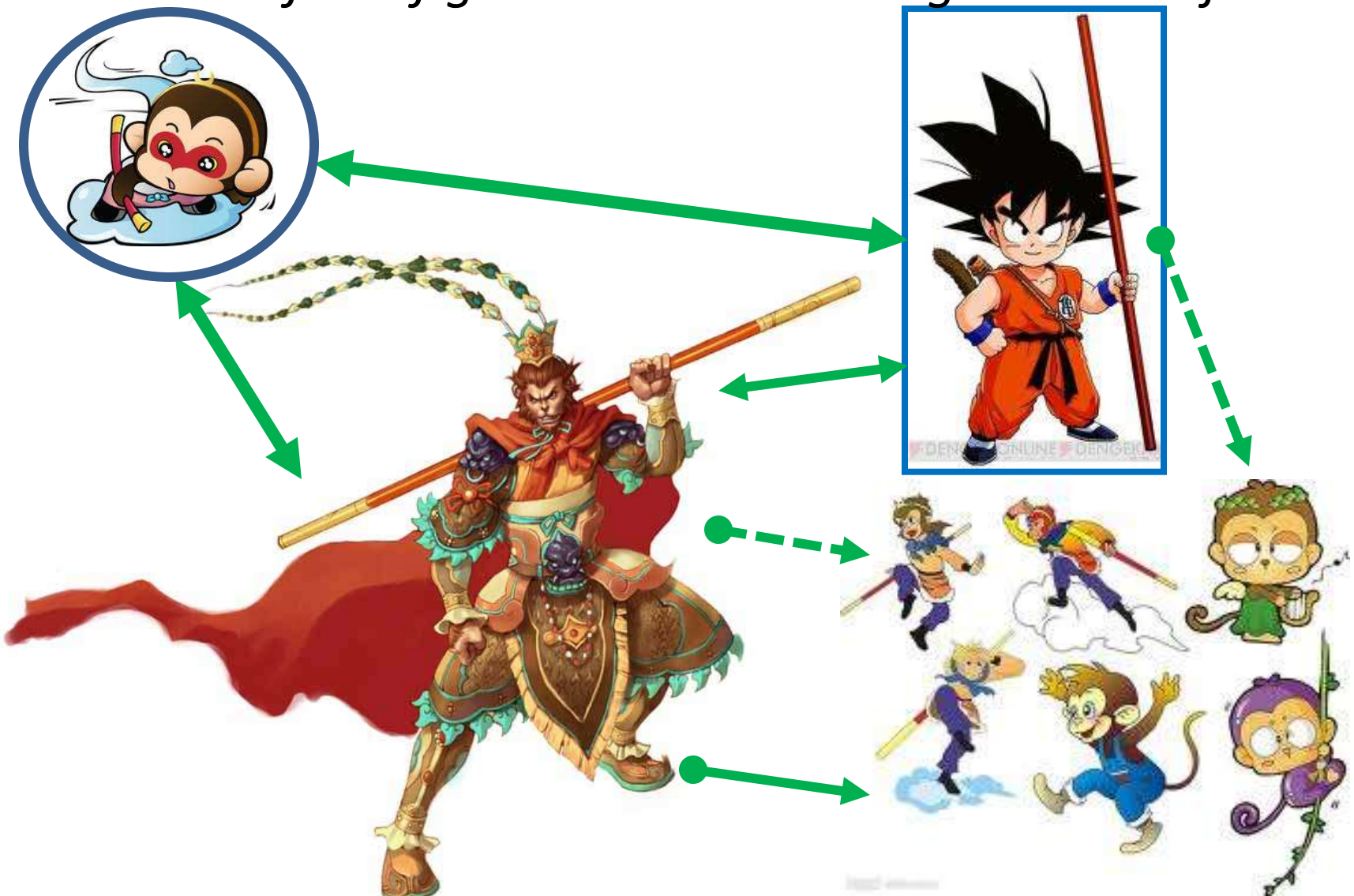
Wu-Kong: WSN Unleashed

A Self-Configurable M2M Management Project



Agenda

- Background: Wireless Sensor Networks for M2M
  - Examples of WSN: *flow@home* and sMAP
  - Sensors, communication, programming sensors, and middleware
- WuKong: Virtual MiddleWare for M2M

Motivation Example - *flow@home*

A smart home utility: Flow Monitoring at Home

- **What to monitor**
  - Human, with or without known identity
  - Kids, maybe with size or person identification
  - Animal, pets with known (tag), or unknown identity (mouse, snake, …)
  - Intruder, unknown identity (should trigger video capture)
  - Valuable objects, with RFID tags
- **Probably the most important utility in many smart home applications**
  - Security system at day time, vacation or evening out
  - Safety coverage for fire, earthquake or disaster events
  - Babysitting safety assistance for toddlers and pets
  - Senior citizen home care
  - Energy saving for inactive house zones

flow@home - How

- Houses have heterogeneous sensors, WSN and Internet
  - Motion
  - Pressure (on chairs, floor, etc.)
  - Light
  - Infrared
  - Temperature
  - Audio
  - RFID, cellphone, video
  - Actuating devices
  - etc.
- Use context and movement reasoning engine to collect, reason and record data
- Data can be saved on home server (and cloud), accessed (locally or remotely) with adequate authorization

A Usage Scenario, *flow@home*

- Smart Home with a collection of heterogeneous sensors
  1. Sensor devices are distributed and installed (richly-populated initially)
  2. Master polls, discovers and identifies connected devices
  3. Master makes configuration plan according to app/user policy specification and context information
  4. Master reports/interacts with app/user to make deployment decisions
  5. Additional sensors are installed if necessary (go back to Step 2 if necessary)
  6. App is started with operational WSN

Advanced Usage Scenario, *flow@home+*

- Smart Home deployed with new sensors for new mission
  1. New sensor devices are added and new apps are ready to go
  2. Master polls and identifies connected old and new devices
  3. Master makes configuration plan according to app/user policy specification
  4. Master reports/interacts with app/user to make deployment decisions
  5. New sensors are loaded with configuration settings, while some old sensors are re-programmed with new code if necessary
  6. App is started with an expanded WSN

flow@home – Policy Selection

- After sensors are physically installed, they can be used for different purposes
- Depending on the scenario, users can set policies like:
  - Senior safety
  - Child safety
  - Pet safety
  - Disaster safety
  - Vacation security
  - Day and night time security
  - Party event (adult, children)
  - Energy saving
- We want to build the *policy-driven self-configuration support* for heterogeneous sensors, for concurrent applications, under different policies, at different times (context), for different rooms, by different owners.

The sMAP Project (Dawson-Haggerty et al, Berkeley 2010)

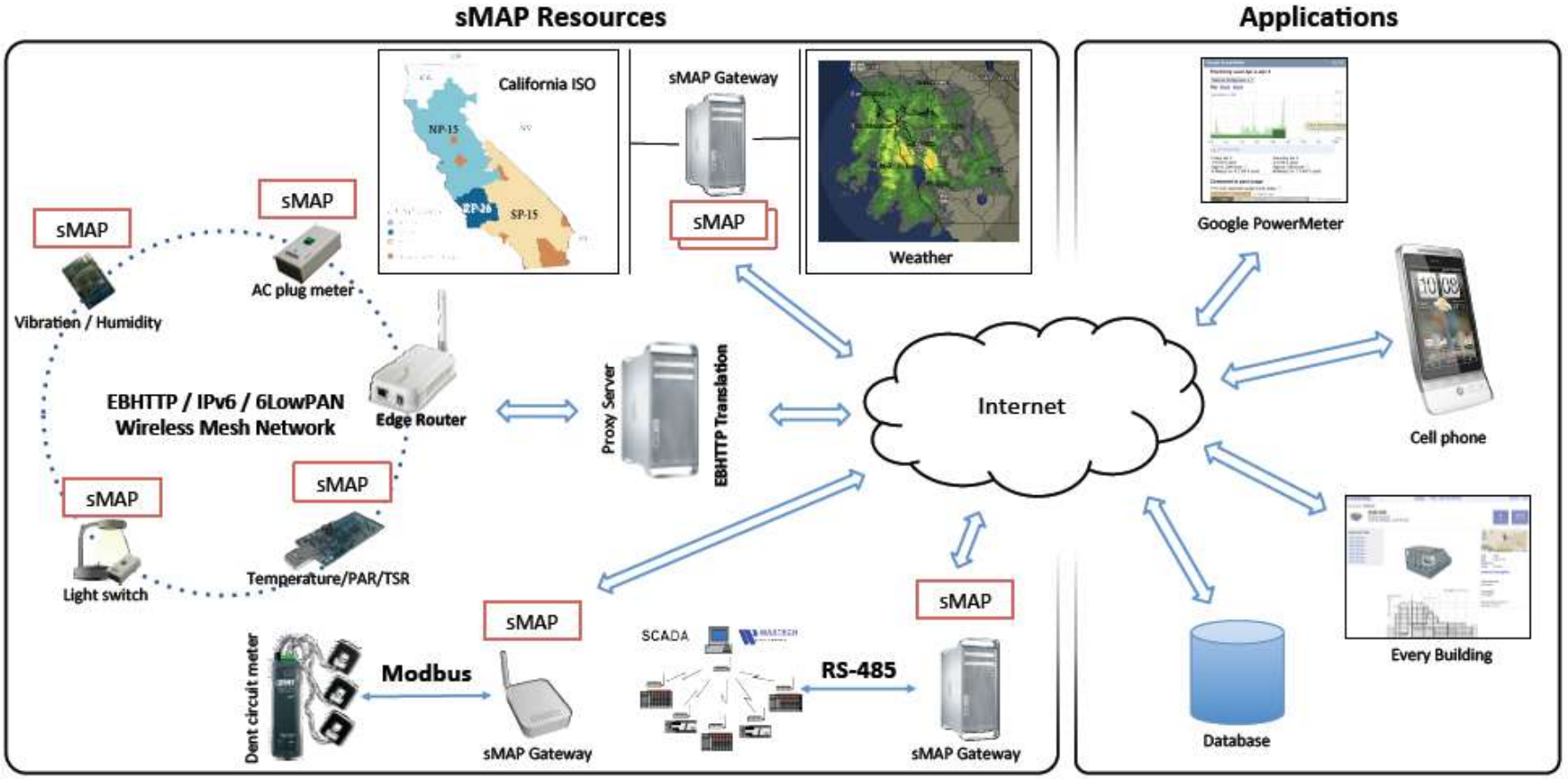


Figure 2. The full diversity of sMAP sources and gateways we have implemented under the aegis of a building monitoring project. Sources include Modbus/RS-485 meters, 6lowpan/IEEE802.15.4 wireless devices, and proxied data from weather and grid sources. sMAP serves as the data interchange layer sitting between those instruments, and applications including a Google PowerMeter gateway, mobile applications, databases, and EveryBuilding. EveryBuilding is a web application for modeling the relationships between spaces in a building, and managing information about the location of all the sense points.

Composition of Wireless Sensor Networks for M2M

- Sensors:
  - The inputs of the M2M networks
  - Sampling the environment using analog/digital devices
- (Wireless) communication:
  - The transmission from the sensors to the gateway/data collectors are mostly conducted via wireless networks.

Versatile Sensor Node

Processing platforms

- ST ARM Cortex-M3

Comm. tech.

- ZigBee, 6LoWPAN, IEEE 802.15.4
- Bluetooth
- Wi-Fi
- GSM/GPRS
- Ethernet
- Sensor Network Protocol (SNP)
- Satellite

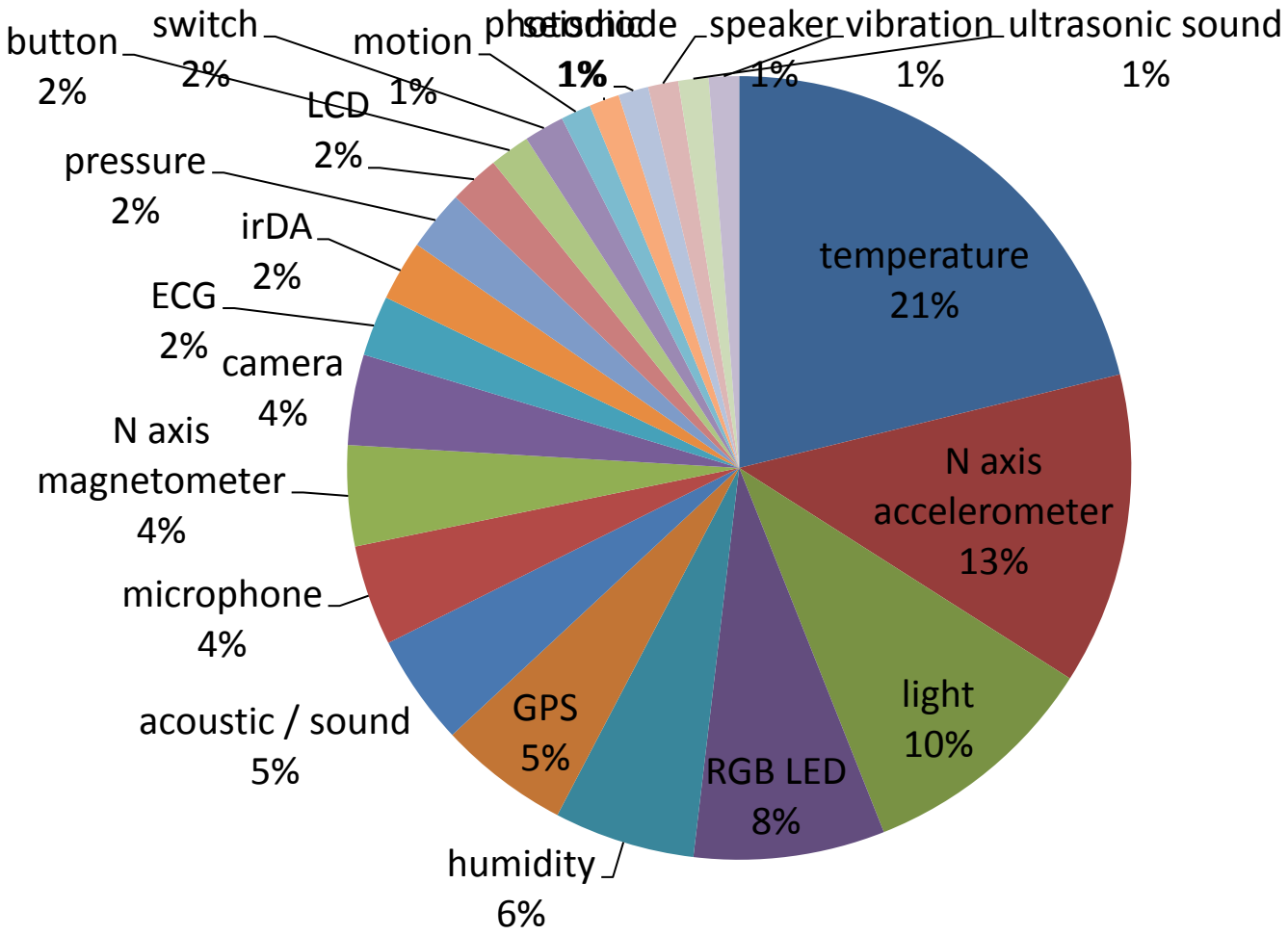
Sensors

- Temperature
- Humidity
- Luminance
- Color
- Reflectance
- Pressure/Force
- Camera
- Optical Detector
- GPS
- Sound
- Accelerometer
- Gas (O_2, CO_2, CO)
- Hall effect
- Motion, presence, range (IR, ultrasonic)
- Capacitive/inductive touch
- Gyroscope
- Compass
- …

Actuators

- Pulse Width Modulation (Light, Motor)
- Switch, Relay
- Servo
- Alarm

What sensors are mounted on the nodes?



What to sense?

absorbed dose	current density	frequency	mass	sound intensity
absorbed dose rate	direction	gesture	mass density	specific energy
acceleration	direction of motion	GPS info	molar energy	specific entropy
activity	dose equivalent	heart rate	molar entropy	specific heat capacity
alcohol	duration	heat capacity	molar heat capacity	specific volume
altitude	dynamic viscosity	heat flux density	moment of force	step frequency
amount of substance	electric charge	humidity	motion	stress
amount of substance concentration	electric charge density	illuminance	percentage	substance presence
angle	electric conductance	image	permeability	surface tension
angular acceleration	electric currency	inductance	permittivity	switch state
angular velocity	electric current	irradiance	plane angle	temperature
area battery charge	electric field strength	kerma	power	thermal conductivity
blood glucose level	electric flux density	length	presence	time
blood oxygen level	electric potential difference	location	pressure	torque
blood pressure	electric resistance energy	luminance	proximity	velocity
body fat percentage	energy density	luminous flux	quantity of heat	volume
capacitance	entropy	luminous intensity	radiance	wave number
catalytic activity	exposure	magnetic field strength	radiant flux	wind speed

Wireless communication

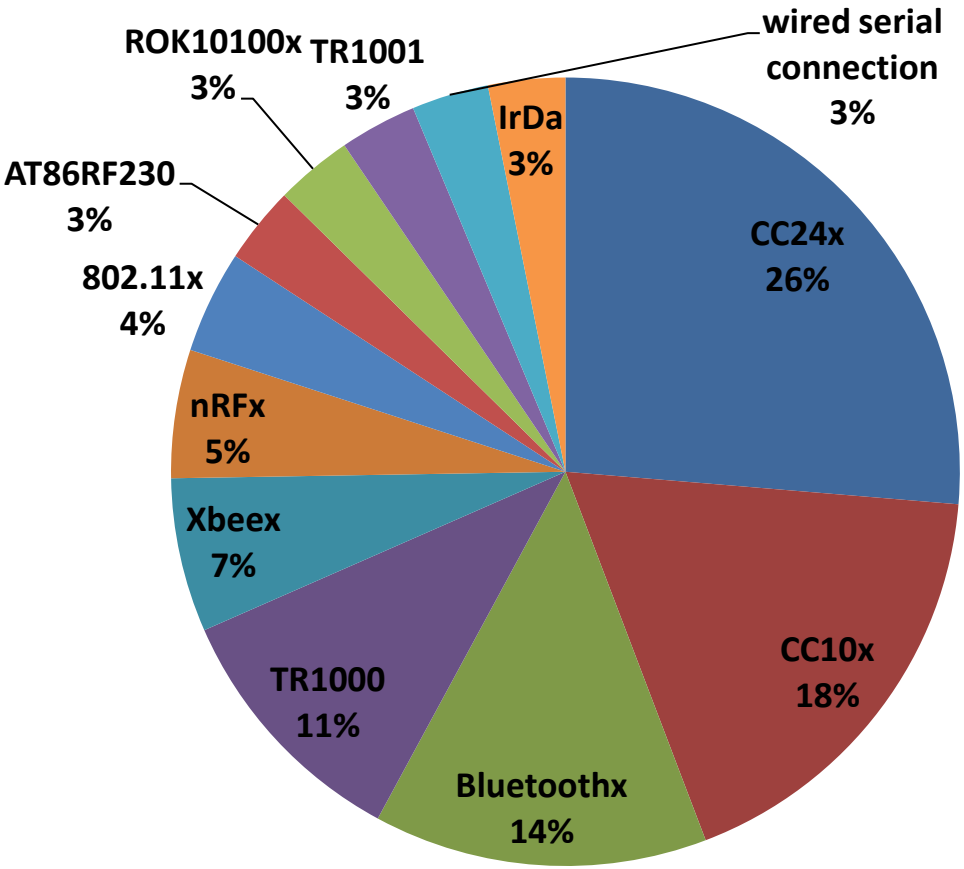
- Standardized wireless communication protocols
  - IEEE 802.15.4 - Low Rate WPAN (PHY & MAC)
  - ZigBee, 6LoWPAN, WirelessHART, Dash7, Wavenis …
  - IEEE 802.15.1 – Bluetooth (BT 3.0 Low Energy Mode)
  - IEEE 802.11x – Wi-Fi
  - Other dedicated communication technologies (Ethernet, GPRS, …) in WSN concept – gateways

- Proprietary wireless communication protocols
  - Z-Wave, ANT, MiWi, SimpliciTI, DigiMesh …

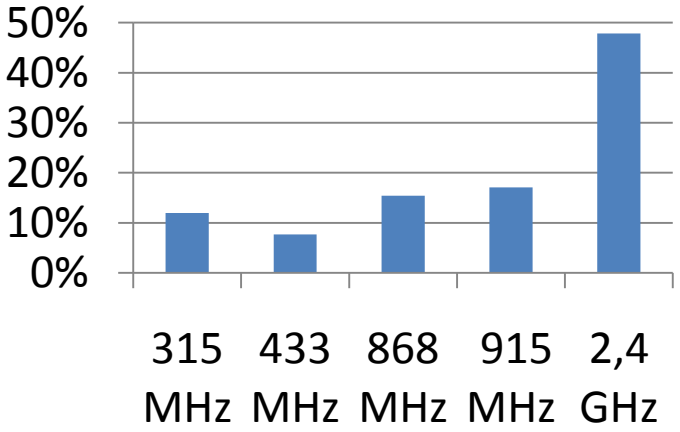
Communication speed

- high data-rates (i.e. > 1000 bps) (Stargate)
  - IEEE 802.11 units
  - Adapted general-purpose computers (Wi-Fi routers)
    - usually constant power supply
  - adapted general-purpose computers (smartphones) and embedded sensor nodes
    - rapidly consume battery power
- medium data-rates (i.e. ~1000 bps) (Smart-its)
  - Bluetooth radios
    - usually do not require constant power supply and can last on battery power for a while
- low data-rates (i.e. ~100-250 bps) (Sentio)
  - data-rates lower than 100 bps (PushPin)
  - data-rates of only few bps (SpotON)
  - dedicated RF interfaces

Which communication modules and frequencies?



Communication modules



frequency bands distribution

Overview as of May 2010

Programming Sensor Nodes

- tool chain: IDE, compiler, debugger
- microcontroller is programmed and executes the code
- radio chip is not programmed, but controlled by microcontroller, usually via SPI which sets/reads registers
- compiled code is loaded to the microcontroller using bootloader or JTAG
- protocol stack may be precompiled and available through API or available as library
- operating system (not needed for simple tasks)
- virtual machine (optional)

Maté Overview

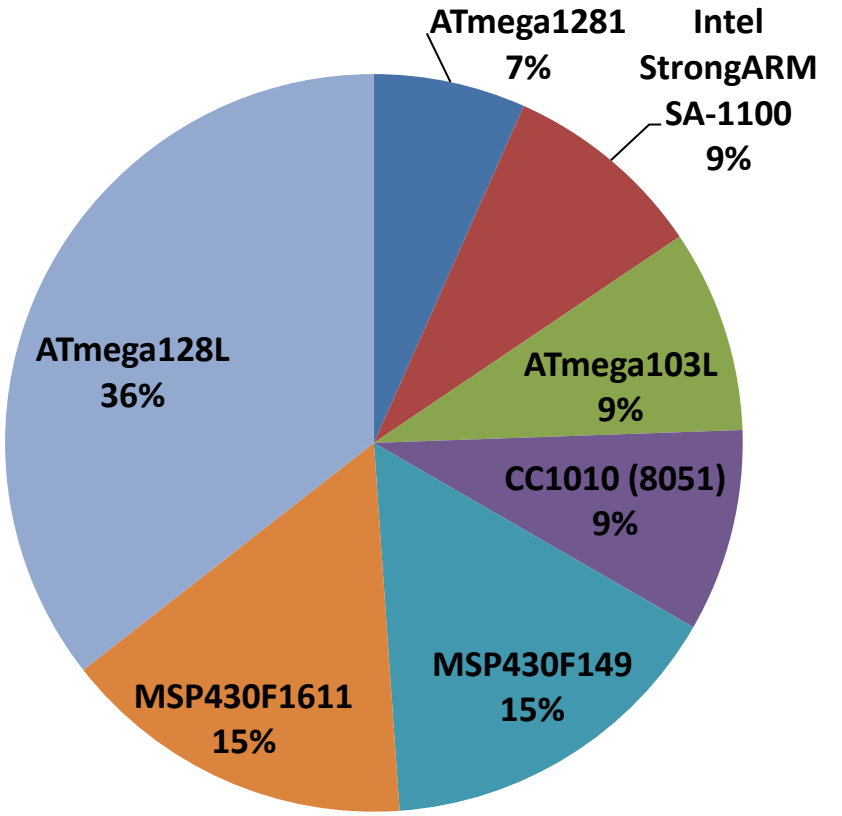
- TinyOS component
- 7286 bytes code, 603 bytes RAM
- Three concurrent execution contexts
- Stack-based bytecode interpreter
- Code broken into 24 instruction capsules
- Self-forwarding code
- Rapid reprogramming
- Message receive and send contexts

Classification by processing and communication capability

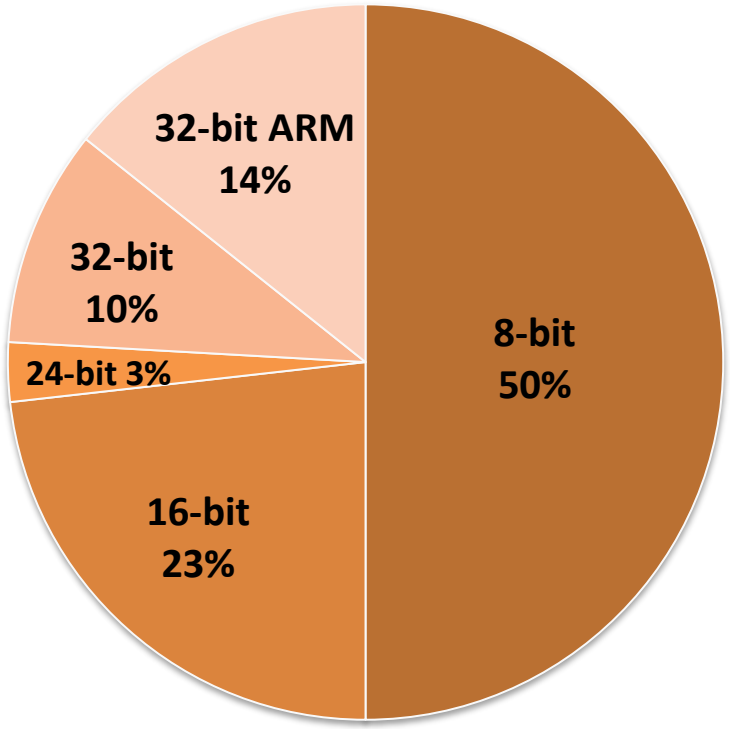
- 125 types of sensor nodes identified in research community
- 118 types of processing units
  - the most popular microcontroller (Atmel ATmega128L, 8 bits)
  - the most popular DSP (Coolflux DSP NxH1200, 24 bits)
  - FPGAs are exceptions (Xilinx XC3S200 Spartan-III)
- 112 types of communication interfaces
  - the most popular radio frequency communication module (TI / Chipcon CC2420, 2.4 GHz)
  - the most popular optical communication module (IrDa transceivers)

Overview as of May 2010

Which processors and how advanced?



Processing units



Architectures

Overview as of May 2010

Sensor Network Programming Requirements

- Small (fit on Motes)
- Expressive (lots of applications)
- Concise (programs short for memory and network bandwidth)
- Resilient (don’t crash the system)
- Efficient (in sensing and communication)
- Tailorable – for application-specific operations
- Simple – in-situ, fast, “autonomous” programming

Maté Overview, Continued

- Three execution contexts
  - Clock, Receive, Send
- Seven code capsules
  - Clock, Receive, Send, Subroutines 0-3
- One word heap
  - `gets`/`sets` instructions
- Two-stack architecture
  - Operand stack, return address stack

MIDDLEWARE

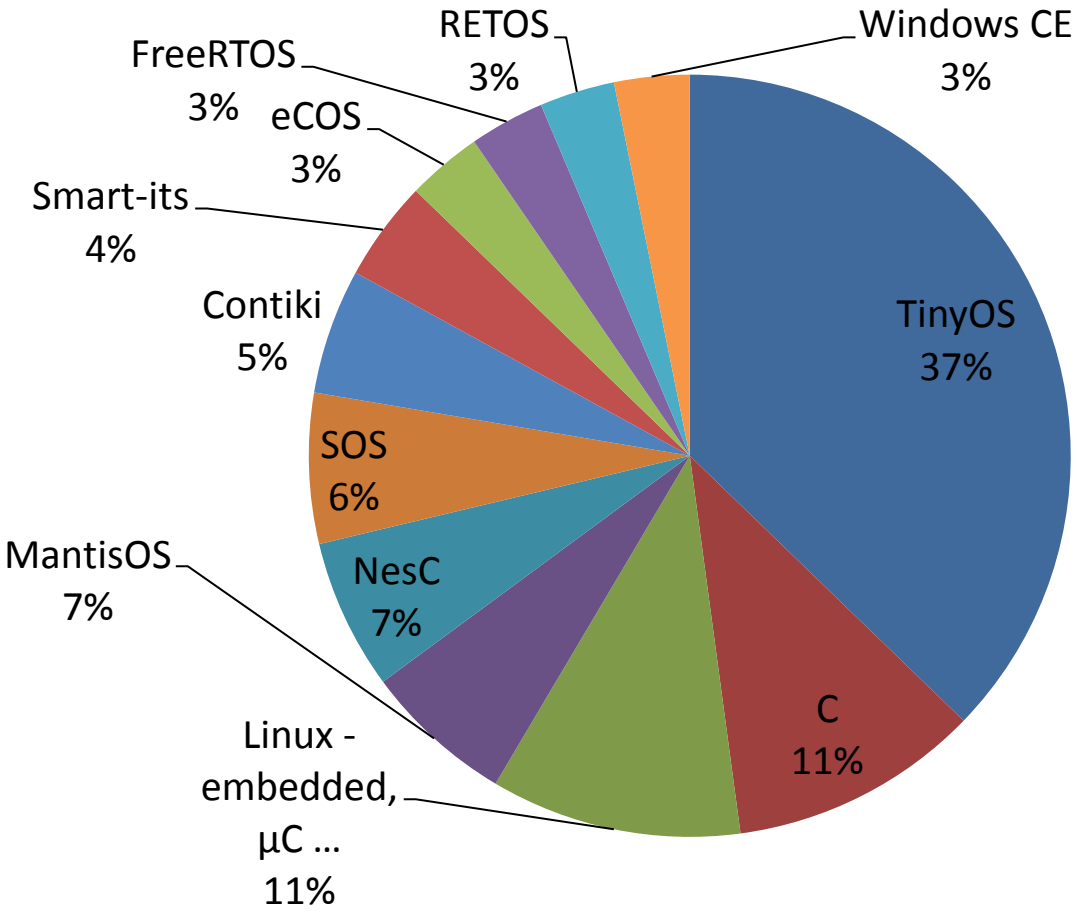
- Is a software layer which provides an abstraction of whatever lies under for use by whatever lies above.

- Under the middleware can lay hardware resources of various types or software resources such as firmware, drivers and sometimes also the operating system.

- *"The main purpose of middleware for sensor networks is to support the development, maintenance, deployment, and execution of sensing-based applications"*[1]

[1]Hee-Jin Jeong, Choon-Sung Nam, Dong-Ryeol Shin, "Design and Implementation of Middleware in Sensor Networks Using Publish/Subscribe Model," IEEE International Workshop on Semantic Computing and Applications, 2008 .

Middleware Usage

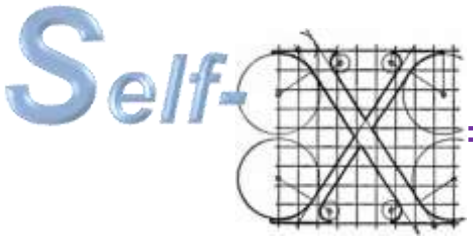


Summary for M2M Application Deployment on WSN

- M2M applications have historically been built with a strong coupling among applications and the underlying network infrastructure.
  - The application-dependent approach is justified by the need to achieve energy efficiency and performance optimization.
- But these applications not only are difficult to develop, but also often result in rigid systems, with M2M's specifically designed to particular scenarios and unique target environments.
- Such an approach is not desirable, considering the costs of the infrastructure deployment, the long operational lifetime of the network, and its potential capability of serving many, concurrent applications.
- We are far from Rapid Application Development (RAD) for M2M.
  - There is very little software reuse from one WSN to another WSN.

Project Goal: Self-X M2M

- The objective is "zero-cost deployment" (i.e., cost refers to human effort)
  - Problem: manual effort is often the bottleneck in large-scale deployment and long-term sustainability of M2M systems in the field.
  - Proposal: Users of M2M systems do not need to learn how and where to deploy sensor nodes.
- The project is to provide an intelligent and responsive M2M framework
  - Self-X stands for self-configuration, -protection, -healing, and -optimization for sensor nodes and gateways.
- Based on user-defined policy and context, the proposed middleware services automatically perform sensor node and M2M configuration, application deployment, fault handling, and system reconfiguration.



= configuration, protection, healing, optimization

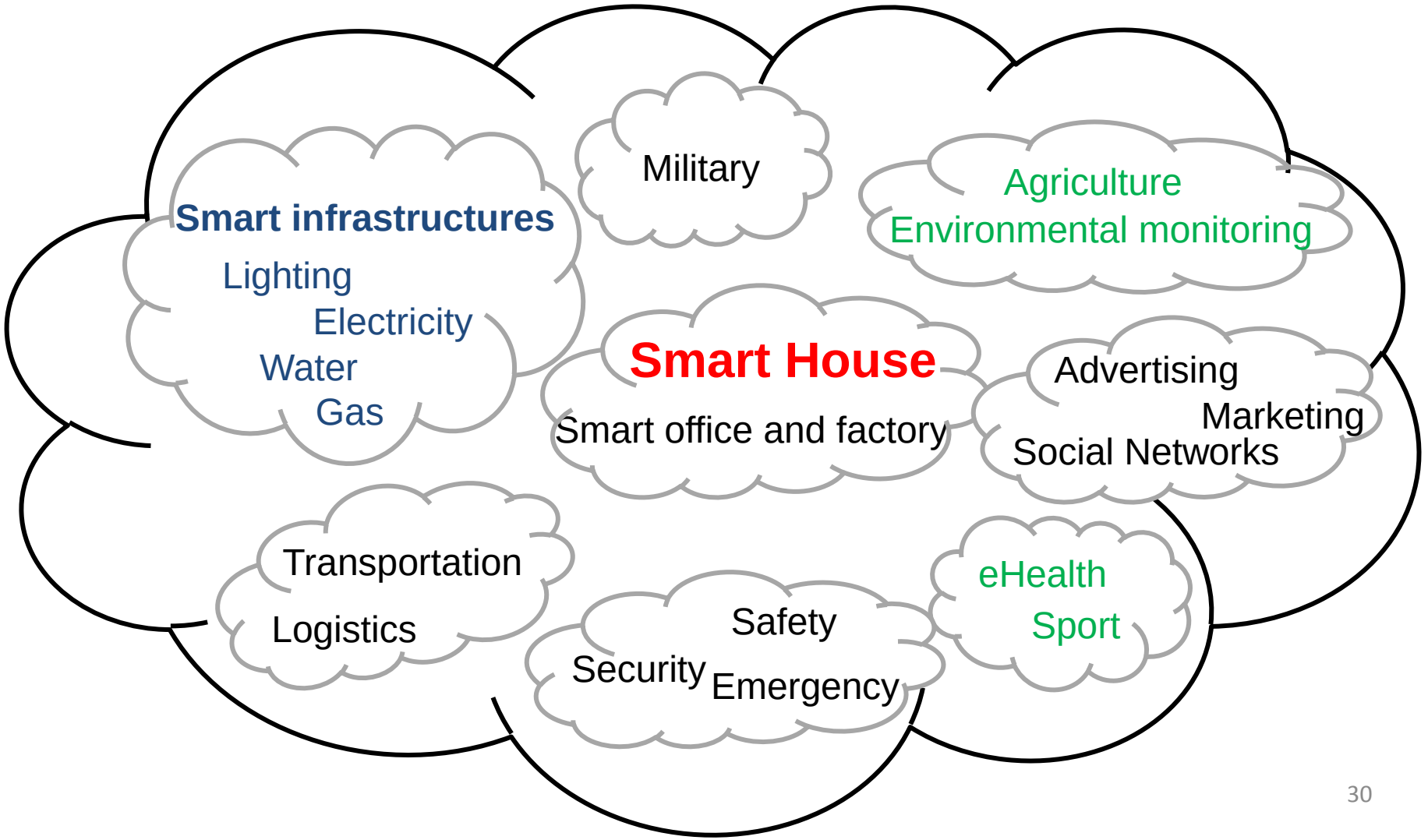
Wu-Kong :
A Classical Chinese Epic Hero

- *Sun Wukong* (悟空), also known as the *Monkey King*, is a heroic character in the classical Chinese epic novel *Journey to the West* (西遊記). In the story, he is a monkey born from a stone who acquires super powers through many masters. He later becomes a loyal personal guard for a monk going to the Western heaven to receive the sacred Buddhism scripture.
- Wukong knows 72 **transformations**, which allow him to transform into various animals and objects
  - he has trouble, however, transforming into other people, because he is unable to hide his tail when excited.
- His hair has magical powers. Each can transform into a **clone** of the Monkey King himself, or various subjects, weapons, or animals.
- Wukong uses a **versatile** golden-banded staff (如意金箍棒), which could change its size, multiply itself, and fight according to the whim of its master.
- Wukong rides on a **cloud** that can travel at an extremely high speed, bringing him virtually anywhere at any time.

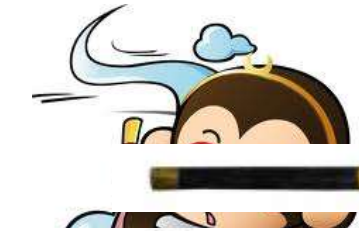
Wu-Kong: Literally, …

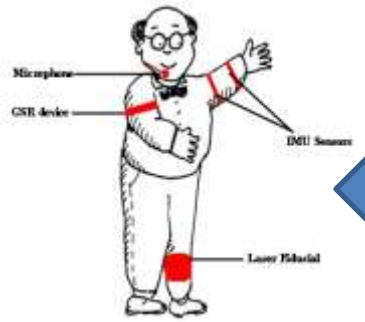
- 悟 (Wu): enlightened
  - For our project: *intelligent*
- 空 (Kong): vanity
  - For our project: *virtual middleware*
- The project is to build an *Intelligent Virtual Middleware* for M2M, that can
  1. recognize and adapt to context and user demand;
  2. configure or transform devices into service components;
  3. deploy the most powerful yet least expensive solutions;
  4. do all of the above without physically access sensors.

M2M Application Areas



Wu-Kong: User Perspective





App (on Cloud)

Master

Communication Media
- Human
- Broadband
- Wireless

User

Sensor Nodes

User (e.g., demanding)

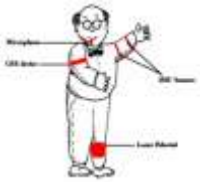
Nodes (e.g., devices) (e.g., sensing user)

Apps (running on user device and/or cloud)
- Physical world sensing and actuating
- Interact with user
- Need only limited computing power to make sense/send data
- Have access to unlimited computing power and intelligence

Gateway (on user side (as one of Nodes) or Master side)
- provides extra computing/connection
- provides backup coordination

Roles, Questions, Challenges

- User (intuitive)



  - How do users define context and high-level policy?
  - Can WSN trust a user and vice versa?
- Apps (concurrent)

  - How do they use, reuse and share sensors?
  - How do users start, pause and kill (if incompatible) apps?
- Master (intelligent)

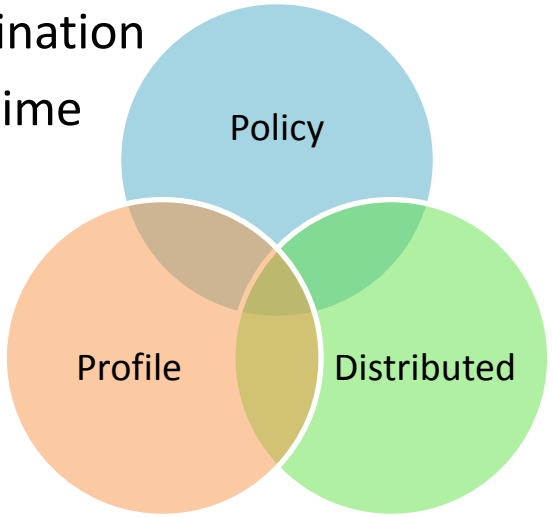
  - How does it know about & control heterogeneous sensors?
  - What‘s the tradeoff between complexity and effectiveness?
- Nodes and network (efficiency)

  - Would it be powerful enough to do all that?

  - How do we handle run-away sensors or network blackout?

Project Innovation: Tri-Framework

- Profile Framework (abstraction for heterogeneous nodes)
  - Device classes (capabilities): discover, share and control
  - Heterogeneous and virtual sensor sharing
- Policy Framework (intelligence for M2M management)
  - Configuration decision and constraint optimization
  - Configuration <-> fault tolerance, security, trust
- Distributed Framework (embedding WSN in cloud)
  - Sensor to Master to cloud distribution and coordination
  - WSN cloud for application access anywhere, anytime



Profile Framework

- Abstraction for heterogeneous WSN nodes so that Master can discover, access and control them
- Related work: uPnP (2006), DPWS (2006), WSDD (2008), TinyDB (2005), sMAP (SenSys 2010)
  - Range of works from detailed description of device services to simple sensor access protocol
  - WSDD defines a lightweight version of DPWS
  - sMAP defines a RESTful interface for universal device access
- Complexity and message size are critical for WSN
- We plan to study how to implement sMAP without HTTP

sMAP details

Policy Framework

- High level specification for M2M management
  - User or app defines an intuitive objective statement
  - Policy interpreter and configuration engine produce the detailed setup for target systems, enabling higher-level thinking/coding
  - By specifying policies declaratively and independent of actual devices, it is possible to change the behavior on-the-fly for better flexibility.
- Related work: Security (SPF), OS Policy, Ponder/Ponder2 (2001/2006), DSN (SenSys 2007), ADAE (SenSys 2010)
- Configuration and constraint satisfaction engine can take many attributes (e.g. context, fault tolerance, security, trust) into consideration for a better, more optimal performance
- Policy IDE, interpreter and context support are needed

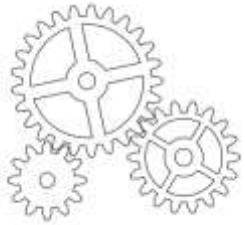
Policy details

Distributed Framework

- Use cloud as part of M2M
  - Sensor <-> Master <-> cloud connection and migration with least energy consumption
  - Security, privacy and trust management across node domains
  - Need to consider both efficiency, energy and security
  - Related work: SpartanRPC (2010), SC^3 for u-Life (Korea)
- Embedding WSN in cloud
  - Sensor-cloud for applications to access sensors anywhere, anytime
  - Service-oriented composition by selecting sensor services from sensor-cloud
  - Need energy and sharing consideration
  - Related Work: sMAP (2010), LiveE! (Japan), NWSG (Singapore)

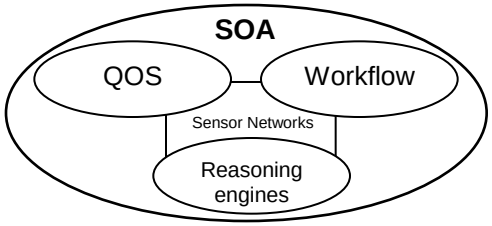
KONG Virtual Middleware

(*K*nowledge, *O*peration, and *N*etwork *G*ears)



WSN Unleashed

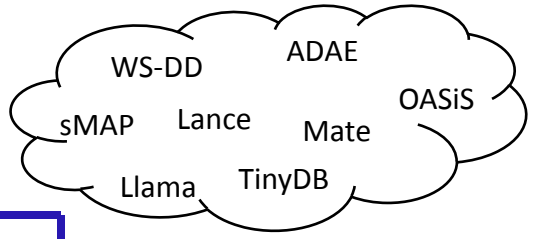
- Cloud Applications
- IDE Tools
- User Services
- Context and policy reasoning



Knowledge Gears

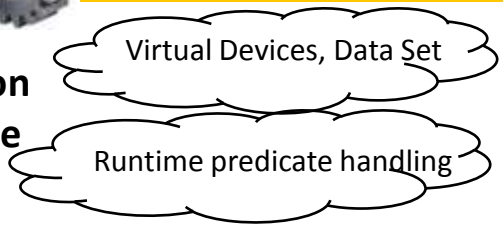
- Self-X capabilities
- WSN composition/reconfiguration
- Fault diagnosis
- Security policy

KONG



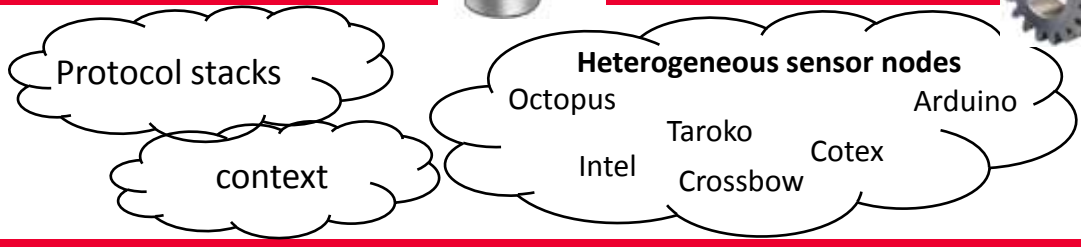
Operation

- Sensor device profile
- Policy and fault detection
- In-situ update & upgrade
- Security/trust keys

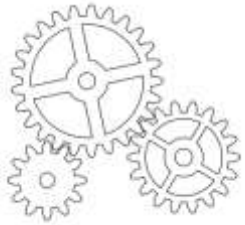


Network

- Sensor Network
- Distributed coordination



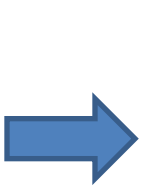
Issues on Virtual Middleware



- Lots of work on Virtual Machine on WSN: Darjeeling(2009), SwissQM(2007), PyMite, etc.
- Trade-off between overhead vs. simplicity & flexibility they provide
- VM is justified since WSN will evolve, be cheaper yet more powerful, but in critical needs of flexibility and intelligence
- Advantage: Code size and programming flexibility
  - Define high-level primitive/policy/abstraction
    - Simpler programming process
  - VM code smaller than native machine code
    - Reduce distribution, energy costs for software updates
- Disadvantage: Execution overhead
  - Increased time and memory requirements for execution
  - Increased energy spent in interpreting the code

Self-X Support

- Configuration protocol needed for the following phases:
  - Detect
  - Identify
  - Setup
  - Deploy



- Self-X includes configuration, protection, healing, optimization
  - Will work with other SIGCAM projects on security, trust, data mining to support self-protection and self-healing
  - Will work with other SIG's on energy and communication optimization

Master Workload Consideration

- There is one Master for each application but many gateways
- There are many sensor nodes, with heterogeneous computing power, sensor capabilities, and connectivity
- Multiple applications may want to share sensor node data:
  - may require one sensor to have different configurations, each for a specific application

  Option 1: master node will translate/convert raw sensor data into app-specific data

  Option 2: some gateways or powerful sensor nodes may share the conversion workload for a subset of sensors and act as their shadow sources

Handling Faulty Master

- Faulty Mater detection & recovery will be conducted by other gateways using distributed fault detection protocols
  - Depending the type of faults (omission, …, byzantine), fault-handling protocols may be simple or sophisticated, cheap or expensive
  - At UCI, we have built the Llama project with faulty monitoring agents and diagnosis reasoning engine on ESB for SOA
- Configuration decisions will be kept on a backup location for ease of recovery, either on a secondary gateway, or cloud server, or both

Programming and In-situu Update

- Load/re-load applications as structures of *high-level* primitives utilizing *lower level* VMW gears
- VMW supports efficient programming primitives
  - Including a bootloader that is responsible for VM upgrade
  - Different VMW classes support progressively powerful gears
- Related work: Maté (2002), SOS (2005)

Research issues

- How to use WSN to access/check all sensors
- Intuitive and consistent user policy specification
- Context and movement reasoning engine
- Real-time trigger and actuator support
- Normal and abnormal event identification
- Reprogramming and deployment transition
- Privacy, safety, security, and convenience tradeoffs
- Many other issues …