Capturing Sensor Data from Mobile Phones using Global Sensor Network Middleware

Charith Pereral, Arkady Zaslavsky, Peter Christen, Ali Salehi and Dimitrios Georgakopoulos

(IEEE 2012)

Presented By-

Anusha Sekar

Summary

- Introduction
- Terms and Concepts
- Mobile Sensors
- Global Sensor Networks
- DAM4GSN Architecture
- GSN Wrapper Life Cycle
- Android Wrappers

Introduction

- Smart phones use a variety of sensors.
- DAM4GSN Architecture to capture sensor data using inbuilt sensors.
- Combine GSN with Android to capture sensor data- achieved using AndroidWrapper.
- Performance Evaluated based on Power Consumption of device.

Terms and Concepts

- Sensors
- Middleware
- Internet of Things
- Global Sensor Network
- DAM4GSN Architecture
- Wrapper

Mobile Sensors

- Common Sensors:
- Motion Sensors- Accelerometer, gravity, linear accelerometer
- Position Sensors- Orientation, geomagnetic field, proximity
- Environment Sensors- Light, pressure, humidity, temperature

- Sensors extended using PAN.
- PAN connects sensing devices to mobiles.
- GSN used as data stream processing engine

to retrieve data from sensors.

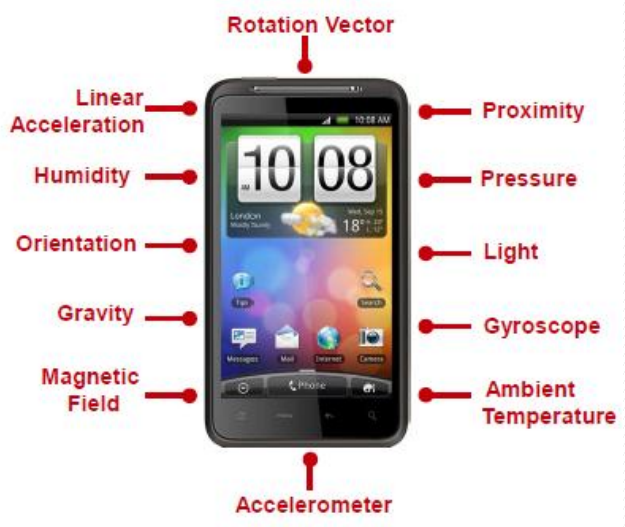


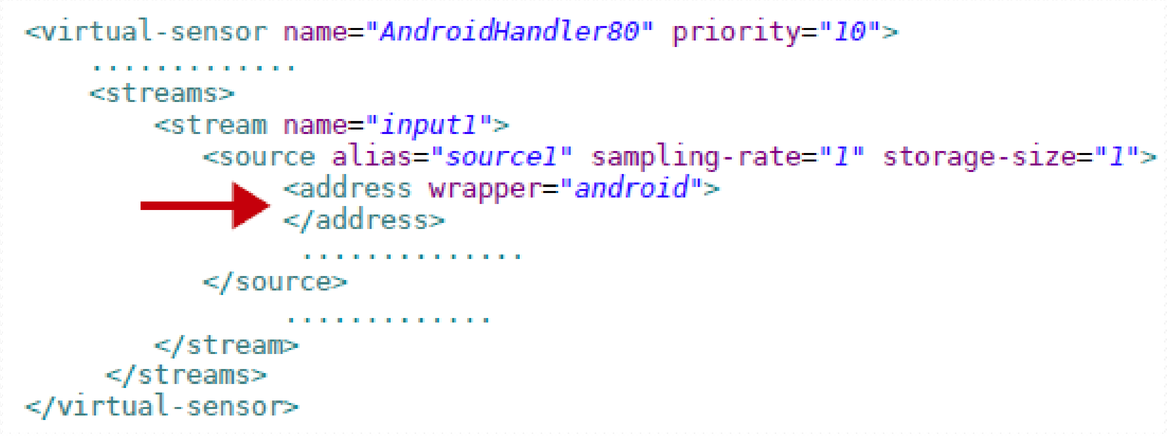
Fig. 1. Sensors in Mobile Phones

Global Sensor Network

- GSN- provides middleware to address challenges of sensor data integration and distributed query processing.
- Design of GSN based on simplicity, adaptivity, scalability, light-weight implementation.
- Features:
  - Simplifies process of connecting sensor devices to applications
  - Allows to integrate, discover, combine, query and filter sensor data.
- Virtual Sensor- like a Data producer. Ex: wireless camera, mobiles.
- Wrapper- Java code that acquires data from a device.
  - Transforms raw data into GSN standard data model.
  - Wrapper classes extend to AbstractWrapper class.
- GSN provides wrappers for all TinyOS sensors.

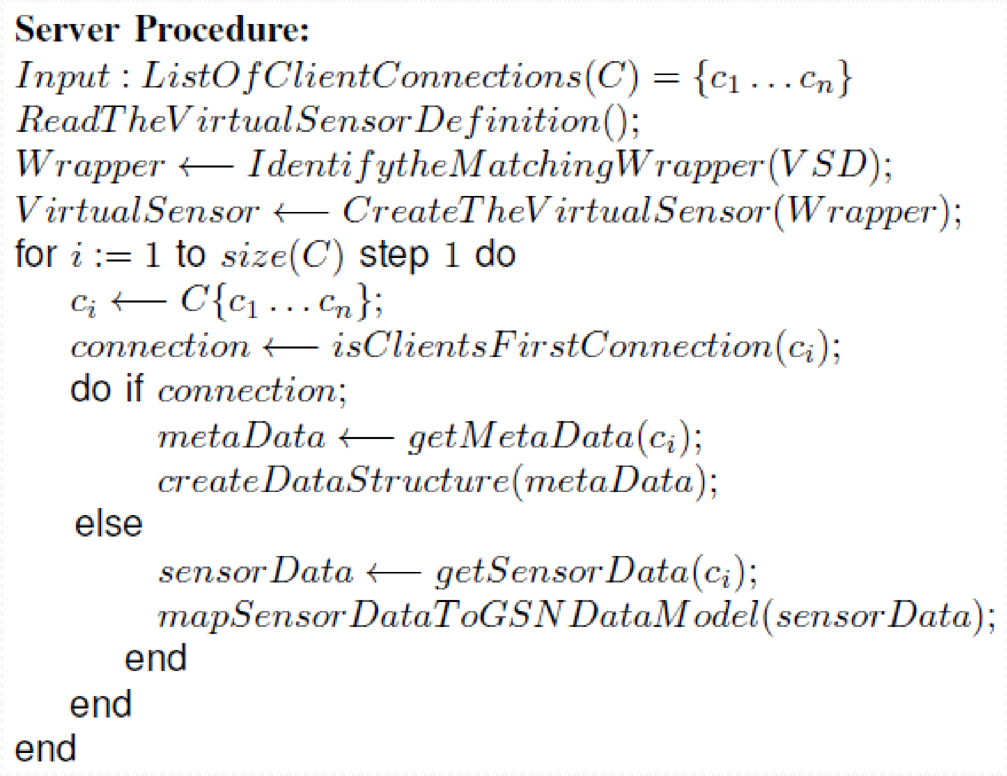
DAM4GSN Architecture

- **Server Configuration**- Explains how GSN server needs to be configured to collect data
- Steps-
  - Develop wrapper to retrieve data from mobiles.
  - Define a Virtual Sensor.
- Virtual Sensor Definition (XML file)- provides information to GSN to create a Virtual Sensor.



```
<virtual-sensor name="AndroidHandler80" priority="10">
    ..............
    <streams>
        <stream name="input1">
           <source alias="source1" sampling-rate="1" storage-size="1">
                <address wrapper="android">
                </address>
                 ...............
           </source>
                ..............
        </stream>
      </streams>
</virtual-sensor>
```

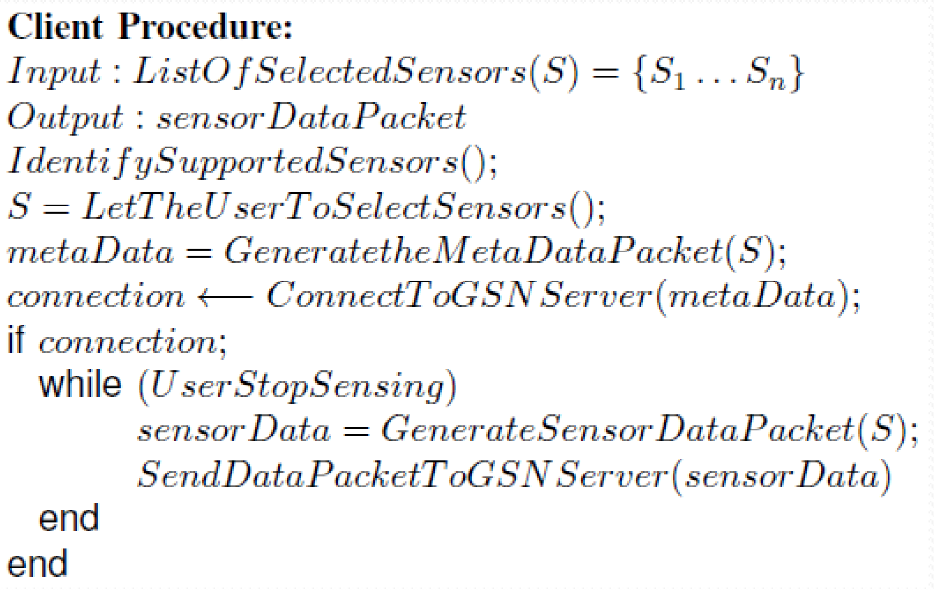
- Phases of communication-
  - Client sends data to GSN Server.
  - GSN Server configures Wrapper and accepts sensor data.
  - Client sends data to GSN Server according to the frequency set.
  - Generic Wrapper-changes internal data structures to suit sensors.



Server Procedure:

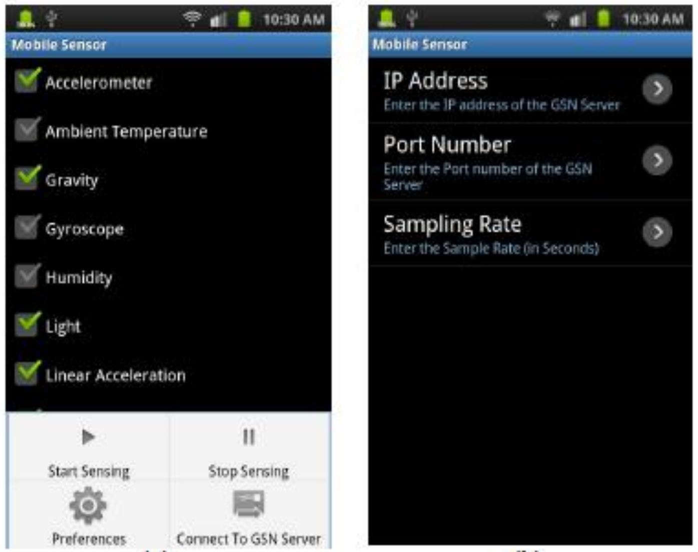
```
$Input : ListOfClientConnections(C) = \{c_1 \ldots c_n\}$
$ReadTheVirtualSensorDefinition();$
$Wrapper \longleftarrow IdentifytheMatchingWrapper(VSD);$
$VirtualSensor \longleftarrow CreateTheVirtualSensor(Wrapper);$
for $i := 1$ to $size(C)$ step 1 do
    $c_i \longleftarrow C\{c_1 \ldots c_n\};$
    $connection \longleftarrow isClientsFirstConnection(c_i);$
    do if $connection;$
        $metaData \longleftarrow getMetaData(c_i);$
        $createDataStructure(metaData);$
    else
        $sensorData \longleftarrow getSensorData(c_i);$
        $mapSensorDataToGSNDataModel(sensorData);$
      end
    end
end
```

- **Client Configuration**- Explains how mobile phones need to configured to read sensor data through sensors and send to GSN Server
- Steps-
  - Identifies the in-built sensors.
  - Sensor enabled if supported by hardware and software.
  - Requires GSN IP address, port number and sensing frequency.
  - User needs to connect device to GSN Server using WiFi or 3G.
  - Connection established based on data sent to GSN Server.
  - Start Sensing.



Client Procedure:

```
$Input : ListOfSelectedSensors(S) = \{S_1 \ldots S_n\}$
$Output : sensorDataPacket$
$IdentifySupportedSensors();$
$S = LetTheUserToSelectSensors();$
$metaData = GeneratetheMetaDataPacket(S);$
$connection \longleftarrow ConnectToGSNServer(metaData);$
if $connection;$
   while $(UserStopSensing)$
         $sensorData = GenerateSensorDataPacket(S);$
         $SendDataPacketToGSNServer(sensorData)$
   end
end
```



(a)

Mobile Sensor

- ✓ Accelerometer
- Ambient Temperature
- ✓ Gravity
- Gyroscope
- Humidity
- ✓ Light
- ✓ Linear Acceleration

Start Sensing | Stop Sensing | Preferences | Connect To GSN Server

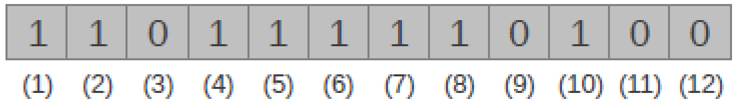
(b)

Mobile Sensor

- IP Address — Enter the IP address of the GSN Server
- Port Number — Enter the Port number of the GSN Server
- Sampling Rate — Enter the Sample Rate (in Seconds)

- **Data Format**- Explains how communication between GSN Server and mobile phone occurs and how data packets are formatted.
- Data format-

➢ Metadata- Establishes connection between GSN Server and mobile.



1	1	0	1	1	1	1	1	0	1	0	0
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)

Fig. 5. Metadata Packet

Metadata packet size does not vary from device to device.

➢ Sensor Data- generated in the same order as the selected sensors.

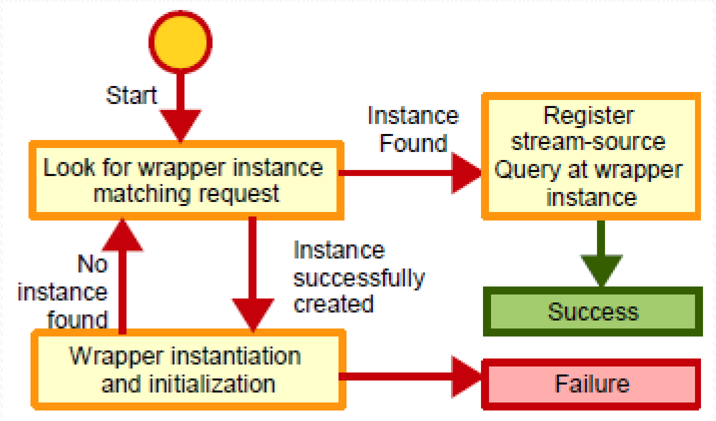
Sensor data size can be varied from 4 bytes to 108 bytes (max of 27 floating values).

Sensor readings from configured sensors appended to data packet.

GSN Wrappers Life Cycle

- Steps-
- VSD file defined->virtual sensor creation process begins.
- Wrapper corresponding to each sensor created by sending a Wrapper Connection request to the wrapper repository.
- Wrapper Connection Request- object that has wrapper name and initialization parameters.

i. Repository looks for instance that matches WCR.
ii. If no match, repository creates wrapper object.
iii. If no match and no appropriate wrapper, it returns false.


Start
Look for wrapper instance matching request
Instance Found
Register stream-source Query at wrapper instance
No instance found
Instance successfully created
Success
Wrapper instantiation and initialization
Failure

Android Wrapper

- Wrappers need to extend to gsn.wrapper.AbstractWrapper
- Methods defined-

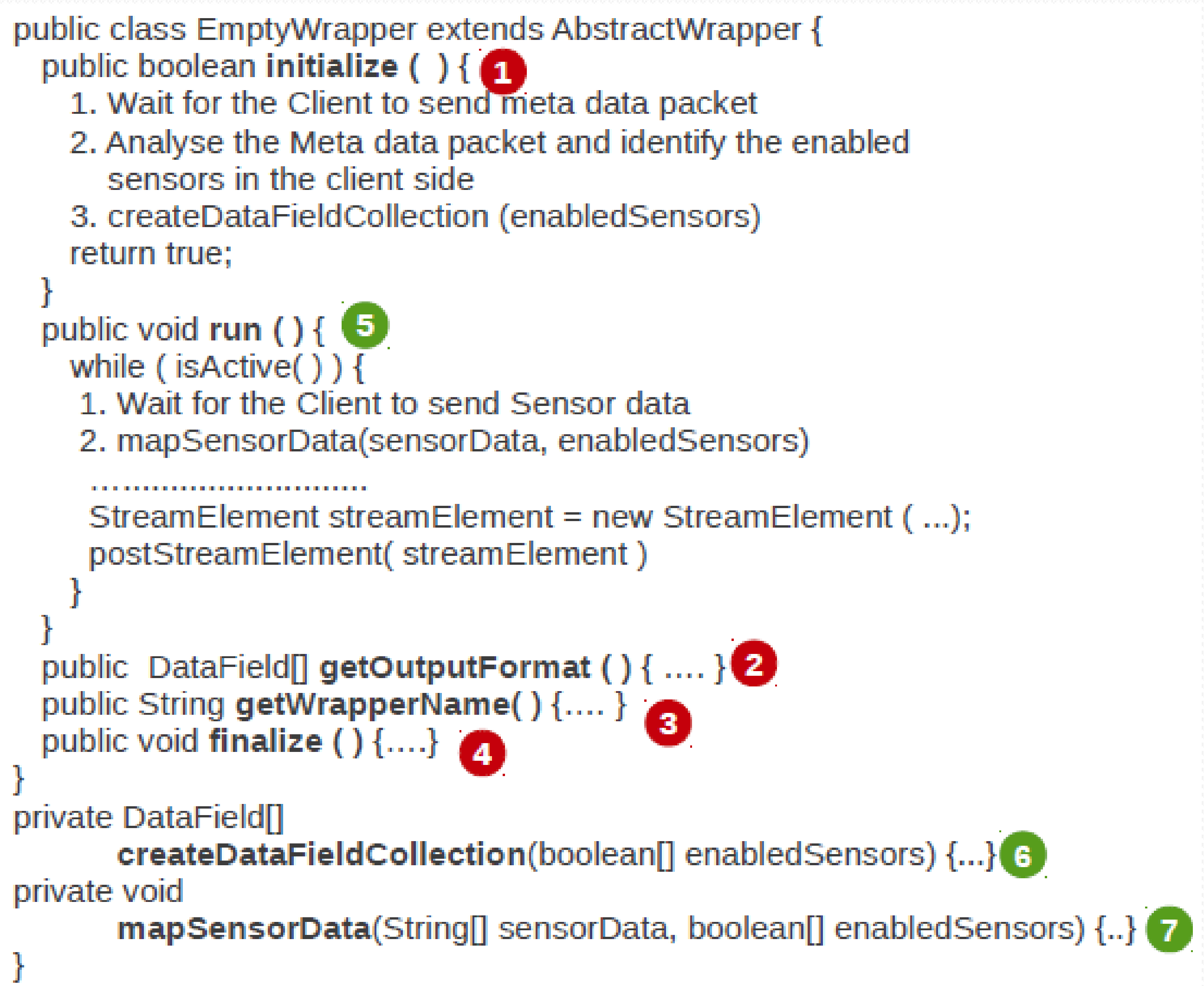
➢ boolean initialize()- called after creating wrapper object->creates a socket and waits for client to send metadata packet->packet is analyzed and identifies client sensors->sensors sent to createDataCollection()

➢ finalize()- called at the end of life cycle to release resources.

➢ getWrapperName()- returns name of wrapper.

➢ getOutputFormat()- describes data structure produced by wrapper.

➢ run()- waits till client sends data->maps received data to GSN data model structure using mapSensordata()->function performed on data.



```
public class EmptyWrapper extends AbstractWrapper {
   public boolean initialize (  ) { ①
      1. Wait for the Client to send meta data packet
      2. Analyse the Meta data packet and identify the enabled
         sensors in the client side
      3. createDataFieldCollection (enabledSensors)
      return true;
   }
   public void run ( ) { ⑤
      while ( isActive( ) ) {
       1. Wait for the Client to send Sensor data
       2. mapSensorData(sensorData, enabledSensors)
        ..............................
        StreamElement streamElement = new StreamElement ( ...);
        postStreamElement( streamElement )
      }
   }
   public  DataField[] getOutputFormat ( ) { .... } ②
   public String getWrapperName( ) {.... }  ③
   public void finalize ( ) {....}  ④
}
private DataField[]
           createDataFieldCollection(boolean[] enabledSensors) {...} ⑥
private void
           mapSensorData(String[] sensorData, boolean[] enabledSensors) {..} ⑦
}
```

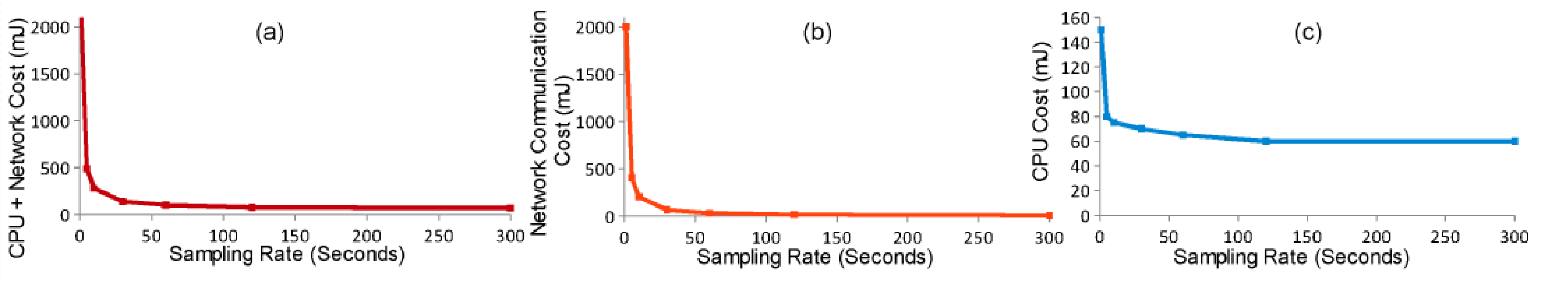
Fig. 7. Android Wrapper

Advantages

- Need not install GSN in every device. Need to develop only a single wrapper on server side->Preserves SCALABILITY.
- Updates need to be done only on client side.
- Need not change wrapper code as it is generalized for all sensors.

Evaluation

- Devices used- Samsung Galaxy S and PowerTutor app.



- From graphs-

➢ Network Communication cost is higher than CPU energy cost.

Future Work

- Automating Wrapper Development
- Building DAM4GSN architecture into the GSN Middleware.
- Compressing data while sending over network to GSN Server.

Questions?