From Reactive to Predictive Flow Instantiation: An Artificial Neural Network Approach to the SD-IoT

Sebastiano Milardo\textsuperscript{1}, Akhilesh Venkatasubramanian\textsuperscript{2}, Krithika Vijayan\textsuperscript{2}, Prabagarane Nagaradjane\textsuperscript{2}, Giacomo Morabito\textsuperscript{3}

1) University of Palermo, Palermo, Italy
2) Sri Sivasubramaniya Nadar College of Engineering, Chennai, India
3) University of Catania, Catania, Italy
Problem Statement
Problem Statement
Problem Statement

- IoT devices and in particular wireless sensor nodes, aim at achieving high efficiency with low power consumption.

- Is it possible to exploit tidal effects to increase the lifetime of a Wireless Sensor Network?

- Is there a simple way to implement such solution?
Our goal:

“We want to make life easier for (network) developers”
SDN

- Control Plane: Forwarding Control logic
  - E.g. Routing protocols

- Data Plane: Forwards the packets/flows according to the specific rules imposed by the control plane
  - IP forwarding, lvl 2 Switching

Network Programmability
# All the packets towards 192.168.101.101:80 should be sent on port 4

```python
msg = of.ofp_flow_mod()
msg.match.nw_dst = IPAddr("192.168.101.101")
msg.match.tp_dst = 80
msg.actions.append(of.ofp_action_output(port = 4))
self.connection.send(msg)
```

Problem solved (?)
SDN: Towards Intelligent Control Planes

- Declarative Programming Languages

```sql
SELECT * from dogs
INNER JOIN owners
WHERE dogs.owner_id = owners.id
```

- Imperative Programming Languages

```javascript
//dogs = [{name: 'Fido', owner_id: 1}, {...}, ... ]
//owners = [{id: 1, name: 'Bob'}, {...}, ... ]

var dogsWithOwners = []
var dog, owner

for(var di=0; di < dogs.length; di++) {
    dog = dogs[di]

    for(var oi=0; oi < owners.length; oi++) {
        owner = owners[oi]
        if (owner && dog.owner_id == owner.id) {
            dogsWithOwners.push({
                dog: dog,
                owner: owner
            })
        }
    }
}
```
SDN: NOS

Northbound - Application Intent Framework
(policy enforcement, conflict resolution)

Distributed Core
 scalability, availability, performance, persistence

Southbound
(discover, observe, program, configure)

OpenFlow
(pluggable, extensible)

More API in future release

Packet-
Optical integ. Host Mobility Proxy ARP Reactive L2 fwding Apps Intent-based L2 fwding Intent calendaring SDN-IP peering
SDN: NOS - Intent
• Ask the NOS to implement a certain property in the controlled network
• Let the NOS install all the required rules to achieve such goal
Proposed Solution

- http://sdn- wise.dieei.unict.it
Proposed Architecture – Predictive SD-IoT

- Performance Specification module (PSM)
- Measurement module (MM)
- Prediction module (PM)
Proposed Architecture – Predictive SD-IoT

- Performance Specification module: it accepts the requirements from the user and translates such requirements into an objective function (e.g. fairness).

- Measurement module: it is based on the ONOS REST APIs which are used to collect the amount of traffic traversing each link of the network.

- Prediction module: it includes the LSTM-ANNs used for predicting network patterns.
We used Long Short-Term Memory ANN in the prediction module as it is regarded as the State of the Art for time series prediction.

In our case we used LSTM-ANNs with 3 layers: 4 neurons in the input layer (one for each variable considered: day of the week, hour of the day, holiday, no. of generated packets) 50 neurons in the hidden layer, and one neuron in the output layer.
Routing Strategy

\[ w'(x, y) = a \cdot w(x, y) + (1 - a) \cdot p(y) \]

- where \( w \) is the weight of the edge between nodes \( x \) and \( y \),
- \( p(y) \) is the amount of packets sent by the node \( y \), as predicted by the LSTM-ANN,
- \( a \) is the tuning parameter imposed by the performance specification module based on the user’s preferences.
Testbed

- 309 wireless sensor nodes
- 37 wireless relay nodes
- 3 gateways

**European Wireless 2018**

- 1,580,807 messages
- from January 1, 2016 to December 31, 2016
Simulations

- Onos
- Mininet
- Cooja
Predictive Flow Instantiation

Algorithm 1: Prediction Algorithm

```
traffic = []
topo = getNetworkTopology()
weightedTopo = setWeights(a=1, b=0, topo)
pThreshold = getThreshold()
while (1) do
    currentTraffic = getTrafficData()
    traffic.append(currentTraffic)
prediction = predictTraffic(traffic)
pCurrent = getPacketsToBeSent(weightedTopo)
newWeightedTopo = setWeights(a, b, topo, traffic)
pPredicted = getPacketsToBeSent(newWeightedTopo)
pRules = getUpdateCost(weightedTopo)
    if (pCurrent > pPredicted + pThreshold + pRules) then
        paths = Dijkstra(newWeightedTopology)
        updateFlowRules(paths)
        weightedTopo = newWeightedTopo
    end if
end while
```

European Wireless 2018
Results
Results

European Wireless 2018
Results

European Wireless 2018
Results

European Wireless 2018
Conclusions

- We have presented a general architecture for an SD-IoT management system based on a LSTM-ANN. We tested our approach on a real dataset inside a simulated environment.

- The proposed solution aims at providing the starting point for a wider declarative, SDN-based, predictive flow rule instantiation system.
Thanks