Realizing Context Services in the Mobility First Architecture

Presented by Richard Martin

Joint work with:
Jon Li, Yanyong Zhang, Marco Gruteser, John-Austen Francisco and the MobilityFIrst team

Outline

Background on MobilityFirst
- GUID overloading
- Delivery Services

Examples:
- Sending to an event
- Routing communications

Realizing Context Services
- Dual stacks
- Implementation strategies

Challenges and next steps

Mobility First: GUID overloading

Globally Unique Identifiers (GUIDs)

-Large ID (few kb) that serves as source/destination

Overload GUIDs to many kinds abstract entities:

- Places, meetings, vehicles, people, sensors, devices, content
- Communication of messages to these entities

Like IP address overloading, but MobilityFirst is overload aware

Service IDs provide type hint

MobilityFirst Delivery Services

Global Name Resolution Service (GNRS):

- Fast translation (low 100's of ms) of GUIDs
  - GUIDs → to list of route-able Network Addresses (NAs)
  - GUIDs → to list of GUIDs (recursive)

GNRS translations + a service ID supports the following primitives:

- Unicast, Multicast, Anycast
- Could build using translation trees

MobilityFirst also provides delay, store and forward services

- Buffer, segment, reassemble large messages
- Send as devices and connectivity become available

We will use these basics to build context services

Defining Context

Environmental states external to the network that:

- Define Entities
- Impact communication

Example states:

- Network:
  - sender, receiver, connection point, channel state
- Spatial-temporal:
  - Time and Location
- Device:
  - Type, energy stored, off/asleep/awake
- Social:
  - In a meeting, busy, free, neighbors

Example: Entity Definition Example

Send a message to an event:

Define “tea time” as a GUID to NA map:

```
If ((# of phones in the break room > 3) AND
     (the time is between 2-5 PM )        AND
     (coffee is pot is > 45°c))

     THEN
          Set the GUID translation to the list of all the
network addresses of phones in the break room.
```

Example: Communication Impact

Routing calls on entry and exit:

```
 If ((my phone enters the conference room)  AND
     (the time is between 8AM-5PM )          AND
     (the source caller is NOT the day care))
     THEN
          Set my personal GUID translation to a voice-mail
NA instead of my Phone's NA.


   If  (my phone exits the conference room)
     THEN
          Set my personal GUID translation to my phone's NA
```

Context Communication Paradigms

Communicate messages to an identity rather than an address
- covered by GUID layer

Communicate messages to an identity on conditional context:
- receive only during work hours, unless from a specific identity
- send a message to whomever is in a car's passenger seat
- send a message to a phone but not when in a moving car

Send a message to an event
- all people at the Winlab's teatime
- anyone at a talk posted on the calendar
- anyone who is in the building that isn't in a meeting

Context Actions

The network's response to changing context:

Add or remove devices from a logical group

Forward messages to another device

Defer messages

Deliver prior deferred messages

Realizing Context

Realizing context requires sensing:

- external to the network
- internal to the device
- network state and conditions

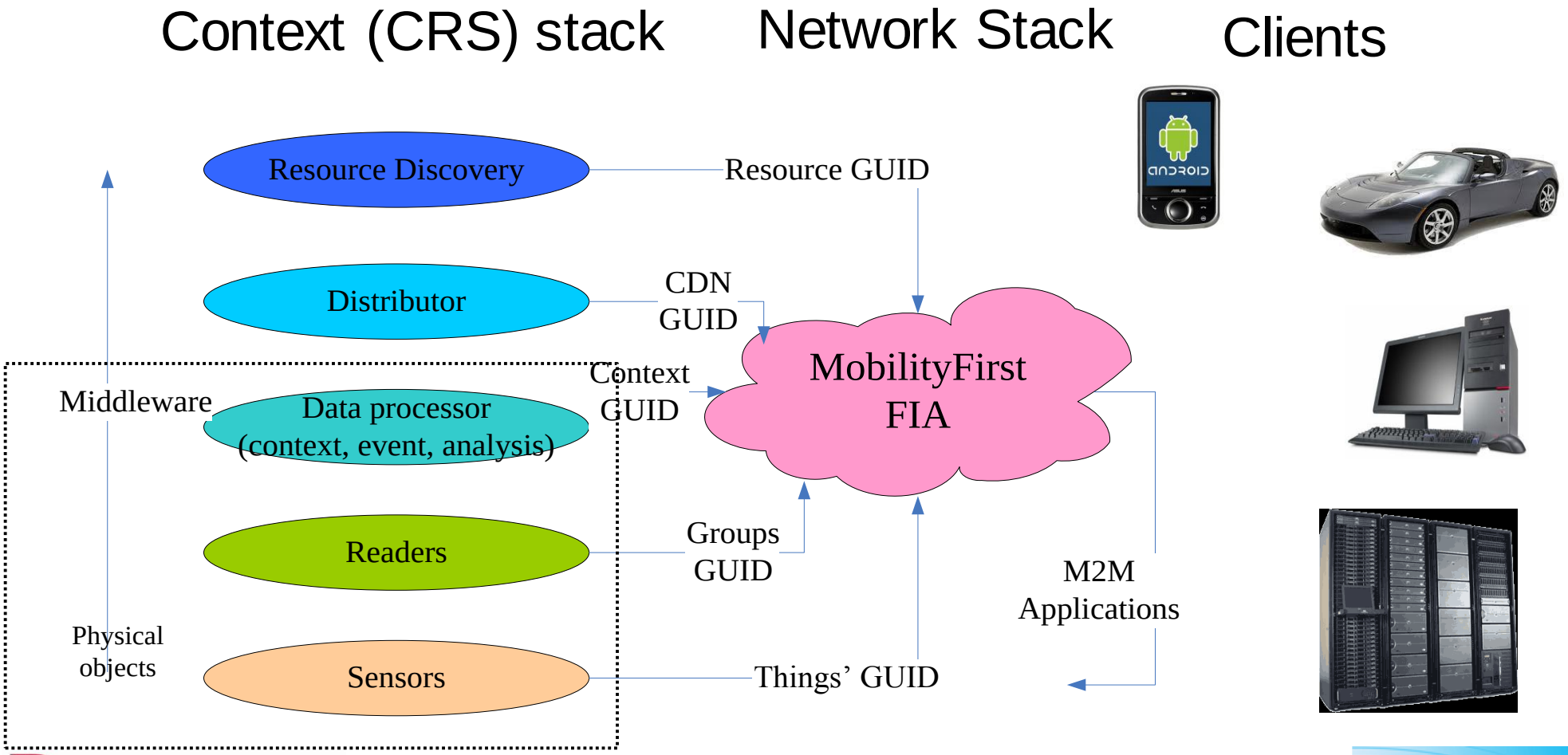
Dual stack approach:

Context Stack provides sensing/logic

GUIDs used at different levels between network and

Context Resolution Service (CRS)

Translation context descriptions into network destinations

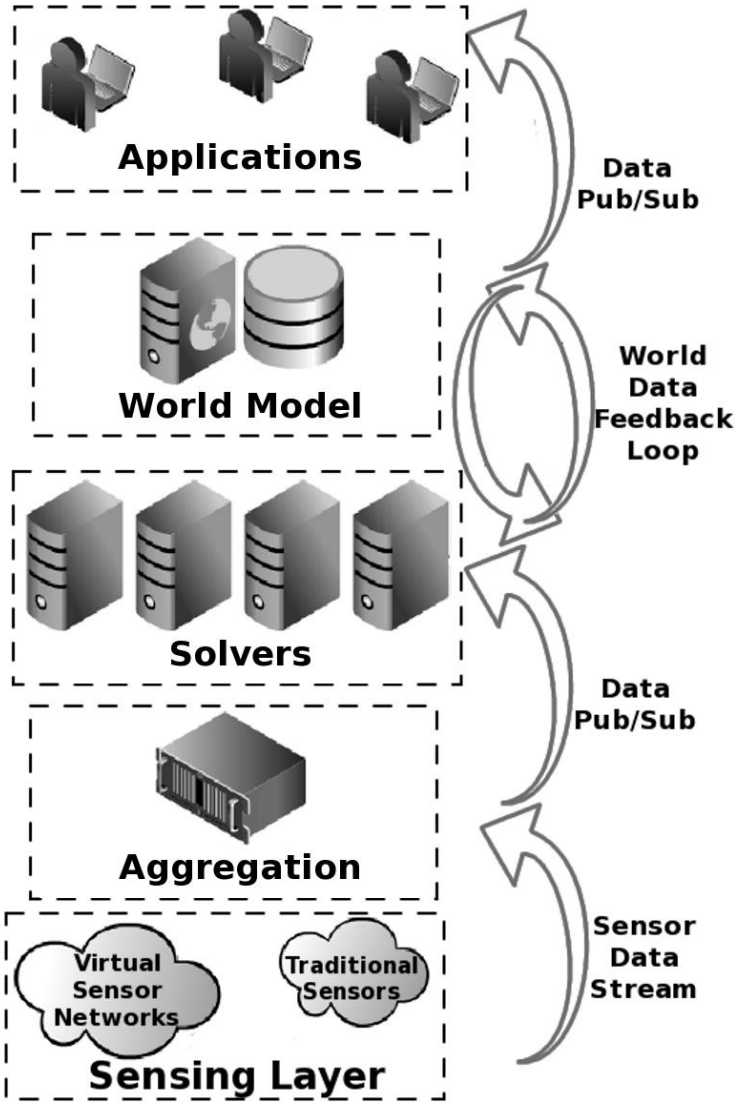


RUTGERS

WINLAB

Octopus Sensing Platform

- Sensors connect to an intermediate layer that hides details
- Solvers build higher-level representations from low-level ones
- A uniform model of the world allows sharing
- Applications run in standard environments in the cloud



CRS: Implementation Strategies

Context Outside the Network: Heavy Lifting *on sending client*

- Client issues a SEND-lookup on a query
- Client talks to any of the Servers from the SEND-lookup GUID list

Context Inside the Network: Heavy Lifting *off of clients*

- Client issues a SEND on a CGUID or query
- CRS computes context and carries out the SEND on the Client's behalf

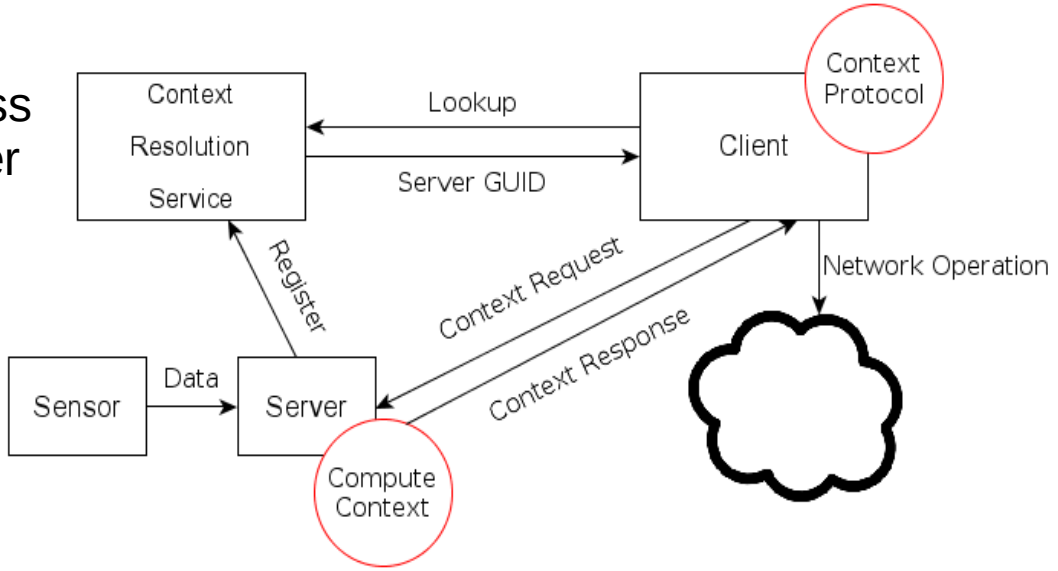
Intentional Receipt: Client only gets *traffic it wants*

- Server/Sensor can issue a SEND with self-described, untargeted data
- Server/Sensor can also issue a GET-lookup for a data description
    - Server/Sensor can SEND to all returned Client GUIDs

Context: Outside the network

Context now:

- Client queries the server address
- Client talks with particular server
- Server computes context
- Server delivers result
- Client interprets result
- Client sends correct NetOp



Aspects:

Heavy lifting *on client*

- Client needs to understand a lot
- Client has to know what to do based on the server result
- Highly decoupled; can be implemented on nearly any data network

Server

- Needs to manage Sensor data and Context computation

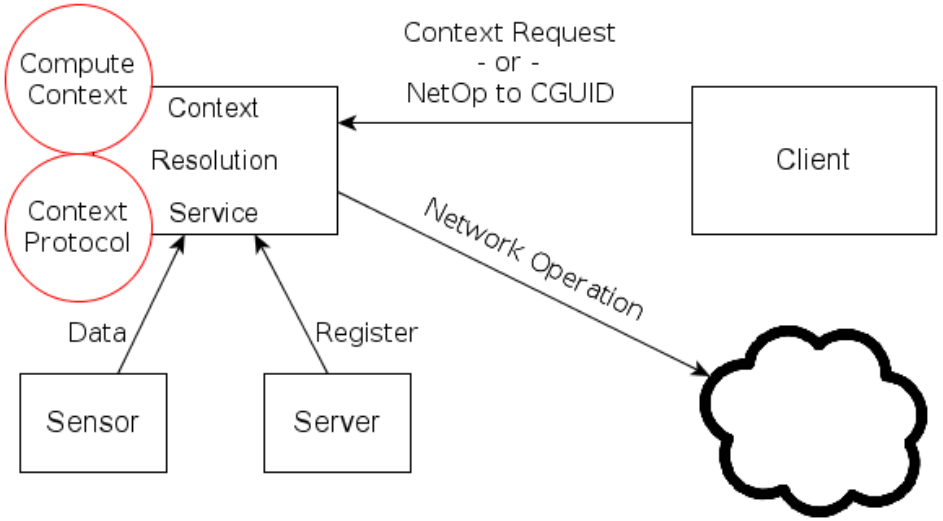
Context Service

- Logically very simple; essentially a database

Context: Inside the network

Context service:

- CRS gets Data
- CRS gets context description
- Client delivers context request
  - -or NetOp on context GUID
- CRS computes context
- CRS dispatches correct NetOp



Aspects:

Heavy lifting *off of client*

- Client does not need to know anything about context
- Client can act on a CGUID without knowing it

Server/Sensor

- Could keep Sensor data local and register as a Sensor itself
- Could specify new operations based on data CRS knows about

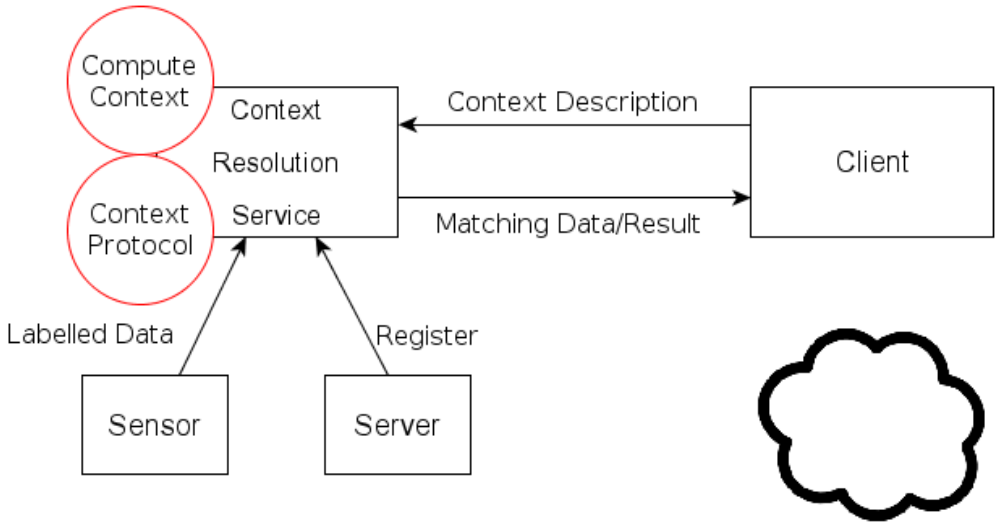
Context Service

- Requires a compute layer
- Inherits problems of coherence and consistency

Context: Intentional receipt

Context Service:

- CRS gets Labeled Data
- Client registers a description
- CRS computes context
- CRS delivers results that match



Aspects:

Client only gets *traffic it wants*

- Client needs to quantify its traffic as context constraints
- Client can use a service to generate these constraints

Server/Sensor

- Deliver data with no target address
- Data carries its own description (delivery as a continuation)

Context Service

- Computes data self-description against client criteria
- Delivers to clients whose criteria match the data self-description

Challenges and Next Steps

Connecting sensing, networking and context services

symbolic names → context expressions →
"tea time" → "if () then {} " →

Incorporating Octopus sensing, CRS and MF

winlab.teapot.temp > 45 → GUID/NA maps