Scaling Down in One Slide

- Target devices (roughly)
  - Small form factor
  - Battery operated
  - Wireless communications

- Strategies
  - Use proxies
  - Avoid communications
The PalmPilot in the Late 90s (Think October 1997)

- 16 MHz Motorola DragonBall 68328
- 1-2 MB of SRAM
  - 32 KB Heap, 32 KB relative jump maximum
- Grayscale display (1-2 bits per pixel)
- Wireless modem
  - Metricom Ricochet at 19200 bps
Top Gun Wingman
A Web Browser for the PalmPilot

- Implemented as a proxy service
  - Communication between proxy and servers
    - HTTP
    - HTML, GIF, JPG
  - Communication between proxy and PalmPilots
    - Customized protocol
      - Application-level framing
    - Pre-arranged text objects, native bit maps
Introducing TACC

- Cluster-based platform for proxy-based services
  - Transformation
    - Distillation, filtering, format conversion
  - Aggregation
    - Collecting and collating data from various sources
  - Caching
    - Both original and transformed content
  - Customization
    - Service tailored for user’s needs or device characteristics
TACC Architecture

- **Front ends**
  - Accept requests, look up users, enqueue tasks

- **Workers**
  - Process tasks
  - Are atomic and restartable

- **Manager**
  - Balances load across workers, starts and reaps workers

- **User profile database**
  - Persistently stores user state
Top Gun Wingman on TACC

- HTML and image processors
  - Convert to pre-arranged text objects, native bit maps
- Aggregators
  - Perform queries for users
- Zip, PalmOS, and Doc processors
  - Expand zip archives
  - But pass through PalmOS databases
    - Including AportisDoc e-books
Top Gun Wingman Performance

Wingman, Palmscape, HandWeb at 57600 bps

Wingman and Netscape at 19200 and 57600 bps
Advantages of the Proxy Approach

- Performance
- Isolation of complexity
  - (Mostly) client-independent back end
  - Transparent functionality upgrades
  - Backward compatibility
  - Support for standard servers and content
- Middleware availability
  - Reusable platform for available and robust services
- Any disadvantages?
The Berkeley Mote (circa 2002)

- Assembled from off-the-shelf components
- 4Mhz, 8bit Atmel CPU
  - 4 K RAM, 128 KB ROM
- 917 MHz RFM radio
  - 50 kbs
- 512 K EEPROM
- Optional sensors
  - Light, temperature, magnetic field, acceleration, sound, power
- Serial bus
Energy Is Precious

- Transmitting 1 bit $\approx$ executing 800 instructions
- In 2000 (with a less powerful CPU)
  - CPU: 5 mA
  - RFM Rx: 4.5 mA
  - RFM Tx: 7 mA
- Also in 2000, 1 battery pack lasts for
  - 35 hours at peak load
  - 1 year at minimum load
Typical Uses for Motes

- Monitor sensor output of collection of motes
  - Building integrity during earthquakes
  - Biological habitat monitoring
  - Temperature and power usage in data centers

- Common requirements
  - Extract data from network
  - Summarize data
TAG: Tiny Aggregation

- Provides declarative interface to data collection
- Distributes and executes queries across network
  - Power-efficient
- Returns *stream* of results
  - Unlike table-based queries for databases
Ad-Hoc Routing for Motes

**Requirements**
- Deliver queries to all nodes
- Provide one or more routes back to root of network

**Tree-based scheme**
- Repeated broadcasts, starting at root mote
- Each mote assigns itself
  - A level representing distance from root
  - A parent, that is, the sender of broadcast
- Periodic repetition for topology maintenance
  - Account for mobility, loss (e.g., battery drainage)
TAG Queries

- SELECT \{agg(expr), attrs\} FROM sensors
  WHERE \{selpreds\}
  GROUP BY \{attrs\}
  HAVING \{havingPreds\}
  EPOCH DURATION \{i\}

- SELECT AVG(Volume), room FROM sensors
  WHERE floor = 6
  GROUP BY room
  HAVING AVG(volume) > threshold
  EPOCH DURATION 30s
Aggregates

- Structure: $agg$ implemented by three functions
  - $f$: merge partial state records
  - $i$: initialize state record for single value
  - $e$: evaluate aggregate from partial state record

- Taxonomy
  - Duplicate sensitivity
  - Exemplary or summary
  - Monotonic
  - Partial state
    - Distributive, algebraic, holistic, unique, content-sensitive
TAG in Action

- **Distribution**
  - Flood network with query (using tree-based routing scheme)
  - Specify interval for receiving results
    - Somewhat less than parent’s

- **Periodic collection**
  - Listen for partial state records
  - Perform sensing & processing
  - Pass up to parent
Grouping

- Each mote is exactly in one group
  - Groups based on one or more attributes
- State maintained separately for each group
  - Records tagged with group
  - Motes merge and forward records for other groups
  - Motes update records for their own group
- Optimization for some monotonic aggregates
  - Inject HAVING clause into network
  - Suppress storage and transmission for unsatisfactory groups
More Techniques

- Snooping to eliminate unnecessary aggregates
  - Works for monotonic, exemplary aggregates
- Hypothesis testing to further suppress aggregates
  - Works for monotonic, exemplary aggregates
  - Also works for summary aggregates
- Caching to overcome losses
  - Works for all aggregates, but smears values
- Propagating to many parents to overcome losses
  - Works for duplicate insensitive aggregates
  - Also works for linearly decomposable aggregates
Performance

Simulation for a realistic environment

Measured count for prototype implementation
Discussion

- High-level interface to TAG
  - Is easier to program
  - Allows for optimizations in network
    - Partial computation
    - Compensation for lossy communications
  - However, effectiveness depends on properties of aggregate
    - Remember the taxonomy?

- Are these techniques limited to sensor networks?