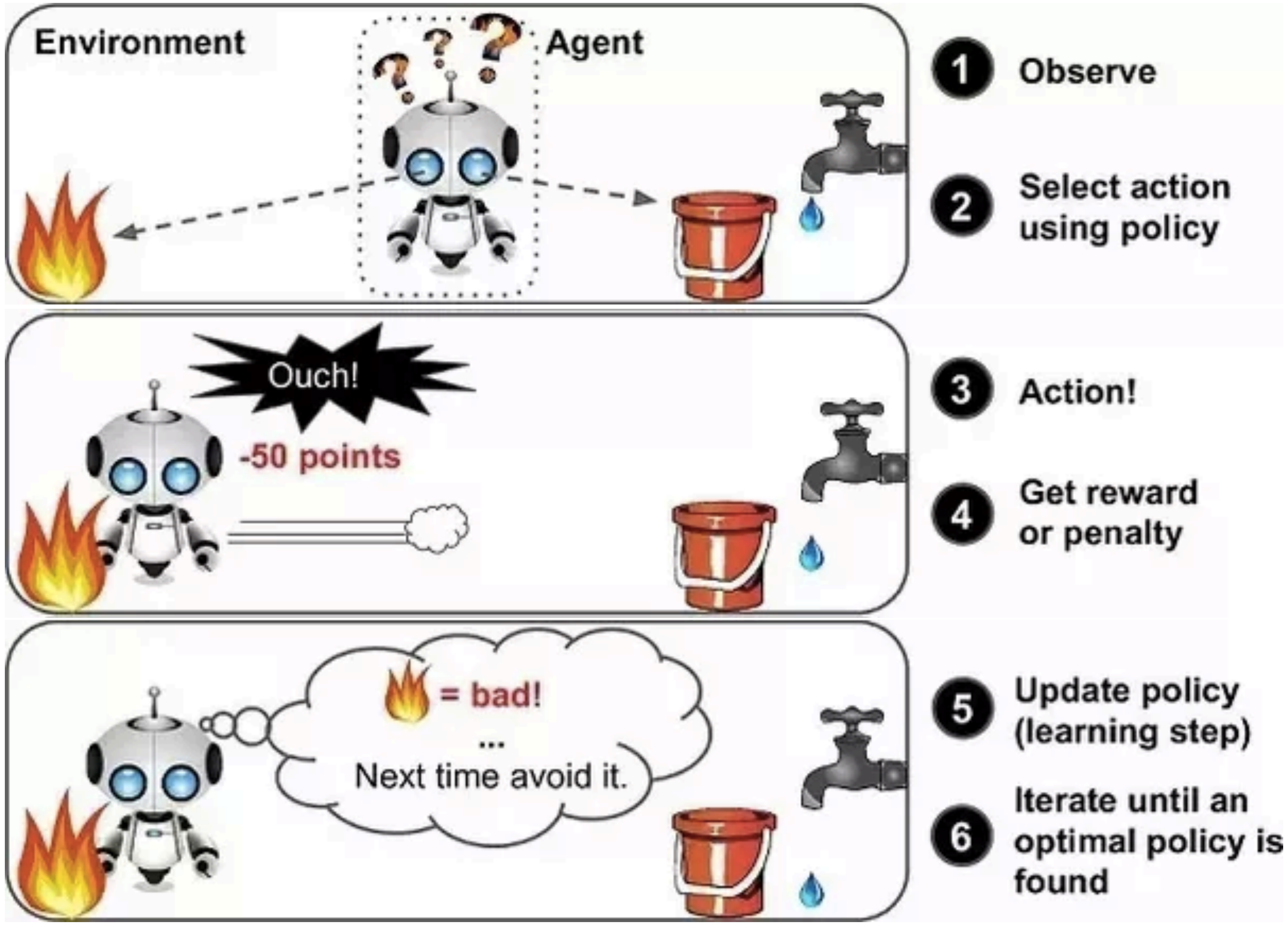
A model for each device?

- Number of devices is growing
- Different behaviors for the same type of device

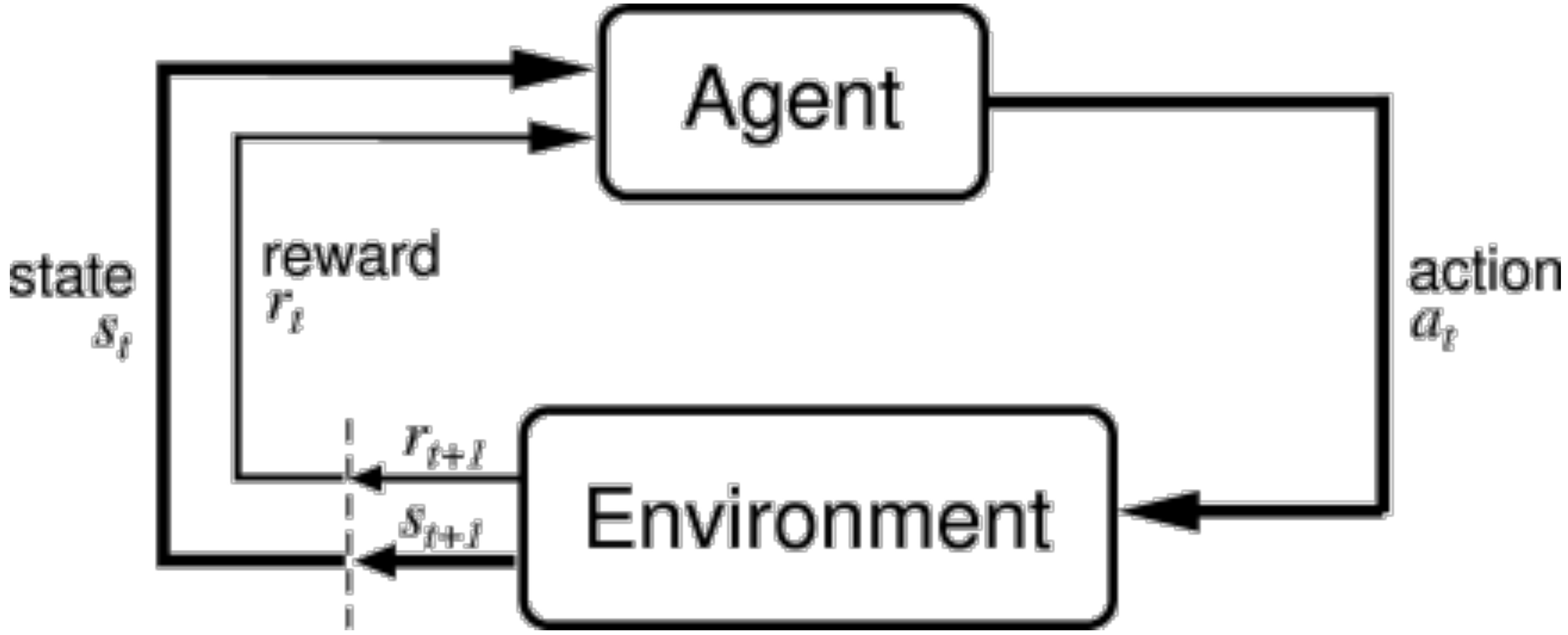
- Too many models

Why don’t we leave the system adapt itself to sensors?

Reinforcement Learning



Reinforcement Learning



- Agent
- A: set of actions
- Q: set of states
- $P_a(s_t,s_{t+1})$: probability of transition to state s_{t+1} from s_t with action a
- $R_a(s_t,s_{t+1})$: expected reward

Plan of Attack

- RFID Sensors: features, protocols and problems
- Hash Function based Protocols
- Reinforcement Learning
- **K-Arm Bandit**
- Q Learning
- Simulating an environment for Reinforcement Learning

A protocol based on K-Arm Bandit

- Agent: Reader
- A: The device which should be queried
- Q: only one state
- R: -regret*|deviceSet|, +reward*|deviceSet|

A protocol based on K-Arm Bandit

Initialize, for a = 1 to k:

$Q(a) = 0$

$N(a) = 0$

Repeat forever:

$$a = \begin{cases} \text{Arg max}_a\, Q(a) & \text{with probability 1-e} \\ \text{Random action} & \text{with probability e} \end{cases}$$

$R = query(a)$

$N(a) = N(a)+1$

$Q(a) = Q(a) + \frac{1}{N(a)}[R-Q(a)]$

A protocol based on QLearning

$$
\begin{aligned}
Q_{n+1} &\doteq Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i .
\end{aligned}
$$

A protocol based on K-Arm Bandit

Initialize, for a = 1 to k:

Q(a) = 0

~~N(a) = 0~~

Repeat forever:

$$a = \begin{cases} \text{Arg max}_a\ Q(a) & \text{with probability } 1\text{-e} \\ \text{Random action} & \text{with probability e} \end{cases}$$

R = query(a)

~~N(a) = N(a)+1~~

$$Q(a) = Q(a) + \cancel{\frac{1}{N(a)}}\alpha[R-Q(a)]$$

A protocol based on K-Arm Bandit

- Agent: Reader
- A: The device which should be queried
- Q: only one state
- R: -0.2*|deviceSet|, +2*|deviceSet|
- α: 0.2
- ε: 0.1

Plan of Attack

- RFID Sensors: features, protocols and problems
- Hash Function based Protocols
- Reinforcement Learning
- K-Arm Bandit
- **Q Learning**
- Simulating an environment for Reinforcement Learning

QLearning

- Multiple Status

$$Q(s_t, a_t) = Q(s_t, a_t) + \eta[(r_{t+1}) + \gamma \underset{a_{t+1}}{max} Q(s_{t+1}, a_{t+1} - Q(s_t, a_t)]$$

$$P(a|s) = \frac{exp[Q(s,a)/T]}{\sum_{b \in A} exp[Q(s,a)/T]}$$ → K-Arm One

Plan of Attack

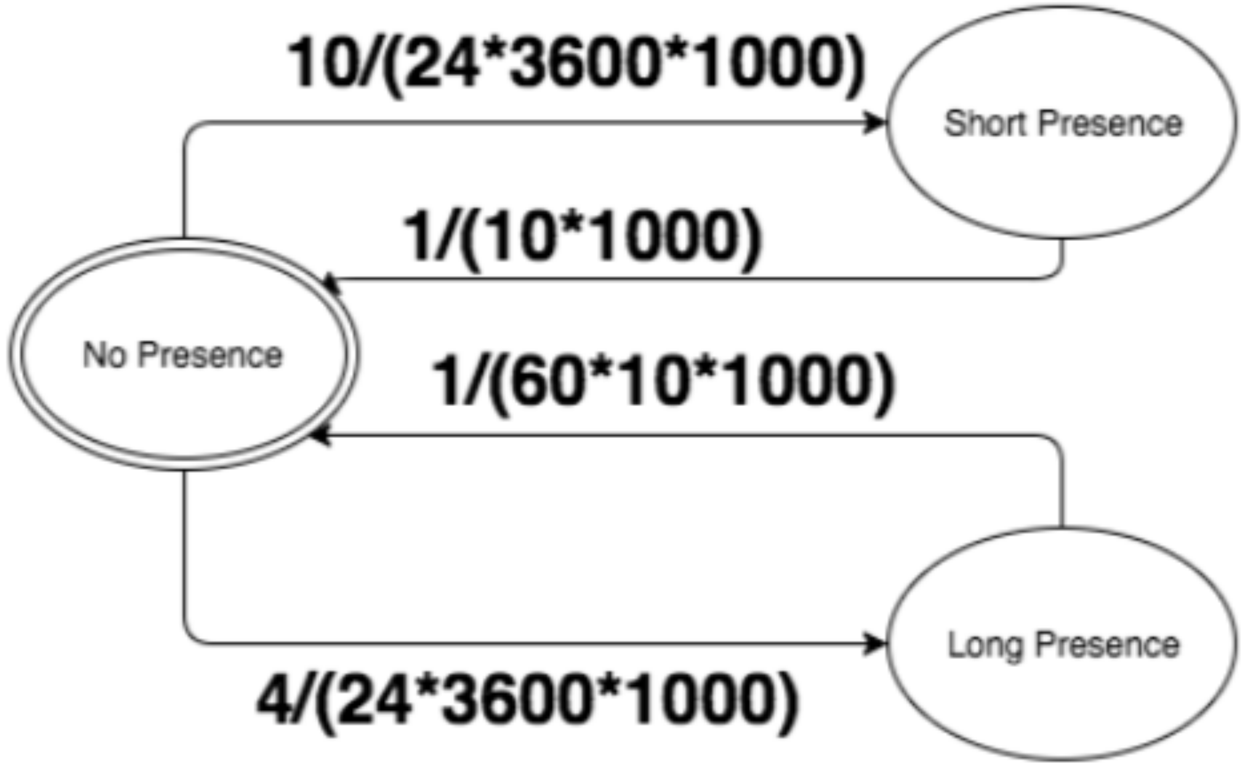
- RFID Sensors: features, protocols and problems
- Hash Function based Protocols
- Reinforcement Learning
- K-Arm Bandit
- Q Learning
- **Simulating an environment for Reinforcement Learning**

Our Network Simulator

- Why?
  - Extensive Simulations
  - Focus on protocols results
- How?
  - Java Software
  - Hardware devices simulators
- Results
  - 30 days long simulations
  - Up to 40 devices

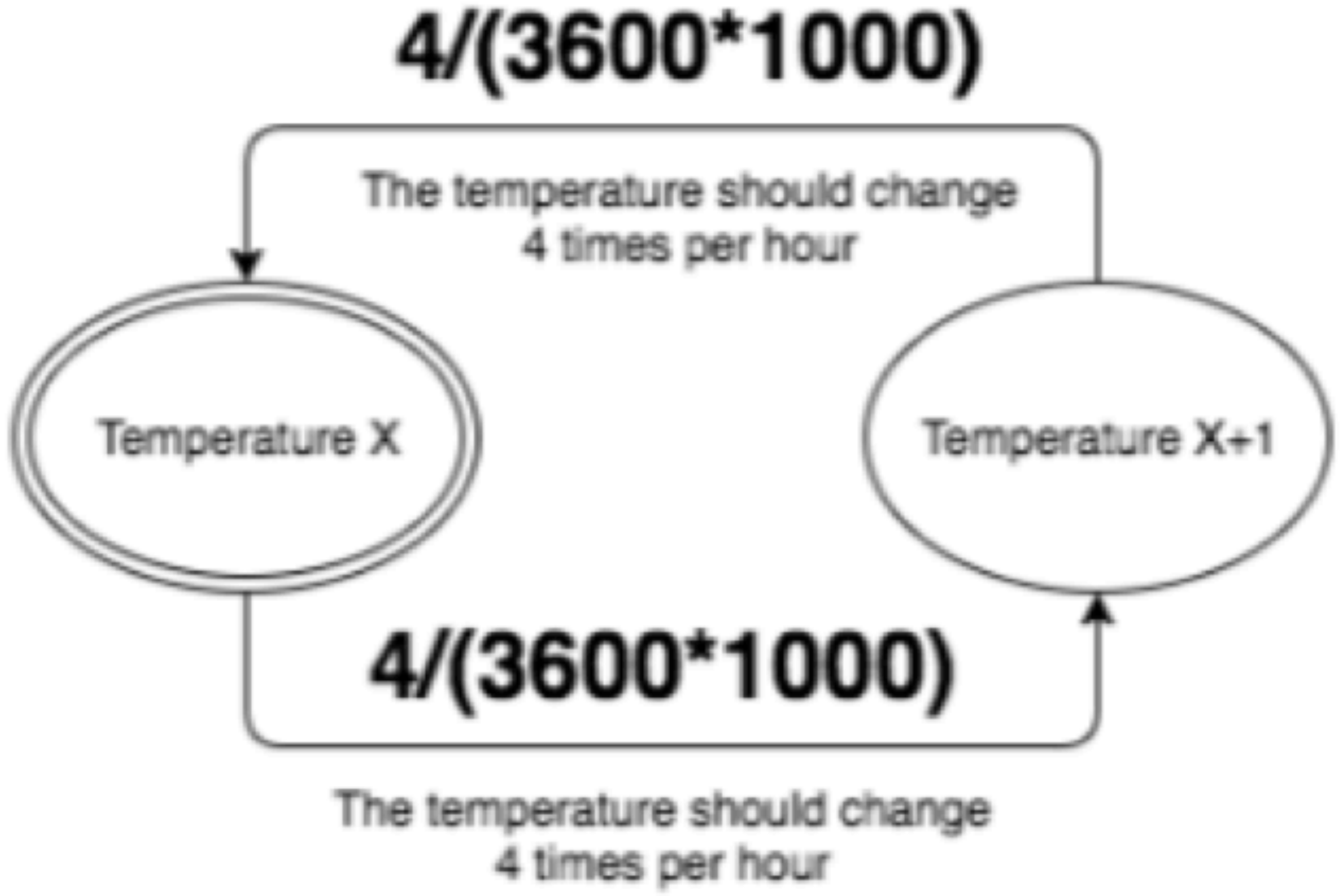
Simulating Devices: presence sensor

- 3 Status
  - No Presence
  - Short Presence
  - Long Presence
- 10 times per day in short presence
- 4 times per day in long presence

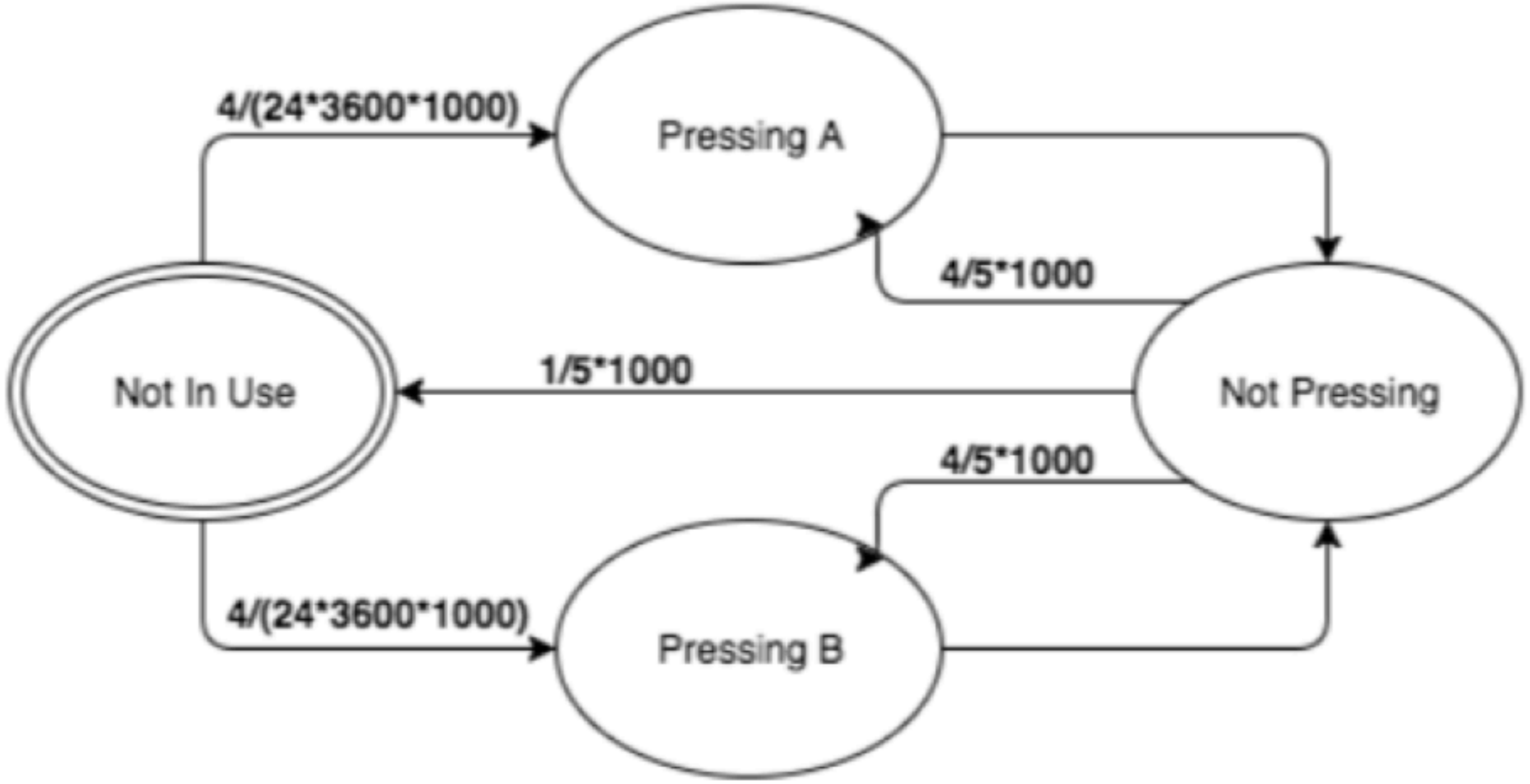


A Status represents the environment condition of a device. We are not interested in collecting the information regarding this status, as it would be impossible, but just in collecting the information regarding the change of status.

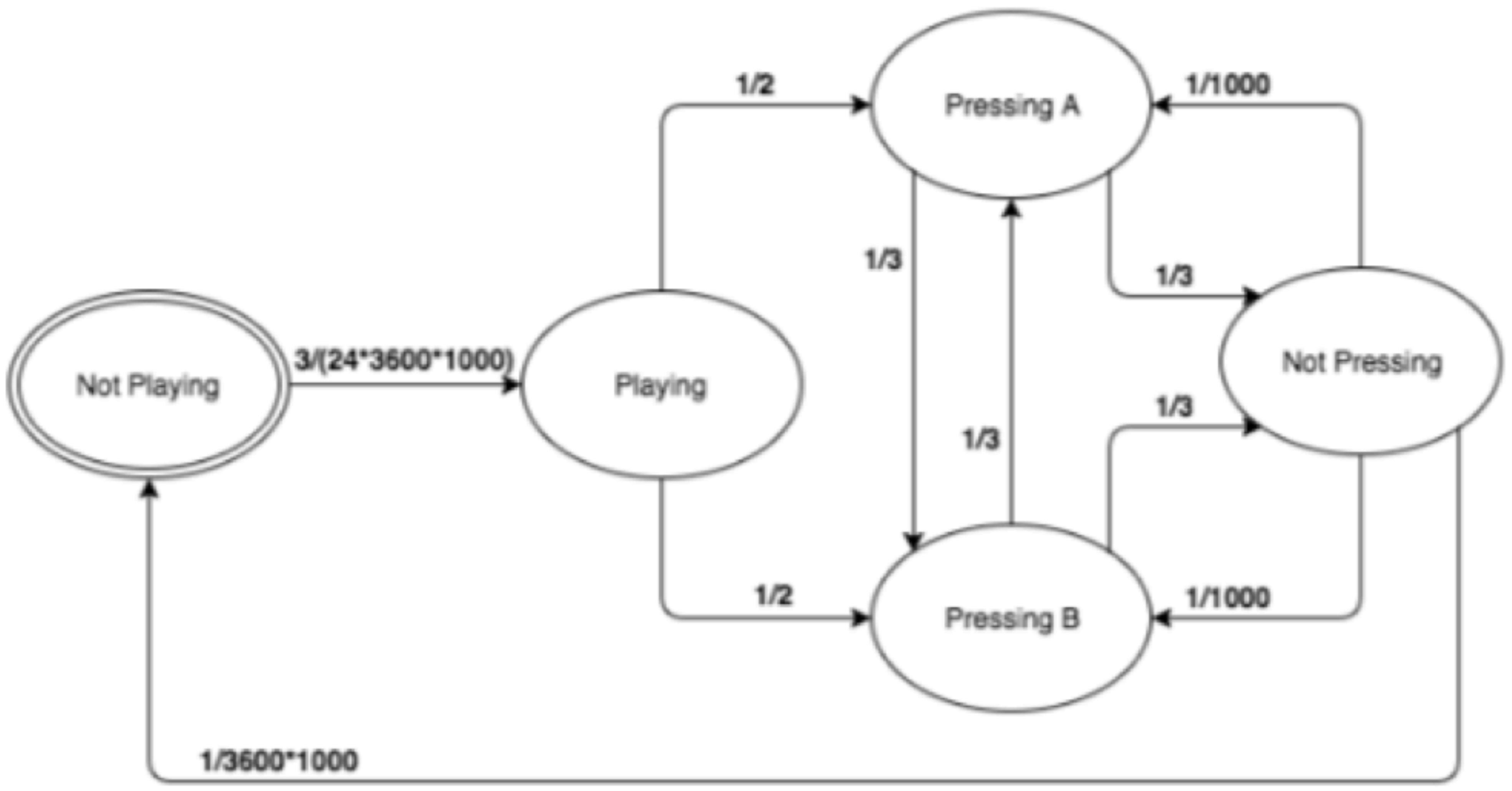
Simulating Devices: temperature sensor



Simulating Devices: TV remote



Simulating Devices: JoyTag



Performance Evaluation

- Metrics
  - Packet delay (Tx+prop+proc)
  - Data Loss Rate
  - Throughput
- Parameters
  - Number of Sensors
  - Slot Lenght
- Factors to study
  - Reward/Regret
  - Learning Rate
  - Epsilon
- Workloads
  - 3 different workloads(number of sensors)
  - 2 different simulation time
- Benchmarks
  - TDMA
  - Optimum

10 Devices – 1 day – 5ms

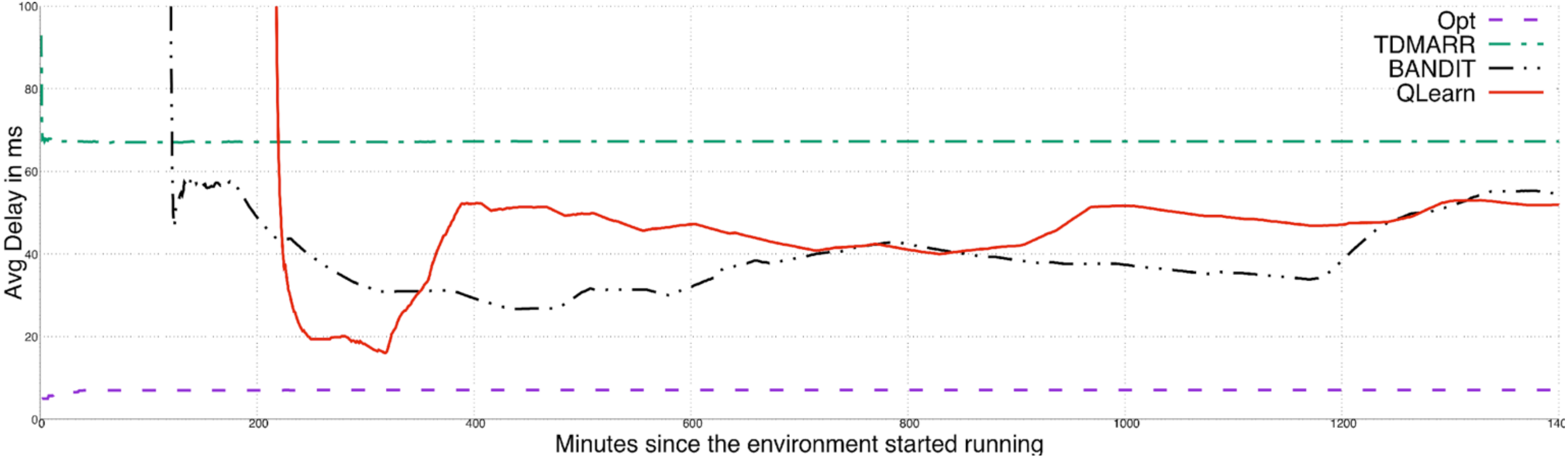
Opt: 7ms

TDMARR: 15ms

Bandit: 8.5ms

Qlearn: 7.7ms

20 Devices – 1 day – 5ms



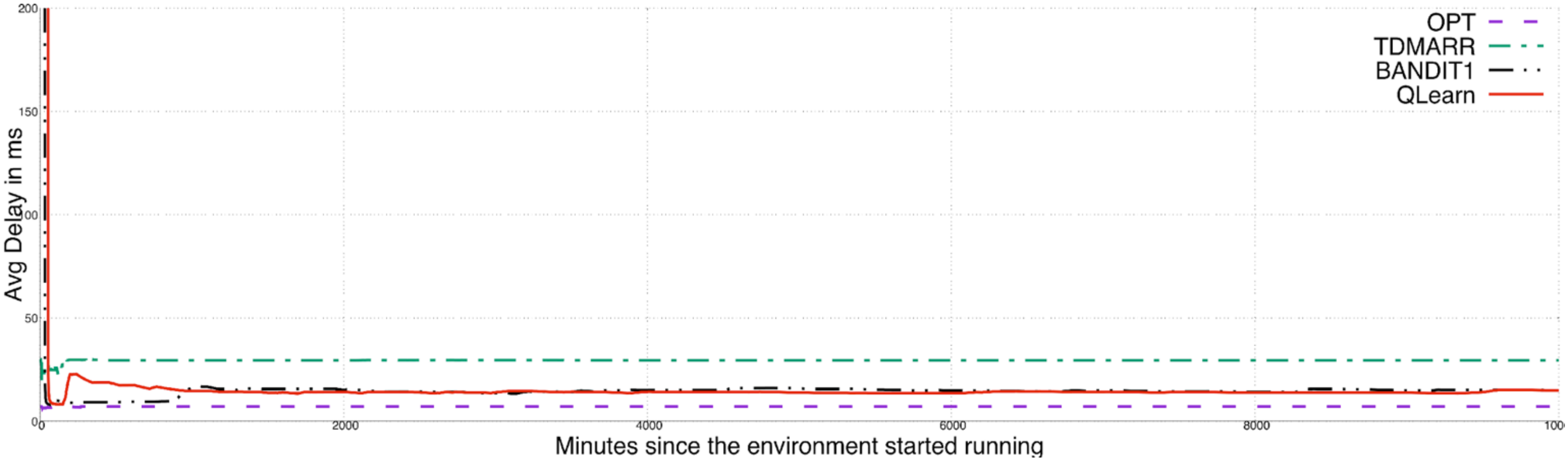
Opt: 7ms

TDMARR: 67ms

Bandit: 55ms

Qlearning: 53ms

10 Devices – 7 days – 10ms



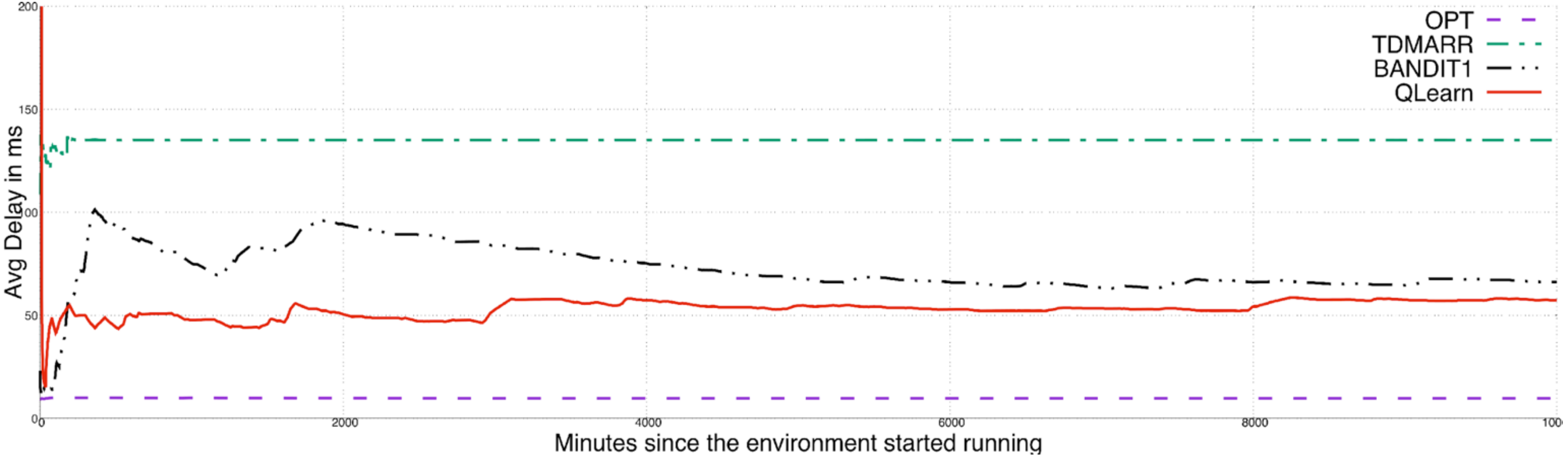
Opt: 7ms

TDMARR: 30ms

Bandit 14.85ms

Qlearning: 14.89ms

20 devices – 7 days – 10ms



Opt: 7ms

TDMARR: 134ms

Bandit: 65ms

Qlearning: 57ms

Sources

- Reinforcement Learning, chapter 1-2

Sutton, R. S., & Barto, A. G. (1998). Introduction to reinforcement learning (Vol. 135). Cambridge: MIT Press.

- SIC - MIC

Chen, Shigang, Ming Zhang, and Bin Xiao. "Efficient information collection protocols for sensor-augmented RFID networks." INFOCOM, 2011 Proceedings IEEE. IEEE, 2011.

- TOP

Qiao, Y., Chen, S., & Li, T. (2013). Tag-ordering polling protocols in RFID systems. In RFID as an Infrastructure (pp. 59-82). Springer New York.