An Associative Broadcast Based Coordination Model for Distributed Processes

James C. Browne
Kevin Kane
Hongxia Tian

Department of Computer Sciences
The University of Texas at Austin

Contents

- Associative Broadcast
- Building an Application
- Example Application: Convex Hull
- Future Work, Summary, and Conclusions

Associative Broadcast

- Just like in IP Multicast,
  - Messages addressed to arbitrarily-chosen numerical group.
  - Nodes subscribe to those groups.
- But addressing is more powerful.
  - Node addresses are tables of attributes ("profiles")
  - Messages are addressed to "profiles."

Profiles

- **Profile: table of attributes and attribute/value pairs**

Name	Optional value
Linux	
Version	“7.0”
Memory	128

Messages and Selectors

- Message: A two-tuple of (Selector, Data)
- Selector: propositional logic expressions over profiles.
  - Existence of attribute
  - Comparison of attribute values
    - Equality
    - Inequalities for ordered value types
  - Compound Boolean (AND/OR)
- Data: Transaction name, control flow protocol, application payload

Selector to Profile Matching

- Examine selector of message
- Consult profile for attributes.
- If selector evaluates true, message passed to application
- Semantics: Selector evaluated when receive primitive called
  - All messages queued until called for

Specifying an Application

- The attribute domain in which the profiles and selectors are specified
- Each component of the application as a state machine
- Rules for binding states to profiles
- Protocols for interactions including message and selector details
- Required common state information

Application Execution Behavior

- Each entity governed by its state machine
- Profile set to correspond to initial state
- When a message is received that satisfies the profile:
  - State machine transitions to new state
  - Executes some action
    - Possibly associatively broadcasting one or more messages
  - Set profile to correspond to new state

Associative Interfaces and Interactions

- *Accepts interface*: set of (profile, transaction, protocol) tuples
  - Represents the services the process provides.
  - Transaction: Method invocation, or unit of work.
  - Protocols: data flow (asynchronous) and call-return (synchronous).
- R*equests interface:* set of (selector, transaction, protocol) tuples
  - Represents the services the process requires.
- These interfaces can, and usually will, change during execution of the system.

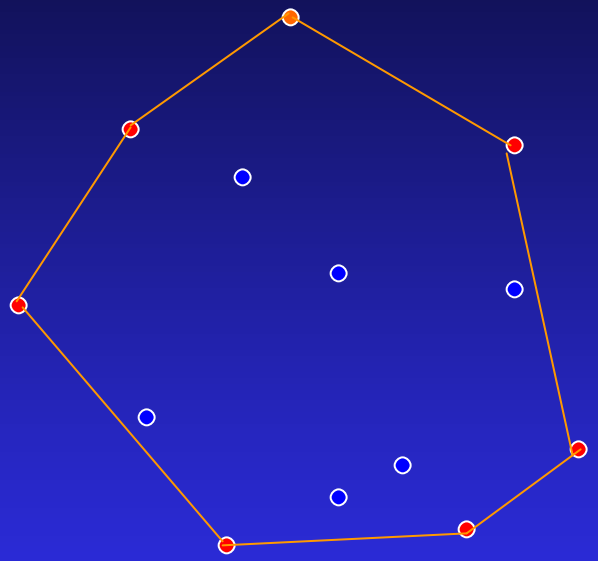
Status

- Working runtime system implementing the communication layer.
- Applications implemented: distributed mutual exclusion, distributed readers/writers, convex hull, some parallel linear algebra algorithms.
- Extension of the associative interface to multiple message conditions. (Data flow firing rules)

Implementation

- Implemented as Java classes on an Ethernet
- Timed Reliable Broadcast
  - Applications require knowledge of maximum network latency
  - Applications require guarantee of reliable delivery of broadcasts
    - Implemented reliable broadcast based on Kaashoek

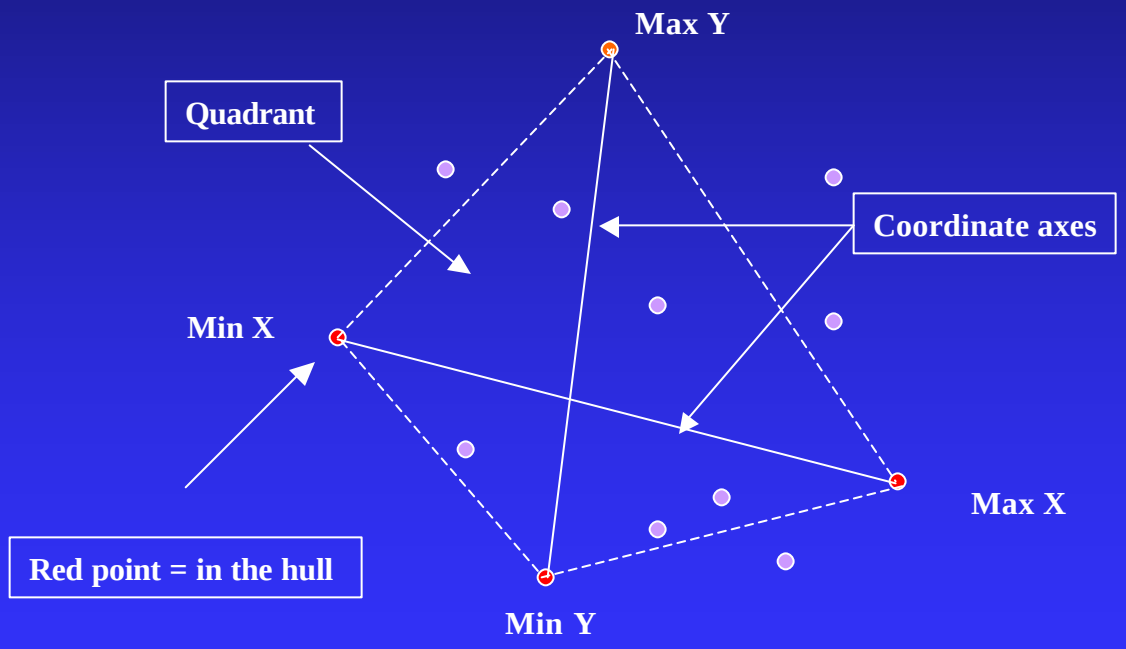
Example: Convex Hull



***Convex hull in 2-D*:** Subset of points that form a convex polygon containing all points in the set.

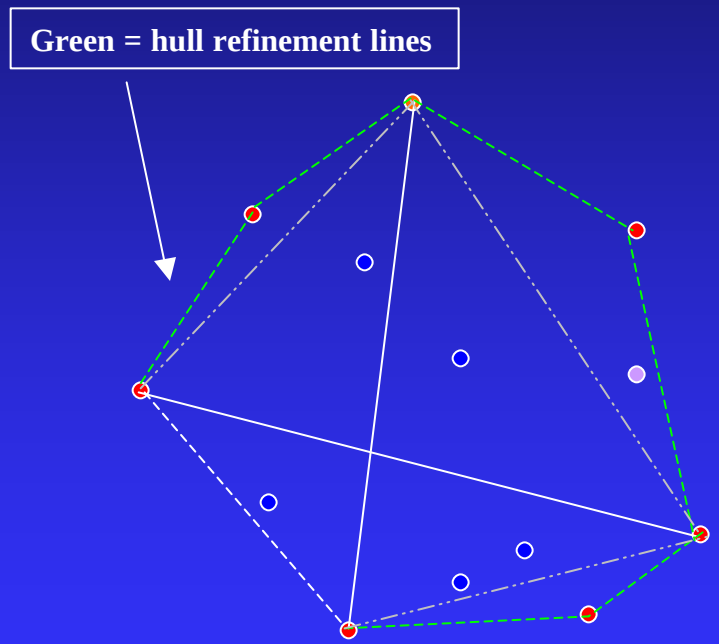
Incremental Convex Hull

- Traditional approach: Guess…



Incremental Convex Hull

- …then refine.



Example: Convex Hull – Algorithm Specification

- Every point represented as a process
- Process common state is the current "incremental" convex hull
- Points coordinate with each other to refine the hull until all points are contained

Example: Convex Hull – Domain Attributes

- Status: Yes/No/Unknown indicating membership in the convex hull
- State: MinX/MinY/MaxX/MaxY/Other
- Quadrant: Computed quadrant value
- X and Y: Point coordinates

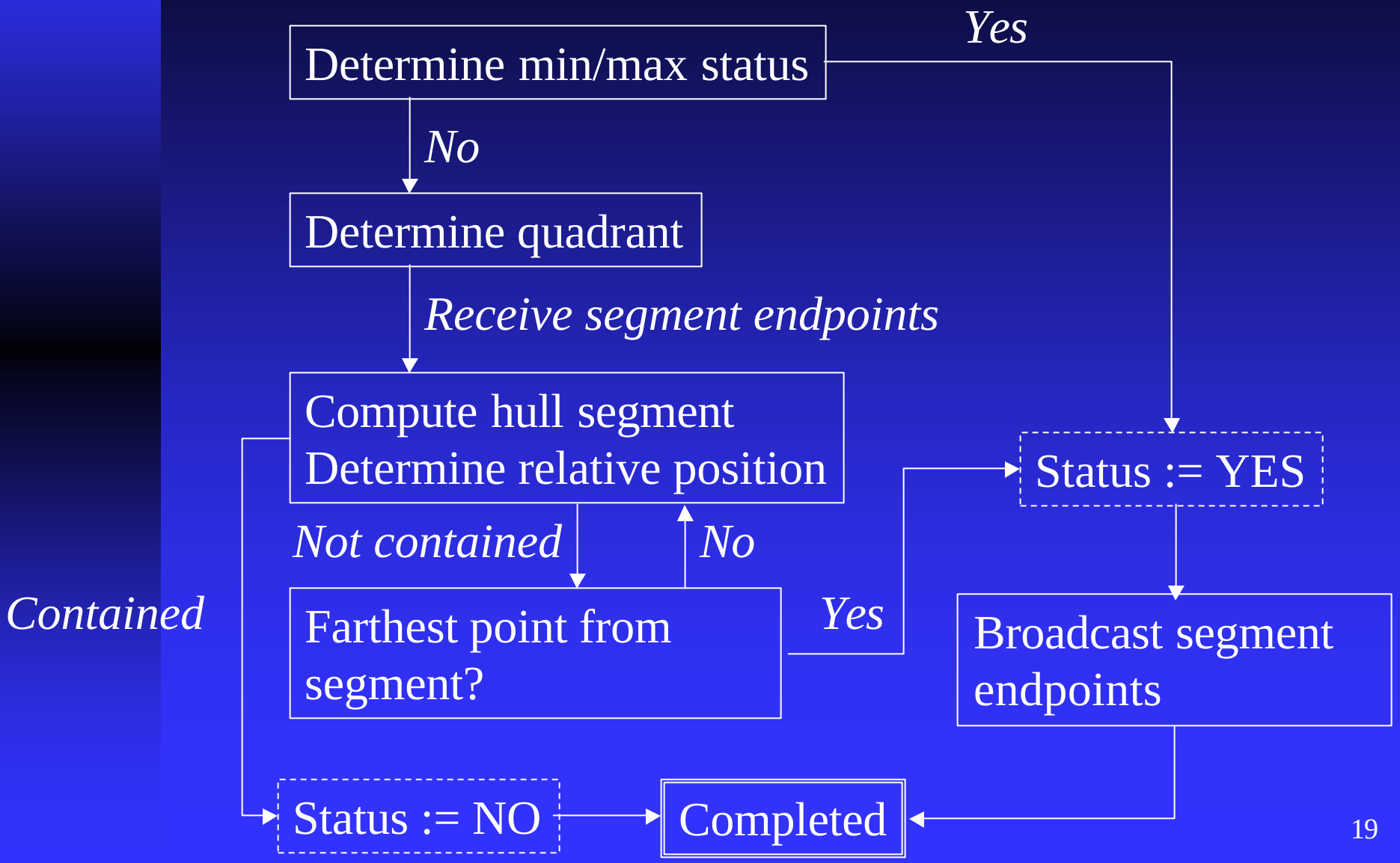
Example: Convex Hull – Algorithm Specification

- Each point
  - Determines whether it is x/y min/max.
  - Determines its quadrant.
  - Computes a line joining “nearest” known hull points,determines position with respect to this line.
  - Continues previous step until membership in hull is confirmed or negated.

Example: Convex Hull – Selector Samples

- Signaling completion for a quadrant:
  - `Quadrant == 1`
- Sending a refinement out:
  - `Status == Unknown && Quadrant == 2 && X >= 2.0 && X <= 10.0`

Example: Convex Hull - States



Properties of Associative Broadcast Coordination and Composition

- Coordination among dynamic sets with dynamic behaviors
- Fault-tolerance and replication follow from coordination model
- Integration of coordination and composition
- Processes are fully distributed and symmetric

Relation to Data-Driven vs. Control-Driven Models

- Combination of data-driven and control-driven models
- Data-driven: State changes caused by message reception
- Control-driven: Interfaces are kept separate, and link into computational code
- Transient Channels

Related Work

- Linda: Pattern matching on tuple extraction
  - Associative Broadcast: Distributed Tuple Space
- Structured Dagger: message-driven programming style
- Splice: Interface-mediated communication, receiver-controlled subscriptions

Future Work

- System definition language
  - Specification of accepts and requests interface
  - Matching of interfaces to ensure all requests can be serviced
  - Libraries of modules to service requests not serviced by the program itself
  - Integration of runtime and compile time composition.