Power of Location & Awareness of Power in Embedded Devices

Managing Power in Dynamic Distributed Environments

Rajesh Gupta
University of California, San Diego

MESL.UCSD.EDU
The Real Culprits

- **Energy awareness**
  - PADS & FORGE PROJECT:
    - Cristiano Pereira, Ravindra Jejurikar, Zhen Ma, Yuvraj Agrawal
    - Vijay Raghunathan, Mani Srivastava, UCLA
    - Jihong Kim, SNU
    - Sandeep Shukla, VT

- **Location awareness**
  - SPATIAL PROGRAMMING
    - R. K. Shyamasundar, TIFR, India

  - DYNAMIC RESOURCE DISCOVERY
    - Jeffrey Namkung
    - Chalermek Intanagonwiwat, Amin Vahdat
Computers with Radios are finding diverse applications and networking over a wide range of environments.
RADIOS ARE EVERYWHERE…

AND SO IS THE NEED FOR POWER
No Moore’s Law For Batteries

- 2-3% growth per year in battery capacity

Source: J. Rabaey, UC Berkeley
Not that we would want it either...

<table>
<thead>
<tr>
<th>Power (Energy) Density</th>
<th>Source of Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Batteries (Zinc-Air)</strong></td>
<td>1050 - 1560 mWh/cm³ (1.4 V)</td>
</tr>
<tr>
<td><strong>Batteries (Lithium ion)</strong></td>
<td>300 mWh/cm³ (3 - 4 V)</td>
</tr>
<tr>
<td><strong>Solar (Outdoors)</strong></td>
<td>15 mW/cm² - direct sun 0.15 mW/cm² - cloudy day.</td>
</tr>
<tr>
<td><strong>Solar (Indoor)</strong></td>
<td>0.006 mW/cm² - my desk 0.57 mW/cm² - 12 in. under a 60W bulb</td>
</tr>
<tr>
<td><strong>Vibrations</strong></td>
<td>0.001 - 0.1 mW/cm³</td>
</tr>
<tr>
<td><strong>Acoustic Noise</strong></td>
<td>3E-6 mW/cm² at 75 Db sound level 9.6E-4 mW/cm² at 100 Db sound level</td>
</tr>
<tr>
<td><strong>Passive Human Powered</strong></td>
<td>1.8 mW (Shoe inserts &gt;&gt; 1 cm²)</td>
</tr>
<tr>
<td><strong>Thermal Conversion</strong></td>
<td>0.0018 mW - 10 deg. C gradient</td>
</tr>
<tr>
<td><strong>Nuclear Reaction</strong></td>
<td>80 mW/cm³ 1E6 mWh/cm³</td>
</tr>
<tr>
<td><strong>Fuel Cells</strong></td>
<td>300 - 500 mW/cm³ ~4000 mWh/cm³</td>
</tr>
</tbody>
</table>

⚠️ Must reduce power, energy consumption.
Low Power Has Been A Design Focus

- Speed power efficiency has indeed gone up
  - 10x / 2.5 years for μPs and DSPs in 1990s
    - between 100 mW/MIP to 1 mW/MIP since 1990
  - IC processes have provided 10x / 8 years since 1965
  - rest from power conscious IC design in recent years
- Another 20X is possible.

<table>
<thead>
<tr>
<th>Processor</th>
<th>MHz</th>
<th>Year</th>
<th>SPECint-95</th>
<th>Watts</th>
</tr>
</thead>
<tbody>
<tr>
<td>P54VRT (Mobile)</td>
<td>150</td>
<td>1996</td>
<td>4.6</td>
<td>3.8</td>
</tr>
<tr>
<td>P55VRT (Mobile MMX)</td>
<td>233</td>
<td>1997</td>
<td>7.1</td>
<td>3.9</td>
</tr>
<tr>
<td>PowerPC 603e</td>
<td>300</td>
<td>1997</td>
<td>7.4</td>
<td>3.5</td>
</tr>
<tr>
<td>PowerPC 604e</td>
<td>350</td>
<td>1997</td>
<td>14.6</td>
<td>8</td>
</tr>
<tr>
<td>PowerPC 740 (G3)</td>
<td>300</td>
<td>1998</td>
<td>12.2</td>
<td>3.4</td>
</tr>
<tr>
<td>PowerPC 750 (G3)</td>
<td>300</td>
<td>1998</td>
<td>14</td>
<td>3.4</td>
</tr>
<tr>
<td>Mobile Celeron</td>
<td>333</td>
<td>1999</td>
<td>13.1</td>
<td>8.6</td>
</tr>
</tbody>
</table>

Source: ISI/USC, DARPA PACC Program
Unfortunately, that is not enough

- There is a bottom line to energy efficiency…
  - Computation cost: 60 pJ/op
  - Minimum thermal energy for communications:
    - 20 nJ/bit @ 1.5 GHz for 100 m
    - 2 nJ/bit @ 1.5 GHz for 10 m
- … but not to the need for energy (or power).
The New Age Computing & Comm.

- New devices with markedly different usage of energy and power (than desktops and laptops)
  - Much wider dynamic range of power demand
    - 6-10X variation in power from sleep to various active modes;
    - Even larger variation in radio power, TX/RX ratio
Need for creative solutions to power management
There are basically two ways to save power...(true for pretty much every thing on-chip)

- **Shutdown** through choice of right system & device states
  - Multiple sleep states
  - Also known as Dynamic Power Management (DPM)
- **Slowdown** through choice of right system & device states
  - Multiple active states
  - Also known as Dynamic Voltage/Frequency Scaling (DVS)
- **DPM + DVS**
  - Choice between amount of slowdown and shutdown
“System Design” for Low Power

- Energy efficiency (has to) cut across all system layers
  - circuit, logic, software, protocols, algorithms, user interface, power supply...
  - Computation versus Communication; Node versus network

- Trade-off between energy consumption & QoS
  - optimize energy metric while meeting “quality” constraint
And thus began our love affair with ‘awareness’

- Knowing an application’s intent one can do a lot of power saving tricks at all levels: architecture, compiler, OS, middleware

- Conversely, if the awareness for power/energy is seeped into all these levels, one can reduce power significantly

- Together they can create a new contract in the computing system!

- Since power is important in radios, things with radios move (or they monitor things that move), location awareness is even more phenomenal for power reduction.
Knowing location can be rewarding...

The sport where you are the search engine.

March Calendar
Every month we release a new desktop calendar in various sizes. You may not have snow in your area (or too much), but here in the Pacific Northwest we have been thinking snow but not getting a whole lot. So this month has a snow theme.
Visit the Goodies page to download the calendar.
posted @ 03/01/2005 8:00 PM Pacific by Jeremy Irish

Site Updates
The Geocaching.com site has recently gone through some cosmetic changes. The existing functionality remains the same.

We hope to add more features to the front page to make it easier to experience the contributions of the geocaching community. This includes tapping into millions of log entries and photos uploaded to the web site during our 4 years of operation.

Thanks to everyone for creating such a great global community and for escaping your entertainment systems and enjoying the great outdoors.
posted 09/05/2005 3:21 PM by Jeremy Irish

Geocaching!
It can also save a lot of power!
In fact, for ad hoc wireless sensor networks

- That is networks with no infrastructural support
  - Compute power, communication (tx, rx) power are interrelated
- The TX power affects
  - Transmission range; Network topology and routing; Network capacity: spatial reuse; Network congestion: interference, near-far; End-to-end delay
  - Tradeoff between power level and network capacity (spatial reuse)
    - Reducing power level reduces average contention at the MAC layer
- Power control protocols
  - Choose power levels that provides for a connected network of wireless nodes (a common power level)
  - Use clustering to group nodes based on geographical proximity
  - Joint clustering, routing and power control
- Many many papers on power efficient routing, data gathering, aggregation…

...all bound by one bad result!
Transport capacity of ad hoc WSN

- Each node can transmit at $W$ bits/sec (bps) over a common wireless channel
- As $n$ increases, so does the multi-hop burden
- As the range $r(n)$ is decreased, each node generates increased burden traffic on other nodes, while the contention is decreased
  - This is a lower limit to range to maintain connectivity

- **Theorem: Best case capacity**
  - Optimally located nodes, destinations, demands for OD-pairs
  - Optimal spatio-temporal scheduling, routes, ranges for each transmission
  - Each node obtains $\Theta\left(\frac{1}{\sqrt{n}}\right)$ bits/sec

- **Theorem: Random networks**
  - Randomly located nodes and destinations
  - Each node chooses same range $r$
  - Each node obtains $\Theta\left(\frac{1}{\sqrt{n \log n}}\right)$ bits/sec

- Only $\sqrt{\log n}$ factor difference between common and different power levels

Energy is lost listening to useless traffic. Many attempts to minimize this listening...
Hence this talk: outline

- What does it mean to be (energy) aware?
  - How is this achieved?
  - An experiment

- Understanding location
  - Does it have anything to do with ‘context’ awareness?
  - Programming with location: spatial programming
  - Infrastructure for location: dynamic resource discovery
Outline:
Bringing energy awareness in application, OS and Middleware

A Application
B OS
C Middleware
What does it mean to be ‘aware’?

- That the application and the services know about energy, power
  - File system, memory management, process scheduling
  - Make each of them energy aware
- How does one make software to be “aware”?
  - Use “reflectivity” in software to build adaptive software
  - Ability to reason about and act upon itself (OS, MW)
Application reflection

- Enables programs to analyze, reason and modify its representation.
- Also enables the run time system to adapt to applications’ behavior by monitoring the meta data.
- Meta data represents resource demands dynamic behavior of the program carrying it.
  - Resources: Memory (R/W, Cache), Processor (IPC)
- Enables energy performance tuning by exploring resource demand variations throughout programs’ execution.
  - Example: Profile of application over memory banks
  - Vary frequency of processor based on IPC demand
Example: Modify workload

An Unmodified Application (UM)

CPU

Disk

Transformed Application:
Better clustering & opportunity for latency Elimination by early re-activation

[Benini 2000]
Example: Rambus DRAM (RDRAM)

- High bandwidth (>1.5 GB/sec), 3.3 volts, 2.5 ns cycle time
- Each RDRAM module can be activated/deactivated independently
- Read/write can occur only in the active mode
- Three low-power operating modes:
  - Standby, Nap, Power-Down

![Diagram showing power consumption in different modes](image)
Approach

- Characterize application offline
  - Divide an application into phases of execution
    - A group of program intervals executing similar code
  - Each phase has similar demand on resources
    - Similar code, similar resource demands (memory, IPC)
- Annotate source code
  - Phase signatures
- Enable OS (and hardware) to recognize signature
  - Smart hardware and/or online learning techniques
- Dynamically tune the power manager
  - As application moves from one phase to another.
Understanding application behavior

Basic Block Frequency Vectors
- A BBFV contains the number of instructions executed per basic block for a given program interval.

Loop Branches Frequency Vectors (LBV)
- Keep track of loop branches, instead of BBs

BBV have been used in performance tuning. We rely on (compressed) LBV

Micro-architectural support through performance counters.
Offline analysis of application

- A data structure with the summary of the information of interest for each phase is attached to the program
  - Fixed location for the program metadata
  - OS support to access metadata.
- Also a signature of each phase is attached to the program.
  - No. of times the loops of the program are executed in the particular phase
Runtime Analysis

- **Challenge:**
  - Detect in which phase the program is running
- **Learning, signature construction, partial matches**
  - Match the signature created offline with a online signature using Manhattan distance.
    - If distance < threshold a match is found.
    - Threshold tells how similar two intervals of execution are.
  - The same technique used for splitting the program in phases offline is applied online but using partial signatures for matching with the offline computed signatures.
Matching signature at runtime

- Use performance counters:
  - Can be programmed to generate an interrupt on specified counts
- ISR provides matching with the meta data and mode changes
  - Every S*10,000 loop branches try a match
  - Phase matching can also be done in hardware
- Notify power manager to trigger proper action
Adaptation for Memory Behavior

- Number of engineering optimizations
  - Frequency of adaptation
  - Granularity of analysis (phase granularity)
  - Tradeoff against cost of adaptation.
Results – Normalized to NAP

Approx. 350K instructions for every 10,000 loop branch instructions / 2 KB of metadata
Results - overheads

- Approx. 350K instructions for every 10,000 loop branch instructions
- Number of instructions executed by the match algorithm at every 10,000 loop branches to match a partial signature (500 instructions per phase)

<table>
<thead>
<tr>
<th># of phases</th>
<th># instructions</th>
<th>overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2,580</td>
<td>0.7%</td>
</tr>
<tr>
<td>10</td>
<td>4,500</td>
<td>1%</td>
</tr>
<tr>
<td>20</td>
<td>8,280</td>
<td>2%</td>
</tr>
<tr>
<td>30</td>
<td>12,060</td>
<td>3%</td>
</tr>
</tbody>
</table>

- Size overhead. 4 bytes per inter arrival estimate per bank / phase. 4 x 16 x 10 = 640 bytes assuming 16 banks and 10 phases.
- The signatures take 1280 bytes for 10 phases. Total of 2KB of meta data
Outline:
Bringing energy awareness in application, OS and Middleware

A Application
B OS
C Middleware
Power Aware OS and Middleware

- Make it adaptive to respond to application requirements
  - and to dynamically smooth the imbalances between demand and availability of energy resource
- Use reflectivity enable dynamic scaling of data, choice of algorithms or parameters
  - (e.g., transcoding algorithms, compression parameters)
- Use brokerage service to negotiate quality demands
Enable Power/Energy Dialogue

- Application can tell OS task information (type, deadlines, WCET estimates)
- OS can update these estimates based on runtime conditions.
- To do this, we need
  - API: Provide ways by which Application, OS and Hardware can exchange energy/power and performance related information efficiently.
  - Middleware: Facilitate a continuous dialogue / adaptation between OS / Applications.
  - HAL: Facilitate the implementation of power aware OS services by providing a software interface to low power devices
Power Aware Parts

- **PA-API (Power Aware API)**
  - interfaces applications and OS making the power aware OS services available to the application writer.

- **PA-OSL (Power Aware Operating System Layer)**
  - implements modified OS services and active components such as a DPM manager.

- **PA-HAL (Power Aware Hardware Abstraction Layer)**
  - interfaces OS and Hardware making the power control knobs available to the OS programmer.
OS Services

- **PA-API - Power aware function calls available to the application writer.**
  - Some functions of this layer are specific to certain scheduling techniques.

- **PA-Middleware - Power aware services**
  - Implemented on the top of the OS (power management threads, data handling, etc...).

- **POSIX - Standard interface for OS system calls.**
  - This isolates PA-API and PA-Middleware from OS.

- **PA-OSL - Power aware OS layer.**
  - Calls related to modified OS services should go through this level. Also isolates OS from PA-API and PA-Middleware.

- **PA-HAL - Power Aware Hardware Abstraction Layer.**
  - Isolates OS from underlying power aware hardware.

- **Modified OS services**
  - Implementation / modification of OS services in a power related fashion. Ex: scheduler, memory manager, I/O, etc.
## Layer Functionality

<table>
<thead>
<tr>
<th>Layer</th>
<th>Function name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PA-API</strong></td>
<td>paapi_dvs_create_thread_type(), paapi_dvs_create_thread_instance()</td>
</tr>
<tr>
<td></td>
<td>paapi_dvs_app_started(), paapi_dvs_get_time_prediction()</td>
</tr>
<tr>
<td></td>
<td>paapi_dvs_set_time_prediction(), paapi_dvs_app_done(), paapi_dvs_set_adaptive_param()</td>
</tr>
<tr>
<td></td>
<td>paapi_dvs_set_policy(), paapi_dpm_register_device()</td>
</tr>
<tr>
<td><strong>PA-OSL</strong></td>
<td>paosl_dvs_create_task_type_entry(), paosl_dvs_create_task_instance_entry(), paosl_dvs_killer_thread(), paosl_dvs_killer_thread_alarm_handler(), paosl_dpm_register_device(), paosl_dpm_deamon()</td>
</tr>
<tr>
<td><strong>PA-HAL</strong></td>
<td>pahal_dvs_initialize_processor_pm(), pahal_dvs_get_frequency_levels_info()</td>
</tr>
<tr>
<td></td>
<td>pahal_dvs_get_current_frequency(), pahal_dvs_set_frequency_and_voltage()</td>
</tr>
<tr>
<td></td>
<td>pahal_dvs_pre_set_frequency_and_voltage(), pahal_dvs_post_set_frequency_and_voltage()</td>
</tr>
<tr>
<td></td>
<td>pahal_dvs_get_lowpower_states_info(), pahal_dvs_set_lowpower_state()</td>
</tr>
<tr>
<td></td>
<td>pahal_dpm_device_check_activity(), pahal_dpm_device_pre_switch_state()</td>
</tr>
<tr>
<td></td>
<td>pahal_dpm_device_switch_state(), pahal_dpm_device_post_switch_state()</td>
</tr>
<tr>
<td></td>
<td>pahal_dpm_device_get_info(), pahal_dpm_device_get_curr_state()</td>
</tr>
<tr>
<td></td>
<td>pahal_battery_get_info()</td>
</tr>
</tbody>
</table>
DVS Related Functions

paapi_dvs_create_thread_type(), paapi_dvs_create_thread_instance()
creates type and instance of a task respectively

paapi_dvs_app_started(), paapi_dvs_app_done()
delimits execution of useful work in a thread. Tell the OS whether the task
has finished execution or not.

paapi_dvs_get_time_prediction(), paapi_dvs_set_time_prediction()
get current execution time prediction for a given thread

paapi_dvs_set_adaptive_param()
set the parameters of the adaptive policy (it will be described later) for a
given task.

paapi_dvs_set_policy()
chooses the policy to be using for DVS
DVS Related Functions (contd.)

paosl_dvs_create_task_type_entry(), ...
create a type and an instance of a thread in the kernel internal tables of
type and instance respectively

paosl_dvs_killer_thread()
kills a thread that missed a deadline

pahal_dvs_initialize_processor_pm()
initialize structures for processor power management

pahal_dvs_get_current_frequency(),
pahal_dvs_set_frequency_and_voltage()  
pahal_dvs_pre_set_frequency_and_voltage(),
pahal_dvs_get_frequency_levels_info()  
pahal_dvs_post_set_frequency_and_voltage()  
functions to switch processor among possible frequencies levels

pahal_dvs_get_lowpower_states_info(),
pahal_dvs_set_lowpower_state()  
functions to switch processor among low power states
DPM Functions

- **paapi_dpm_register_device()**
  - just register the device to be power managed

- **paosl_dpm_deamon()**
  - implements the actual policy for a specific device. This daemon uses PA-HAL functions to decide on how to switch devices among all possible states.

- **pahal_dpm_device_switch_state()**
  - switch device’s state

- **pahal_dpm_device_check_activity()**
  - check whether the device has been idle and for how long. This function needs support from the device driver.

- **pahal_dpm_device_get_info(), pahal_dpm_device_get_curr_state()**
  - gets information about the device and about its current state respectively

- **Others**
  - functions for helping implementing power policies. For example:
    - **pahal_battery_get_info()** – gets battery status
Prototype Implementation

- Platforms
  - eCOS RTOS:
    - open source, Object oriented and highly configurable RTOS (by means of scripting language)
  - Hardware platforms we are currently working with:
    - Linux-synthetic (emulation of eCos over Linux - debugging purposes only)
    - Compaq iPaq Pocket PC, Accelent IDP
Using Power Aware OS

- The scheduler adapts frequency according to the real time parameters passed in as parameter on the thread type.
- The frequency is adjusted by means of slowdown factors (a factor can also speed up the processor if it is > 1).

```c
void main()
{
    mpeg_decoding_t =
    paapi_dvs_create_thread_type(100,30,100,hard);

    paapi_dvs_set_policy(SHUTDOWN | STATIC
    DYNAMIC | ADAPTIVE);

    Selects the DVS policy for all threads
    paapi_dvs_create_thread_instance(
        mpeg_decoding_t, mpeg_decode_thread);
}
...```

```c
void mpeg_decode_thread()
{
    for (; ;) {
        paapi_dvs_app_started();
        /* original code */
        mpeg_frame_decode()
        paapi_dvs_app_done();
    }
}
```
void threshold_policy_deamon(device_info_t dev) {
    unsigned idleness;
    for (;;) {
        /* check for how long the device has been idle */
        idleness = dev->check_activity(dev);
        /* if idle for longer than the threshold switch to next state */
        if ( idleness > dev->check_state()-&gt;threshold ) {
            dev-&gt;check_state()-&gt;switch_state(dev, dev-&gt;check_state,
                dev-&gt;check_state()-&gt;next); }
        /* sleep until next period for checking idleness */
        sleep(dev-&gt;policy_info-&gt;th_policy-&gt;period); }
}
### OS-directed DVS Results

#### Energy Consumption for each scheme

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Taskset A</th>
<th>Taskset B</th>
<th>Taskset C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>0.80</td>
<td>0.90</td>
<td>0.95</td>
</tr>
<tr>
<td>Shut/Static</td>
<td>1.00</td>
<td>1.10</td>
<td>1.20</td>
</tr>
<tr>
<td>Shut/Static/Dyn/Adap (0.95)</td>
<td>0.80</td>
<td>0.90</td>
<td>1.00</td>
</tr>
<tr>
<td>Shut/Static/Dyn/Adap (0.90)</td>
<td>0.80</td>
<td>0.90</td>
<td>1.00</td>
</tr>
<tr>
<td>Shut/Static/Dyn/Adap (0.85)</td>
<td>0.80</td>
<td>0.90</td>
<td>1.00</td>
</tr>
<tr>
<td>Shut/Static/Dyn/Adap (0.80)</td>
<td>0.80</td>
<td>0.90</td>
<td>1.00</td>
</tr>
<tr>
<td>Shut/Static/Dyn/Adap (0.75)</td>
<td>0.80</td>
<td>0.90</td>
<td>1.00</td>
</tr>
</tbody>
</table>

### Taskset Details

<table>
<thead>
<tr>
<th>Task</th>
<th>Application</th>
<th>WCET (us)</th>
<th>Std Dev (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>MPEG2 (wg_sdo_1.mpg)</td>
<td>30700</td>
<td>3100</td>
</tr>
<tr>
<td>T2</td>
<td>MPEG2 (wg_cs_1.mpg)</td>
<td>26300</td>
<td>2100</td>
</tr>
<tr>
<td>T3</td>
<td>ADPCM</td>
<td>9300</td>
<td>3300</td>
</tr>
<tr>
<td>T4</td>
<td>FFT</td>
<td>15900</td>
<td>0</td>
</tr>
<tr>
<td>T5</td>
<td>FFT (gaussian distribution)</td>
<td>13600</td>
<td>800</td>
</tr>
</tbody>
</table>
Energy Aware Middleware in FORGE

[Jointly with Nalini Venkatsubramaniam, UC Irvine.]

- Use reflective middleware services to continuously monitor and keep track of application needs
  - Use a rule base and director service to carry out its functions.

- Example:
  - Multimedia streaming from a server to a set of mobile “nodes”
  - Use a proxy server to adapt video stream to specific nodes

- Node device: sends device info to proxy, connects video stream and network parameters to lower layers
- Proxy: admission control, real-time transcoding, network traffic regulation.
Experiments using iPAQs demonstrate viability of dynamic adjustments of video quality

- based on changing battery conditions and client devices.
- [Mohapatra et al, ACM MM03]
Example

- **T₀**: 3 users (1, 2, 3)
- **T₁**: user 4 joins
  - System readjusts quality levels
- **T₂**: residual power on user 1 decreases
  - Quality level is decreased for 1
- **T₃**: user 5 joins
  - All levels go down to accommodate 5
- **T₄**: user 3 finishes streaming
  - Quality levels rise back
- **T₅**: user 1 finishes streaming
  - Even higher levels

Radu Cornea, UCI
Hence this talk: outline

- What does it mean to be (energy) aware?
  - How is this achieved?
  - An experiment

- Understanding location
  - Does it have anything to do with ‘context’ awareness?
  - Programming with location: spatial programming
  - Infrastructure for location: dynamic resource discovery
Understanding Location

- Enabled by technology advances
  - Outdoor, indoor, absolute, relative, assertive, …
  - Supported by ubicomm community (Location Stack)
- Programming with location
  - Spatial programming: how to use the location information
- Infrastructure for location
  - Dynamic resource discovery and binding
- Using location-awareness to minimize listening, duty cycle.

<table>
<thead>
<tr>
<th>Intentions</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contextual Fusion</th>
<th>Arrangements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Handling</td>
<td></td>
</tr>
<tr>
<td>Layers (Non-Location)</td>
<td>Fusion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measurements</th>
<th>Sensors</th>
</tr>
</thead>
</table>
Isn’t this another ‘context’?

- Yes, it is just another context among:
  - **Computing context**: network connectivity, communication cost, communication bandwidth, nearby resource
  - **User context**: user profile, location, social situation
  - **Physical context**: lighting, noise, traffic condition, temperature
  - **Time Context**: time of a day, week, month, season of a year

- But really, the application experience has shown [Abowd]
  - Few contexts other than location have been used
  - Context history is rarely used
  - Any reliance on the user to explicitly provide context information, proves to be obtrusive and inconvenient.
Location awareness

- Programming with spatial references
- Infrastructural support for location
  - Transparent location updates
  - Monitor resources in their environment
    - Virtual resource naming, that is, resources referred to based on their expected location
    - Reactive to changes in the resource availability
    - Control resource usage: access timeouts.
Spatial Programming

- Programming abstraction using Reactive Mobile Communicating Processes (RMCPs)
  - RMCPs define observables among processes
  - Communication measures, termination, clock, location, resources
- Our conceptual view of the distributed system as a federated wireless network
- Location information can be geometric, symbolic, logical, key-value etc.
RMCP Briefly

- Network of reactive processes that communicate through asynchronous communication mechanisms
- Remote communication is suspensive and observable
- Communication is observable along with location information
- Implemented on top of Esterel
RMCP

- A program, P, is a set of Modules, M, with prefixed identifiers
- A module, M, consists of a reactive program S and set of tasks, T
  - Tasks may return either signals or messages
  - [M] is encapsulated in a black box with only external interfaces as observables
- Reactive commands are extended by a set of Hook commands that capture interaction between modules
  - send(m) to Mi = send message m to module Mi (asynchronous send, nonblocking)
  - receive(x) from Mi = receive a message and assign it to variable x from module Mi. (blocking)
  - location commands as match_ontology operators
    - Match_loc(a1, b1), match_range(a1, b1), match_containment(a1, b1), check_comm( location A1), find_loc(a1)
  - If [match_ontology] -> S fi
    - Basic location matching, S or Skip
  - Classical guarded selection commands with guards as:
    - Receive from M; Boolean location commands; Boolean load; time delay
\[ P := A_1 : M_1 \quad \text{//} \quad A_n : M_n \]
\[ M := S, T \quad | \quad [M] \quad (* [M] \text{ denotes that one can observe only the external interface *}) \]
\[ (* \text{This is the classical hiding operator *}) \]
\[ T := \text{task Task-name} (x_1, \ldots, x_n) \]
\[ | \quad \text{task Task name} (x_1, \ldots, x_n) \quad \text{return} \quad (y_1; \ldots; y_n) \quad | \quad T, T \]
\[ S := \quad H \quad | \quad R \]
\[ H := \text{Comm} \quad | \quad \text{Location} \quad | \quad \text{Load} \quad | \quad \text{Select} \]
\[ \text{Comm} := \text{send} \quad (m) \quad \text{to} \quad M_i \quad | \quad \text{receive} \quad (x) \quad \text{from} \quad M_i \]
\[ \text{Location} := \quad \text{match.loc}(a_1, b_1) \quad (* \text{coordinate matching of } a_1 \text{ and } b_1 *) \]
\[ | \quad \text{match.range}(a_1, b_1) \quad (* \text{Is } b_1 \text{ within the range of } a_1? *) \]
\[ | \quad \text{match.containment}(a_1, b_1) \quad (* \text{Is } b_1 \text{ contained within } a_1? *) \]
\[ | \quad \text{match.source.within}(\text{source.name}, a_1) \quad (* \text{Is the resource source.name within } a_1? *) \]
\[ | \quad \text{check.comm(location } a_1) \quad (* \text{check the communication status at location } a_1 *) \]
\[ (* \text{The above } \text{match operations are grouped under the class Match.ontology *}) \]
\[ | \quad \text{find.acoustic}(self, a_1) \quad (* \text{estimates distance with } a_1 \text{ through acoustic signals *}) \]
\[ | \quad \text{find.loc}(a_1) \quad (* \text{find coordinates of } a_1 *) \]
\[ | \quad \text{find.range}(a_1, b_1) \quad (* \text{find the distance of } b_1 \text{ from } a_1 *) \]
\[ | \quad \text{findall.range}(a_1, \text{source.type}, L) \quad (* \text{find all the resources of source.type within a range of } L \text{ from } a_1 *) \]
\[ \text{Load} := \text{bandwidth}(\text{self}, M_2) \quad (* \text{bandwidth of communication from } \text{self} \text{ to } M_1 * ) \]
\[ \text{Selection} := \text{if} \quad \text{[match.ontology]} \quad \rightarrow \quad S \]
\[ \quad | \quad \text{else} \quad \text{skip *}) \]
\[ \text{if} \]
\[ \text{if} \]
\[ \quad | \quad b_1; g_1 \quad \rightarrow \quad S_1 \]
\[ \quad | \quad b_2; g_2 \quad \rightarrow \quad S_2 \]
\[ \quad | \quad \ldots \]
\[ \quad | \quad b_m; g_n \quad \rightarrow \quad S_n \]
\[ \text{fi} \]
\[ g := \quad \text{receive from } M \quad | \quad \text{Location} \quad | \quad \text{R-load} \quad | \quad \text{Time} \]
Spatial Programming and Its Use in Power Management

- Consider purely spatial phenomena such as hidden terminal and exposed terminal problems

**Hidden Terminal Problem**

- A talks to B
- C senses the channel, but cannot hear A due to out of range
- C talks to B and collides with transmission of A at B
- Collisions cause bw/energy wastage
Spatial Programming and Its Use in Power Management

- Consider purely spatial phenomena such as hidden terminal and exposed terminal problems

Exposed Terminal Problem

- B talks to A
- C wants to talk to D
- C senses channel and finds it to be busy
- C stays quiet when it could have transmitted.

emplace Underutilization of channel
Hidden Terminal

- The radios that cannot talk to each other can interfere with each other.

Possible solutions

- **PHY solutions**
  - Reduce carrier range, so listen from farther radios than they can interfere with (causes “near-far” problem)
  - Process each interfering signal to determine which one is destined at the receiver (cognitive radios)

- **MAC layer solutions**
MAC Layer Solutions

- Basic idea
  - Hidden terminal problem
    - absence of carrier does not mean an idle medium
  - Exposed terminal problem
    - presence of carrier does not mean a busy medium
  - So, get rid of CS in CSMA/CA = MA/CA or MACA
    - Intelligent use of RTS/CTS
  - The beauty is that availability of location information at the TX, RX makes it possible to program such solutions in RMCP.
    - [Gupta, Shyamasundar, ASIAN04] for examples.
Clever Use of RTS, CTS

- Dialogue between sender and receiver
  - Sender sends RTS
  - Receiver (if free) send CTS
  - Sender sends DATA
  - Collision avoidance through (intelligent) RTS/CTS
    - When you overhear an RTS/CTS address to another station, inhibit your own TX “long enough”

- Collisions are still possible
  - RTS packets can collide but not as bad as DATA collision.

A sends RTS
B sends CTS
C overhears CTS
C inhibits its own TX
A successfully sends DATA to B

B sends RTS to A (overheard by C)
A sends CTS to B
C cannot hear A’s CTS
C assumes A is either down or out of range
C does not inhibit its own TX to D
Computers with radios present challenges in energy efficient processing.

- Location information is key to this efficiency improvement.

System architectures are changing to enable high hardware integration, and yet allow for reasonable software development across multiple heterogenous processors.

Yet, not fast enough. With the coming multiple processor cores on chip, the interaction of application behavior with core utilization will be even more critical to maintain energy profile.

Novel services – such as location awareness – may make it possible to achieve extreme duty cycles.

- Instrumented wide-area spaces
- Personal area spaces
- Internet end-points
- In-body, in-cell, in-vitro spaces