IOX
INTERNET OF EVERYTHING INTERCONNECTION ENVIRONMENT
ANTONIO CISTERNINO (CISTERNI@ITC.UNIPI.IT, TWITTER: @CISTERNI)
UNIVERSITY OF PISA
Key Pillars to an OpenFog Architecture

Security:
- Trust
- Attestation
- Privacy

Scalability:
- Localized cmd, ctrl, & processing
- Orchestration & Analytics
- Avoidance of network taxes

Open:
- Resource visibility & control
- White box decision making
- Interop & Data normalization

Autonomy:
- Flexible
- Cognition & agility
- Value of data

RAS:
- Reliability
- Availability
- Serviceability

Programmability:
- Programmable SW/HW
- Virtualization & multi-tenant
- App Fluidity

Hierarchy:
- Fully cloud enabled
- Computational & System
- Autonomy at all levels

Agility:
- Tactical & strategic decision making
- Data to wisdom
FOG NODES = SEMANTIC ROUTER
### MOTIVATION AND GOALS

**IoT**
- interconnecting small devices and sensors with Internet technologies

**Assumption**
- http(s) + REST + JSON will drive most of the message passing for IoT

**Need**
- Control routing of JSON messages

**Goal**
- Define a (IoT) vendor independent runtime for processing JSON messages on whitebox IoT gateways and network switches
ARCHITECTURE

• IoX is responsible for:
  • Receiving JSON msgs
  • Vendor modules take action:
    • Act on the device
    • Fwd message
    • Send message to IoT devices

• Nodes can offer light or heavy computing capabilities (IoT devices, IoT gateways and network switches vs. PCs and servers)
IOX ARCHITECTURE

• Runtime
  • .NET + F# + Suave
  • CLI with F# interactive
  • Event based processing with evReact

• Vendor modules register for http prefixes

• Module isolated in process using .NET core capabilities

• inject message processors in the processing runtime

• User can express global rules for inspecting JSON messages

• Modules can offer HTML5 UIs

• Action may include Interop with host enviroment (i.e. network switch)
module example

type HelloWorldModule() =
  inherit Module("hw", "Example module")

override this.OnLoad() =
  let hello = this.RegisterEvent("/hw/hello")
  let chat = this.RegisterEvent("/hw/chat")
  let bye = this.RegisterEvent("/hw/bye")

  let net =
    +( (!(hello |-> fun arg -> arg.Result <- OK "Hello dear")
        -
        +( !(chat |-> fun arg ->
            let msg = match arg.Context.request.query.[0] with "msg", Some m -> m | _ -> ""
            arg.Result <- OK (sprintf "I disagree on %s" msg)
          ) / ![bye] )
          -
          ( !(bye |-> fun arg -> arg.Result <- OK "Bye bye")
        )
    )

  this.ActivateNet(net) |> ignore
A WORKING DEMO

- Control the Relay to open the door
AGGREGATE PROGRAMMING ON IOX
CONCLUSIONS

• We believe that HTTP/REST/JSON message routing will be central to IoT
• IoX aims at simplifying implementing the message processing infrastructure
• We are investigating how to simplify the implementation of Fog nodes
• Repo is available on GitHub
• Follow #IoX on Twitter and on unipi-itc/iox on GitHub