Trust Assessment in 32 KiB of RAM: Multi-application Trust-based Task Offloading for Resource-constrained IoT Nodes

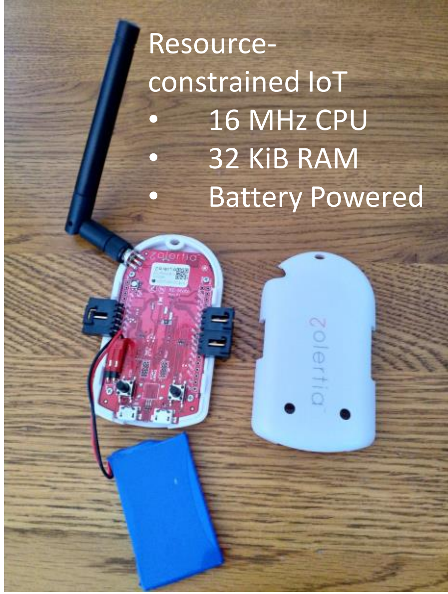
Matthew Bradbury, Arshad Jhumka and Tim Watson

02:10-03:50 and 12:10-13:50 UTC, 22nd March 2021

Dependable, Adaptive, and Secure Distributed Systems Track of the Symposium of Applied Computing

Introduction

- Wireless IoT devices are useful for deployment when physical access to infrastructure is restricted (costly, untrusted, unavailable).
- These devices are constrained (limited CPU, RAM, data storage) to maximise lifetime when battery powered.
- These devices will have expensive tasks that they need to perform.
- As the devices are constrained, expensive tasks can be offloaded to Edge nodes with greater capabilities.
- Which Edge node is chosen for these tasks to offload?



Multi-access Edge Computing (MEC)

- A fair amount of investigation has been done for resource-rich systems (e.g., vehicular/cellular networks)
- The same solutions will not translate to resource-constrained IoT systems
  - Communication
  - Security layer
  - Edge selection approaches

This Talk

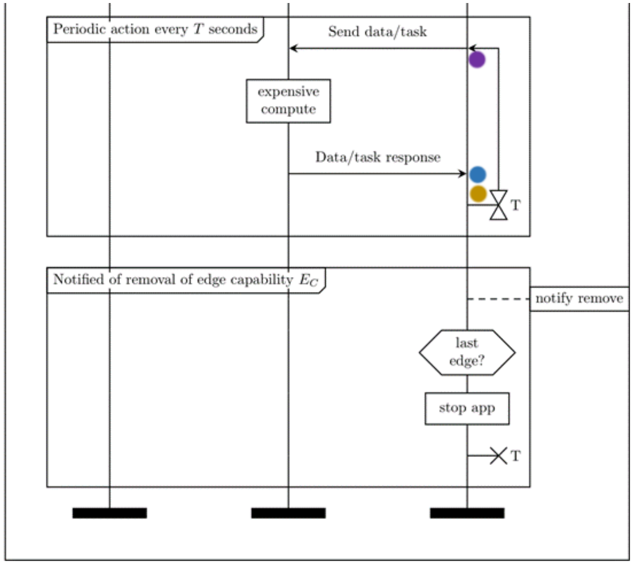
1. Introduce an example trust model
2. Describe the middleware to support trust-based task offloading, including disseminating information required by different trust models
3. Examine results from a deployment, looking at:
    1. Cryptographic operation costs
    2. RAM/Flash usage
    3. Middleware overhead (in terms of bytes sent and received)

Trust-based Task Offloading

- There are several low-memory trust models suitable for use in assessing trust in edge nodes
- BRS – two counters α (number of “good” interactions) and β (number of “bad” interactions), ranking = $\alpha / (\alpha + \beta)$
- The challenge is that in order to know how much memory is available for the trust models, the middleware supporting task offloading needs to be implemented and measured.

Example Trust Model

- Assess trust independently on each IoT node for multiple applications (edge capabilities)
- Aim to answer three questions:
    1. Did an edge acknowledge and accept a task?
    2. Did that edge provide a timely result for the task?
    3. Was the task's result *correct*?
- The trust model cannot store a complete list of all these interactions due to limited memory



Algorithm 1 Update state based on a situation and interaction

▹ a is an application, s is a situation, i is an interaction

function UPDATE(a, s, i)
  for $m \in M(a)$ **do**
    if RELEVANTINTERACTION(a, s, i, m) **then**
      $o \leftarrow f_{a,m}^{\text{opinion}}(s, i)$
      if o = Successful **then**
        $\mathcal{T}_m(e, a).\alpha \leftarrow \mathcal{T}_m(e, a).\alpha + 1$
      else
        $\mathcal{T}_m(e, a).\beta \leftarrow \mathcal{T}_m(e, a).\beta + 1$

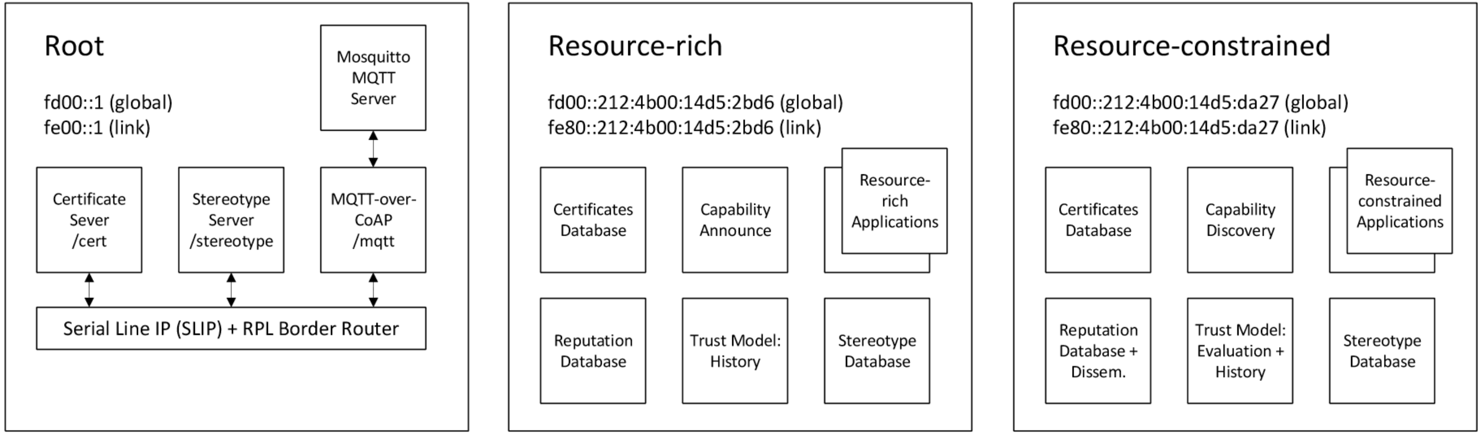
Example Trust Model

- Maintain three Beta distributions:
  - For every edge, did that edge respond that they had accepted a task
  - For every edge, did that edge provide a result for a task
  - For every capability on every edge, was the result returned for that application *correct*.
- Calculate trust by finding the weighted sum of the expected value of these distributions
- By default, start each distribution at (1, 1)
- Allow distributions to be initialised using stereotypes
- Update the distributions when interacting with an Edge

Middleware for Trust-based Task Offloading

Required functionality:

1. Ability to supply digital certificates to IoT devices without them
2. Discovery of capabilities of edge nodes
3. Request Stereotypes of edge nodes
4. Disseminate reputation
5. Application request/response



Message Protection – OSCORE (RFC8613)

- Tasks may contain sensitive information, so messages need to be protected
- Typically, would do so with DTLS, but some recent issues were identified with multiple implementations [1]
- Decided to use OSCORE which provides confidentiality, integrity and authenticity protection for CoAP messages
- Plan for the use of Group OSCORE (draft-ietf-core-oscore-groupcomm-10) for multicasted messages that need non-repudiation
- OSCORE only protects some header fields

No.	Name	E	U
1	If-Match	x	
3	Uri-Host		x
4	ETag	x	
5	If-None-Match	x	
6	Observe	x	x
7	Uri-Port		x
8	Location-Path	x	
9	OSCORE		x
11	Uri-Path	x	
12	Content-Format	x	
14	Max-Age	x	x
15	Uri-Query	x	
17	Accept	x	
20	Location-Query	x	
23	Block2	x	x
27	Block1	x	x
28	Size2	x	x
35	Proxy-Uri		x
39	Proxy-Scheme		x
60	Size1	x	x
258	No-Response	x	x

E = Encrypt and Integrity Protect (Inner)
U = Unprotected (Outer)

Figure 5: Protection of CoAP Options

(From RFC8613)

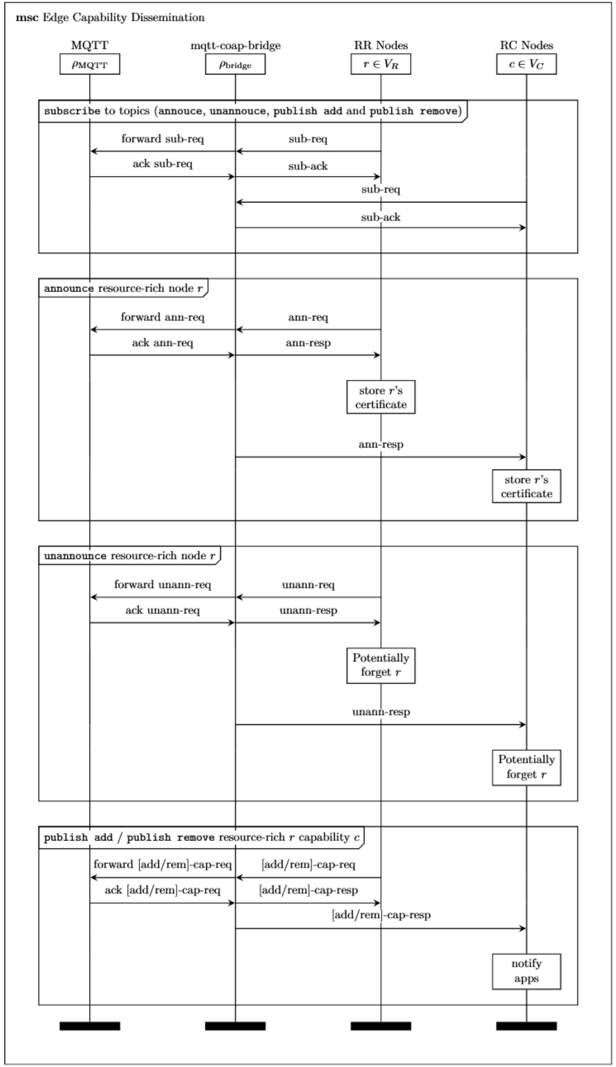
[1] P. Fiterau-Brostean, B. Jonsson, R. Merget, J. de Ruiter, K. Sagonas, and J. Somorovsky. 2020. Analysis of DTLS Implementations Using Protocol State Fuzzing. In 29th USENIX Security Symposium. USENIX Association, Boston, MA, 2523–2540.

PKI - Digital Certificates

- Lightweight EC digital certificate using secp256r1
- Inspired by XIOT certificates in [2]
- X.509 are too large for these systems (ASN.1 less efficient encoding than CBOR)
- These systems may not have a global view of time (due to cost of time synchronisation) = For now, certificates do not expire

```
Certificate = [
    tbscertificate   : TBSCertificate,
    signature        : bytes .size 64
]
TBSCertificate = [
    serial_number    : uint,
    issuer           : bytes .size 8,
    validity         : [notBefore: uint,
                        notAfter: uint],
    subject          : bytes .size 8,
    stereotype_tags  : StereotypeTags,
    public_key       : bytes .size 32
]
StereotypeTags = [
    device_class     : uint
]
```

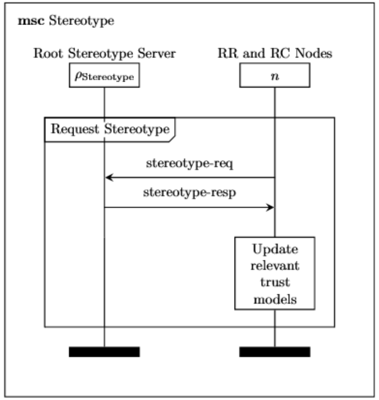
[2] J. Höglund, S. Lindemer, M. Furuhed, and S. Raza. 2020. PKI4IoT: Towards public key infrastructure for the Internet of Things. Computers & Security 89 (2020), 101658. https://doi.org/10.1016/j.cose.2019.101658



Capability Discovery

- Fits well with a publish/subscribe protocol
- IoT devices subscribe to capabilities
- Edge nodes publish capabilities
- MQTT would be a natural fit, but it uses TCP, which required too much RAM
- MQTT-SN uses UDP but is not provided by Contiki-NG
- MQTT-SN would also not be protected with OSCORE
- An MQTT-to-CoAP bridge was implemented

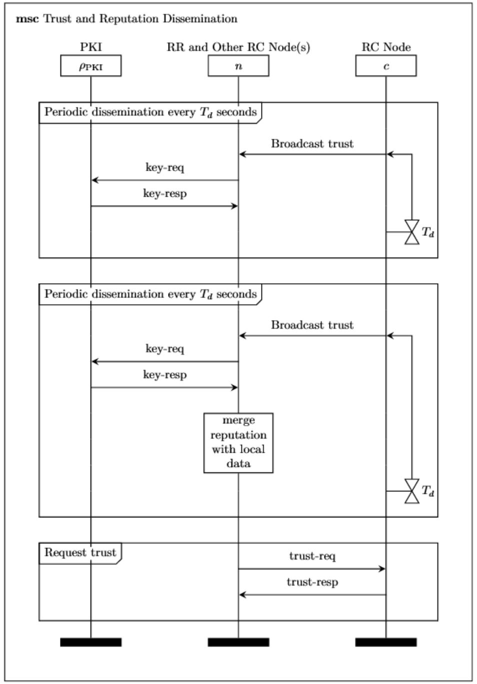
Stereotypes



- Trust models can make use of stereotypes to bootstrap new entrants
- Avoids needing to “take a risk” on an unknown entity
- Assumption: Stereotypes are in the same language as the trust model
- Limitation: The implementation only uses stereotypes to describe an edge, not the application it runs

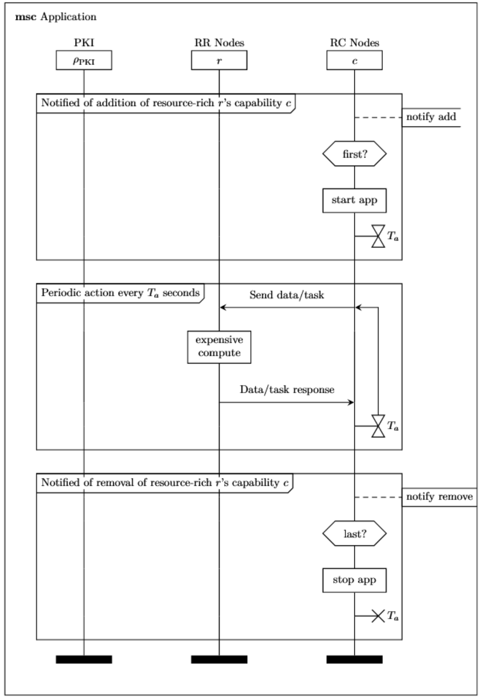
Reputation Dissemination

- Reputation is very useful for trust models
- Needs to provide non-repudiation, cannot allow an IoT device to claim they previously sent a different reputation
- Two modes supported:
  - Periodic dissemination
  - Request current views on an specific edge node



Application

- Example of periodic task submission
- Triggered by edge nodes notifying they have the appropriate capability
- Aperiodic applications also possible
- Task information may need to be private, so confidentiality guarantees from OSCORE are important



Cryptographic Operation Performance

- Elliptic curve signing, verifying and Diffie-Hellman is very expensive
- AES-CCM is much faster, so use it for the majority of communication
- Use ECC for shared secret derivation and to sign reputation dissemination messages
- ECC facilitated by co-processor so CPU can continue working while performing ECC operations

Operation	Mean Cost	Units
SHA256	637 ± 11.6	ns/B
EC Sign (sepc256r1)	360 ± 0.04	ms
EC Verify (sepc256r1)	711 ± 0.03	ms
ECDH	344 ± 0.02	ms
AES-CCM-16-64-128 Encrypt	0.94 ± 0.01	µs/B
AES-CCM-16-64-128 Decrypt	1.01 ± 0.01	µs/B

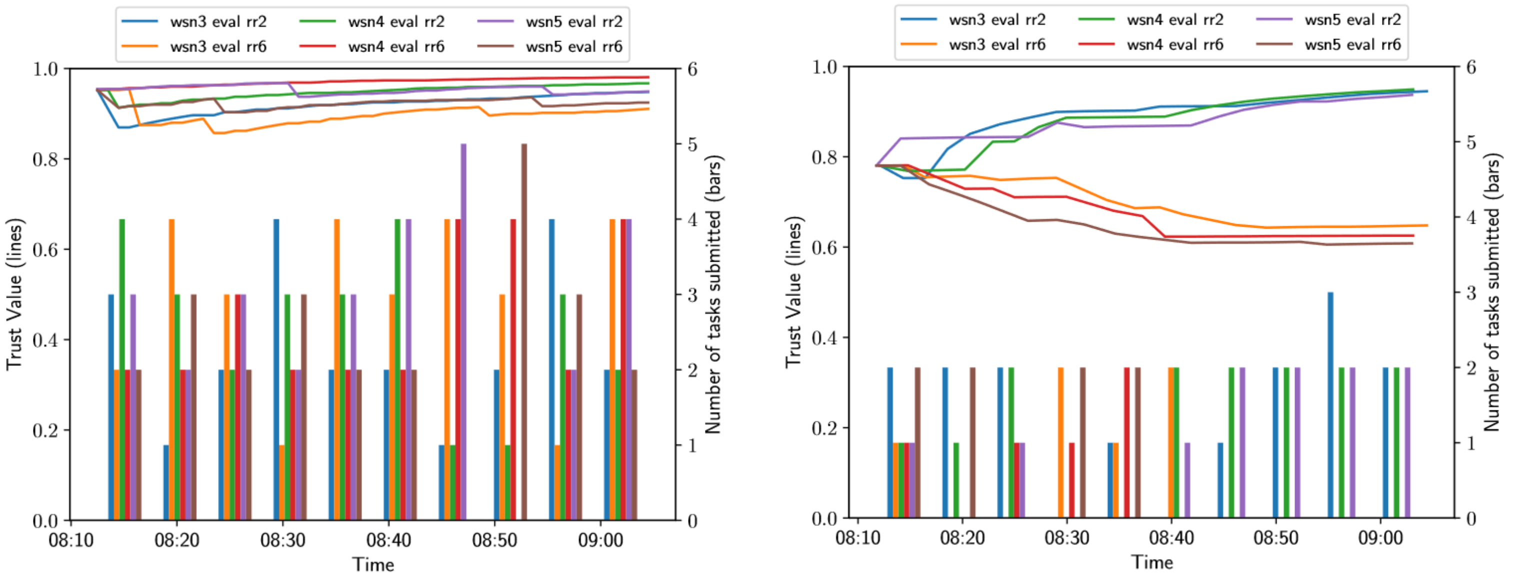
RAM and Flash usage

- Nearing RAM limit of hardware
- Lots of Flash remaining
- Trust model and ECC crypto support are both expensive in terms of RAM
- Will work on optimisations when needed
- Highlights benefits of design decisions (e.g., MQTT-over-CoAP)

	Flash		RAM	
Category	(B)	(%)	(B)	(%)
applications/monitoring	1 388	1.2	353	1.2
applications/routing	3 868	3.3	474	1.6
contiki-ng	7 280	6.2	846	2.9
contiki-ng/cc2538	14 556	12.4	2 356	8.0
contiki-ng/coap	8 556	7.3	2 388	8.1
contiki-ng/net	26 824	22.9	8 232	27.8
contiki-ng/oscore	5 512	4.7	1 010	3.4
newlib	26 415	22.6	2 534	8.6
system/common	3 188	2.7	37	0.1
system/crypto	6 210	5.3	5 173	17.5
system/mqtt-over-coap	1 490	1.3	503	1.7
system/trust	11 846	10.1	5 659	19.1
Total Used	117 133	100	29 565	100
Total Available	524 288		32 768	

Name	Count	Entry (B)	Total Size (B)
Certificates	12	288	3 456
Stereotypes	5	24	120
Edges	4	52	208
Edge Capabilities	12	28	336
Peers	8	32	256
Peer Edges	32	32	1 024
Peer Edge Capabilities	96	16	1 536

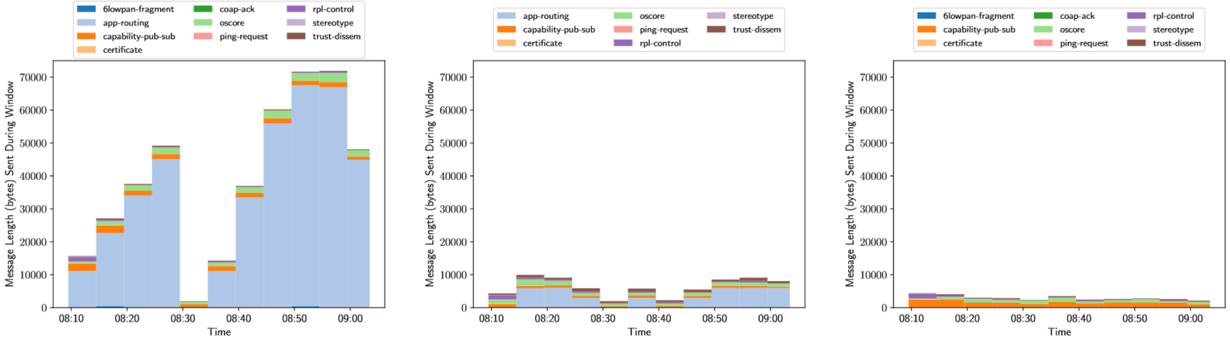
Results



(a) Monitoring (b) Routing

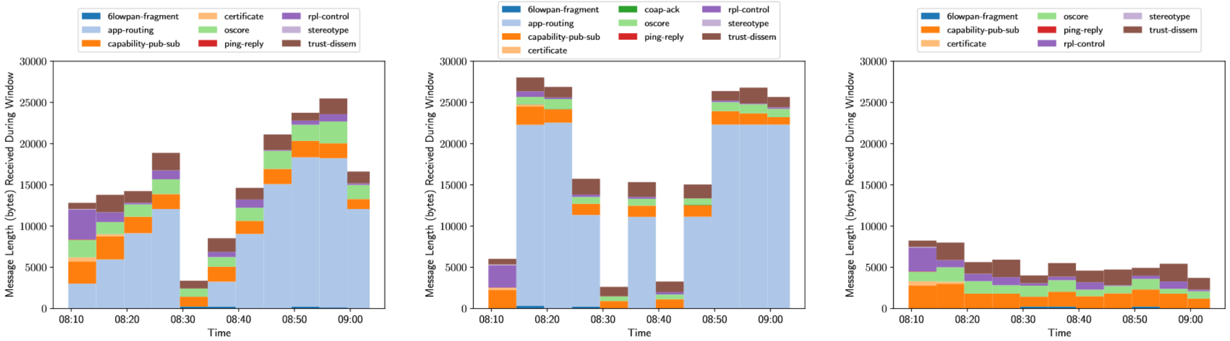
- 1 root node, 2 edge nodes and 3 IoT nodes; two applications
- Both edge nodes always "good" for monitoring, rr6 always bad for routing
- Trust model eventually excludes tasks from being sent to rr6

Results



(a) rr2 (b) wsn3 (c) rr6

Figure 9: Length of messages sent over 5 min windows



(a) rr2 (b) wsn3 (c) rr6

Figure 10: Length of messages received over 5 min windows

- Overhead:
- Trust dissemination: 17% Tx, 5% Rx
- Maximum 50% Tx, 27% Rx
- Challenges with tooling

Conclusions

- A common assumption in the agent-based systems community is that “more information” == “better trust model”
- With these resource constraints it is not feasible to do so
- Trust models need to work within a few KiBs of RAM and will only have limited information from the middleware

For the future:

- Consider providing additional features used by trust models (e.g., witness statements)
- Investigate attacks on the middleware that can impact trust evaluation and which edge is selected for task offloading

Acknowledgement

- This work was supported by the PETRAS National Centre of Excellence for IoT Systems Cybersecurity EPSRC Grant EP/S035362/1.
- https://petras-iot.org
- You can find out more about the project at:
  - https://petras-iot.org/project/evaluating-trustworthiness-of-edge-based-multi-tenanted-iot-devices-team
  - https://mbradbury.github.io/projects/project-6-TEAM

Thank you for listening!