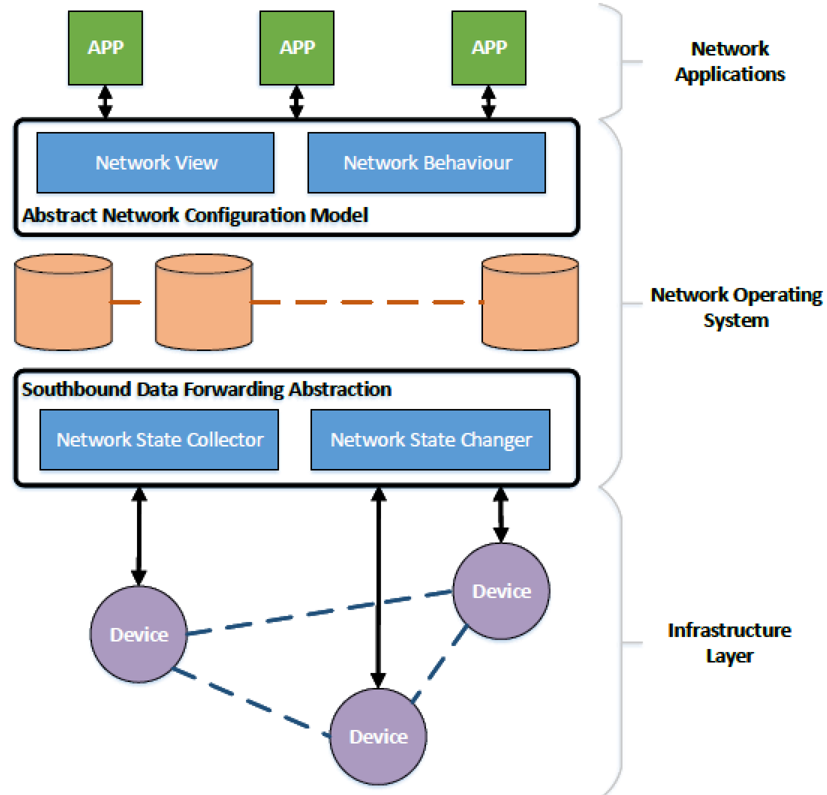
Evolving SDN for Low-Power IoT Networks

Michael Baddeley
PhD Candidate, University of Bristol
Toshiba Research Europe Ltd.

Authors: Michael Baddeley, Reza Nejabati, George Oikonomou, and Dimitra Simeondou at the University of Bristol, and Mahesh Sooriyabandara at Toshiba Research Europe Ltd.

Context: What is SDN?



Compare SDN to the OS on a computer:

- Network Applications => OS Applications.
  - Specify network behaviour.
- Network Operating System => Computer OS.
  - Compiles behaviour to network state.
- Infrastructure Layer => CPU/Mem. instructions.
  - Applies network state to generic devices.

… it provides **Network Programmability**

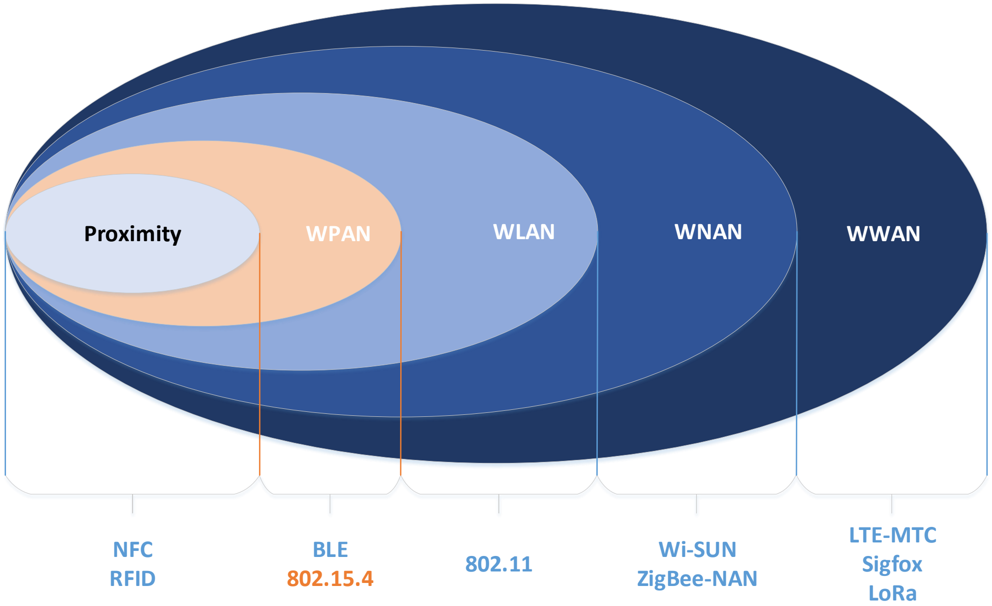
Context: Low-Power IoT (IEEE 802.15.4)

IEEE 802.15.4 forms the basis of many low-power IoT protocols:

- 6LoWPAN, ZigBee, WirelessHART, Thread, ISA100.11a

Low-Power and Lossy Networks:

- Low data-rate (250kbps).
- Extremely low-power (<15mA to TX).
- Multi-hop mesh (10s to 100s of nodes).
- Used for data collection/sensor networks.



Motivation: Why bring them together?

1. **Network (Re) configurability**
   - How do we scale and adapt (extremely) large IoT networks as needs and requirements change?
2. **Global and centralized knowledge**
   - How to we identify issues within the mesh and find optimal solutions to these issues?
3. **New business models and new solutions**
   - How do we slice the network resources to provide and operate a multi-tenant environment?

Challenge: SDN in a Constrained Network

SDN assumes:

- Low-latency controller communication.
- Reliable links.
- Dedicated control channel.
- Large flowtables.
- Real-time network state.

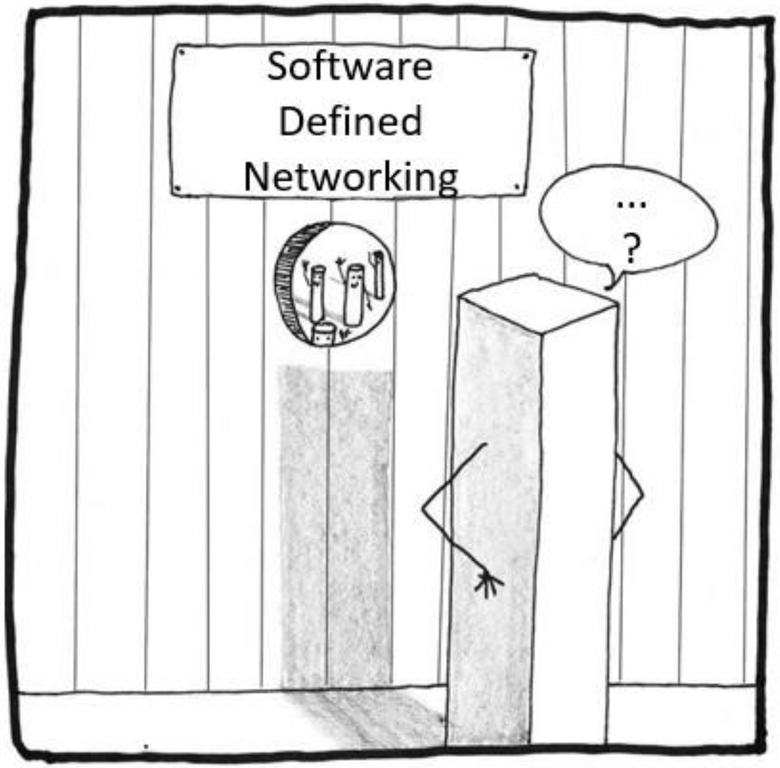
IEEE 802.15.4 offers:

- Constrained Devices
  - Small memory footprint (KB not GB!).
  - Limited energy.
- Constrained Links
  - Wireless, low-power, and lossy.
  - Max frame size of 127B.
- Mesh Topology
  - Motes need to self-organise (dist. Protocols).
  - "Downwards" communication is hard.
  - Mobility + dead branches.

Challenge: Square peg, round hole

Question: How do we apply a high-overhead architecture in an extremely constrained environment over a multi-hop mesh topology?

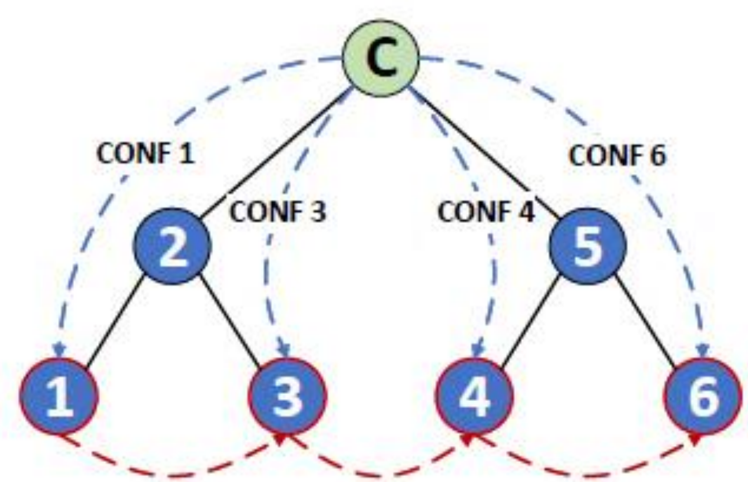
Answer: With difficulty…



Challenge: Maintaining Node/Controller Link

There needs to be a link between the controller and network nodes:

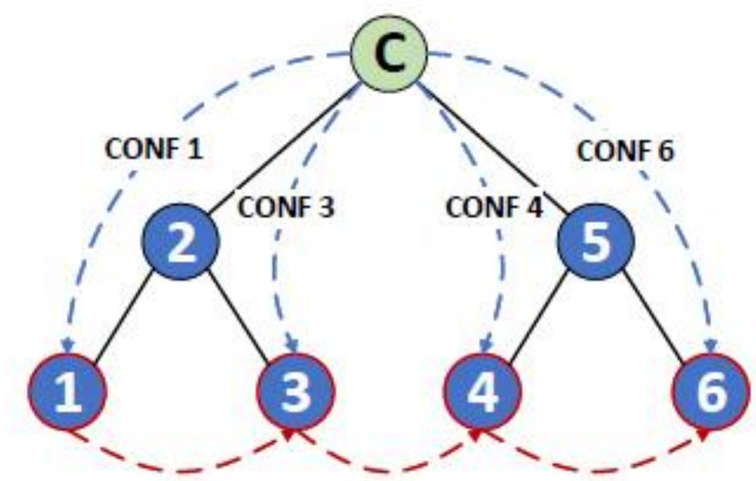
- Routing Protocol for Low Power and Lossy Networks (RPL)
- Self-organising, self-healing.
- Nodes route through their parent.
- Designed for robust *upwards* collection of low-rate sensor data.
- *Downwards* or *point-to-point* communication can be difficult.



Challenge: Maintaining Node/Controller Link

This is an issue for SDN <u>configuration</u> of the network:

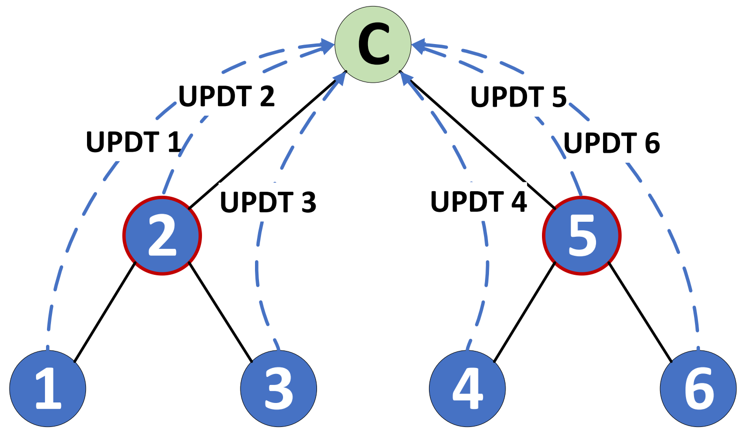
- Messages from the controller to the rest of the network need to navigate *downwards* along the RPL topology, across multiple branches.
- This can result in replication of control messages as the controller tries to configure nodes in the network.



Challenge: Maintaining Node/Controller Link

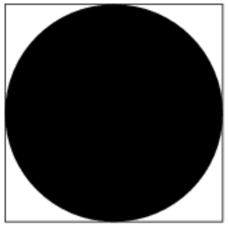
This is an issue for SDN data collection (for network state information):

- SDN data collection for network state can be excessive (depending on application needs)
- Nodes further up the tree need to serve messages from children, exacerbating energy loss.
- Increases contention with other control and application protocols (e.g. RPL control messages: DIS, DIO, DAO).

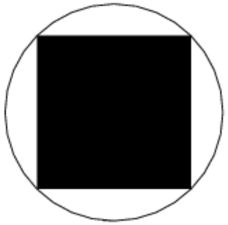


Approach: Get the peg to fit the hole

- Change the peg…



- Change the hole



μSDN: Lightweight SDN for Contiki

Design principles:

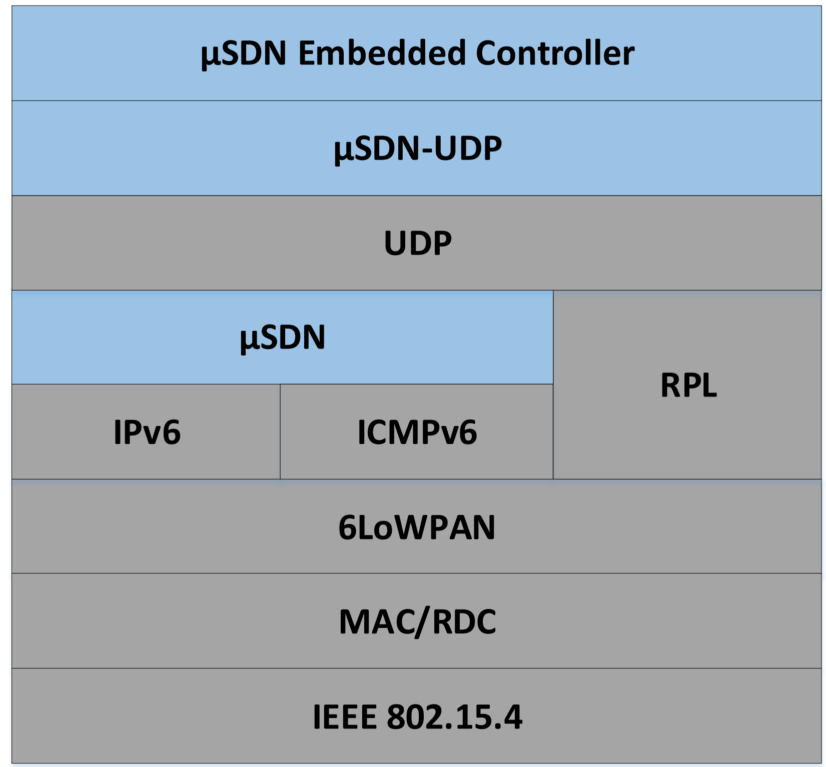
- Minimize memory footprint
- Lightweight control protocol
- Interoperability with existing stack
- Embedded controller at DAG root

Objectives:

- Workable SDN for constrained networks

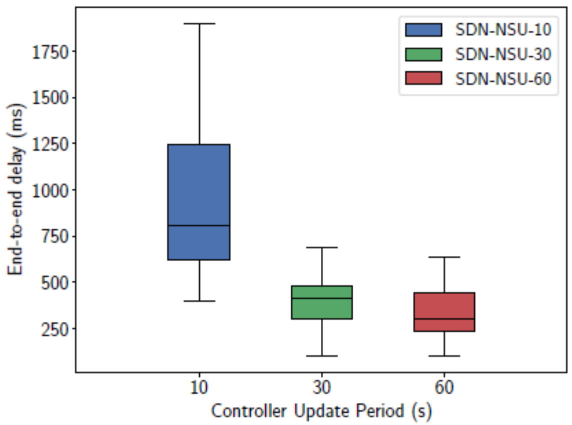
Challenges:

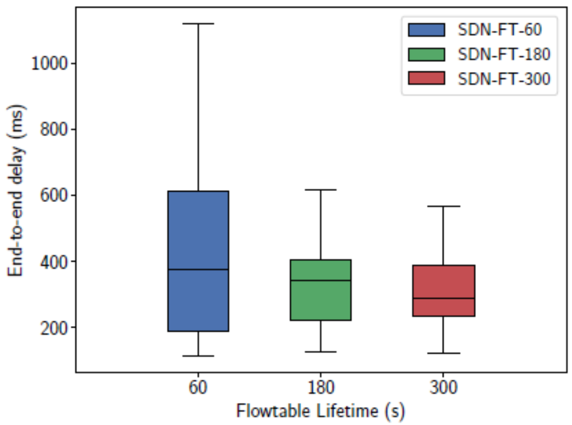
- Reduce the SDN overhead (delay + jitter)
- Reduce flowtable lookups (processing delay)
- Reduce flowtable size (memory limitations)



µSDN: Cost of SDN Overhead

Packet Type	Direction	Behavior	Description
Node State Update (NSU)	UP	Periodic	Updates the controller with node information
Flowtable Query (FTQ)	UP	Intermittent	Requests flowtable instructions from controller
Flowtable Set (FTS)	DOWN	Intermittent	Sets an entry in a node's flowtable
Configuration (CONF)	DOWN	Initial	Configures a node's non-flowtable settings





The rate of NSU (constant bit-rate) and FTQ/FTS (variable bit-rate) traffic patterns can severely affect application-layer flows in terms of end-to-end delay and jitter.

µSDN: Optimize the Stack

Protocol Optimization:

- Eliminate fragmentation
- Reduce packet frequency
- Match on byte array/index

Architectural Optimization:

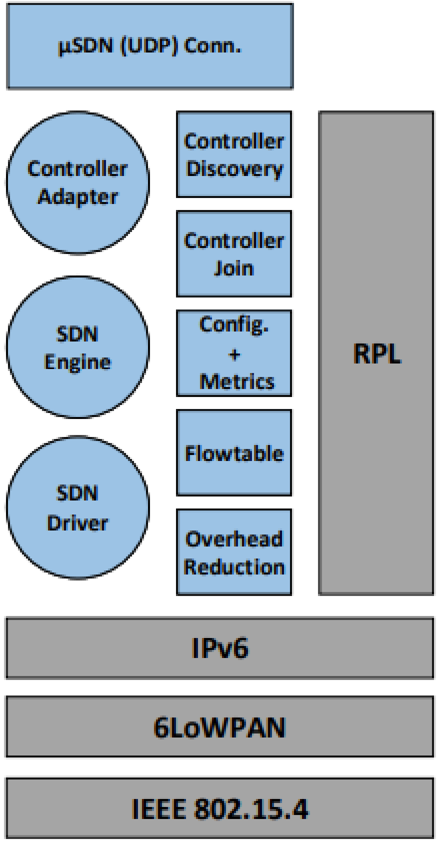
- Use source routing
- Throttle control requests
- Refresh flowtable entries

Memory Optimization:

- Re-use flowtable matches/actions
- Reduce buffer sizes

Controller Optimization:

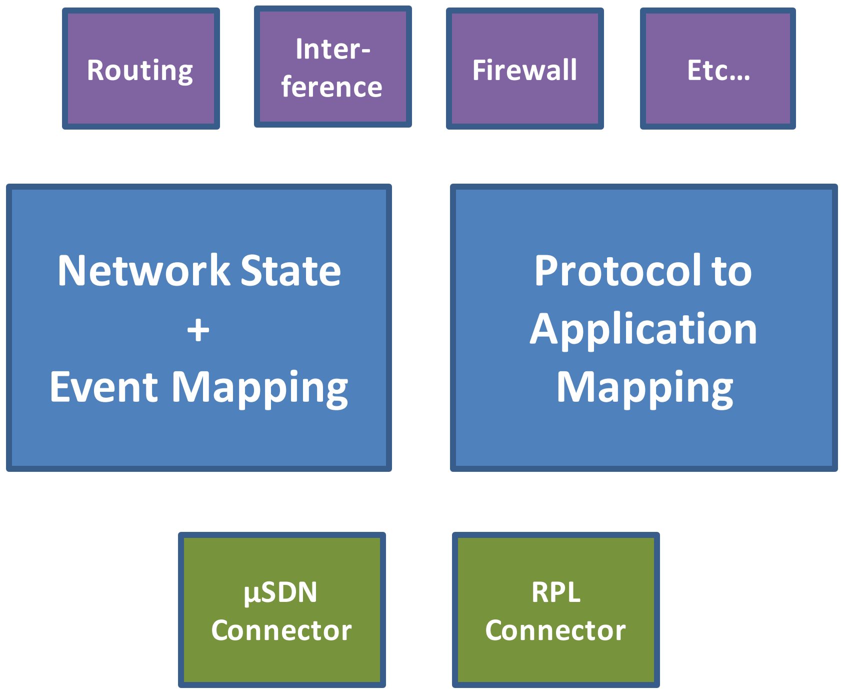
- Reduce controller response times by including an embedded controller within the mesh for simple tasks.



µSDN: Embed the Controller Within the Mesh

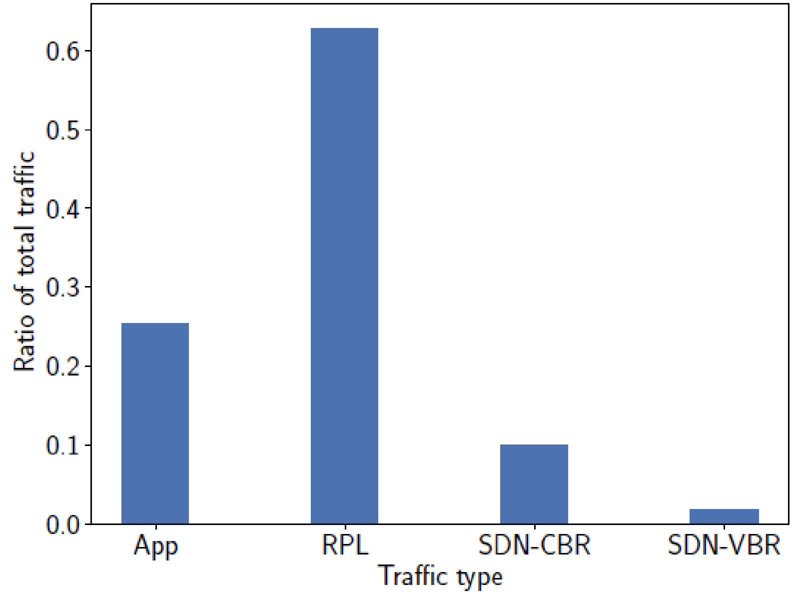
Embedded SDN Controller:

- Implemented in Contiki.
- Application API:
  - Programme network functions.
- Connector API:
  - Multiple southbound protocols.
- Applications can update network state.
- Applications can subscribe to network state.
- Applications can map to protocol connectors.



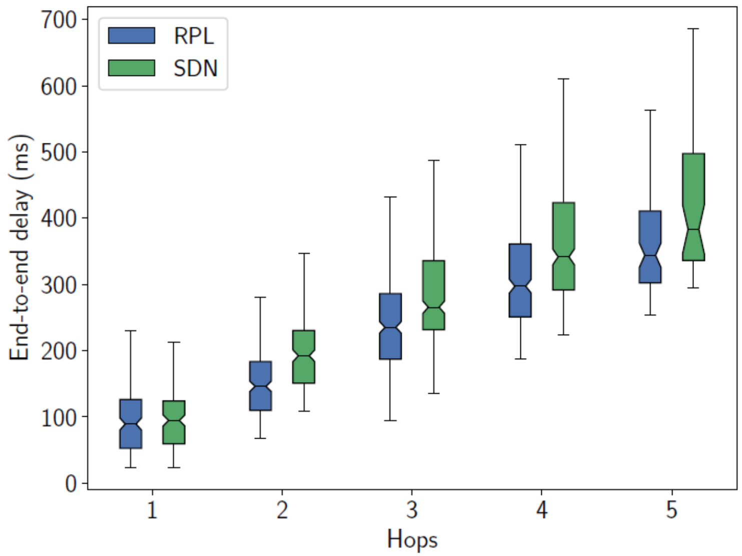
µSDN: Minimal SDN Overhead

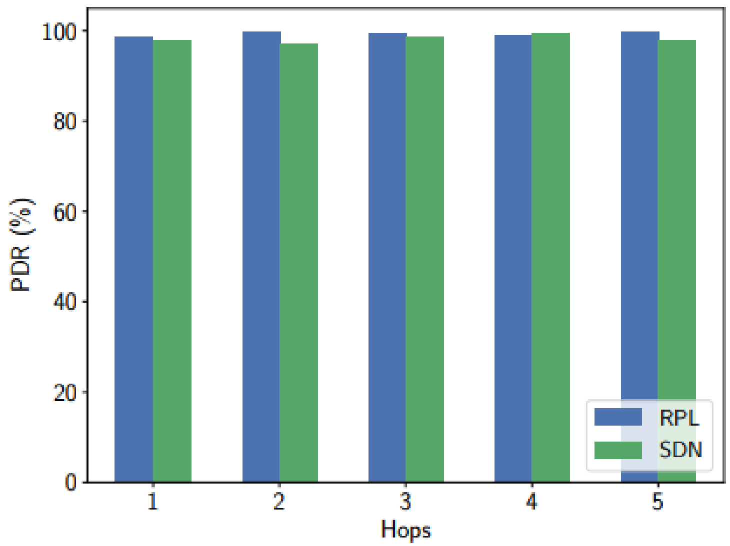
Parameter	Setting
Duration	1h
MAC Layer	ContikiMAC [17]
Transmission Range	100m
Transmitting Nodes	All
Receiving Node	Root/Controller
Network Size	30 Nodes
Packet Send Interval	60 - 75s
Link Quality	90%
Radio Medium	UDGM
RPL Mode	Non-Storing
RPL Route Lifetime	10min
RPL Default Route Lifetime	∞
μSDN Update Period	180s
μSDN Flowtable Lifetime	10min



All evaluation was performed using ContikiMAC (an energy saving MAC layer) on a 30-node network, comparing µSDN against a solely (Non-Storing mode) RPL-based network. In the µSDN network, with traffic reduction techniques, Constant Bit Rate (CBR) overhead (180s) and Variable Bit Rate (VBR) (10min) overhead combined makes up ~13% of the total network traffic.

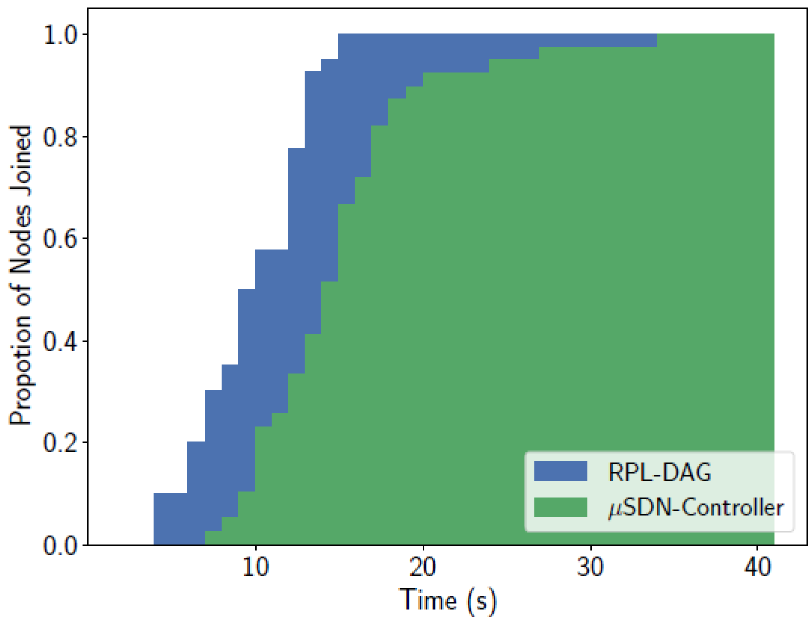
µSDN: Minimal SDN Overhead

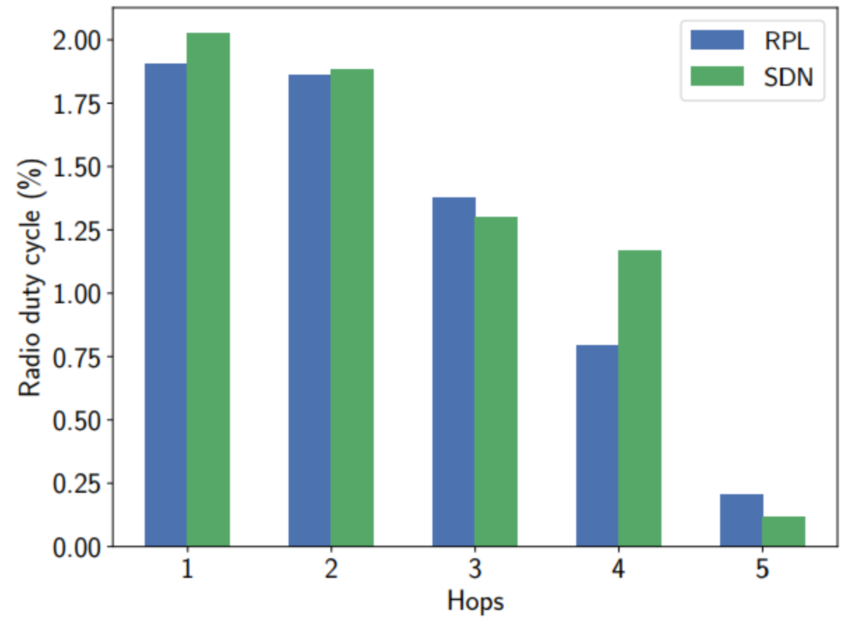




End-to-end delay and Packet Delivery Ratio (PDR) of application flow latency, with a packet sent towards the sink node at a variable rate of 60s – 75s. With optimization of the SDN stack, similar delay and latency is achieved for application traffic, in comparison to a solely RPL-based network.

µSDN: Minimal SDN Overhead





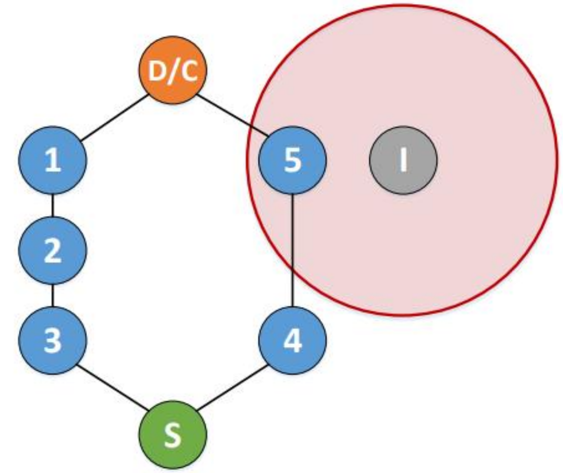
Association time and Radio Duty Cycle (RDC) for a 30-node network. With optimization of the SDN stack, results are similar to a solely RPL-based network.

Use-Case: Reroute flows under interference

Setup:

- Source node *S* sends data from two applications to the DAG Root / SDN Controller at rates of 0.25s and 10s.
- Interference is generated on the same channel as the network every 100ms for a duration of 15ms.
- SDN controller monitors incoming messages and instructs *S* to send *Flow 1* (a critical flow) along a different route if the delivery rate is < *X*.

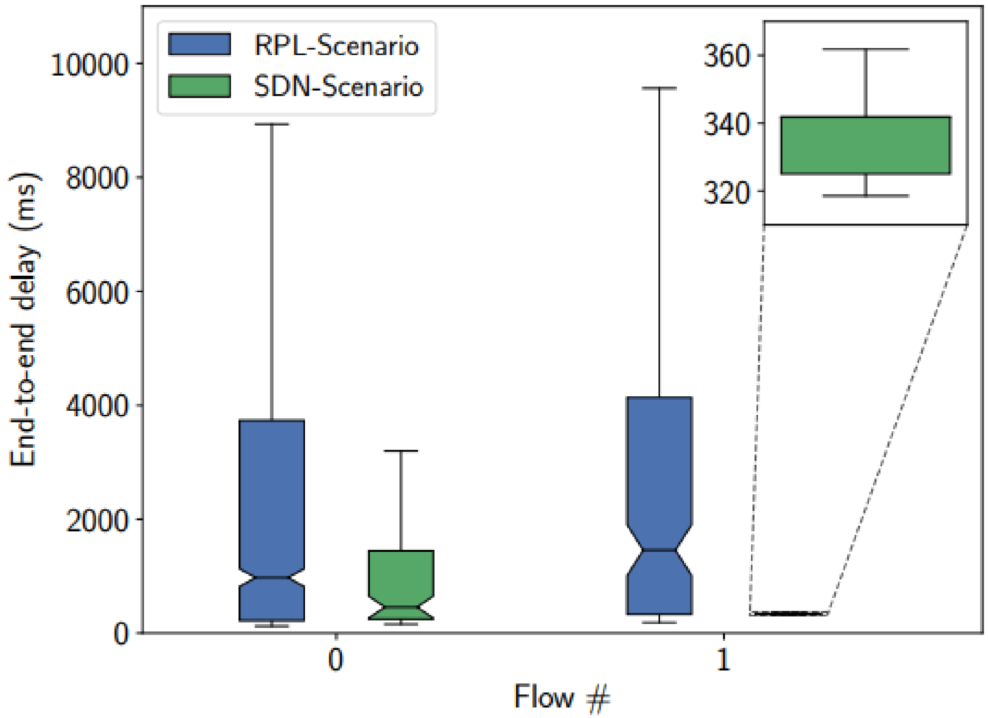
Parameter	Setting
Interference Period	100ms
Interference Duration	15ms
Flow F_0 Bit Rate	0.25s
Flow F_1 Bit Rate	10s



Use-Case: Reroute flows under interference

Results:

- Under RPL, *Flow 0* and *Flow 1* experience severe delay and jitter.
  - Interference is intermittent so RPL cannot self-heal.
- Under SDN, *Flow 0* and *Flow 1* are no longer in contention.
  - *Flow 0* continues to experience some interference.
  - *Flow 1* is rerouted and is no longer subject to interference.



Conclusions

You can provide programmable low-power IoT with minimal SDN overhead:

- Optimize the SDN stack.
- Eliminate control message fragmentation.
- Eliminate unnecessary transmissions.
- Use source-routing on control messages.
- Embed the controller.
- µSDN codebase will be publicly available **soon**!

Time Scheduled Channel Hopping (TSCH) based networks:

- SDN concepts are a a big part of 6TiSCH (IPv6 over IEEE 802.15.4-2015 TSCH).

Larger Networks:

- How do we move from 100s -> 1000s of nodes?

Node/Controller communication is essential, but RPL overhead is excessive:

- Are there other ways to provide this link but retain robustness/mobility?

Questions?