IOX
ROUTING MESSAGES THROUGH THE FOG

ANTONIO CISTERNINO (CISTERNI@ITC.UNIPI.IT, TWITTER: @CISTERNI)
IT CENTER, UNIVERSITY OF PISA
FOG AND IOT

Public cloud

Private cloud

GW
### MOTIVATION AND GOALS

<table>
<thead>
<tr>
<th><strong>IoT</strong></th>
<th>• interconnecting small devices and sensors with Internet technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ASSUMPTION</strong></td>
<td>• http(s) + REST + JSON will drive most of the message passing for IoT</td>
</tr>
<tr>
<td><strong>NEED</strong></td>
<td>• Control routing of JSON messages</td>
</tr>
<tr>
<td><strong>GOAL</strong></td>
<td>• Define a (IoT) vendor independent runtime for processing JSON messages on whitebox IoT gateways and network switches</td>
</tr>
</tbody>
</table>
**ARCHITECTURE**

- **IoX is responsible for:**
  - Receiving JSON msgs
  - Vendor modules take action:
    - Fwd message
    - Send message to IoT devices
- **Nodes can offer light or heavy computing capabilities (IoT gateways and network switches vs. PCs and servers)**
• 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)
open Suave
open Suave.Filters
open Suave.Successful

let app =
    choose
    [ GET ==> choose
        [ path "/hello" ==> OK "Hello GET"
        path "/goodbye" ==> OK "Good bye GET" ]
    POST ==> choose
    [ path "/hello" ==> OK "Hello POST"
    path "/goodbye" ==> OK "Good bye POST" ] ]

startWebServer defaultConfig app
REACTIVE PROGRAMMING WITH EVREACT

```ocaml
1: let net = 
2:   +(  
3:     (!md |-> fun e -> printfn "Mouse down @(%d,%d)" e.X e.Y)  
4:     (+(!mm) |-> fun e -> printfn "Mouse move @(%d,%d)" e.X e.Y) / [!mm; !mu]  
5:     - !mu |-> fun e -> printfn "Mouse up @(%d,%d)" e.X e.Y  
6:   )
```
```haskell
module 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
```
EARLY CONSIDERATIONS

- Using names and metadata seems impractical to control flow of information
- Overlaying of different network topology is becoming mainstream (SDN)
- Security should be enforced not only in communication, but also on route processing and message content
- Identity based filtering does not scale to the size
- Content and annotations should be added all along the way

[Graph showing estimated number of installed IoT devices by sector from 2014E to 2019E, with projections for billions of devices. The graph is sourced from BI Intelligence Estimates.]
A MOTIVATING EXAMPLE: TEMPERATURE SENSOR

• A temperature sensor on a window: public info
• The same sensor on a patient body: confidential
• Identity based control seems not to scale to the IoT size, we need semantic rules
• A distributed logic for controlling messages with user defined policies
• Keep automation (msg -> reaction -> message) as close as the device
A (ALMOST) WORKING DEMO

- Control the Relay to open the door
CONCLUSIONS

• We believe that HTTP/REST/JSON message routing will be central to IoT

• Security is central and should be first class in the message processing

• IoX goal is to become an Open Source, cross platform runtime reference for routing and processing IoT messages based on standard protocols contributing to make the IoT/Fog more than a buzzword

• Early bits will be posted on GitHub

• Follow #IoX on Twitter