Title
Tenet: An Architecture for Tiered Embedded Networks (SYS 8)

Permalink
https://escholarship.org/uc/item/7s0013nx

Authors
Omprakash Gnawali
Ben Greenstein
Ki-Young Jang
et al.

Publication Date
2006
How do we build sensor network systems that are robust and easy to program?

Large scale sensor networks will be tiered
- Traditional approach with application specific code on one-tier network of impoverished motes and in-network processing in leads to complex software systems that are hard to program, debug, and manage.
- Tenet has two classes of nodes
  - Mote-class nodes (e.g. MicaZ, TelosB) interact with the real world i.e sense/actuate and run the same general purpose image even across different applications.
  - More resourceful masters (PC/Stargates) manage clusters of mote-class nodes (perhaps 25-100). Masters are more amenable to complex application level programming and debugging.

Tenet: An Architectural Principle for Tiered Embedded Networks
Multi-node data fusion functionality and complex application logic should be implemented only on the masters, since the cost and complexity of implementing this in motes outweighs the performance benefits of doing so.

Design Principles of Tenet
- Asymmetric Task Communication
  - Any and all communication from a master to a mote takes the form of a task. Any and all communication from a mote is a response to a task.
- Addressability
  - Any master in a Tenet can communicate with any mote or master in that Tenet. Any mote in a Tenet can communicate with at least one master in that Tenet.
- Task Library
  - Motes provide a limited library of generic functionality, such as timers, sensors, simple thresholds, data compression, and FFT transforms. Each task activates a simple subset of this functionality.

Tenet API and Applications

The Tenet Stack

Transport API
- Transport API provides functions to send tasks and collect results from the network.
  ```c
  config_transport(host, port)
tid send_task(task description)
response* read_response(timeout)
delete_task(tid)
  ```

Tenet Program
- Include "tenet.h"
- ``main()``
- ``config_transport(localhost, 19998)``
- ``tid = send_task("issue(1Hz)->sample(0x55, LIGHT)->send()")``
- While (not (tid timed_out)) {
  if (response) {
    response = read_response(1000 ms)
    light = find_attr(response->attrlist, 0x55)
    printf("Node %d reports light value %d\n",
     response->sender, light->value)
  } else {
    tid timed_out = TRUE;
  }
}

Tasking API
- Tasking API provides functions to describe a task to run on the motes.

System
- reboot(), send(type), local_address(), global_time(), routing_parent()

Sensor/Actuator
- sample(...) actuate(channel, argtype, value)

Tasks
- issue(gtime, abs, repeat)
- delete_taskif(arg, argtype)
- delete_activated_taskif(arg, argtype)

Attribute Management
- delete_attributeif(arg, argtype, attr)
- store(attr1, attr2)
- restore(attr, attr2)

Operations on Attributes
- Arithmetic: add(), subtract(), multi() etc
- Compare: less(), greater(), equal() etc
- Logical: and(), or() etc
- Bitwise: bit_and(), bit_or() etc
- Statistics: avg(), stdl(), min() etc

Other
- count(attr, init_value, rate)

Applications
- Ambient Structural Vibration monitoring
  - Continuous structural monitoring and event detection
  - "sample(3 channels, 20 Hz) → send(stream)"
    (http://enl.usc.edu/projects/bridge/)

- Pursuit Evasion Game
  - Pursuer robots estimate the location of evaders and corral them.
  - "sample(0xaa, RSSI) → less(0xaa, 125, 0xbb) → delete_activated_task(0xbb) → send()"
    (http://enl.usc.edu/projects/peg/)