DynPolAC: Dynamic Policy-based Access Control for IoT Systems

Mehdi Karimibiuki, Ekta Aggarwal, Karthik Pattabiraman and Andre Ivanov

December 6, 2018

The 23rd IEEE Pacific Rim International Symposium on Dependable Computing

Taipei, Taiwan

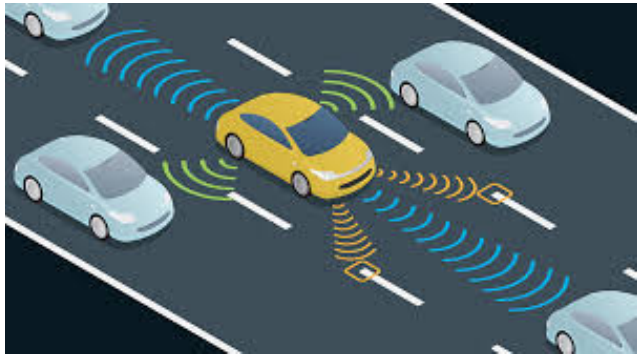
Motiation: IoT Space

- The number of IoT systems are growing
- About 26 devices per person

200 billion IoT devices

Motivation: mobile IoT by 2020

7 million unmanned aircraft systems (UAS)



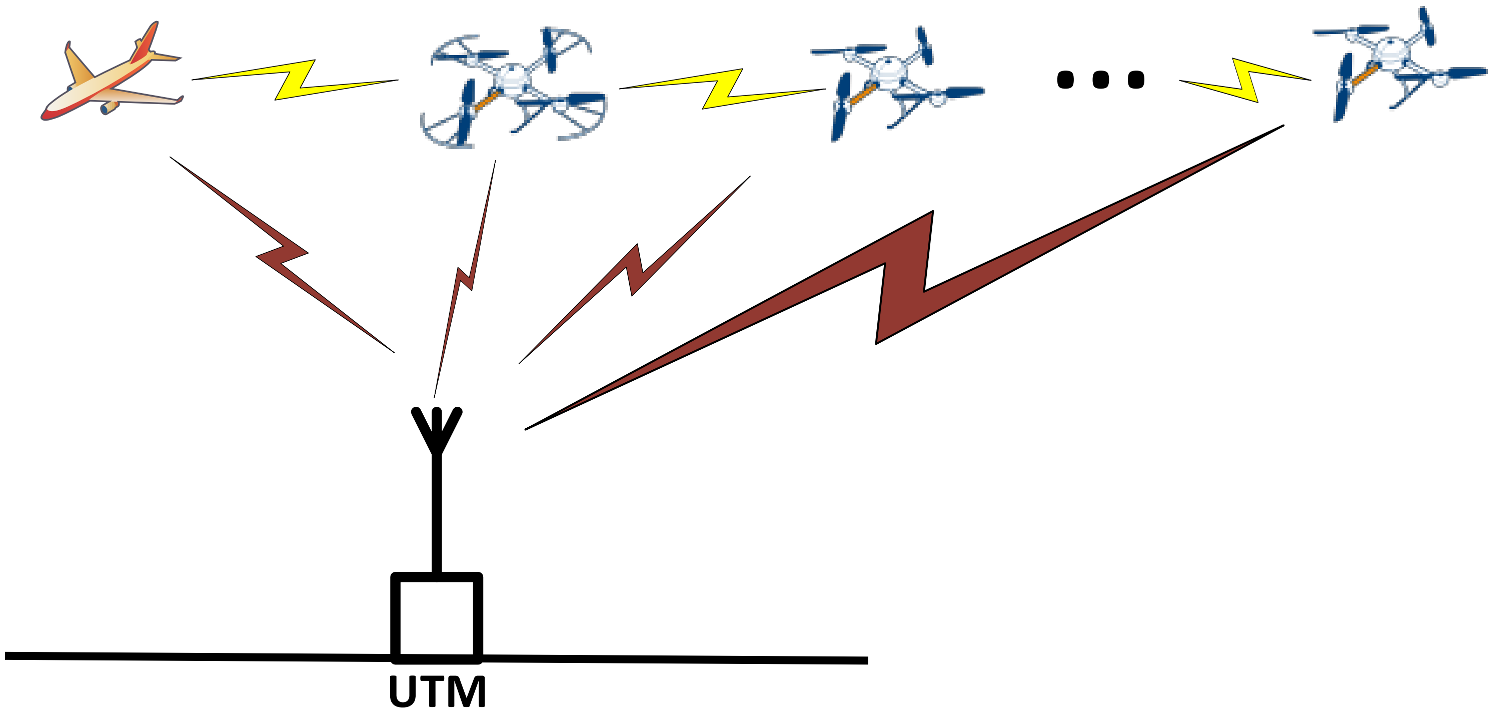
10 million connected vehicles

1 in 4 cars are autonomous by 2030

Autonomous IoT: Drones

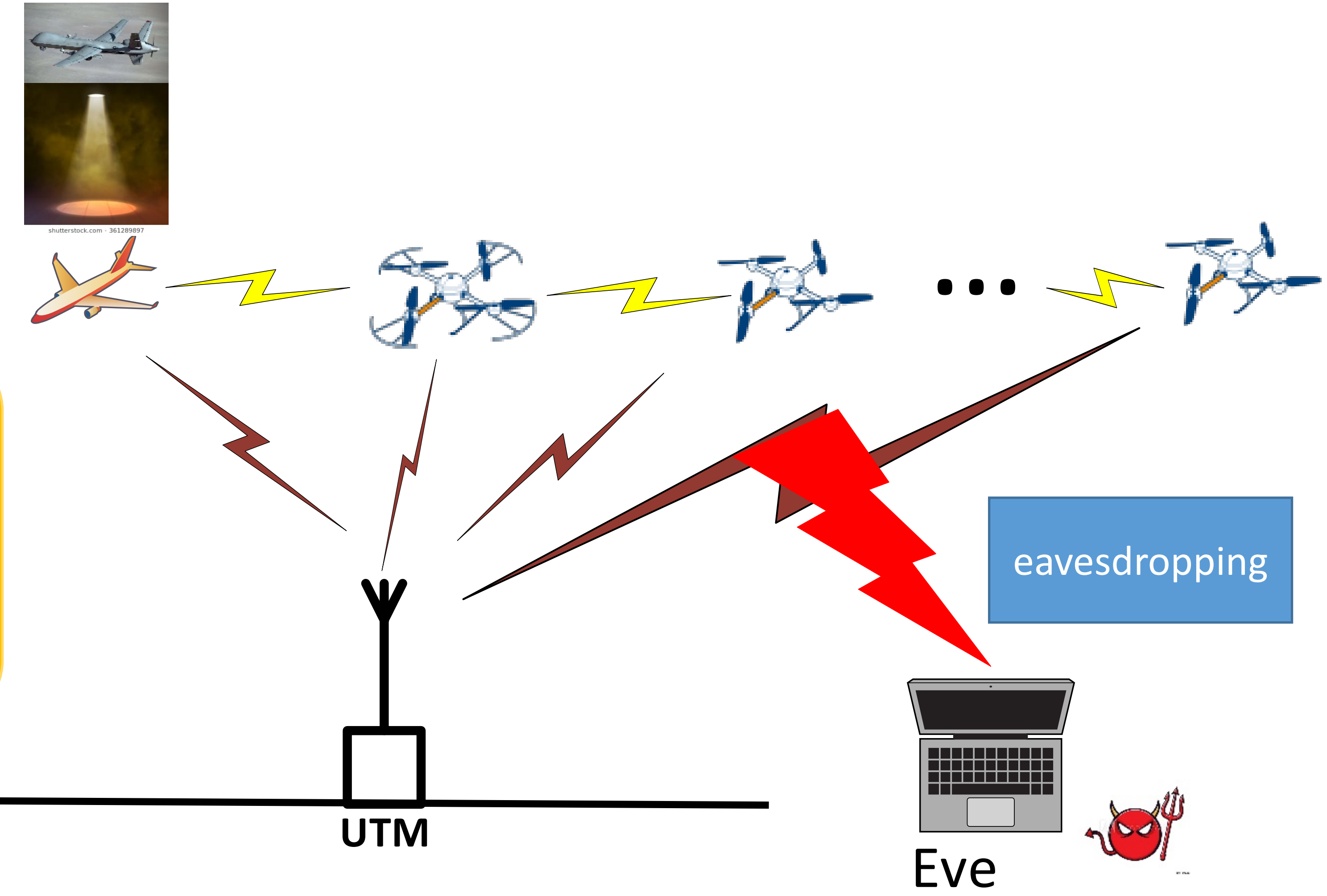
(1) Moving objects have higher levels of interaction than stationary networks

(2) with linear growth of IoT nodes, communication between them grow quadratically $\frac{n(n-1)}{2}$



Problem: Malicious Attacks

spy and snoop



eavesdropping

Our Goal: Develop an authorization scheme for highly interactive IoT systems

Challenges

High interaction means fast authorization required

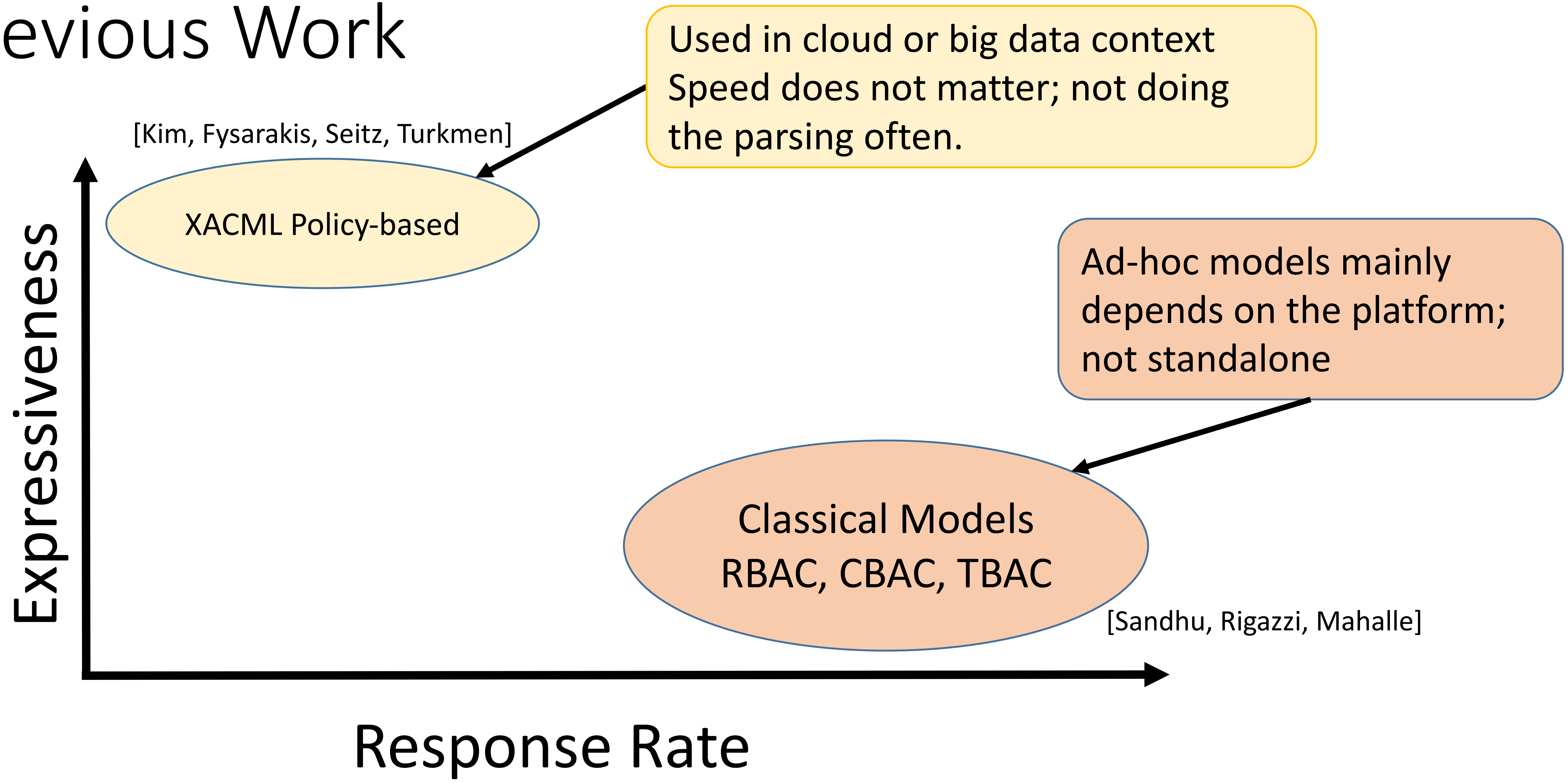
Dynamic IoT nodes are constrained systems

➢Weight limitation

➢Power consumption

Communication in congested networks could build up quadratically → so does authorization

Previous Work



Used in cloud or big data context
Speed does not matter; not doing the parsing often.

[Kim, Fysarakis, Seitz, Turkmen]

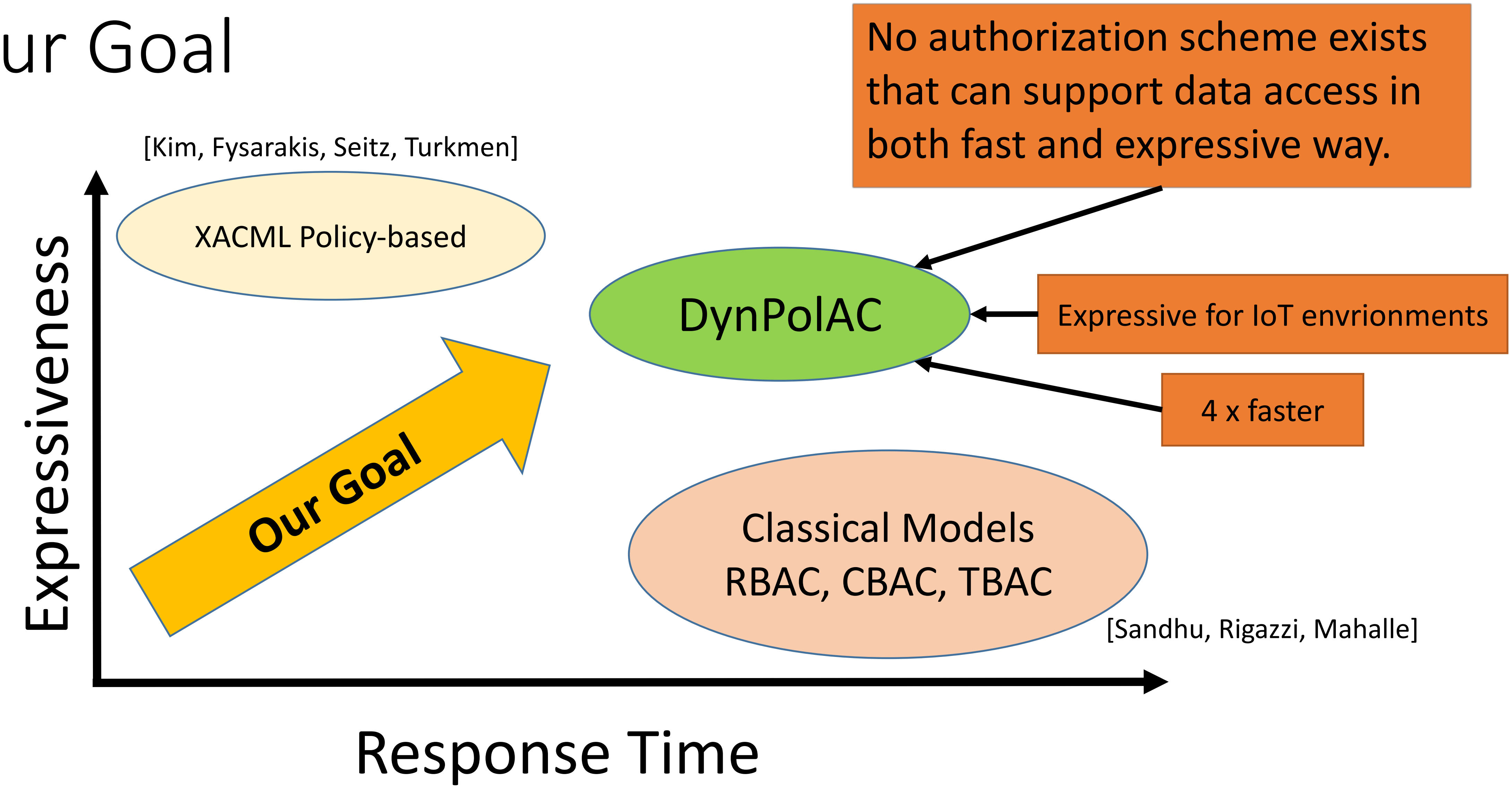
XACML Policy-based

Ad-hoc models mainly depends on the platform; not standalone

Classical Models
RBAC, CBAC, TBAC

[Sandhu, Rigazzi, Mahalle]

Our Goal



No authorization scheme exists that can support data access in both fast and expressive way.

[Kim, Fysarakis, Seitz, Turkmen]

XACML Policy-based

DynPolAC

Expressive for IoT envrionments

4 x faster

Our Goal

Classical Models
RBAC, CBAC, TBAC

[Sandhu, Rigazzi, Mahalle]

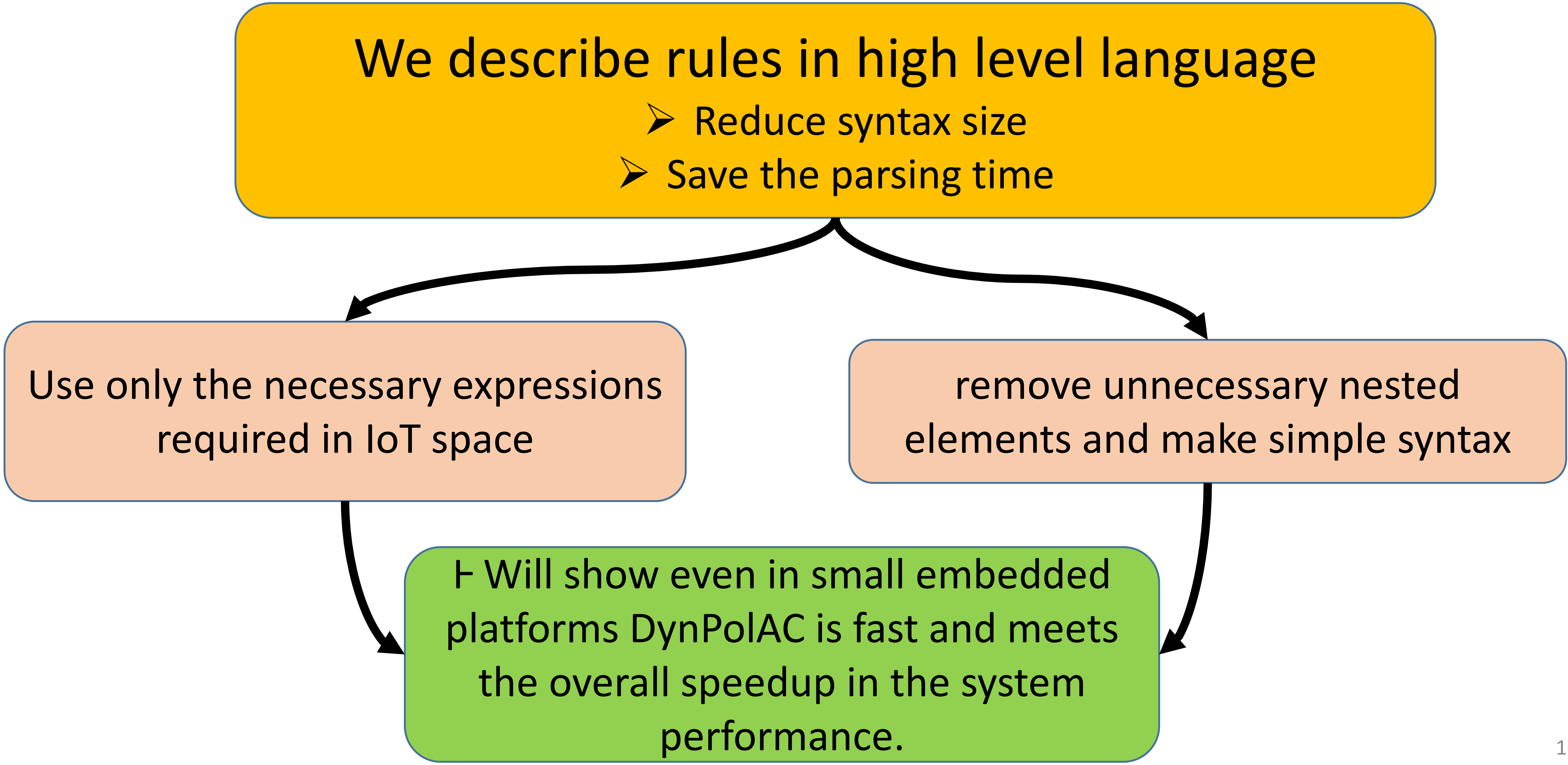
Expressiveness

Response Time

Outline

- Motivation
- Approach
- evaluation

DynPolAC: Key Insight



We describe rules in high level language

- ➢ Reduce syntax size
- ➢ Save the parsing time

Use only the necessary expressions required in IoT space

remove unnecessary nested elements and make simple syntax

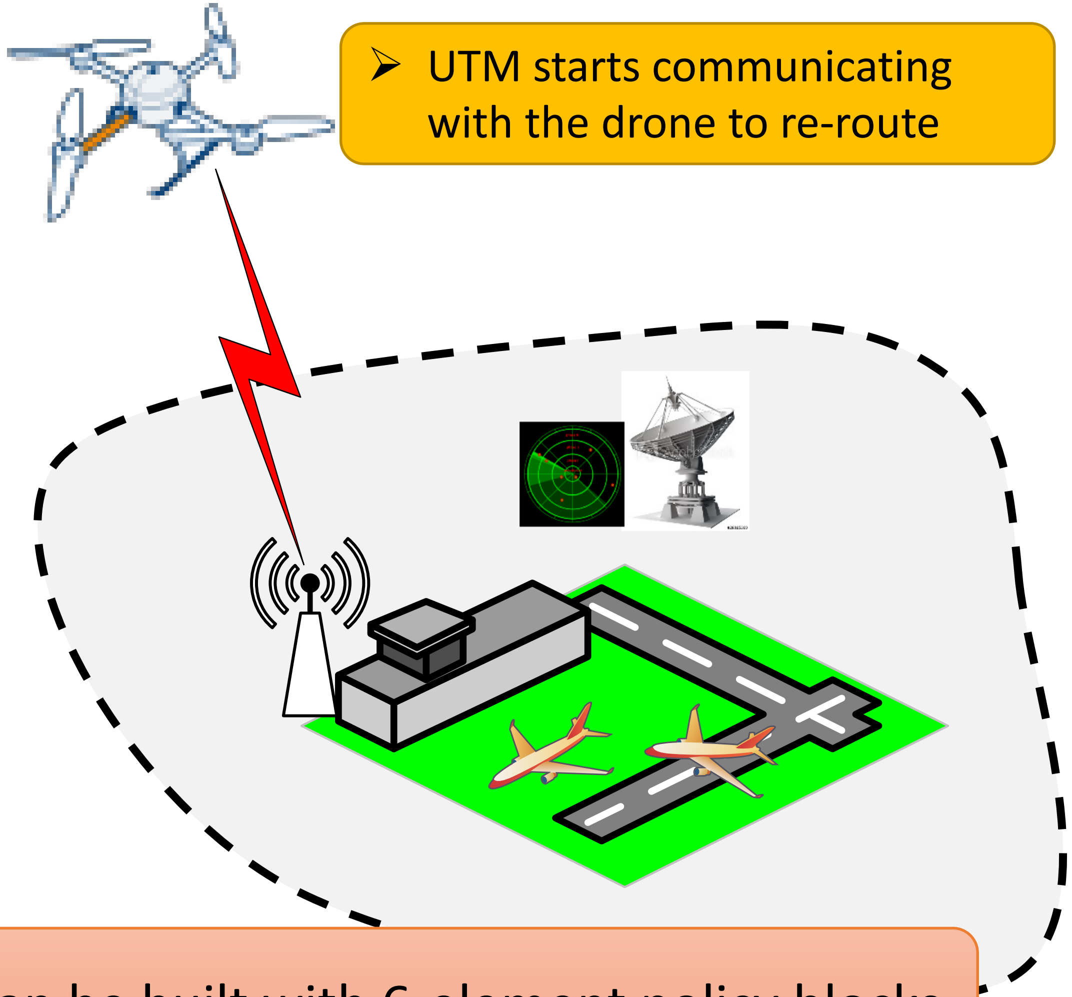
⊦ Will show even in small embedded platforms DynPolAC is fast and meets the overall speedup in the system performance.

DynPolAC: No-fly Zone

We can construct rules with 6 primitives only

1. Access type: Permit
2. Data type: coordinates
3. Drone name: Friendly
4. Time: ALL
5. User: UTM
6. Group: Airport

- ➢ Radar sees an unknown asset
- ➢ UTM starts communicating with the drone to re-route



- ➢ Spatial, temporal, and role-based expressions can be built with 6-element policy blocks

DynPolAC: Comparison

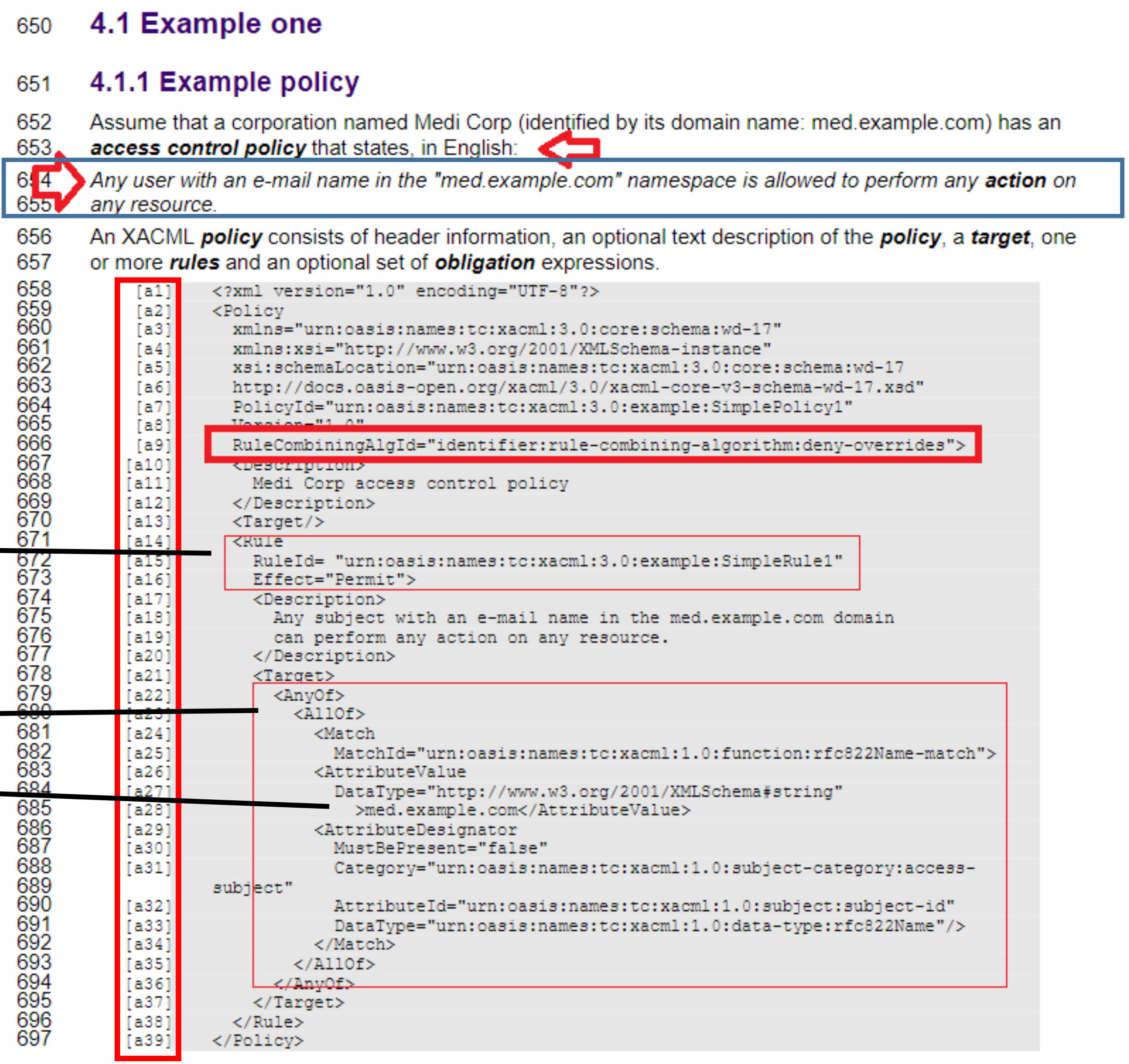
Let's see how rules look in previous model?

DynPolAC

```
1.  <policy>
2.    <rule>access</rule>
3.    <attributes>
4.       <type>email</type>
5.       <vendor>MediCorp</vendor>
6.       <time>ANY</time>
7.       <user>ANY</user>
8.       <group>med.example.com</group>
9.    </attributes>
10. </policy>
```

removed unnecessary nested elements, still 6 primitives, made simple syntax

10 vs. 39



4.1 Example one

4.1.1 Example policy

Assume that a corporation named Medi Corp (identified by its domain name: med.example.com) has an ***access control policy*** that states, in English:

*Any user with an e-mail name in the "med.example.com" namespace is allowed to perform any **action** on any resource.*

An XACML ***policy*** consists of header information, an optional text description of the ***policy***, a ***target***, one or more ***rules*** and an optional set of ***obligation*** expressions.

```
[a1]   <?xml version="1.0" encoding="UTF-8"?>
[a2]   <Policy
[a3]     xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
[a4]     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a5]     xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
[a6]     http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
[a7]     PolicyId="urn:oasis:names:tc:xacml:3.0:example:SimplePolicy1"
[a8]     Version="1.0"
[a9]     RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
[a10]    <Description>
[a11]      Medi Corp access control policy
[a12]    </Description>
[a13]    <Target/>
[a14]    <Rule
[a15]      RuleId= "urn:oasis:names:tc:xacml:3.0:example:SimpleRule1"
[a16]      Effect="Permit">
[a17]      <Description>
[a18]        Any subject with an e-mail name in the med.example.com domain
[a19]        can perform any action on any resource.
[a20]      </Description>
[a21]      <Target>
[a22]        <AnyOf>
[a23]          <AllOf>
[a24]            <Match
[a25]              MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
[a26]            <AttributeValue
[a27]              DataType="http://www.w3.org/2001/XMLSchema#string"
[a28]                >med.example.com</AttributeValue>
[a29]            <AttributeDesignator
[a30]              MustBePresent="false"
[a31]              Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
       subject"
[a32]              AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[a33]              DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
[a34]            </Match>
[a35]          </AllOf>
[a36]        </AnyOf>
[a37]      </Target>
[a38]    </Rule>
[a39]  </Policy>
```

Reference: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf

Outline

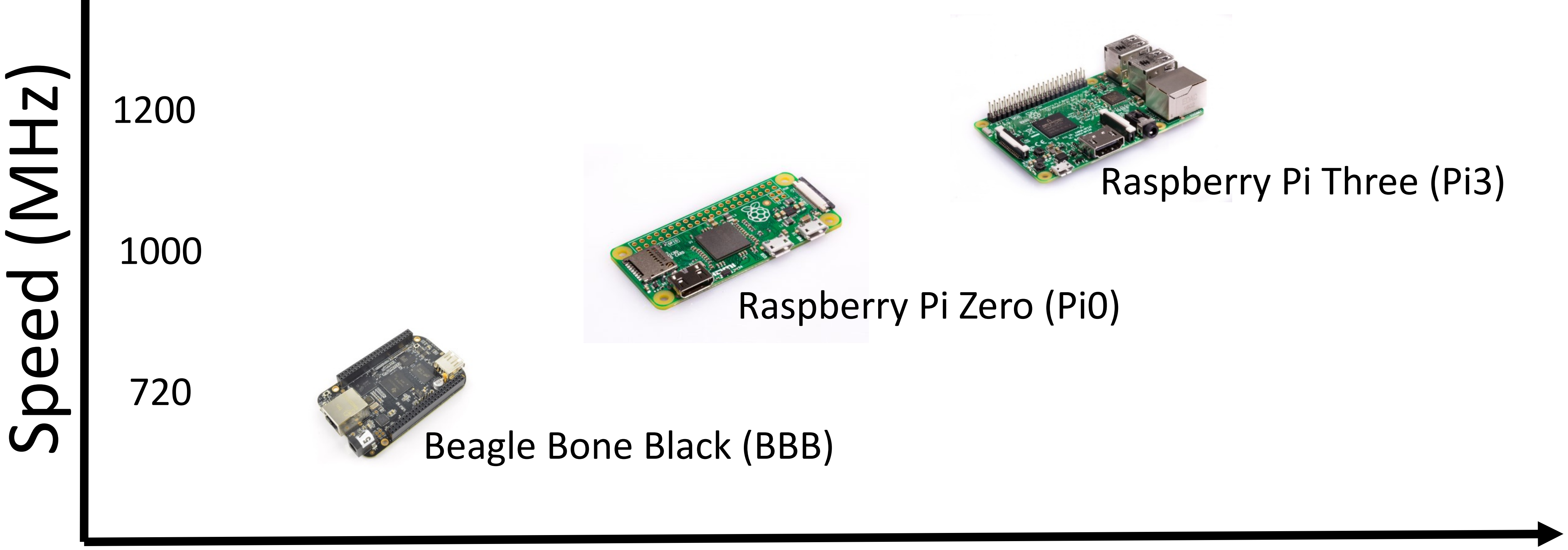
- Motivation
- Approach
- evaluation

Research Questions

- RQ1. At micro-level, what is the processing time improvement?
- RQ2. At system-level, what is the response time?
    - ➢Check stability condition
        - o Can it meet requests in interactive environments?
    - ➢Sensitivity analysis
        - o what is the bottleneck in extreme scenarios?
- RQ3. What is the memory overhead?

Experimental Setup

Instrument DynPolAC in 3 different embedded platforms



Speed (MHz)

1200

1000

720

Raspberry Pi Three (Pi3)

Raspberry Pi Zero (Pi0)

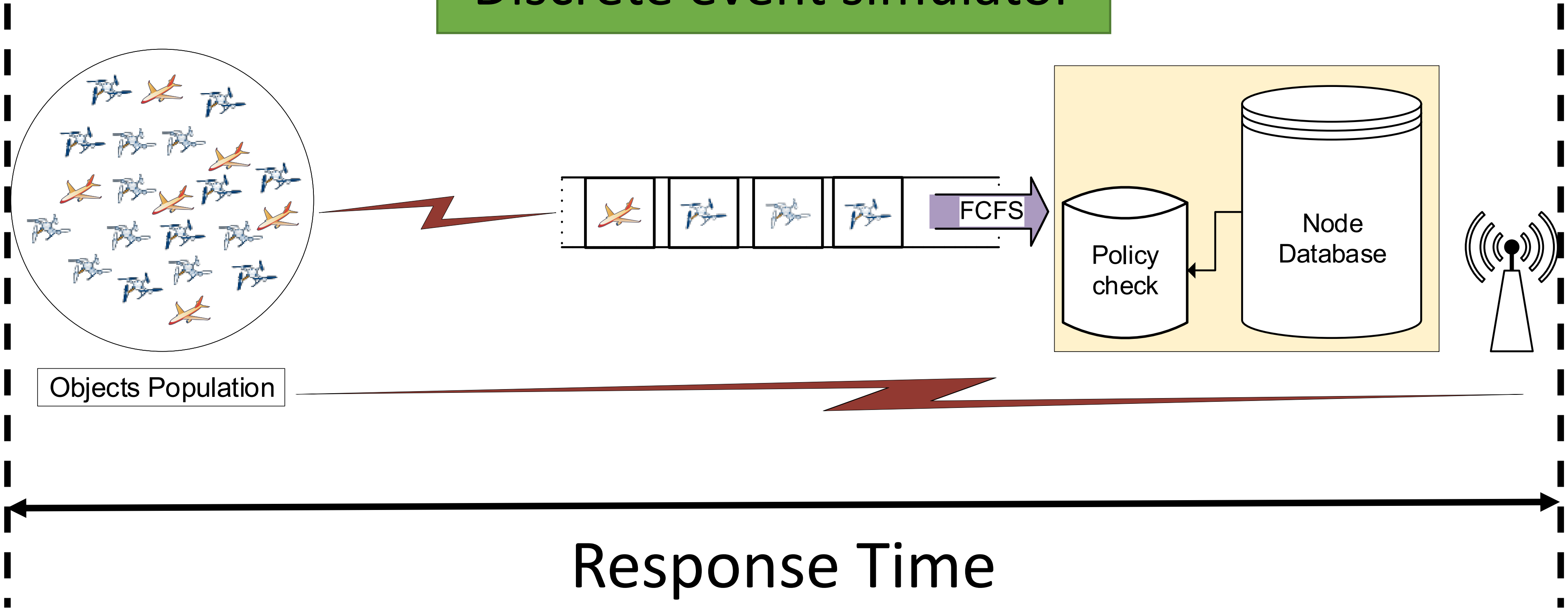
Beagle Bone Black (BBB)

Goal: show the homogeneity of our results in different platforms

Experimental Setup

System Study: Emulate an interactive IoT environment

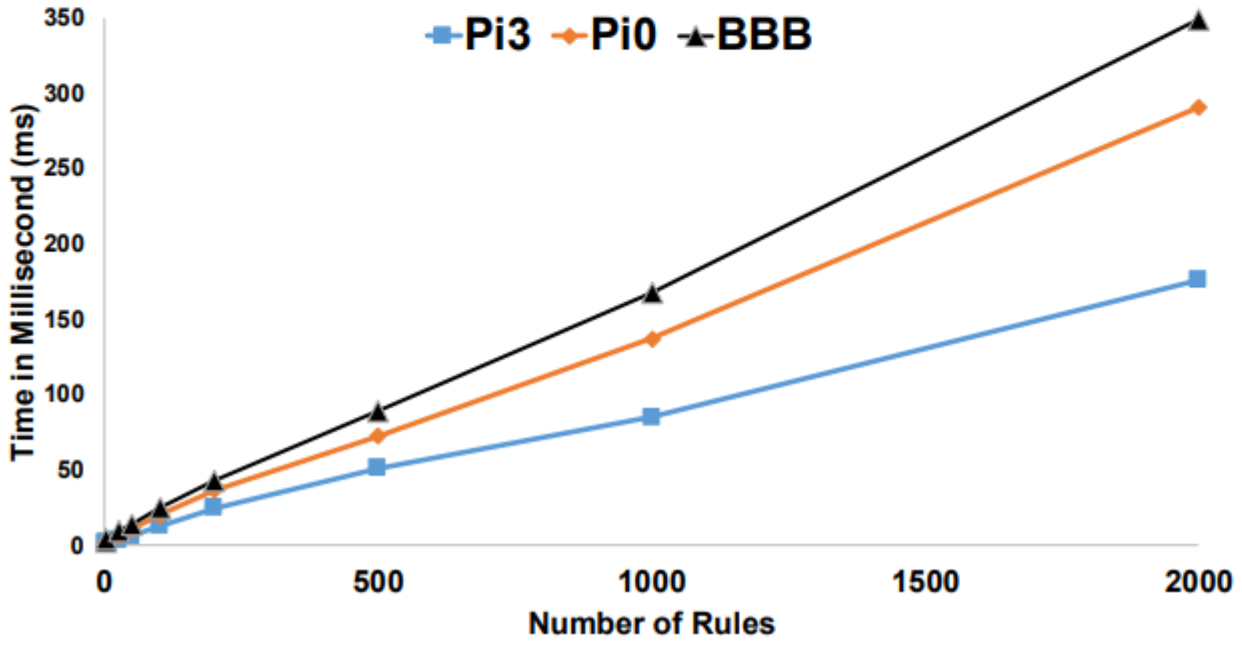
Discrete event simulator



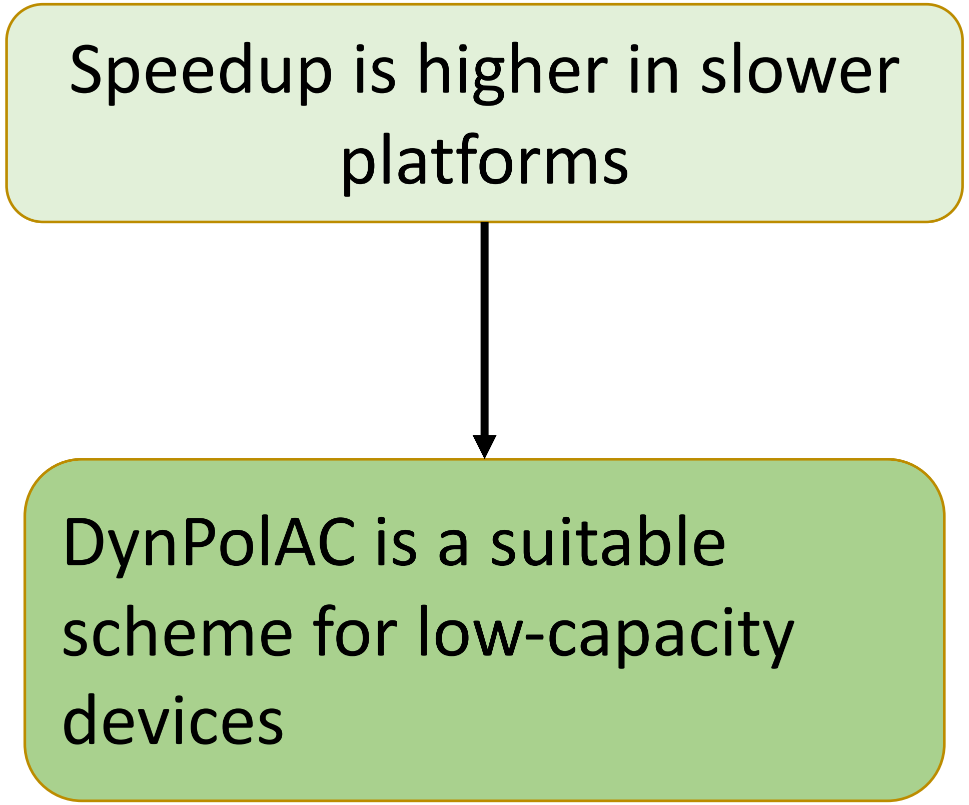
Response Time

RQ1. Processing Time

Rules by DynPolAC syntax are parsed and processed in milliseconds
Less than half seconds

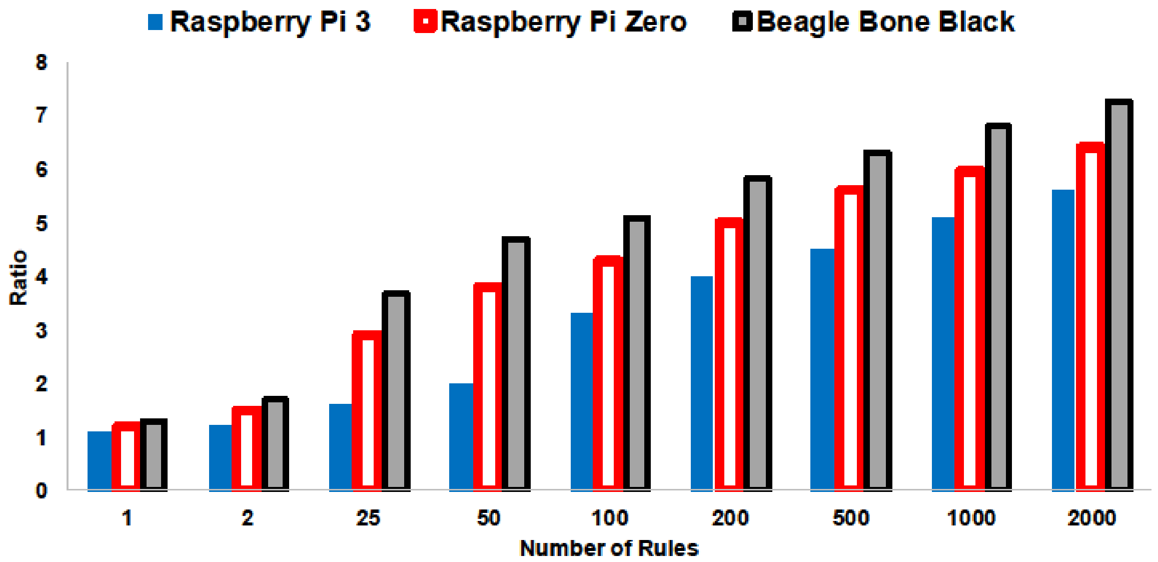


RQ1. Comparison



Speedup is higher in slower platforms

DynPolAC is a suitable scheme for low-capacity devices

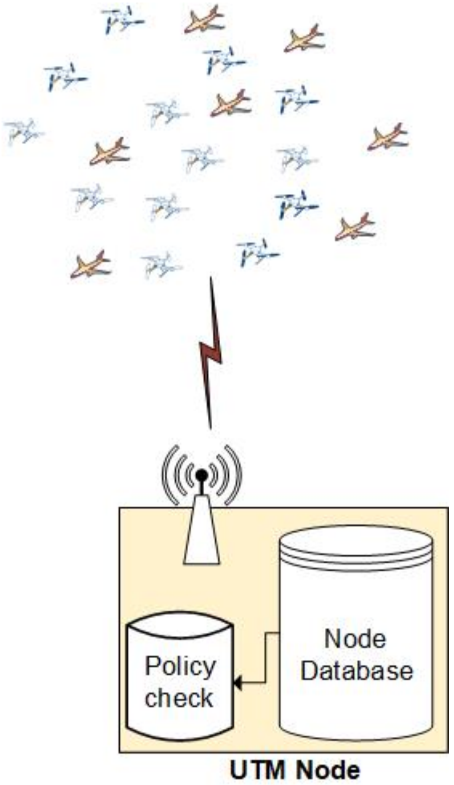


✓On Average 4x process improvements

✓up to 7.27x speedup

RQ2. System Stability Condition

Parameter	Unit	value
Arrival rate (λ)	1/s	1 - 8
Size of query	Bytes	200 – 5K
Size of policy	No. of rules	1 - 2000

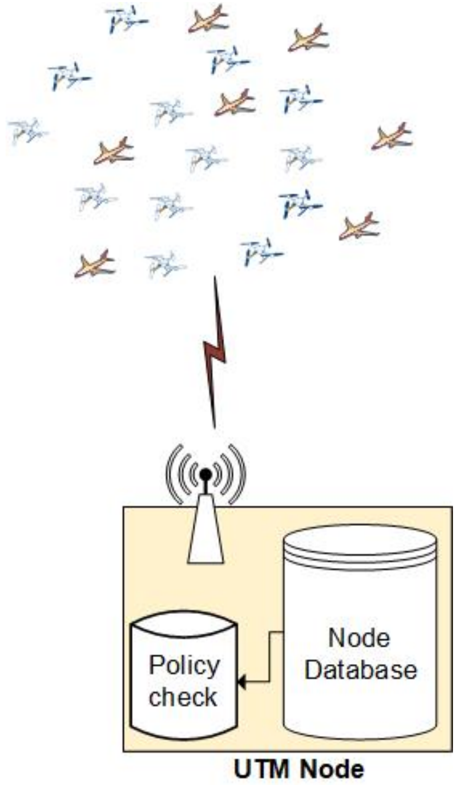


Goal: measure system performance by calculating the response time

The end-to-end time of a drone to initiate the request until the reply is received.

RQ2. System Stability Condition

Parameter	Unit	value
Mean Arrival rate (λ)	1/s	4
Size of query	Bytes	200 – 5K
Size of policy	No. of rules	1 - 2000



simulation	response time (ms)	Response rate (μ)
No-policy	178	5.6
DynPolAC	245	4.1
XACML	840	1.2

DynPolAC satisfies the stability condition being right above the threshold of 4.

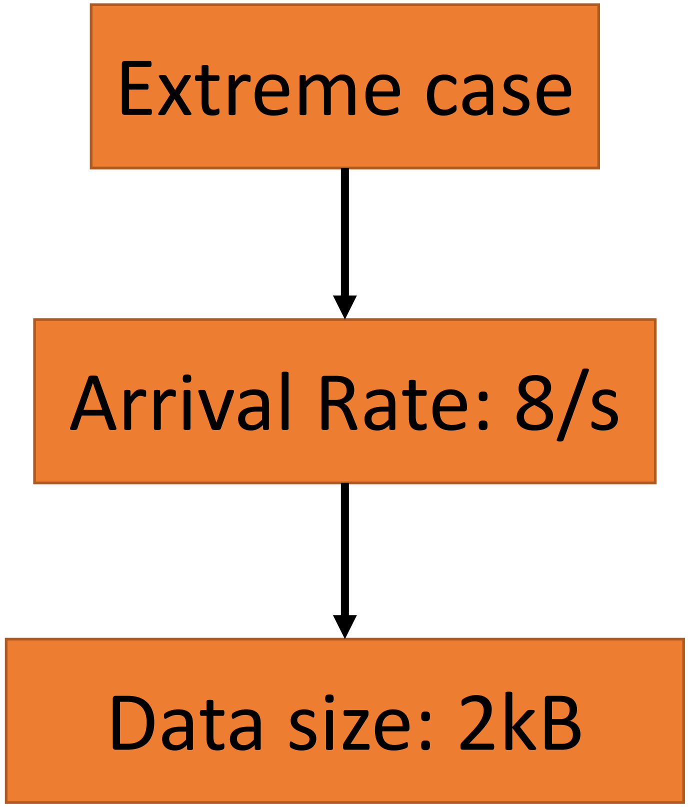
RQ2. Stability Condition

simulation	response time (ms)	Response rate (μ)
No-policy	178	5.6
DynPolAC	245	4.1
XACML	840	1.2

XACML-based policy systems experience instability, so it cannot keep up requests!!

DynPolAC improves the overall response time by 70%.

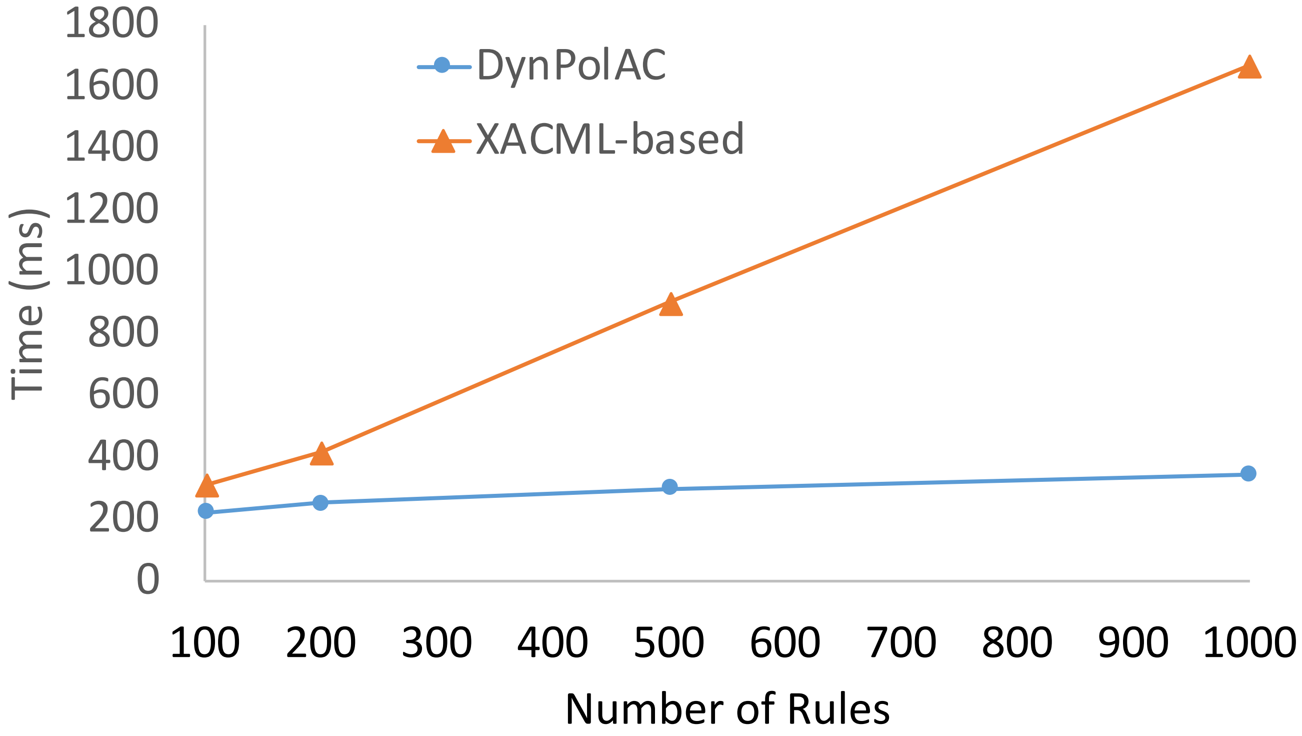
RQ2. Sensitivity Study



Extreme case

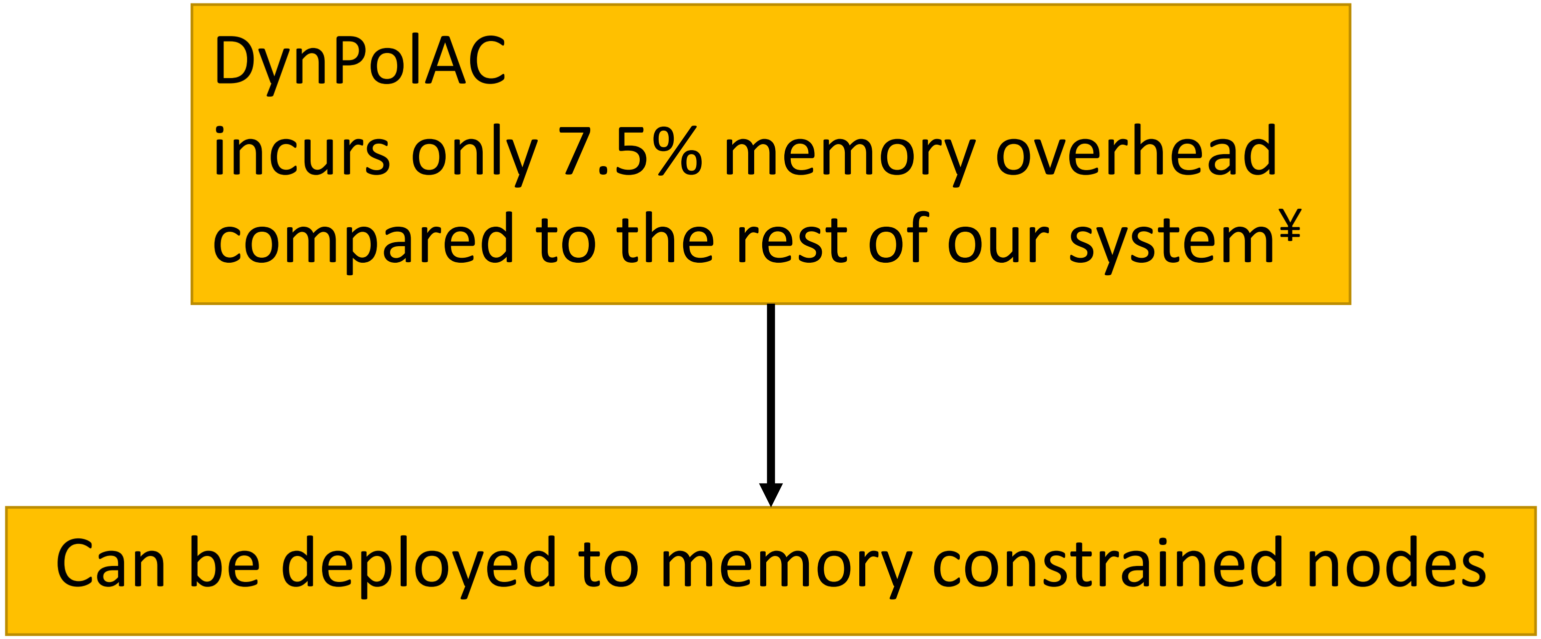
Arrival Rate: 8/s

Data size: 2kB



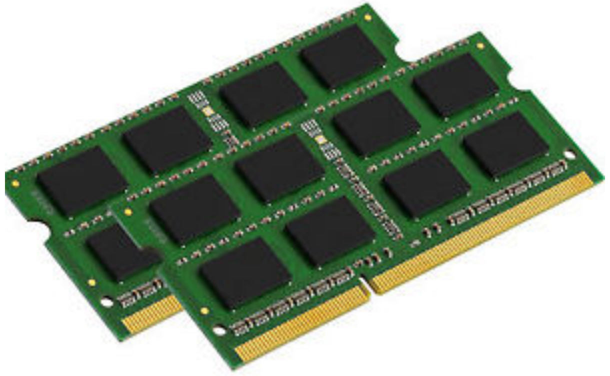
Sweep the number of rules

RQ3. Memory Overhead



DynPolAC
incurs only 7.5% memory overhead compared to the rest of our system[¥]

Can be deployed to memory constrained nodes



[¥] Karimibiuki, Mehdi, and André Ivanov. "MiniCloud: a mini storage and query service for local heterogeneous IoT devices." *Proceedings of the 8th International Conference on the Internet of Things*. ACM, 2018.

Summary

- Looked at a scenario of dynamic IoT system
    - DynPolAC is the solution to securely authenticate dynamic objects
- Insight: DynPolAC has a crisp language selection
    - high-level language to express very low-level parameters.
    - expresses similar rules compared to previous work.
    - Suitable for constrained IoT nodes with only 7.5% overhead.
    - Up to 7.28x speedup achieved, 4x on average.
- DynPolAC guarantees system stability.

Mehdi Karimi; Email: mkarimib@ece.ubc.ca

Download DynPolAC: https://github.com/DependableSystemsLab/DynPolAC