MOSDEN: A Scalable Mobile Collaborative Platform for Opportunistic Sensing Applications

Prem Prakash Jayaraman , Charith Perera , Dimitrios Georgakopoulos, and Arkady Zaslavsky

smart devices are everywhere

figure from the net

opportunistic sensing

autonomous collaborative sensing

takes advantage of a population of users

measures large-scale, group phenomenon

figure from the net

related work

great applications, but

monolithic, application-specific architectures

custom hardware, protocols

no interoperability

interoperability

could allow arbitrary application development

hook into as many devices as necessary

leverage existing hardware

goals

autonomous (online and offline)

scalable

interoperable

support efficient sensor data capturing, processing, storage and sharing

MOSDEN - MObile Sensor Data ENgine

Separate data collection, processing and storage to application logic

easy development for distributed collaborative crowdsensing application

Support for autonomous functioning - self-management

component-based system supporting domain specific models and algorithms

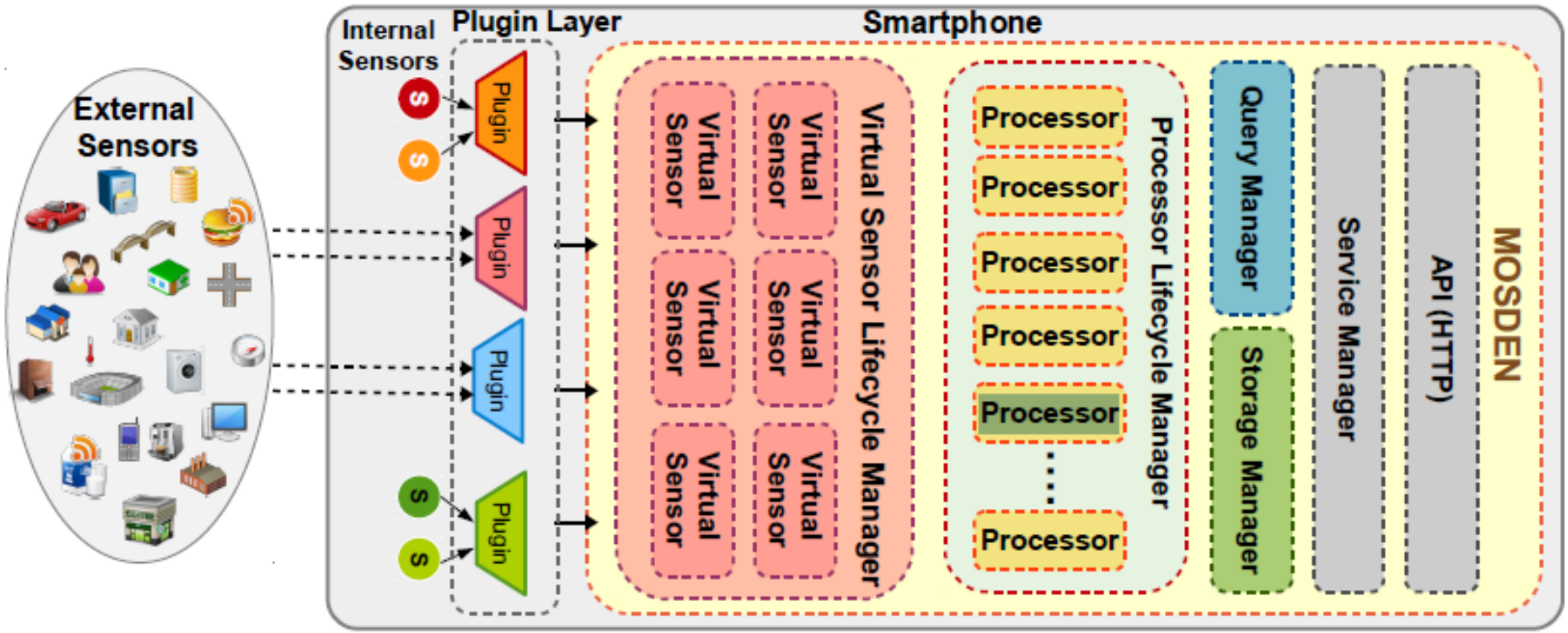

figure from the paper

virtual sensors

accelerometer
gyroscope
magnetometer
front and rear cameras
NFC
barometer
speaker
microphone
proximity
light sensor
Bluetooth
GPS
WiFi + cellular
humidity
temperature

figure from the net

```
<DataFields>
   <DataField>
      <name> accelerationX_axis_incl_gravity </name>
      <type> double </type>
      <description> Acceleration force along the X axis
      (including gravity)measures in m/s2.
       </description>
   </DataField>
   <DataField>
      <name> accelerationY_axis_incl_gravity </name>
      <type> double </type>
      <description> Acceleration force along the Y axis
      (including gravity)measures in m/s2.
       </description>
   </DataField>
   <DataField>
      <name> accelerationZ_axis_incl_gravity </name>
      <type> double </type>
      <description> Acceleration force along the Z axis
      (including gravity)measures in m/s2.
       </description>
   </DataField>
</DataFields>
```

figure from the paper

proof of concept implementation

MOSDEN runs in the background

connects to GSN server, registers available sensors.

OR

server pings request for devices

client receives a data request

collects, processes, and stores the data

remote server app queries the instance for the data

push, or pull

experiment

2 setups. One as GSN server, one as MOSDEN.

2 types of apps: streaming (restful, good connection) and burst (burst, bad connection. New connections each transfer).

tested each client having 30 virtual sensors

results

Storage requirements – linear to time/number of sensors

CPU Usage - about the same

30 (restful) vs 40 (push) % (push requires more)

Resftul stakes more memory for more devices

Burst has more delay after request. Esp. on a mobile phone

however, platform/os optimizations not considered by the authors

conclusion

It's a good platform. Evaluation was a bit interesting, though stats/results can't give strong conclusions

future work

Not testing with large number of devices

When building such applications, devs need to make many decisions still (push/connected, frequency, etc)

How else could management improve?

Assuming MOSDEN is actually the best, scalable etc (features are good), what is lacking? What things could improve dev speed?