Title

Triggering on Area: A Systems Approach

Authors

Benjamin Greenstein
Deborah Estrin
Eddie Kohler
et al.

Publication Date

2003

Triggering on Area: A Systems Approach

Ben Greenstein, David Culler, Deborah Estrin, Eddie Kohler
Laboratory for Embedded Collaborative Systems – http://lecs.cs.ucla.edu

Introduction: Complexity of application mandates a macroprogramming environment

Application Challenges

- **Homogenous region identification and area calculation**
  - Identify *homogenous regions*
  - Compute the area of homogenous regions in a purely distributed fashion
  - Trigger when a region satisfies a user-configurable predicate
- **Application complexity**
  - Routing: *Spanning tree* formation
  - Local area calculation: *Delaunay triangulation* or inverse neighborhood
  - Aggregation: In-network weighted count
  - System services: Packet transmission, ADC sampling, timers

Architectural Challenges

- **Efficiency vs. programmability**
  - Continuing challenge to implement complex *distributed applications* on systems that have been designed from the ground up for efficiency
  - Today's application developers still need to be skilled kernel hackers
- **Flexibility**
  - Devise a flexible system so that components can be swapped out to achieve alternate functionalities
- **Macroprogrammability**
  - Provide a framework for the development of sophisticated distributed evaluation and actuation applications
  - Provide user-level configuration via *wiring* and *parameterization*

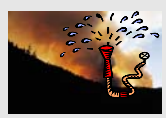
Problem Description: Distributed discovery of large, homogenous regions

Detection, Evaluation, Action

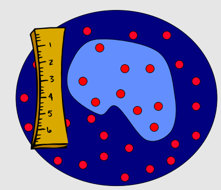
When a data set cut from space/time is determined to contain a target phenomenon, take action

- A fire has been detected
- Is it a campfire or a forest fire?
- If the latter, locate any sprinklers nearby and turn them on. If there are no sprinklers, alert a human

OR



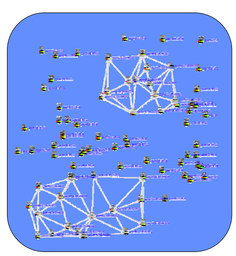
- Homogenous regions are connected subgraphs of the communication graph in which nodes' sensor values fall within a predefined range
- We compute the area of such regions



Proposed Solution: Architect a flexible stack of system services to form a cohesive application

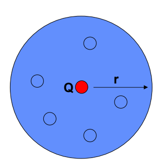
Delaunay Triangulation

- Defined as the subdivision of the network into triangles such that triangle vertices are nodes in the graph and the circumcircle of every triangle contains no vertices
- Used to compute area
  - Assign each triangle to the vertex with the lowest ID
  - Build an aggregation tree to sum the area of the triangles
- Requires location information
- Is consistent for any $\triangle ABC$ in the triangulation when AB, BC, and AC are either:
  - Communication links
  - Present in the "neighborhood" of each vertex.
- Tends to underestimate area because it yields the tightest polygon enclosing the region, even though edge nodes might very well represent space beyond that polygon



Inverse Neighborhood

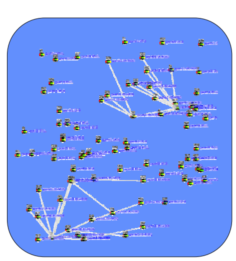
- Also used to compute area
  - Derive an estimate of a node's covering area using its transmission area divided by the number of communicating neighbors it has
  - Build an aggregation tree to sum covering areas of all nodes in region
- Requires no location information
- Communication area estimate has a huge impact on the result (unless used only to compare relative sizes)
- Inverse neighborhood can overestimate area, because nodes near edge of network have fewer neighbors



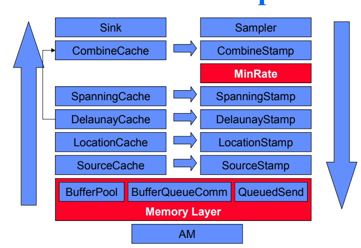
$$A_Q = \frac{\pi \times r^2}{N+1}$$

Spanning Tree

- Finding the optimal tree is equivalent in complexity to computing all-pairs shortest paths
  - Cannot be solved greedily; optimal graph does not necessarily contain optimal substructure
- In terms of depth, random is no worse than twice the optimal
- Approximations to optimal are possible
  - Construct convex hull and choose node near center of mass as root
  - Form k random trees and choose best one
- Implemented a greedy formation algorithm
  - If in region, set region ID to local address, parent to NULL, and hop count to zero, then advertise
  - If a node receives a message with a lower region ID or the same ID and lower hop count than its parent, then switch parent pointer to the source
  - Version number incremented when parent dies or changes to a new region to prevent count-to-infinity loops



Architectural Components

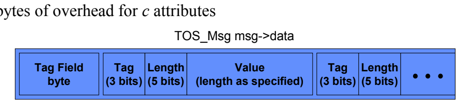


Minimum Rate

- Drives soft-state mechanisms of the various layers
- Generates NULL packets at a minimum rate
  - Checks over period to see if any packets came through, and if not, generates one
- Soft-state data gets on this "elevator" on its way down the stack
- Fewer header bytes sent; fewer timers needed; reduced duplication of attributes
- Eventual consistency, however, isn't always timely

Tag/Value Packet Format

- Add as many attributes as needed or as can fit in data portion of *TOS_Msg*
  - Limited to eight different types of attributes in system
- Constant time inserts, tests for inclusion, linear reads
  - *putAttr()* and *getAttr()* provided as inline for write and read
  - *Byte 0* of data portion contains bit field for test
- *1+c* bytes of overhead for *c* attributes



Memory Layer

- Buffer pool
  - Buffer allocation and guards handled in memory layer instead of in every component
  - *alloc(), free(), copyAlloc()*
- Implicit deallocation
  - Occurs when *SendMsg.send ()* fails and immediately after upcalls to *SendMsg.sendDone()* and *ReceiveMsg.receive()*
- Queued send absorbs bursts
- Sits directly above AM layer