

Resource-Aware Programming in the **Pixie Operating System**

Sravanthi Kota Venkata

New Challenges and Applications

Sensor networks increasingly used for data intensive applications:

- Structural monitoring: vibrations, seismic response
- Geophysical monitoring: earthquakes, fault zones, volcanoes
- Biomedical monitoring: EKG, EEG, movement, physical activity

Need to squeeze the most out of very limited resources

Adapt to change in power and resource budget

Existing sensor node OS's don't provide much help in terms of resource management

Pixie

New operating system for sensor networks that supports:

- A resource aware programming model based on resource tickets and brokers

Intended to make it easy to write adaptive applications

- With such limited sensor node resources, application must contend with varying resource conditions!

Fundamental challenge: How to enable resource awareness without placing undue burden on the programmer?

Outline

- Pixie OS Architecture
- Programming model: Resource tickets and brokers
- Evaluation: Adaptation to bandwidth and energy variations
- Related work and conclusions

Pixie OS

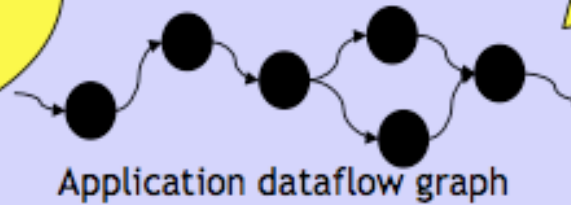
Pixie OS design goals

- Resources as first-class entity
- Direct application knowledge of available resources
- Fine-grained control over resource consumption

The Pixie OS Architecture

Brokers help application manage resource requests and adapt to changing conditions

Applications structured as a dataflow graph of **stages**



Bandwidth broker

Storage broker

Energy broker

Allocators control access to resources and estimate availability

Pixie OS

CPU allocator

Radio allocator

Sensor allocator

Flash allocator

Energy allocator

CPU

radio

sensors

flash

energy

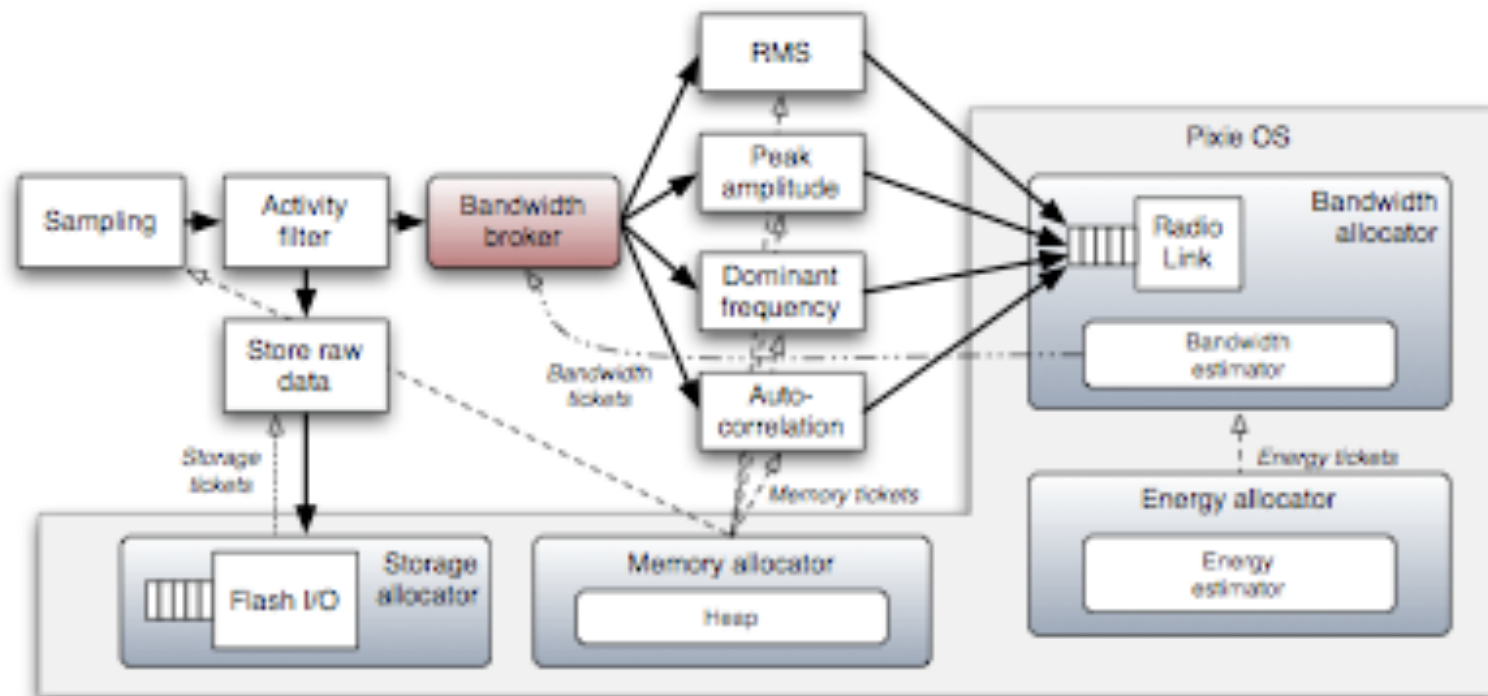
Pixie OS

Key design features

- Dataflow programming model: Application structured as a graph of stages
- Resource tickets: Primitive to represent fine-grained resource reservations
- Resource brokers: Implement policies to manage resources on behalf of the app

Prototype implemented in NesC Supports limited backwards compatibility with TinyOS

Dataflow programming model



- An example Pixie application for limb monitoring using wearable sensors

Resource tickets

Core abstraction for resource management in Pixie

- Ticket $\langle R, c, t \rangle$ represents right to consume c units of resource R until the expiry time t .

Basic resource ticket operations:

- **Redeem**: Performed by app when ticket is used
- **Forfeit**: Application gives up the ticket
- **Revoke**: Resource allocator reclaims resources (can happen prior to expiry time!)
- **Join**: Combine two tickets into one
- **Split**: Split a ticket into two separate tickets

Granularity depends on resource variability

- e.g., Bandwidth tickets would have a short expiry time; storage tickets need not expire.

Resource allocator

Each physical resource has a corresponding allocator

- Allocators for energy, memory, flash storage, and radio
- Implicit allocator for CPU – the scheduler

Allocators estimate available resource, and allocate tickets.

- No policy: Always allocate ticket if the resources are available
- Policies handled by higher-level abstractions (e.g., brokers)

Allocators also enforce ticket redemption

- e.g., To transmit a packet must have corresponding bandwidth ticket

Resource estimation

Storage and memory are straightforward

- Allocators track total flash/memory consumption

Bandwidth: Radio link estimates packet transmission delay

- Measure total time for packet transmission plus ARQ (if used)

Resource Brokers

Resource tickets are intended to be low-level and fine-grained

- Very flexible and powerful, but sometimes difficult to use

To simplify app design, resource brokers used

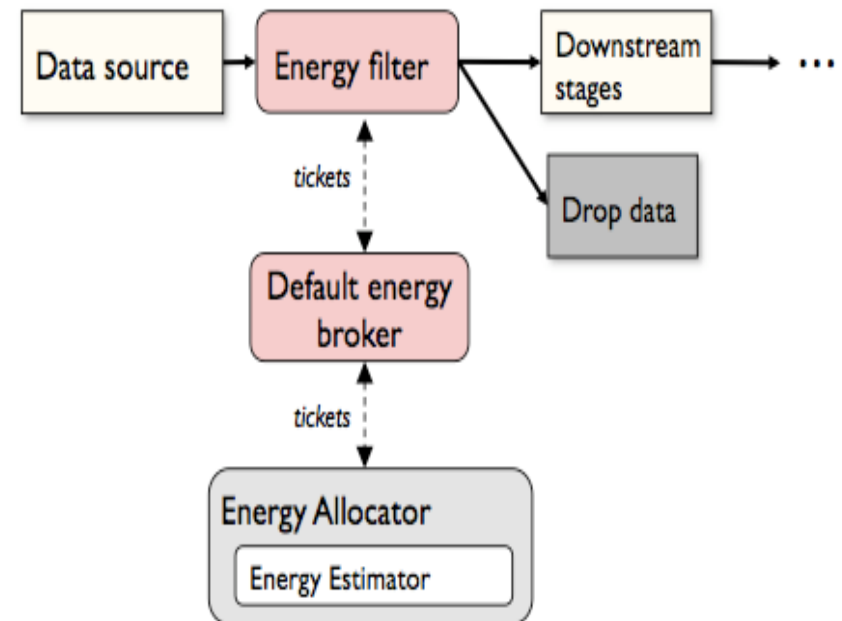
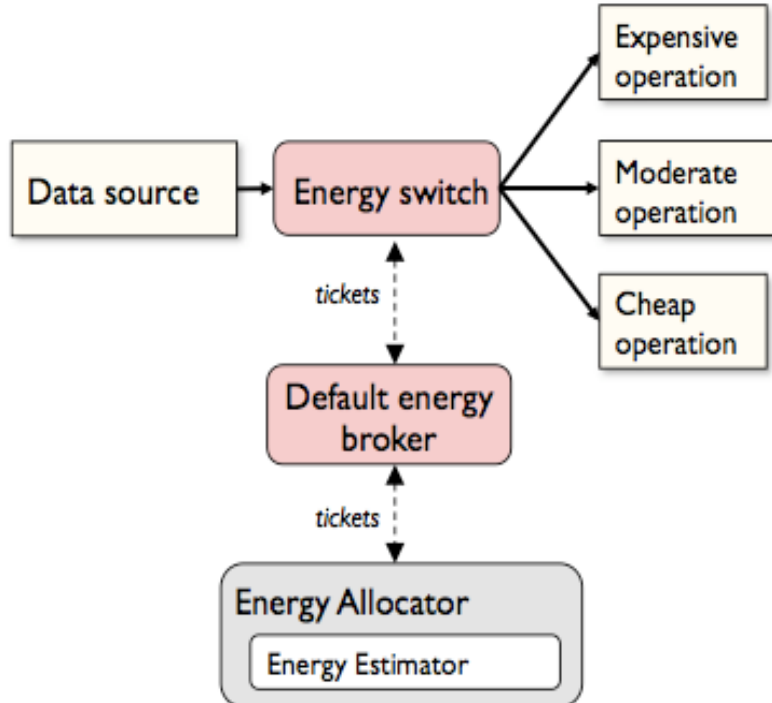
- Brokers request and manage resource tickets on behalf of the application
- Each broker implements some policy and provides a high-level API to the app

Brokers are specialized stages

- Can interpose directly on the application's dataflow path
- Can inspect, redirect, discard, or refactor data

Pixie Brokers

- Bandwidth broker
 - In terms of pps (packets per second)
- Energy broker
 - App stages specify *energy quanta and priority*
 - Broker delivers energy tickets to stages according to priority and energy schedule



Programming Benefits

- Dataflow programming model maps well onto application structure
- Tickets offer a great deal of flexibility and power
- Brokers decouple resource management policies from mechanisms
- Resource-awareness is pervasive in the programming model

Evaluation Summary

Microbenchmarks

- Pixie's overhead is low compared to TinyOS

Accuracy of resource estimation

- Pixie accurately estimates available resources

Bandwidth adaptivity

- Pixie enables rapid adaptation to changing radio link characteristics

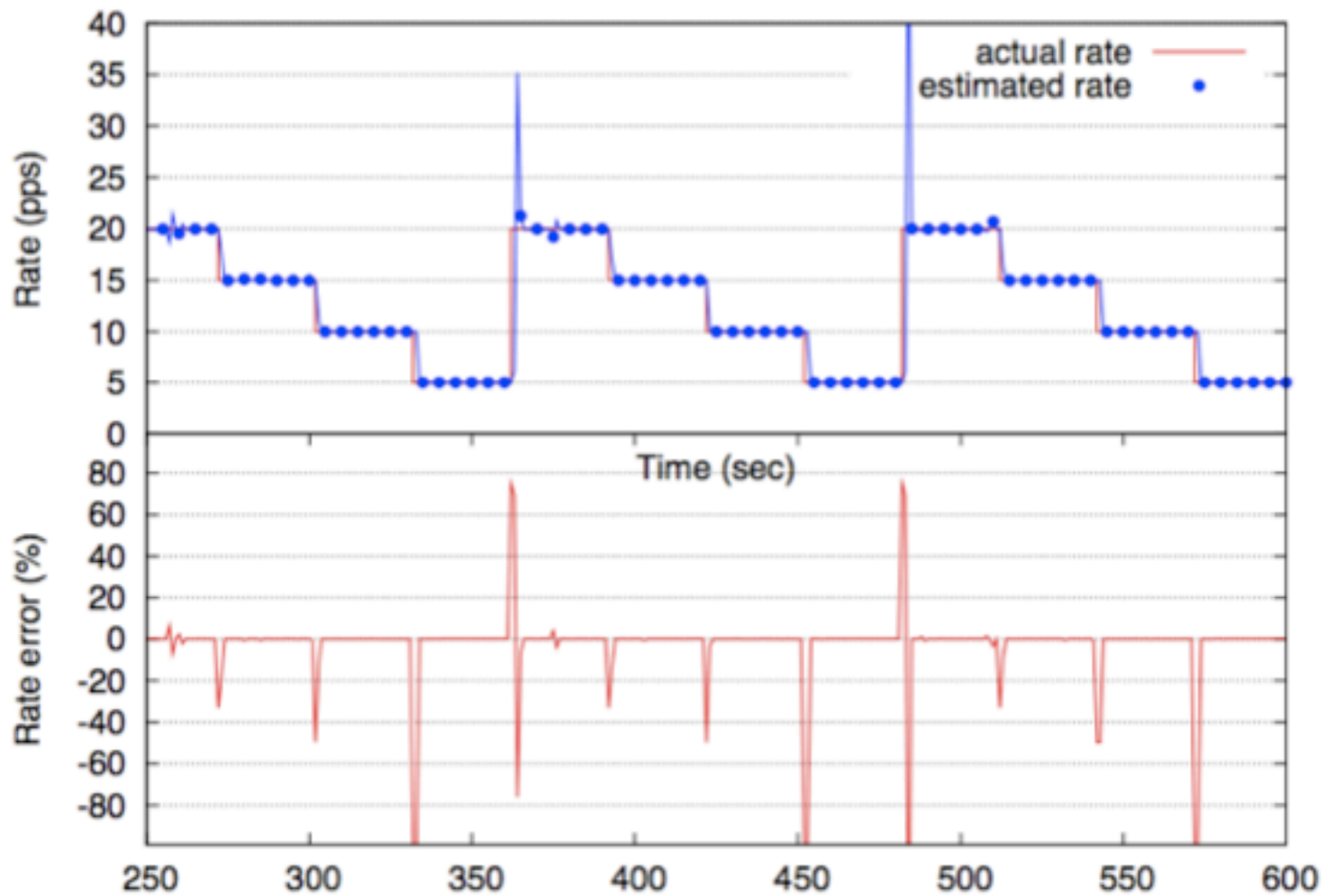
Energy adaptivity

- Pixie applications degrade gracefully as lifetime target is increased

Combined bandwidth and energy adaptivity

- Can jointly optimize app behavior based on multiple resource constraints

Bandwidth Estimation Accuracy



Conclusions

Pixie is a new OS to facilitate resource aware programming:

- Staged dataflow programming model to simplify application design
- Resource tickets expose resource requests and grants from the system
- Resource brokers mediate between application code and the underlying OS
- Runtime estimation of CPU, radio bandwidth, energy, and storage availability

Future Work

- Next steps: Coordinated resource management across a network

Questions...