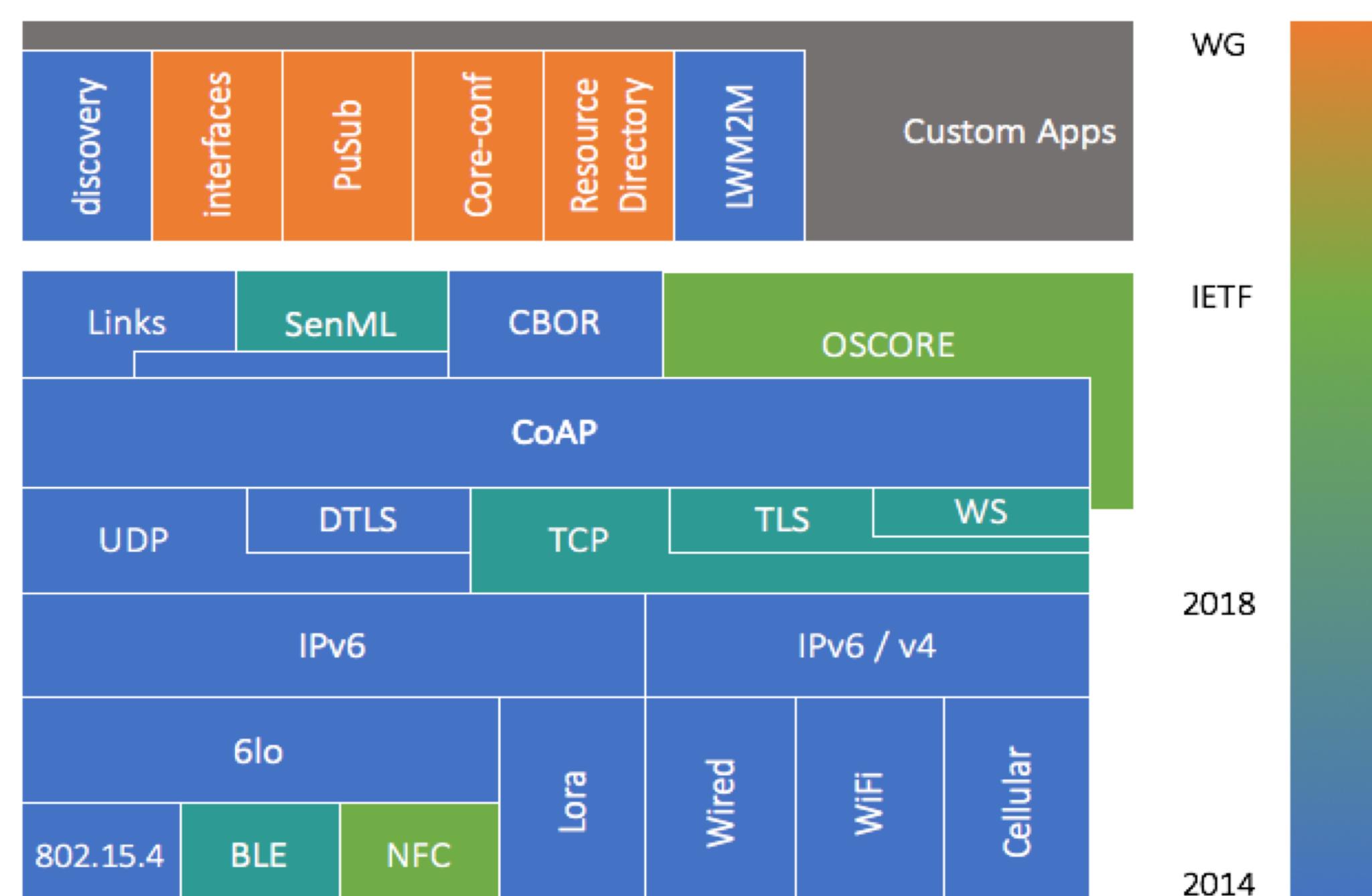


AUTOMATED DISCOVERY OF COAP DEVICES

Concept

Discovering CoAP endpoints is a well-established procedure and much necessary for the real IoT. We analyze all that are available and built a crawler on them.

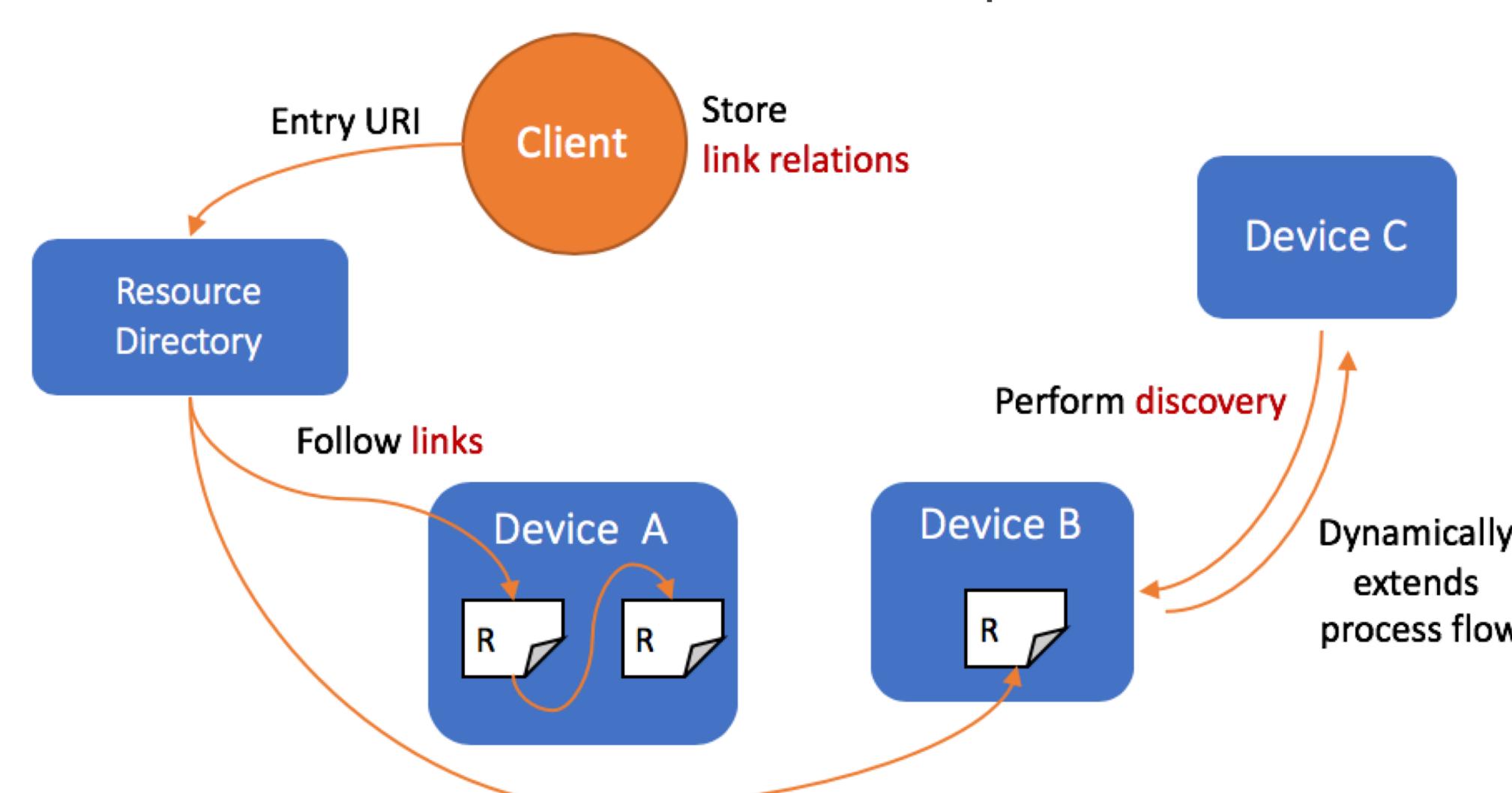
Automated Discovery of CoAP Devices is the outcome of that work.



Background

The **Constrained Application Protocol (CoAP)** has been designed by the IETF as an universal transfer protocol with uniform identifiers for resources, methods for enabling resources interactions, resources discovery mechanisms, and security extensions CoAP endpoints use a typed link specified in "Web Linking" (RFC5988). All endpoints have an URI called "`/well-known/core`" that exposes their resources.

Another service is the **Resource Discovery**, which provides links for the resources hosted by CoAP servers, as well as other metadata such as attributes and possible link relations.



Making the discovery automatic needs the application to be structured meeting the **REST** design principles. The principles from **HATEOAS** allow to create an underlying continuous process of functionalities discovery. Similar to that of a **crawler**, a simple peer would then:

1. Receive requests over `/well-known/core` from devices with pre-configured connections;
2. Answer providing the list of resources hosted;
3. Extract the IP address from the request just received and start a "**Mutual Discovery**" process;
4. Receive list of resources hosted by another device, process them and create new links via the "**hostedby**" relation type.

Evaluation

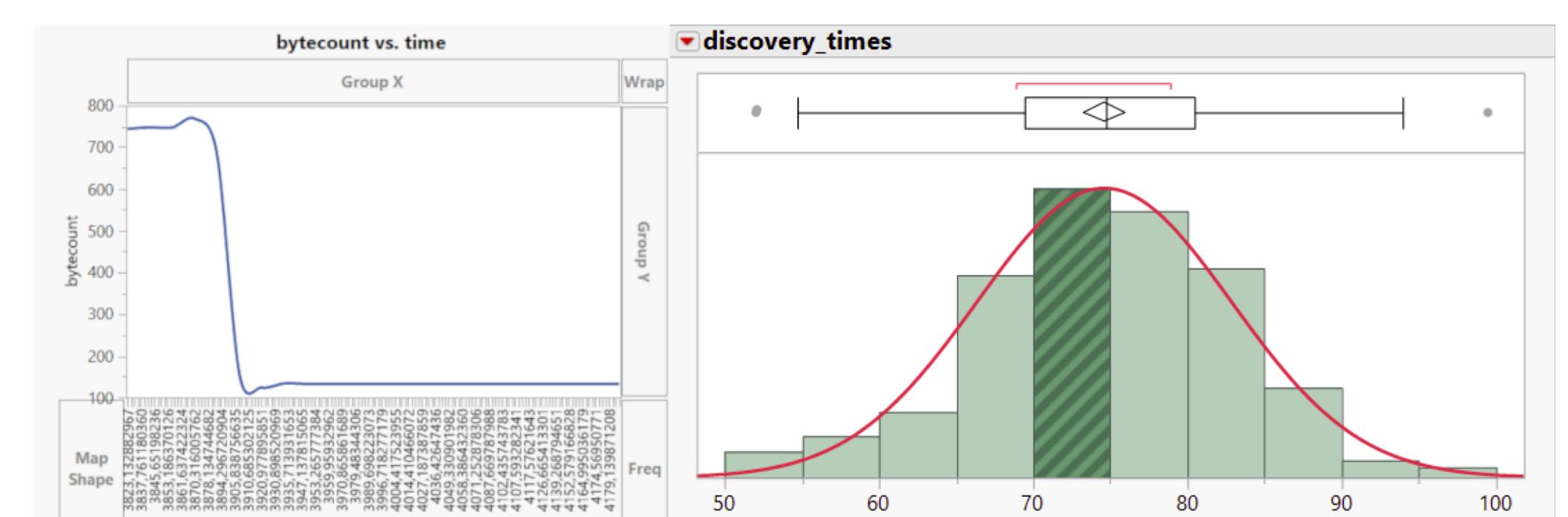
We emulate IoT devices that run CoAP C/S and host few resources. Each of these devices engages one or more discovery mechanisms and builds virtual connections with other nodes. This process is periodically triggered, new devices that are deployed at run-time can discover nodes already present on the network and build relations with them.

End-points perform the discovery using the well-known URI as entry point. The discovery can be performed either via **Unicast** or **Multicast**.

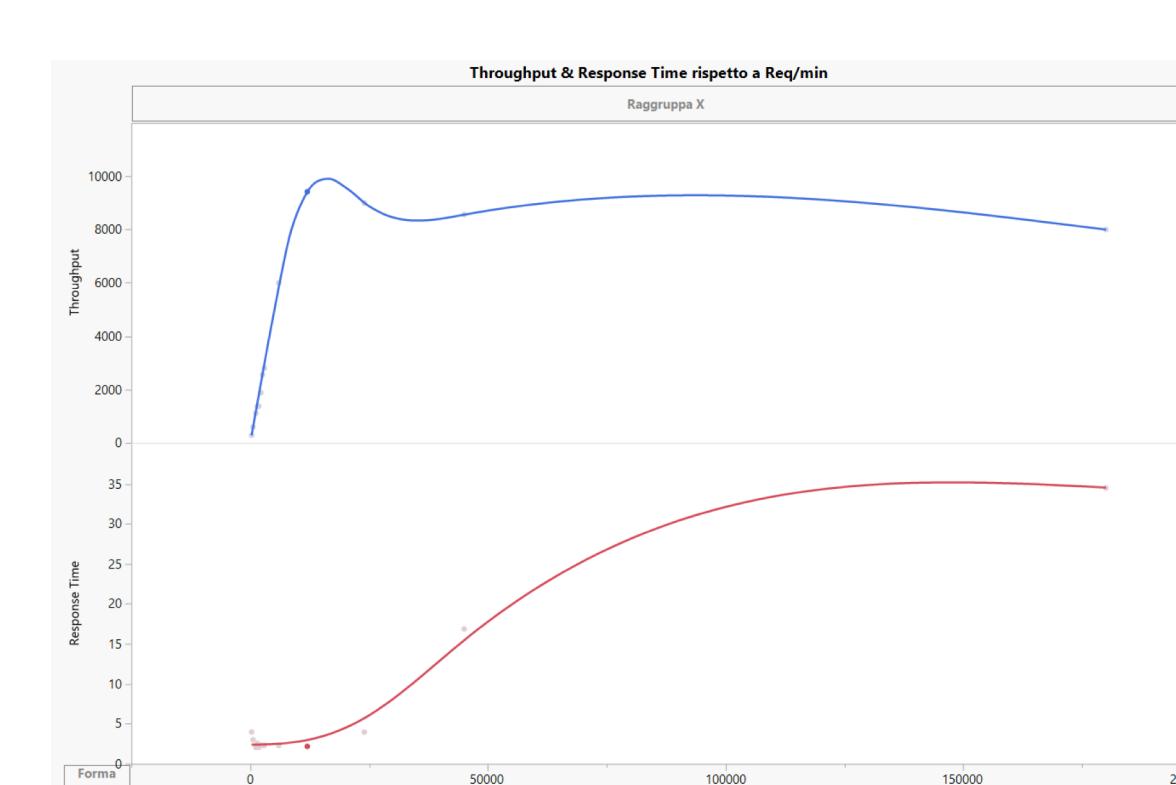
Discoverability		
	RD	Multicast Peer
Unicast Interface	✓	✓
Multicast Interface	✓	✗
Discovery Mechanism		
Multicast CoAP	✗	✓
CoRE RD	✗	✓
Research Across Peers	✗	✓
Pre-built relations	✓	✓
Mutual Discovery	✗	✓

The crawler tries every discovery technique, shows the devices discovered, their resources and **the link paths** that things have built over time.

The testbed is built on **docker**, with each device having its own IP network interface. When using **docker-compose** we will have multicast communication too. Each node will expose port 5683 for UDP connections over CoAP. Resources are simulated. The workload characterization is that of 150 simple peers and a multicast peer.



Bytes exchanged (left) - Distribution of discovery times mean (right). The Multicast Peer takes about 74 seconds to discover 150 devices.



We run 12 experiments, starting from 300 requests/minute and ending with 180000 requests/minute.

Knee Capacity happens between 12000 req/min and 24000 req/min.

Conclusion

This work makes use of common standards and protocols, presenting an autonomous way for devices to discover other endpoints as well as autonomously update their own state based on that. Future work will be invested on the crawling algorithm to make it more elaborated.

References



git clone
<https://git.overleaf.com/17326933pxvtfztcfrv>