The Network as a Computer with IPv6 Segment Routing: a Novel Distributed Processing Model for the Internet of Things

Andrea Mayer[1], Emanuele Altomare[1], Stefano Salsano[1],
Francesco Lo Presti[1], Clarence Filsfils[2]

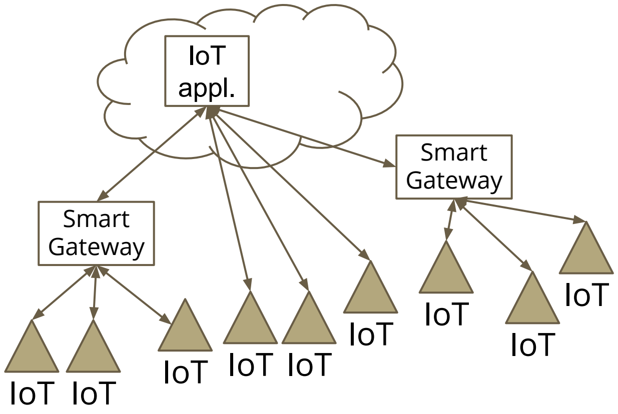
(1) University of Rome Tor Vergata, (2) Cisco Systems

1st International Workshop on Next-Generation Operating Systems for Cyber-Physical Systems (NGOSCPS)
April 15, 2019 at CPS-IoT Week, Montreal, Canada

stefano.salsano@uniroma2.it

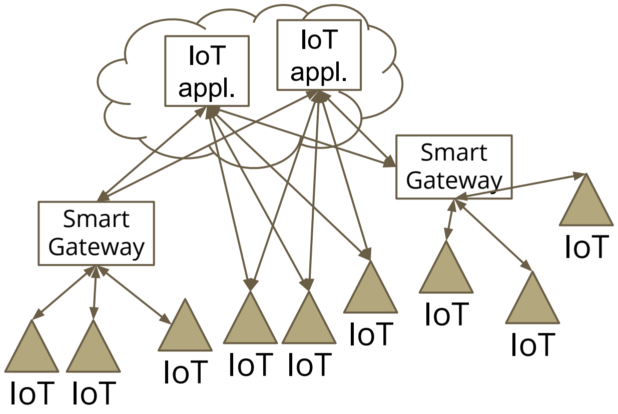
IoT applications

- Current cloud based IoT applications are based on a “star” model: the cloud applications interacts with all the “things”, typically with the help of smart gateways and using publish subscribe



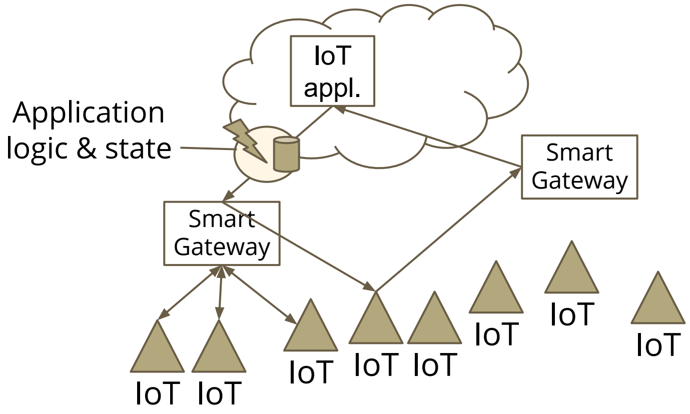
IoT applications

- Current cloud based IoT applications are based on a “star” model: the cloud applications interacts with all the “things”, typically with the help of smart gateways and using publish subscribe
- Multiple IoT application may need to interact with the “things”:
a lot of messages - complex coordination - need to sync the execution state and the versions of the applications



New processing model

- Assuming that it is possible to exploit distributed processing in the “things/gateways”, can we envisage a different model?
- The application logic and the state is transferred “on the fly” with IP packets

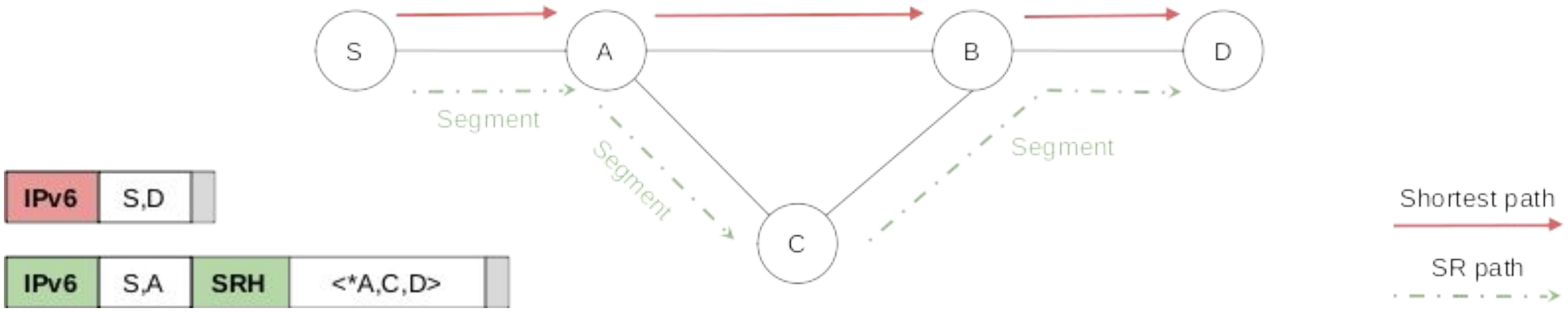


New processing model

- The control program is not decomposed but it “travels” among **computing units** (of course this is possible for “small” processing tasks!)
- All nodes that are capable of computing can be part of the computation
- The packets arrive to the computing unit in the order specified by the program itself
- The network layer (IPv6 with Segment Routing) provides support to this model

IPv6 Segment Routing (SRv6)

- Segment Routing is a kind of "source routing";
  - a sequence of "waypoints" can be specified in the packet header
- Segment Routing for IPv6 includes the sequence of IPv6 addresses of nodes to be visited in a "segment list" in the Segment Routing Header (SRH)
- Under IETF standardization, several implementations are already available (including in Linux Kernel)



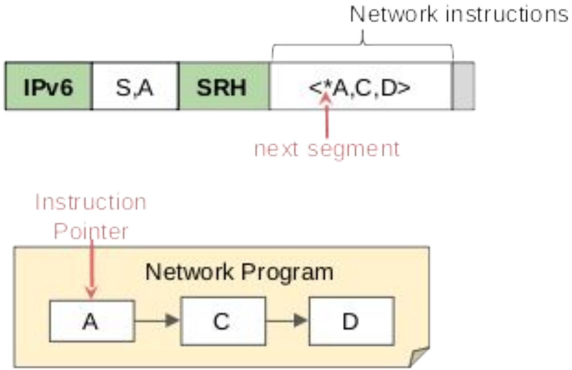
IPv6 Segment Routing (SRv6) and the "Network Programming" model

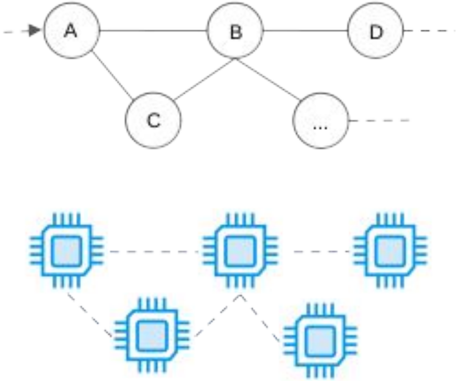
- The IPv6 Segments can represent network instructions to be executed at a given node (this is referred to as the "Network Programming" model)
- Examples of network instructions: encapsulate, decapsulate, forward on a specific interface, apply a firewall ACL
- The 128-bits IPv6 addresses are split in a **Locator** part used to route the packet to the node and a Function part that identifies the function to be executed at the node

128 bits (16 bytes) IPv6 address

IPv6 Segment Routing (SRv6) and the “Network Programming” model

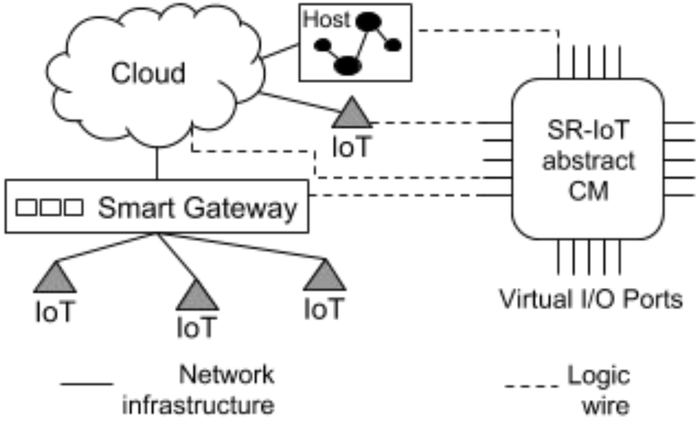
- ❖ The Segment List can be seen as a “Network program”, where the next segment is the Instruction Pointer and a network node is a CPU that executes the instruction





Proposed SR-IoT processing model

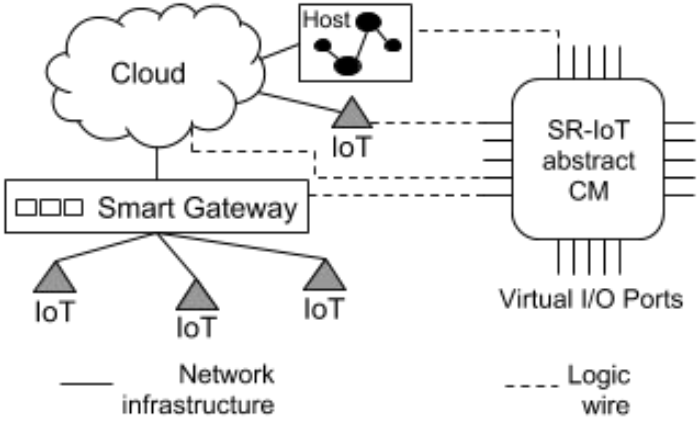
- With the proposed SR-IoT, we extend the Segment Routing network programming model, considering the Functions as “operation codes” of a processor ISA (Instruction Set Architecture)
- The whole IoT infrastructure is seen as a logical machine with I/O ports (corresponding to the ports of IoT devices), that can be programmed through an Instruction Set Architecture



Proposed SR-IoT processing model

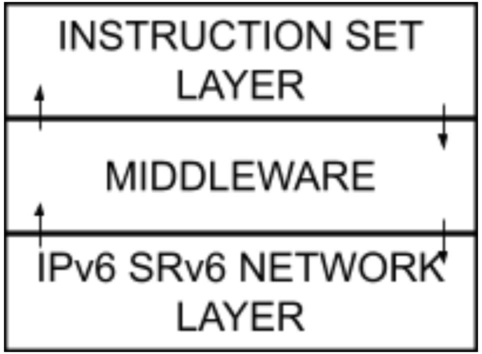
- The developer can write a processing task consisting of reading/writing I/O ports and making arbitrary computations (taking decisions).

- These operations are mapped to physical computing units and the compiled processing task is represented into an SRv6 packet.



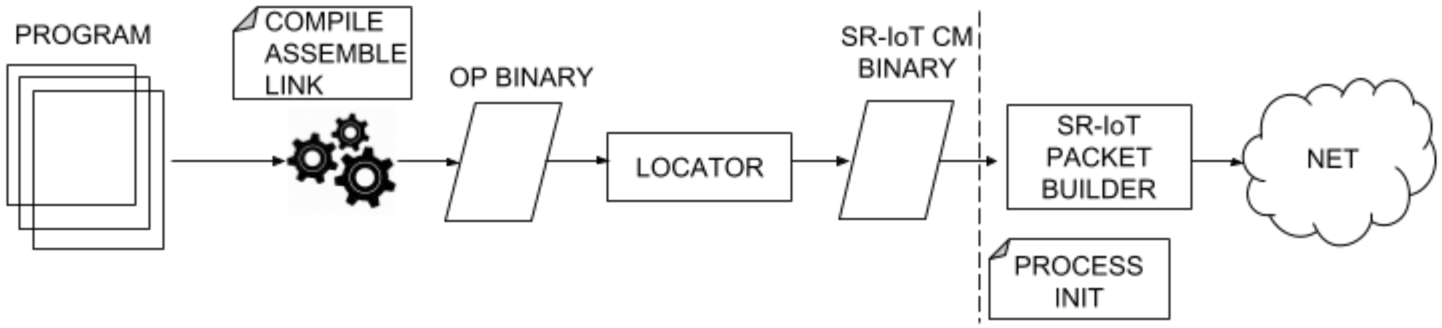
Prototype of SR-IoT (work in progress)

- Instruction Set Architecture (ISA) of Atmel AVR microcontroller (Arduino)
- We use the SimAVR emulator on Linux to emulate the AVR microcontroller
- We are building a middleware that coordinates the execution of process tasks providing the mapping into the location of computing units and the serialization/de-serialization of processes into SRv6 packets



Prototype of SR-IoT (work in progress)

- We are extending the AVR Toolchain that includes the tools needed to the developer to compile a program (on a single device), to interact with our middleware in a distributed IoT scenario
  - Open source tools: avr-gcc, avr-gdb, avr-objdump,

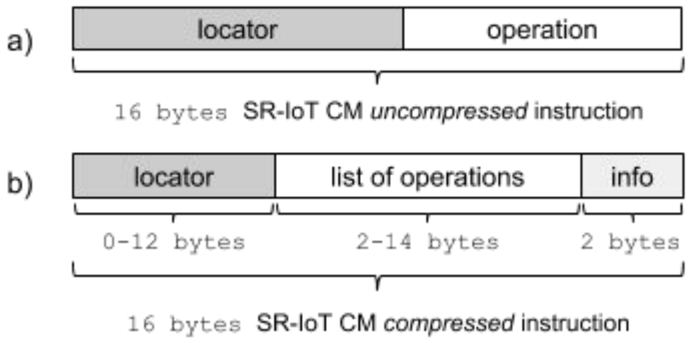


Space constraints

- An SR-IoT packet corresponds to a process
- It includes the program in the SRv6 segment list, and the serialization of registers, stack and RAM
- All need to fit in less than 1500 bytes ! (Assuming 1500 MTU)

Space constraints

- All need to fit in less than 1500 bytes ! (Assuming 1500 MTU)
- For TinyAVR microcontrollers, RAM used can be as low as 128 or 256 bytes
- Using an IPv6 address (16 bytes) to represent the location and the operation, for each operation is a waste. We designed more efficient solution to encode operations in the IPv6 segment list. For example, 100 instructions over 10 different nodes can be represented with 320 bytes.



Discussion

- A program of 100 instructions can be represented with 320 bytes, leaving 1000 bytes for RAM, stack and CPU registers… what can we do with these processes?
- Example: reading temperatures from N sensors, evaluating an average, then setting actuators in M sensors, all with a self-contained packet/process
- How to cope with packet loss? Retransmission, but tasks need to be idempotent
- The middleware with packet serialization / de-serialization can be easily implemented in smart gateways. Is it worth implementing this model also in “tiny” IoT devices?