Self-stabilization and eventual consistency in replicated real-time databases

Sanny Gustavsson, Sten F. Andler
University of Skövde, Sweden
Presentation outline

• Replication in DeeDS
• Conflict detection and resolution
• Convergence and stabilization
• Challenges and ongoing work
Fundamental questions

• What does self-healing mean to us?
  – Error processing part of fault tolerance
    • Self-inflicted “errors”
    • Restoring global consistency

• What part of the self-healing problem?
  – Detecting and resolving inconsistencies
  – Proving system convergence
Fundamental questions, cont.

• What applications are we targeting?
  – Embedded, distributed real-time systems

• What are our main ideas?
  – Applying self-stabilization theory to a new area
  – Framework for resolving conflicts and proving convergence
DeeDS: **Distributed Active Real-time Database System**

- Embedded & distributed real-time systems
  - Predictable resource requirements
  - Fault tolerance
  - Example application: unmanned helicopters (WITAS)
- Full data replication
  - Central for communication, fault tolerance, availability
  - *Eventual* rather than immediate (mutual) consistency
  - Local predictability
DeeDS: replication protocol

- Supertransaction: consistent; relaxed isolation
- Subtransactions: single node; relaxed global consistency
Conflict detection and resolution

• Conflict detection
  – E.g., double-booked flight seats
  – Performed using version vectors

• Conflict resolution
  – Requires a policy and a mechanism
  – The policy could be simple and generic ...
    • E.g., use value with lowest timestamp
  – ... or complex and application-specific
    • E.g., “Book one of the passengers on a later flight, and compensate her by giving her a free trip in the future.”
Convergence and stabilization

• Convergence: correctness criterion
  – Eventual consistency
    • A quiescent system eventually stabilizes
  – Tolerance of inconsistency
    • Applications must be aware of possible inconsistencies
  – Similar to self-stabilization

• Conflict resolution protocol must guarantee eventual consistency
  – Self-stabilization theory may help
Convergence and stabilization, cont.
Convergence and stabilization, cont.

• Conflict resolution protocol
  – Consists of a policy and a mechanism

• Requires that
  – The mechanism correctly implements the policy
  – The system stabilizes in a finite number of steps
    • In a quiescent system
    • The mechanism never takes the system to an inconsistent state
Challenges and differences

• Complexity
  – CR protocols are application-specific and complex
  – Is it still feasible to use self-stabilization theory?
• Conflict resolution is explicit; self-stabilization generally implicit in the algorithm
• Stabilizes from inconsistencies introduced by applications in normal operation rather than from faults
Ongoing work

• Application of self-stabilization theory to a new area: conflict resolution in eventually consistent databases
• A formal method for analysis of conflict resolution protocols
  – Specification of state spaces
  – Specification of conflict resolution policies
• Proofs for generic conflict resolution protocols