Title
Kairos: A Macro-Programming Model for Wireless Sensor Networks

Permalink
https://escholarship.org/uc/item/6k57n215

Authors
Ramakrishna Gummadi
Nupur Kothari
Ramesh Govindan
et al.

Publication Date
2005
Kairos: A Macro-programming Model for Wireless Sensor Networks

Ramakrishna Gummadi, Nupur Kothari, Ramesh Govindan, Todd Millstein

http://kairos.usc.edu

Motivation

- Macroprogramming: Allow all nodes to be programmed as a single unit
- Global program behavior captured as a single sequential task on a centralized memory model
- No need for explicit parallelization or synchronization code
- Challenge is designing the compiler and runtime components that generate and implement an equivalent concurrent distributed version ...
- … and optimize resulting distributed execution at various levels (programmer-level, compiler-level, compiler-network level, and network-level) for the primary constraint of traffic economy in WSNs

Implementation

- Python based implementation of Kairos on Stargates, with motes as radio interfaces.
- Uses Python’s extensibility and embedding APIs to execute a source-level Python script at every node.
- 24 node test-bed of 16 Stargate motes + 8 Mica2Dots hanging off a multi-port PC serial card
- Emstar for E2E routing, topology management, and reliability
- Logical multi-hops over a single physical hop, using S-MAC as the MAC layer.
- Three representative programs: Data Routing Tree, Localization and Vehicle Tracking

Future Work

- Mote implementation: A Kairos compiler which can output nesC + Kairos runtime for motes
- Generic Failure Recovery: Automated recovery mechanisms in presence of various classes of failures
- Various levels of performance optimizations
- Exploiting Heterogeneity, Hierarchy, and User-level Energy/Resource Management

Mechanisms

- Four main programming primitives in Kairos:
  - first-class datatypes: node, node_list (iterator on nodes) for topology independent programming
  - get_neighbors()@node to obtain current one-hop neighbors of a node
  - var@node to synchronously access node-local data
  - a time_queue abstraction for temporal actions
- Kairos compiler translates the centralized sequential program into a node-localized version specialized at runtime by a node’s state.
- Kairos runtime present at every node enforces inter-node control coordination and data coherence using explicit messages
- Eventual Consistency semantics for distributed control flow synchrony and data coherence

Results

- In the Vehicle Tracking application, loose synchrony (a form of eventual consistency) provided by Kairos without the timed_queue abstraction is insufficient
- For the Data Routing Tree application, a correct routing tree of better quality is found, incurring only twice the byte overhead and 30% more convergence time than the One Phase Pull scheme of Directed Diffusion.