Dribble - A learn-based timer scheme selector for mobility management in IoT

Authors:
Bruno P. Santos,
Paulo H. L. Rettore,
Luiz F. M. Vieira,
Antonio A. F. Loureiro

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

UFOP
Universidade Federal de Ouro Preto

Introduce yourself, our university and the title of this work.

Agenda

1. Introduction
2. Background
3. Dribble
4. Evaluation
5. Conclusion and Future work

Pass through the content

Introduction

Contextualization

- Smart devices are part of our daily routine

Smart devices have already been part of our everyday life.
They are in everywhere from our body/pocket to attached in infrastructures.
When such devices are connected to the Internet, we consider it as a extension of the current Internet or Internet of Things.

Mobility is a key challenge!

- IoT - Challenges
  - Internet adaptations
  - Heterogeneous devices
  - Constrained resources (Energy, CPU, Memory…)
  - Mobility

IoT rise up several challenges because:
Smart device demands adaptation to operate on the Internet
They are heterogeneous and have different capabilities of Energy, CPU, Memory…
Also, the mobility is a key challenging aspect.

Introduction

Mobility is a key challenge!

- IoT - Challenges
  - Internet adaptations
  - Heterogeneous devices
  - Constrained resources (Energy, CPU, Memory...)
  - **Mobility**
- We are interested in handle Mobility
  - Key aspect for mobile and wireless environment
  - Mobility from routing protocol lens

In this work, we are interested in handle mobility
Focusing on Mobility from routing protocol perspective

Introduction

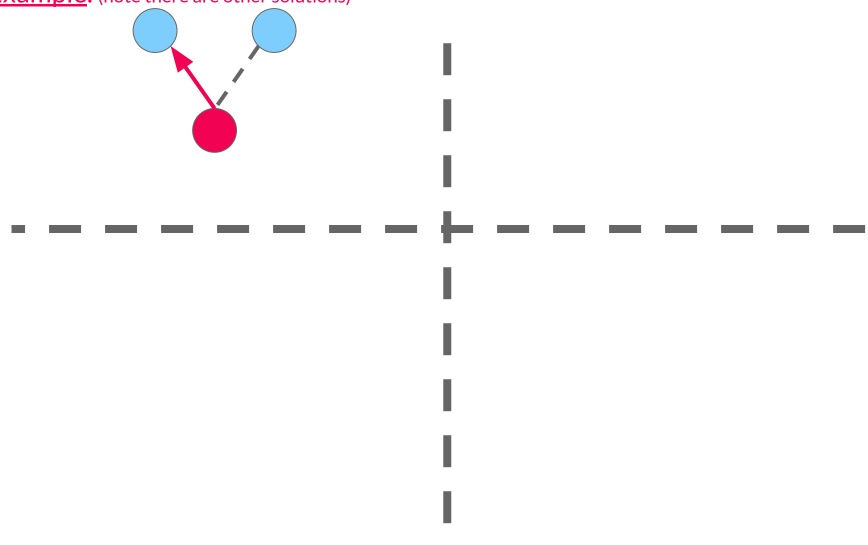
Routing under mobility events

- Mostly of routing protocols for mobile IoT have one timer scheme
  - It governs the communication structure construction and maintenance

Mostly routing protocols have a timer scheme to create and maintain routing structure

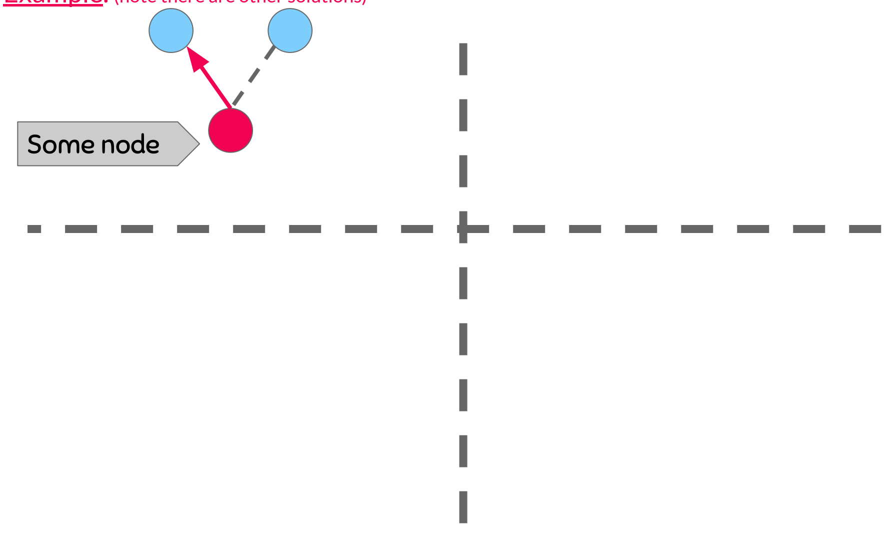
Routing under mobility events

Example. (note there are other solutions)



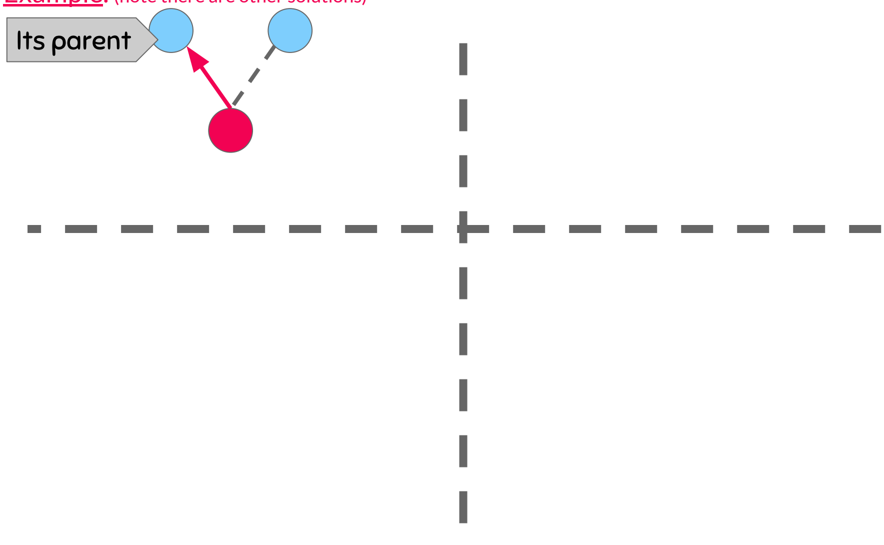
Routing under mobility events

Example. (note there are other solutions)



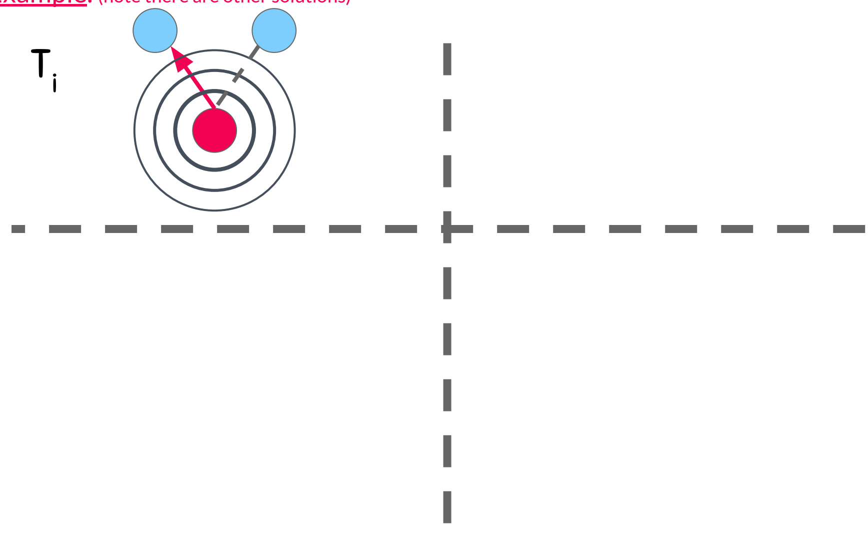
Routing under mobility events

Example. (note there are other solutions)



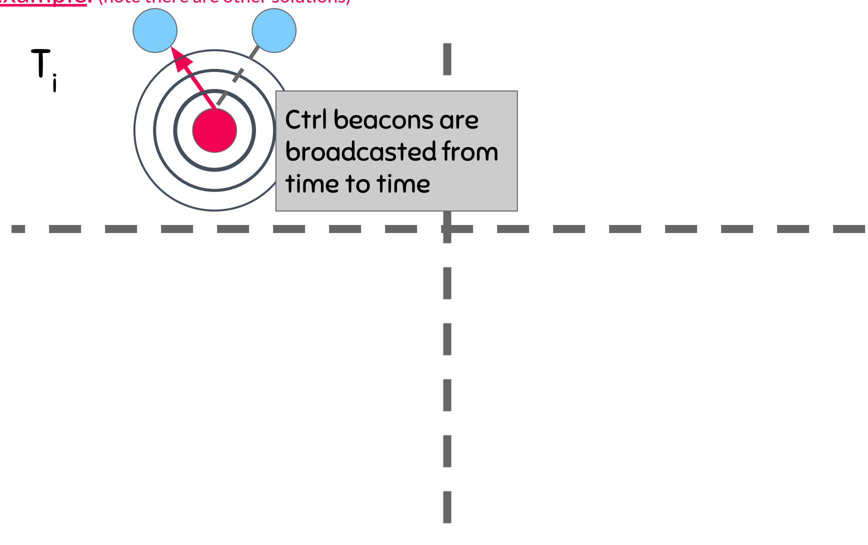
Routing under mobility events

Example. (note there are other solutions)



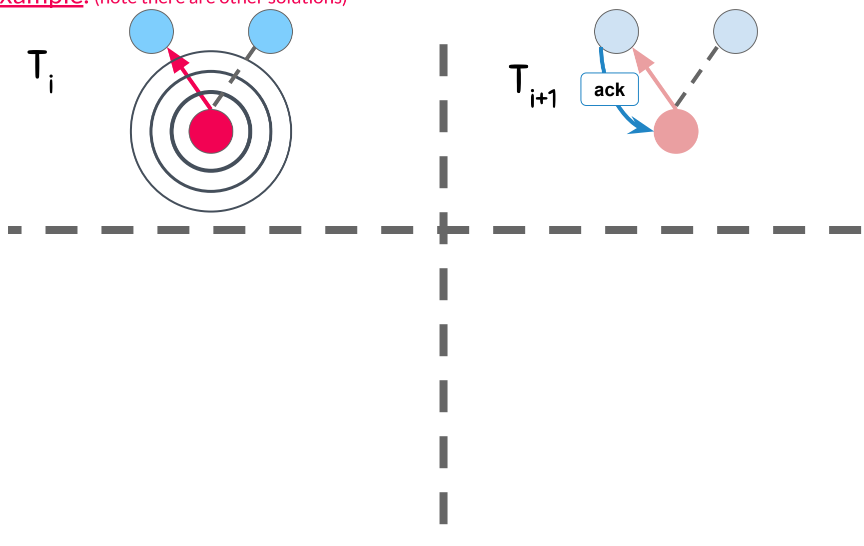
Routing under mobility events

Example. (note there are other solutions)



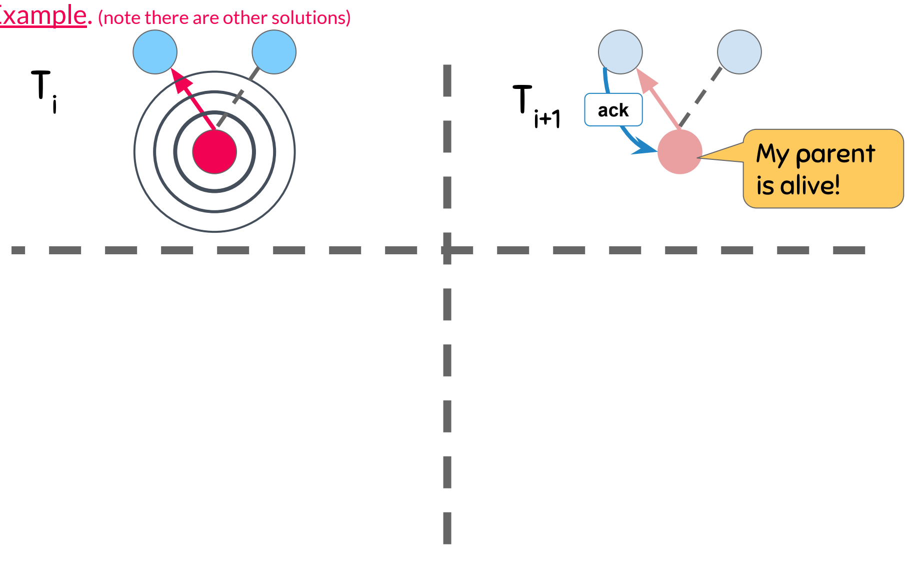
Routing under mobility events

Example. (note there are other solutions)



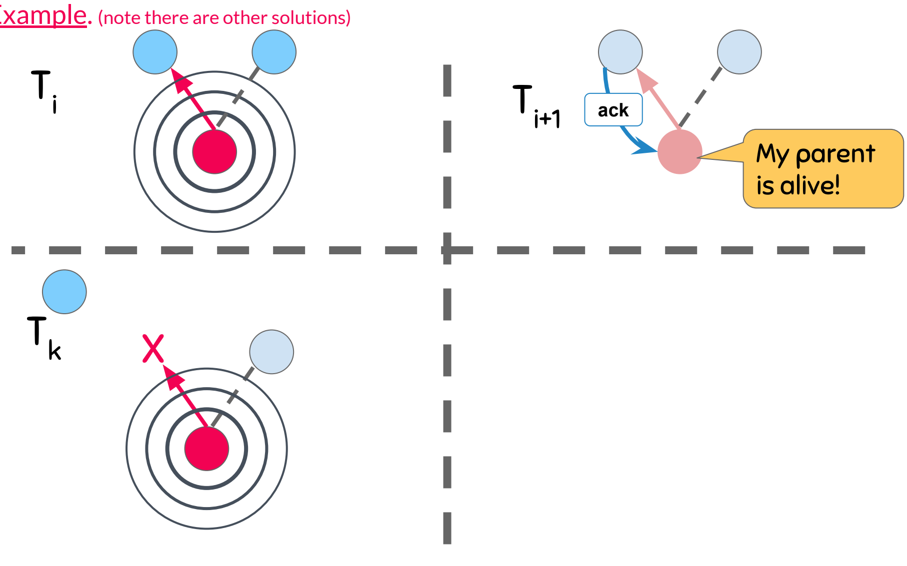
Routing under mobility events

Example. (note there are other solutions)



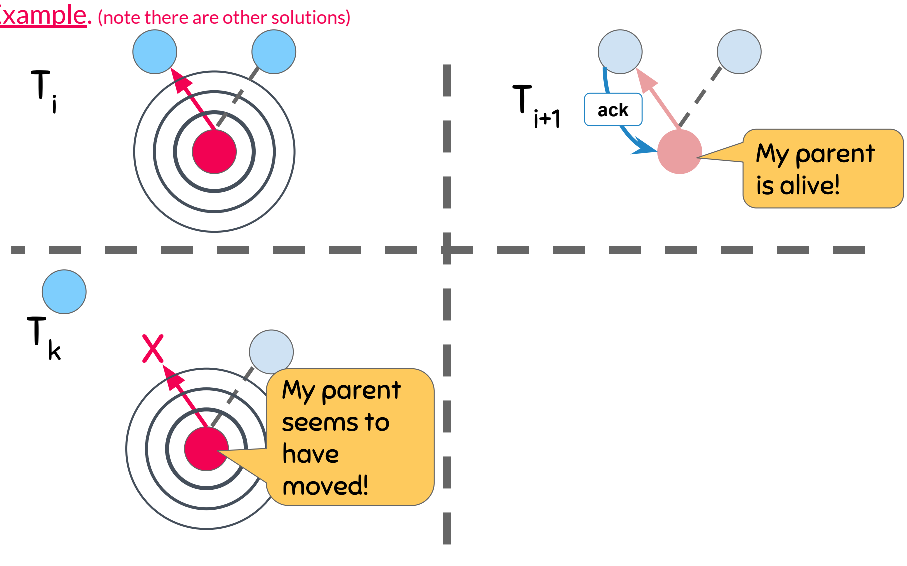
Routing under mobility events

Example. (note there are other solutions)



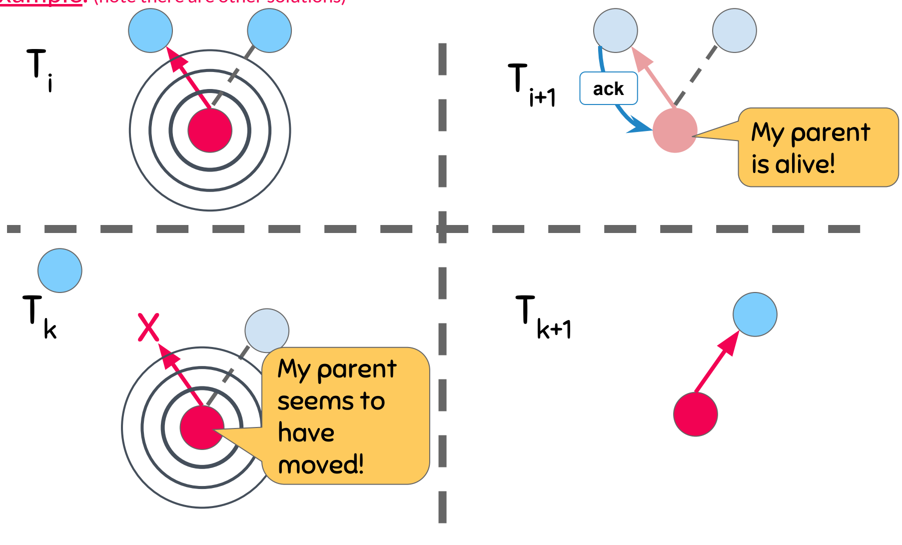
Routing under mobility events

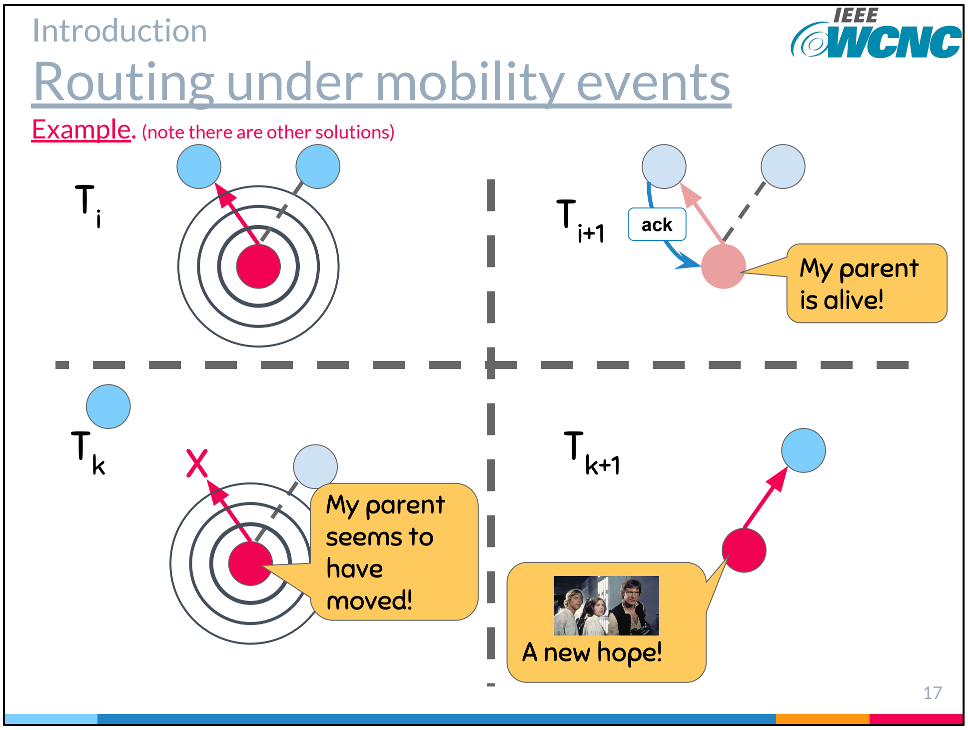
Example. (note there are other solutions)



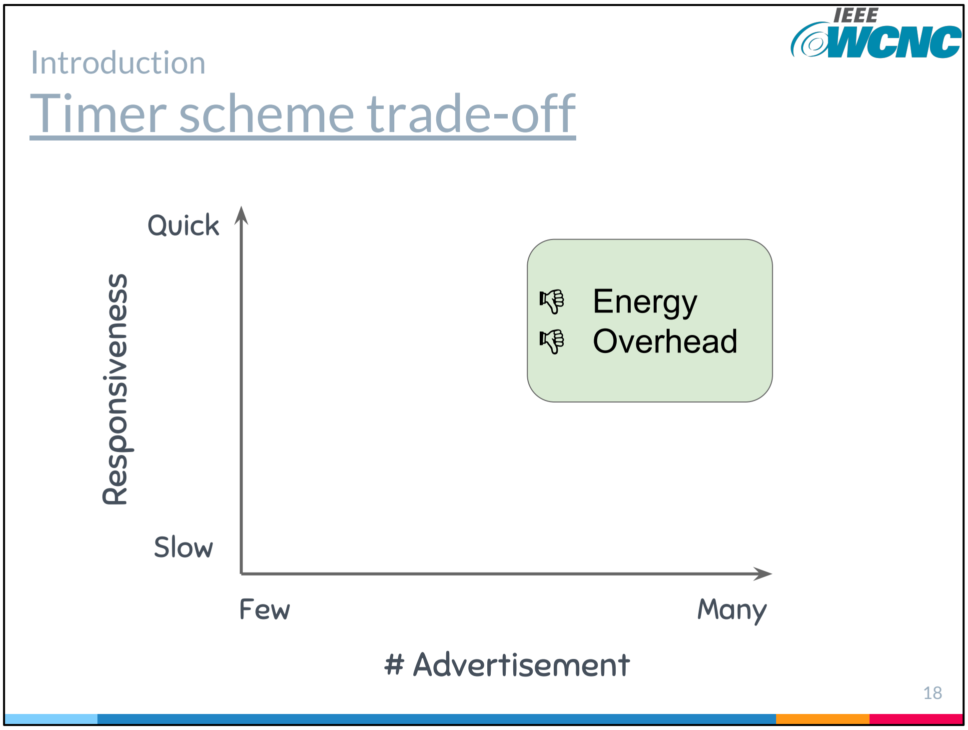
Routing under mobility events

Example. (note there are other solutions)

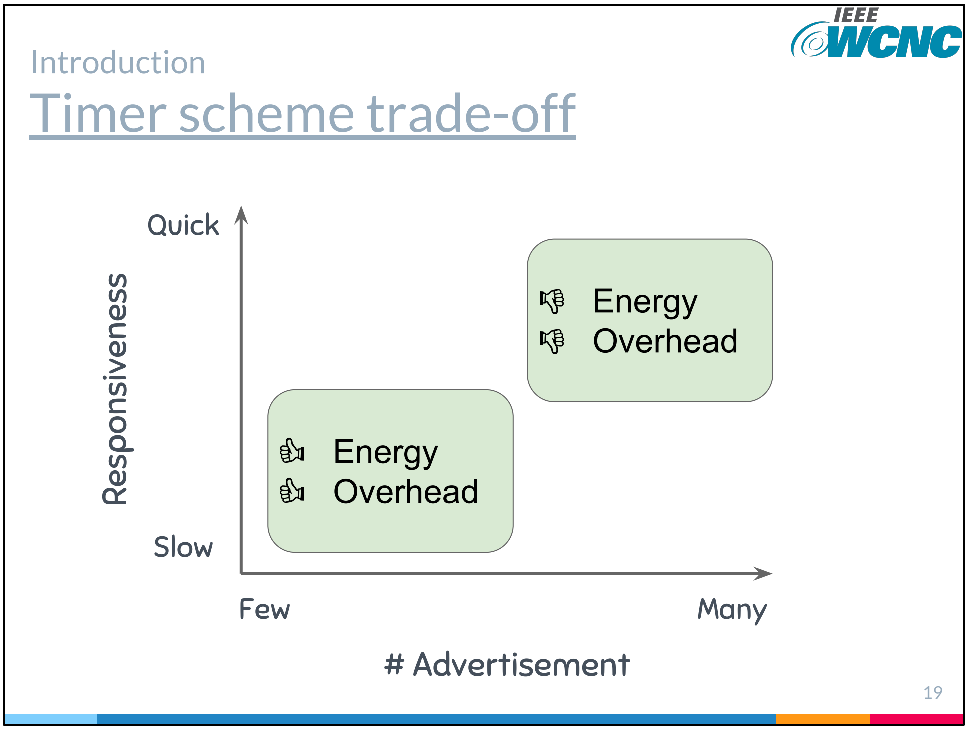




For example.
Here, we have a routing structure from red node to blue one.
From time to time, beacons (heatbeat) are broadcasted to check reachability to the parent.
After a heartbeat received, the parent answer with a ACK packet. And then, the red node knows that the parent is alive.
In Time k the red node tries again, but it did not receive a ACK. Then, the red node can trigger a new route mechanism.



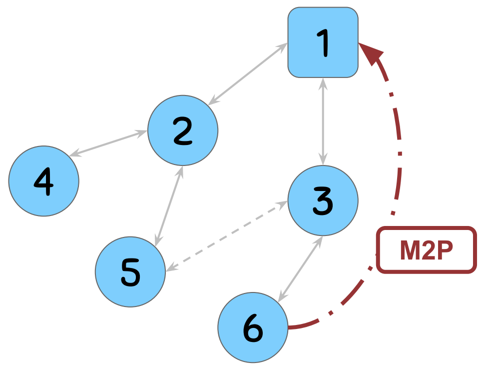
Here we face a basic timer scheme trade-off.
If the timer scheme is greedy to send beacons to quick catch topology changes, we will waste to much energy and introduce channel overhead.
However, if the timer scheme is conservative by sending few beacons, it will spend less energy and introduce low overhead to the channel, but it will be slow to catch topology changes.



Here we face a basic timer scheme trade-off.
If the timer scheme is greedy to send beacons to quick catch topology changes, we will waste to much energy and introduce channel overhead.
However, if the timer scheme is conservative by sending few beacons, it will spend less energy and introduce low overhead to the channel, but it will be slow to catch topology changes.

Background

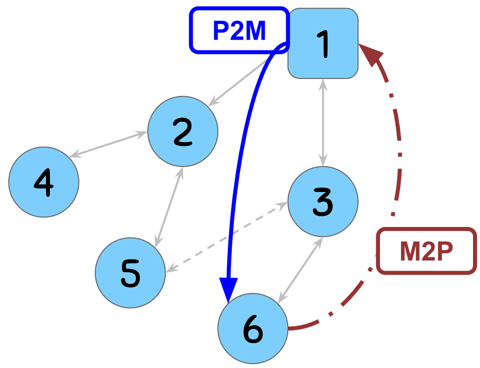
IoT routing in a nutshell



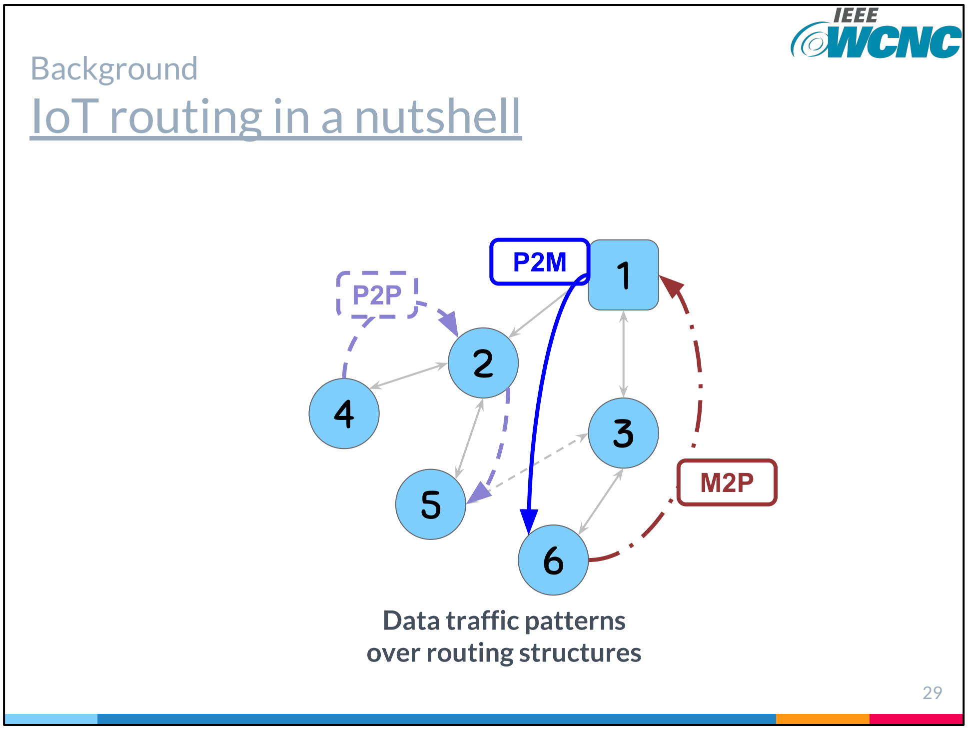
Data traffic patterns over routing structures

Background

IoT routing in a nutshell



Data traffic patterns
over routing structures



Usually, IoT routing protocols use such timer scheme to build and maintain routing structures to provide the following data traffic patterns: multipoint-to-point, point-to-multipoint, and point-to-point

Background

IoT routing in a nutshell

- Literature routing protocols
  - RPL (*de facto* the state-of-the-art)
    - Several RPL adaptations for mobile scenarios
      - Co-RPL, MRPL, MMRPL, ERPL...
  - Mobile Matrix
  - Hydro
  - XCTP
  - ...

RPL is the state-of-the-art routing protocol for IoT
Also, there are several RPL optimizations for mobile scenarios.
They typically change the timer scheme to handle mobility.
Also, from the literature there are others routing protocols like Mobile MAtrix, Hydro and XCTP for mobility environments

Background

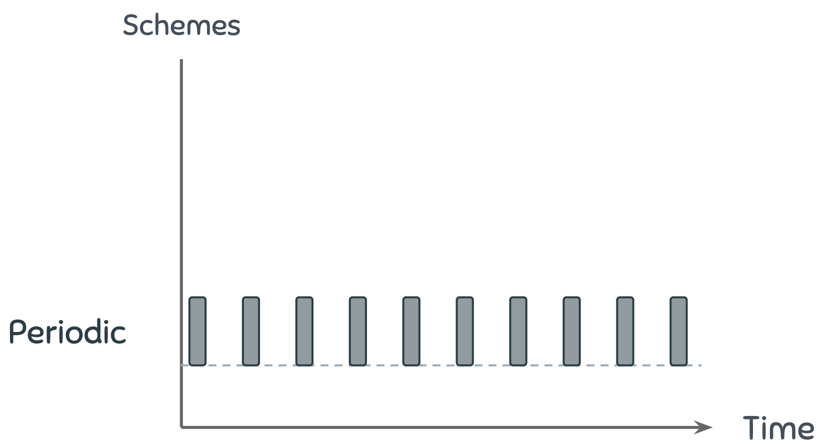
Dealing with mobility and link dynamics

- Timer schemes
  - Control advertisements
  - Govern the communication structure construction and maintenance
- What timer schemes are most commonly used?

Those routing protocols make use of Timer schemes…
But, what timer schemes are most commonly used?

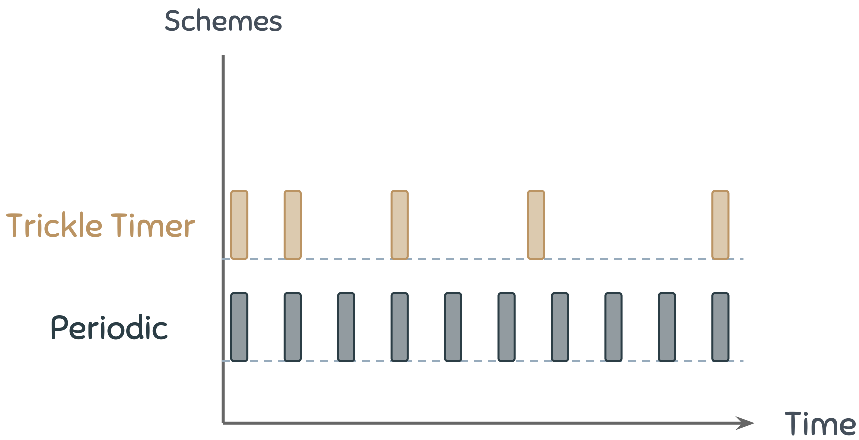
Background

Dealing with mobility and link dynamics



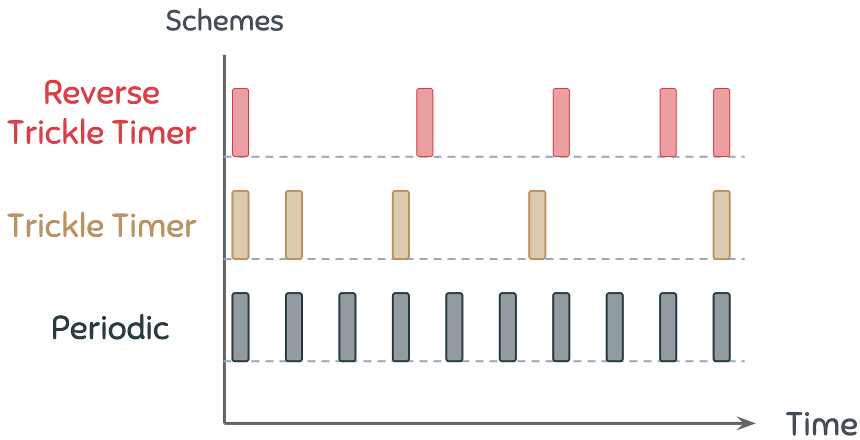
Background

Dealing with mobility and link dynamics



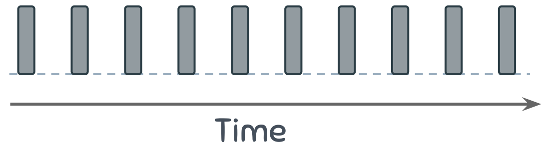
Background

Dealing with mobility and link dynamics



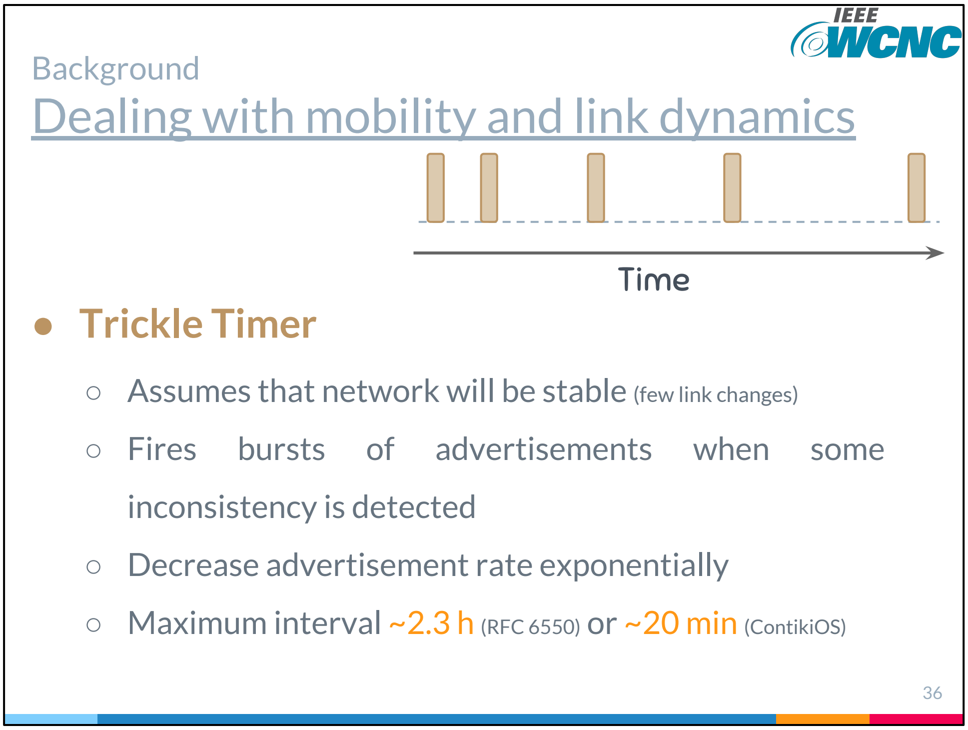
Background

Dealing with mobility and link dynamics



- **Periodic**
  - Large interval
    - 👍 Low channel and energy usage
    - 👎 Slow responsivity
  - Small interval
    - 👎 High channel and energy usage
    - 👍 Quick responsivity

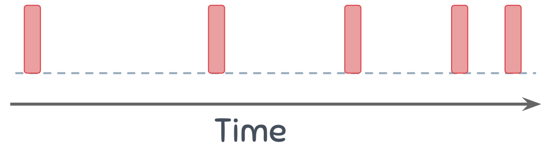
Self explanatory



Self explanatory

Background

Dealing with mobility and link dynamics



- **Reverse Trickle Timer**
  - The “opposite” of Trickle Timer
  - Assumes that as long as a node remains connected to a parent, it is likely that node will move away
  - Increase advertisement rate exponentially
  - Authors use ~20 min in their experiments

Self explanatory

Background

Dealing with mobility and link dynamics

1. **Reverse Trickle Timer,**
2. **Trickle Timer,**
3. **Periodic.**

- Such schemes assume:
    - 👎 Only one scheme governs the entire network

Self explanatory

Background

Dealing with mobility and link dynamics

1. Reverse Trickle Timer,

2. Trickle Timer,

3. Periodic.

- Such schemes assume:
  - Only one scheme governs the entire network
  - All devices follow the same mobility pattern

Self explanatory

Dribble

A learn-based timer scheme selector for mobility management in IoT

- It learns the IoT device mobility pattern
- Automatically assign a proper timer scheme
  - Better balance the timer scheme trade-off

We create Dribble timer scheme selector that learns the devices mobility patterns and automatically assign a proper timer scheme

Dribble

How it works...

Start with a default timer scheme

Ex:
Trickle Timer

Dribble

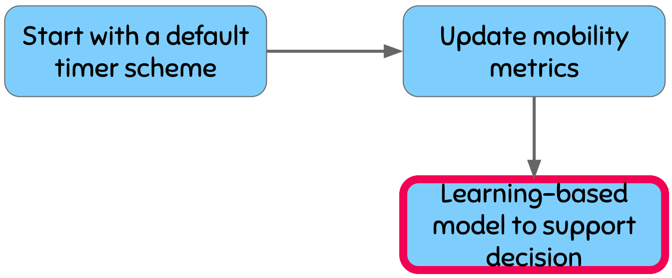
How it works...

Start with a default timer scheme → Process mobility metrics log

Ex:
- Speed,
- GPS,
- Travel Distance,
- Visit Time,
- Interconnection Time

Dribble

How it works...



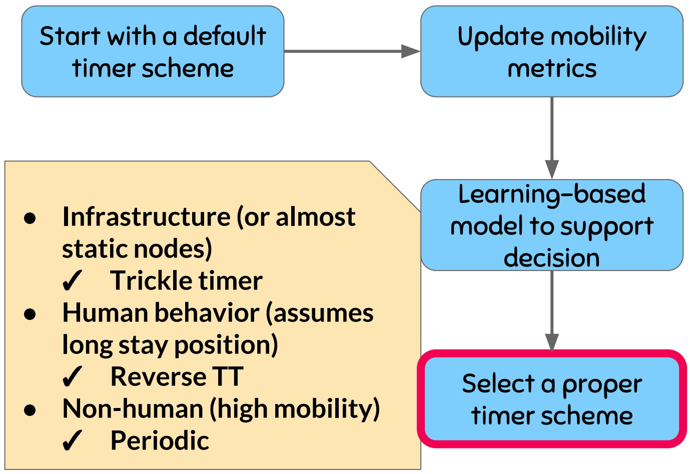
- We've tested
  - Supervised and unsupervised models
  - But we have labeled data
- Multi-Layer Perceptron classifier as learning algorithm

$$f : R^m \rightarrow R^p$$

m is the mobility metrics
p is the mobility patterns

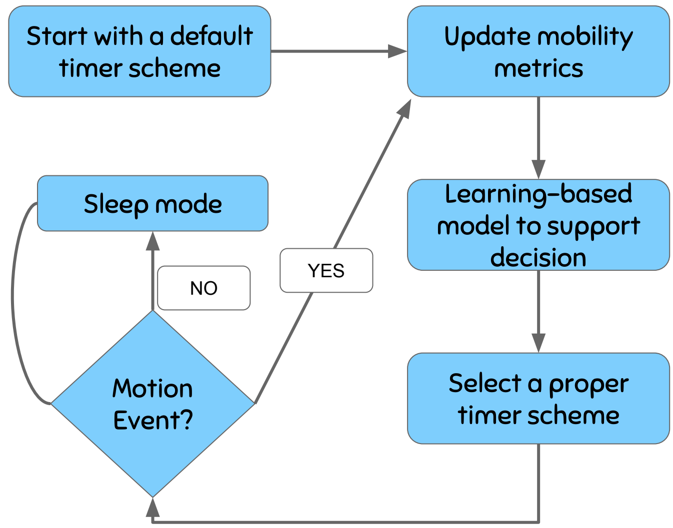
Dribble

How it works...



Dribble

How it works...



Evaluation

Experimental environment

- Sinalgo simulator
- RPL as routing protocol
  - Tree data traffic enabled: M2P, P2M, and P2P
  - Storing mode
  - ETX as Objective function

Self explanatory

Simulation setup	
Duration	15 days

Self explanatory. Focus on the orange one.

Simulation setup	
Duration	15 days
# nodes	200
Base station	1 (center)
Distribution	Random

Self explanatory. Focus on the orange one.

Simulation setup	
Duration	15 days
# nodes	200
Base station	1 (center)
Distribution	Random
DIM	1500m x 1500m (campus)

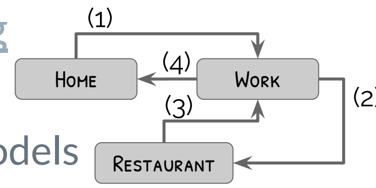
Self explanatory. Focus on the orange one.

Simulation setup	
Duration	15 days
# nodes	200
Base station	1 (center)
Distribution	Random
DIM	1500m x 1500m (campus)
Radio Range	100 (m)
Transmission Model	CC2420-like
# random topologies	15
Timer schemes	
Trickle and Reverse Trickle timers	Min = 1s, Max = ~20 min
Periodic	60s

Self explanatory. Focus on the orange one.

Evaluation

Device mobility modelling

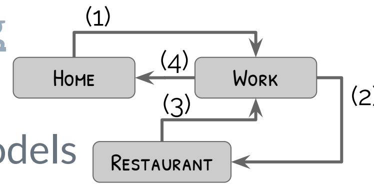


- We use two mobility models
  - Group Regularity Mobility model (GRM)
    - Human-like

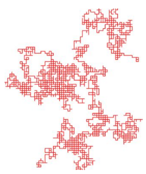
Humans follow cyclical mobility pattern: Home-work-restaurant-work-home. For this mobility pattern, we use GRM model

Evaluation

Device mobility modelling



- We use two mobility models
  - Group Regularity Mobility model (GRM)
    - Human-like

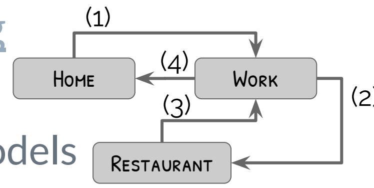


  - Cyclical Random Waypoint Mobility Model (CRWP)
    - Non-human

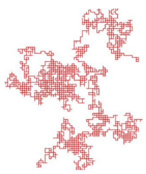
For non-human mobility pattern, we use a extension of RWP, called Cyclical RWP.

Evaluation

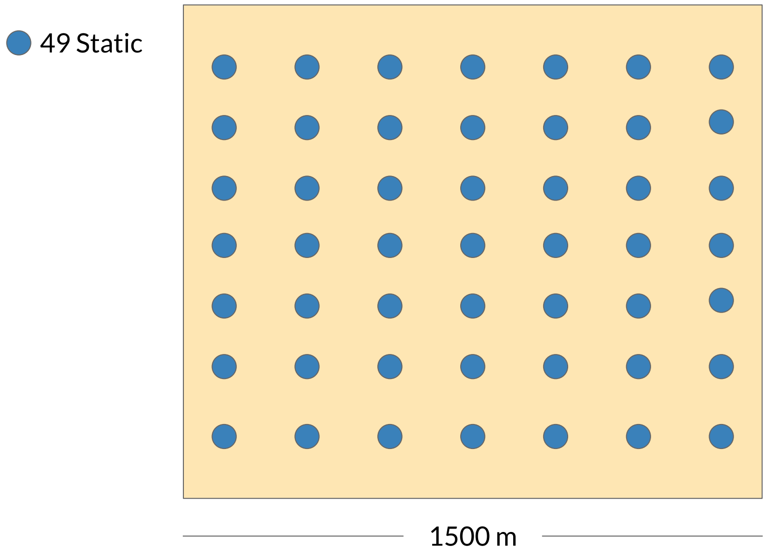
Device mobility modelling



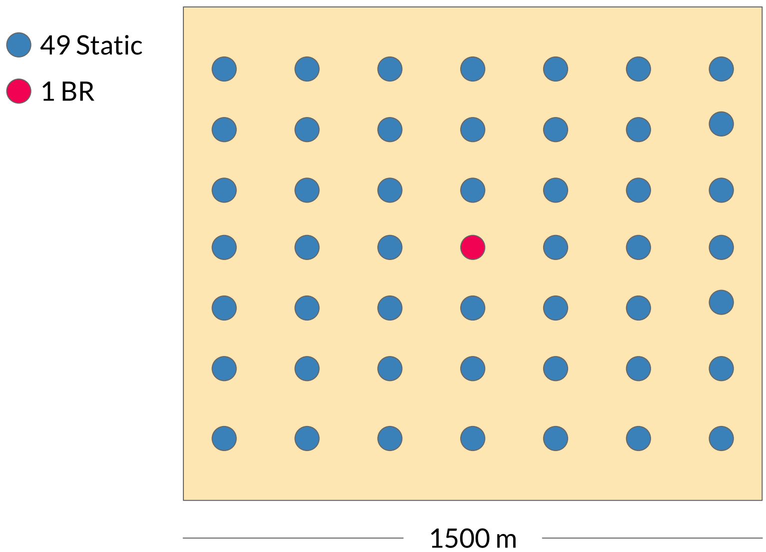
- We use two mobility models
  - Group Regularity Mobility model (GRM)
    - Human-like
  - Cyclical Random Waypoint Mobility Model (CRWP)
    - Non-human
- Some static nodes to represent the infrastructure



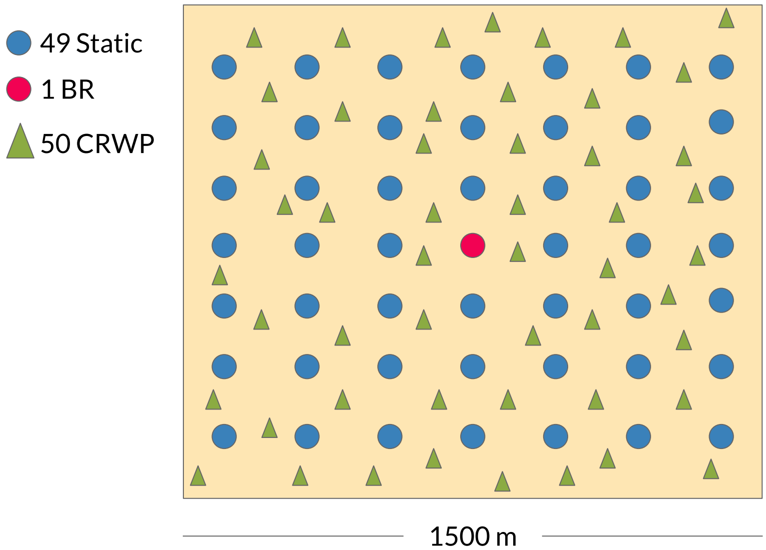
Also, some static nodes to represent the infrastructure.



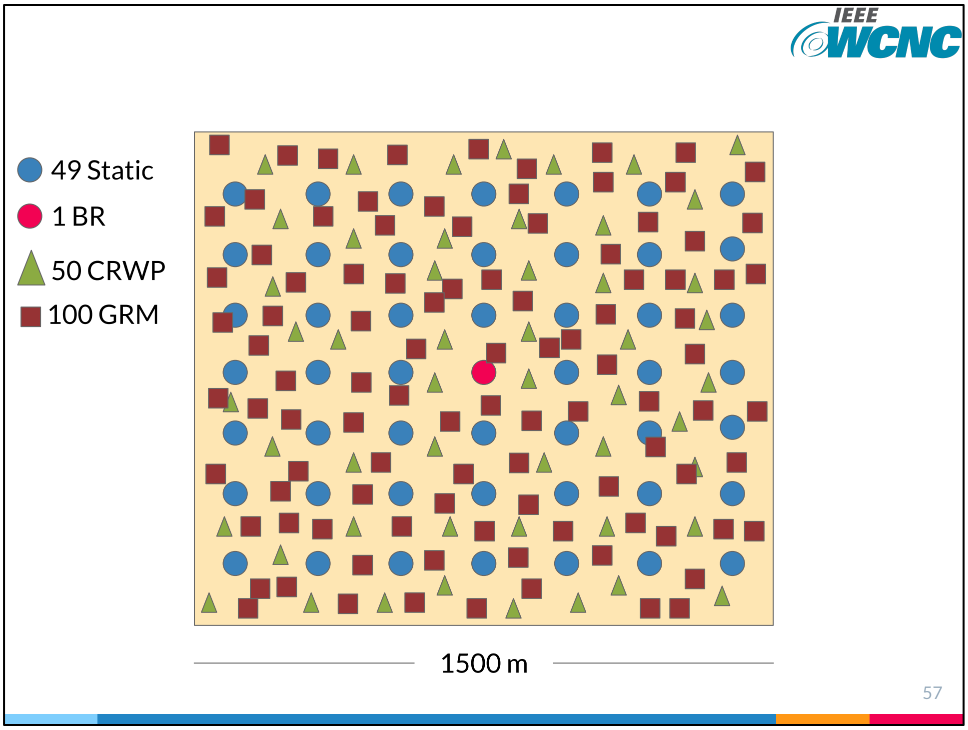
self-explanatory



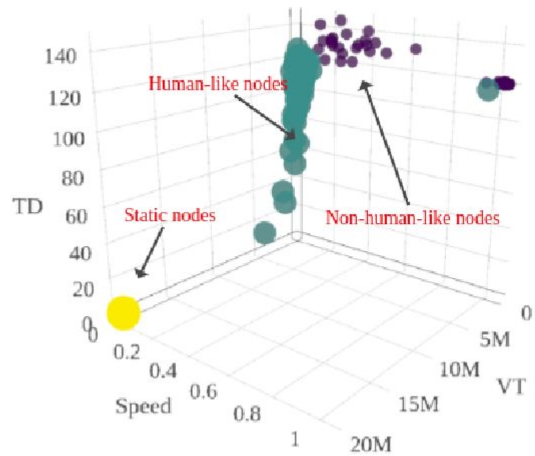
self-explanatory



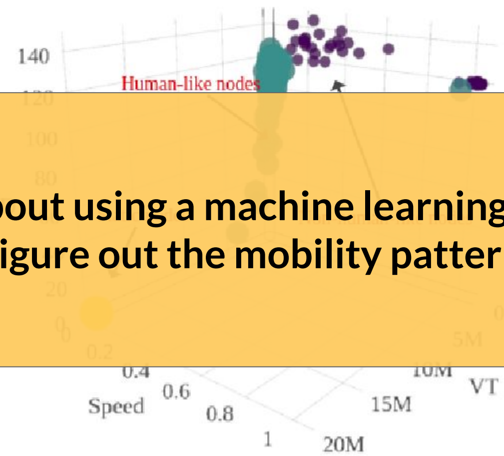
self-explanatory



self-explanatory



Travel distance, Speed, Visit time (mobility metrics from one of our mobility scenarios)



What about using a machine learning model to figure out the mobility pattern?

Self explanatory

Neural Network (Multi-Layer Perceptron) Architecture and parameters	
Architecture	1 Hidden layer with 100 neurons

Self explanatory focus on the orange one

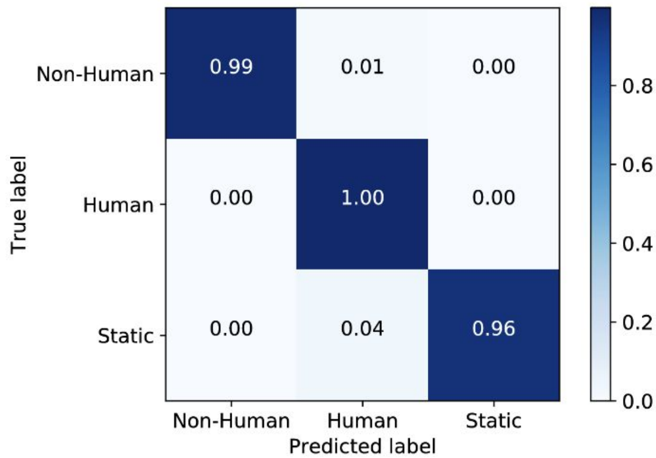
Neural Network (Multi-Layer Perceptron) Architecture and parameters	
Architecture	1 Hidden layer with 100 neurons
Activation	Rectified linear unit function
Learning rate	Constant
# epochs	500
Weight optimization	Adam
Train dataset	10 random topologies
Validation model	10-fold cross-validation

Self explanatory focus on the orange one

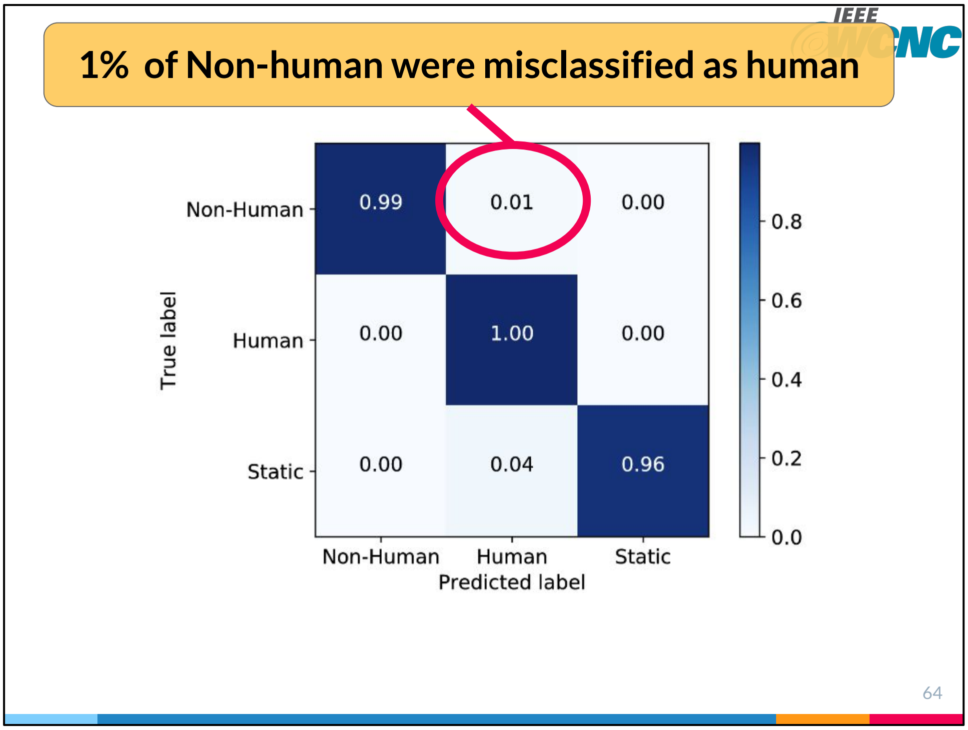
Neural Network (Multi-Layer Perceptron) Architecture and parameters	
Architecture	1 Hidden layer with 100 neurons
Activation	Rectified linear unit function
Learning rate	Constant
# epochs	500
Weight optimization	Adam
Train dataset	10 random topologies
Validation model	10-fold cross-validation

	Precision	Recall	F1-score	Support
Non-Human	1	0.99	0.99	165
Human	0.98	1	0.99	317
Static	1	0.96	0.98	171
Avg/Total	**0.99**	**0.99**	**0.99**	**653**

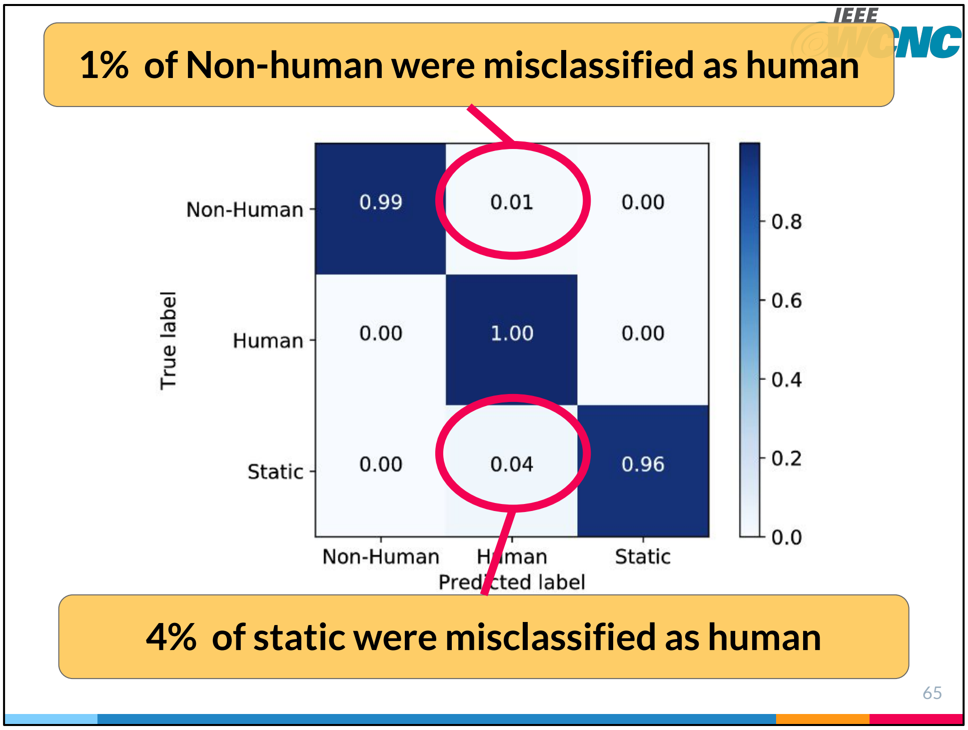
Self explanatory focus on the orange red



Self explanatory



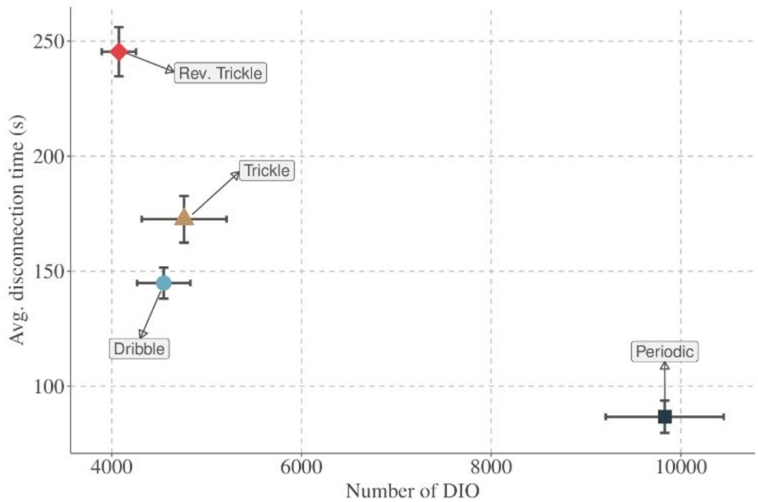
Self explanatory focus on the orange one



Self explanatory focus on the orange one

Evaluation

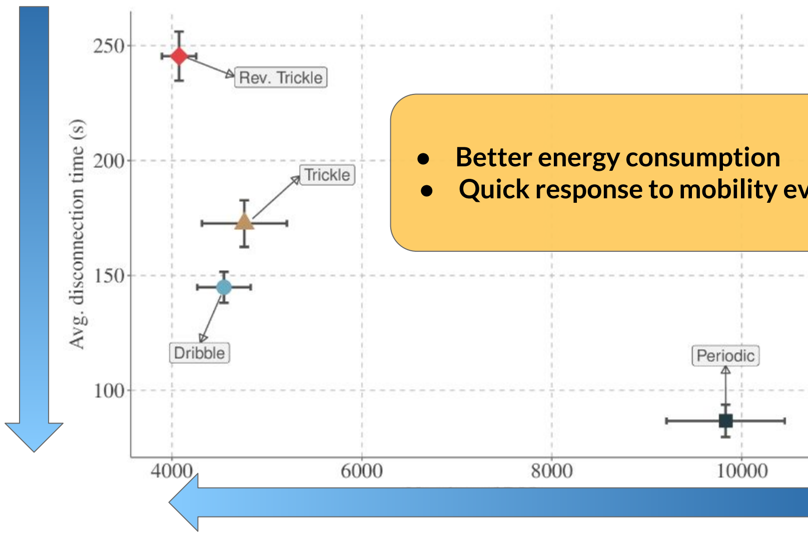
Trade-off balance



Self explanatory

Evaluation

Trade-off balance

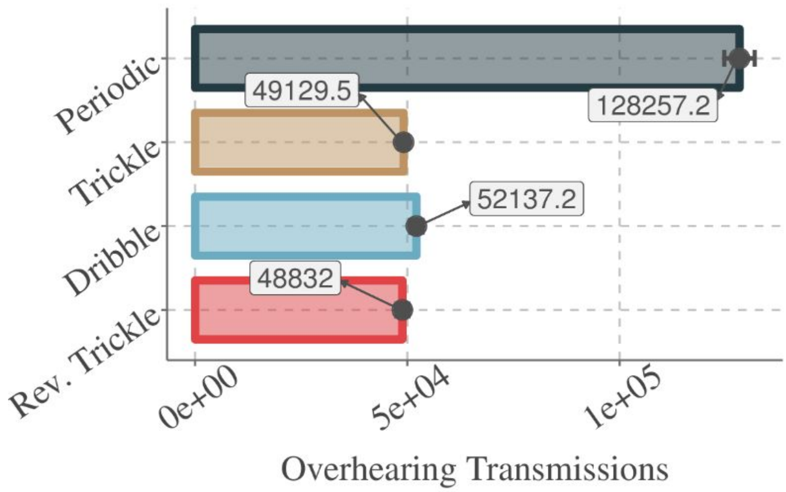


- **Better energy consumption**
- **Quick response to mobility events**

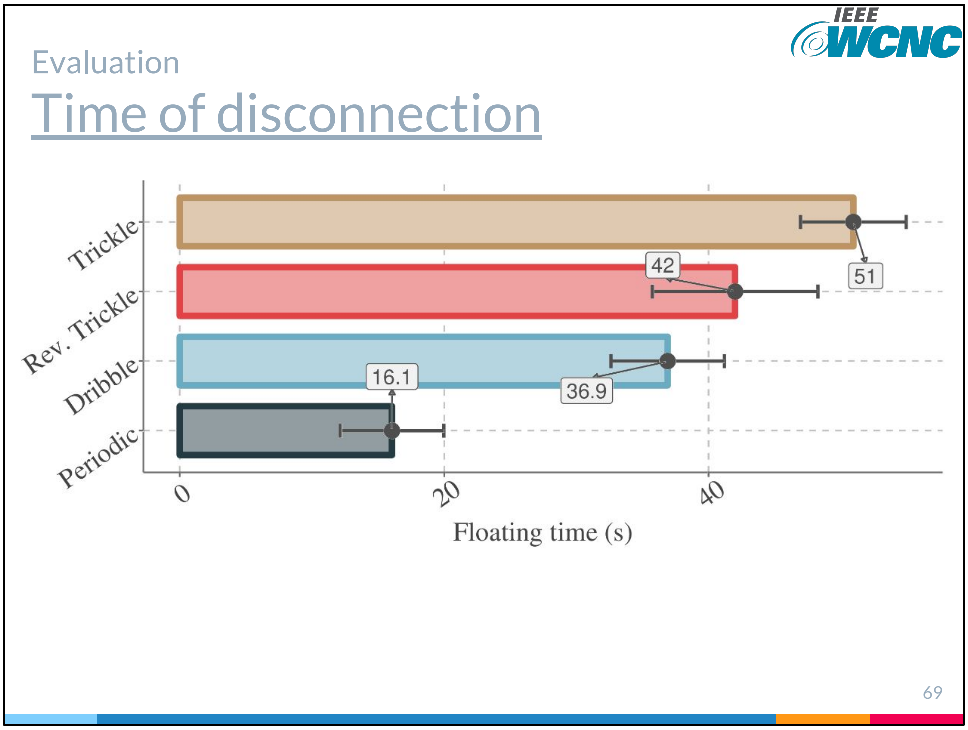
Self explanatory focus on the orange one

Evaluation

Overhead



Dribble shows similar overhead as Trickle and Rev. TT which are overhead efficient



Dribble presents shorter disconnection time than Trickle. Also, as expected, it shows higher disconnection time than Periodic. But Dribble spend less energy and introduce less overhead.

Conclusion and future work

- We have proposed Dribble
  - A learn-based time scheme selector
  - It sets a custom timer scheme given the mobility pattern of a IoT device
  - Also, Dribble presented a better timer scheme trade-off balance

Self explanatory

Conclusion and future work

- We intent to extend Dribble to support:
  - Automatic parametrization of timer schemes
  - Automatic way to associate mobility patterns to timer schemes.

Self explanatory

Thanks!

Any questions?

You can find us at:

- bruno.ps@dcc.ufmg.br
- rettore@dcc.ufmg.br
- lfvieira@dcc.ufmg.br
- loureiro@dcc.ufmg.br