UCLA
Posters

Title
System Supoort for Coordinated Imaging for Sensor Networks

Permalink
https://escholarship.org/uc/item/02h4x5wp

Authors
Thanos Stathopoulos
Lewis Girod
John Heidemann
et al.

Publication Date
2004

eScholarship.org Powered by the California Digital Library
University of California

Center for Embedded Networked Sensing

System Support for Coordinated Imaging for Sensor Networks

Thanos Stathopoulos*, Lewis Girod*, John Heidemann** and Deborah Estrin*
*CENS Systems Lab **USC/ISI

Mote Herding: The next step in Tiered Systems

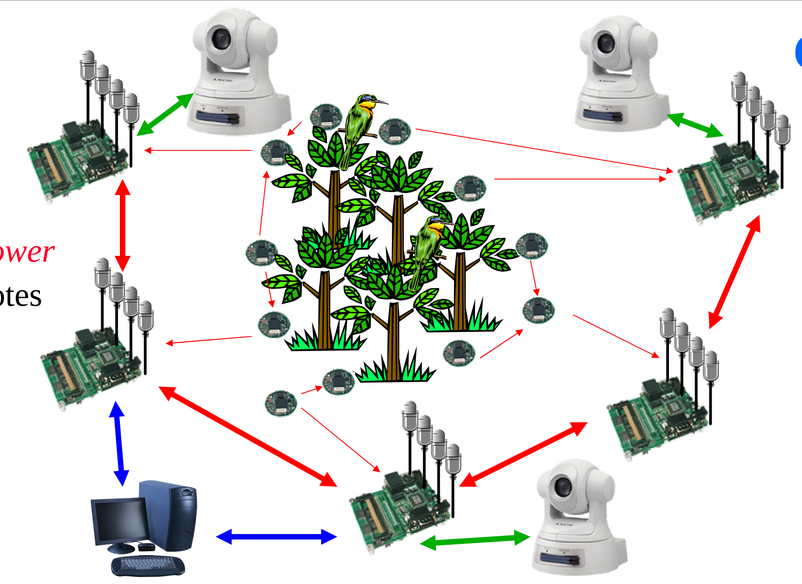
Mote Herding

Tiered systems: Collections of *motes* and *microservers* working together

- **Motes** help achieve the desired *spatial density*
- **Microservers** increase the system's *processing power*
- **Efficient** and cohesive *interoperation* between motes and microservers

Why not **statically** assign motes to stargates?

- Network and environmental *dynamics*
- Dealing with *failures*
- Needs to be done every time



Coordinated Triggered Imaging

Traditional imaging systems: 24/7 monitoring

- **Constant** human *supervision*
- **Large** amount of *data* that has to be processed *manually*
- **Significant** power consumption
- The system should only take a picture when *required* to do so

Why not just 'a sensor triggers a camera' ?

- Nontrivial camera selection based on *latency*, *reaction time*, *orientation*
- Significant system *heterogeneity*

Mote herding is a new *design approach* and **coordinated triggered imaging** is an *example application* illustrating this new approach

System and Design Issues

Mote Herding is…

- **Registration Protocol**
  - Network registration: Binding between the *mote* and its *herder*
  - Motes try to *always* be *attached* to a herder
  - Logical registration: Binding between the *sensor* and the *actuator*
  - Network and Logical registration in general don't overlap
- **Microserver Information Sharing**
  - Allows microservers to present a *high quality global state view* to the application due to increased *spatial* and *temporal resolution*
  - Enables seamless data flow from *sensors* to *actuators* with herders acting as proxies
  - System state exchange allows for *resource handoff*

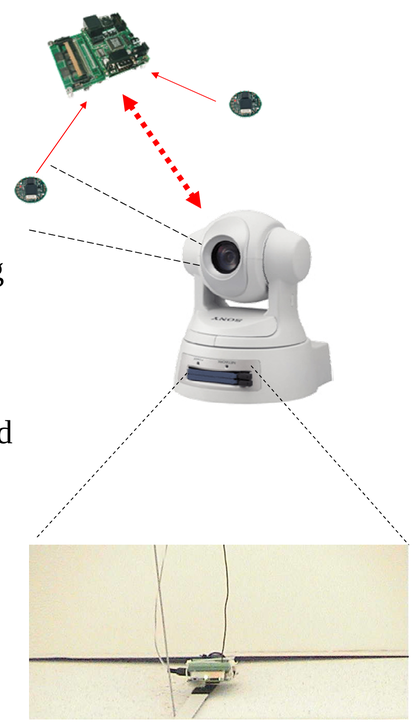
Mote Herding enables…

- **Coordinated Triggered Imaging**
  - Associates mote *locations* and camera *fields of view*
  - Seamless *data flow* from a collection of *motes* to a collection of *cameras*
  - Resource location: find the most *appropriate* camera in terms of *coverage*, *reaction* time and *orientation*
  - Resource contention**:** camera *load balancing*
- **System Health Monitoring**
  - Monitor the status of all system components: *stargates*, *motes* and *actuators*
  - Take appropriate action if necessary: *hand off* motes to different herders
- **Custody Transfer and Storage Management**
  - Microserver *storage capacity* allows them to act as *custodians* in a particular custody hierarchy

Preliminary Implementation Details

Demo Setup

- **Hardware**
  - Sony RZ30N pan-tilt-zoom *steerable* camera
  - A *stargate* connected to a camera via *ethernet*
  - A mica2 mote attached to the stargate acting as a *motenic*
  - A pair of mica2 motes with *PIR sensors* acting as the triggering input
- **System goal**
  - When a mote's PIR sensor detects activity, steer the camera and take a *picture* of the mote's *surrounding area*
- **Mote Herding**
  - The *stargate* is the *herder* while the *motes* are the *herd*
  - Association: how does the *stargate* know *where* to point the camera?



Application Setup

- **Mote software**
  - **Data transport** is achieved using a diffusion like *multihop tree* protocol with the sink running on the stargate
  - **Interfaces** to the stargate part of the system are provided through *EmTOS / motenic*
  - **Data** sent from a mote to the sink after a PIR sensor *interrupt* occurs
- **Controlling application**
  - Binds mote *locations* to camera *orientation* and *zoom*
  - Issues *steering commands* to the camera driver based on *data* coming to the *multihop sink*
  - Receives *image data* through a *sensor client*

Camera control

- **Camera driver**
  - Fully *integrated* with the *EmStar framework*
  - Uses *cURL* in *asynchronous* mode to handle HTTP requests and replies
  - Provides *status devices* that contain information on all camera components and configurations
  - Provides *query devices* for all actuation commands
  - Can be controlled from the *command line* or through a *query device client*
  - Imaging provided through a *sensor device* as well as a jpeg file
  - Supports *non-blocking* operations
  - Supports *multiple concurrent requests* through a *serialization scheduler*

Conclusions and future work

- **Mote herding** is a *set of tools and services* that act as a middleware component
  - Enables *seamless interoperation* of motes and stargates
  - Masks *resource heterogeneity* under common *abstractions*
  - Provides *higher-layer* services that applications can use
- **Future work**: build a complete coordinated triggered imaging system that utilizes mote herding and *extend* the concept to *other classes* of applications