Continuous Consistency Management
in Distributed Real-Time Databases with Multiple Writers of Replicated Data

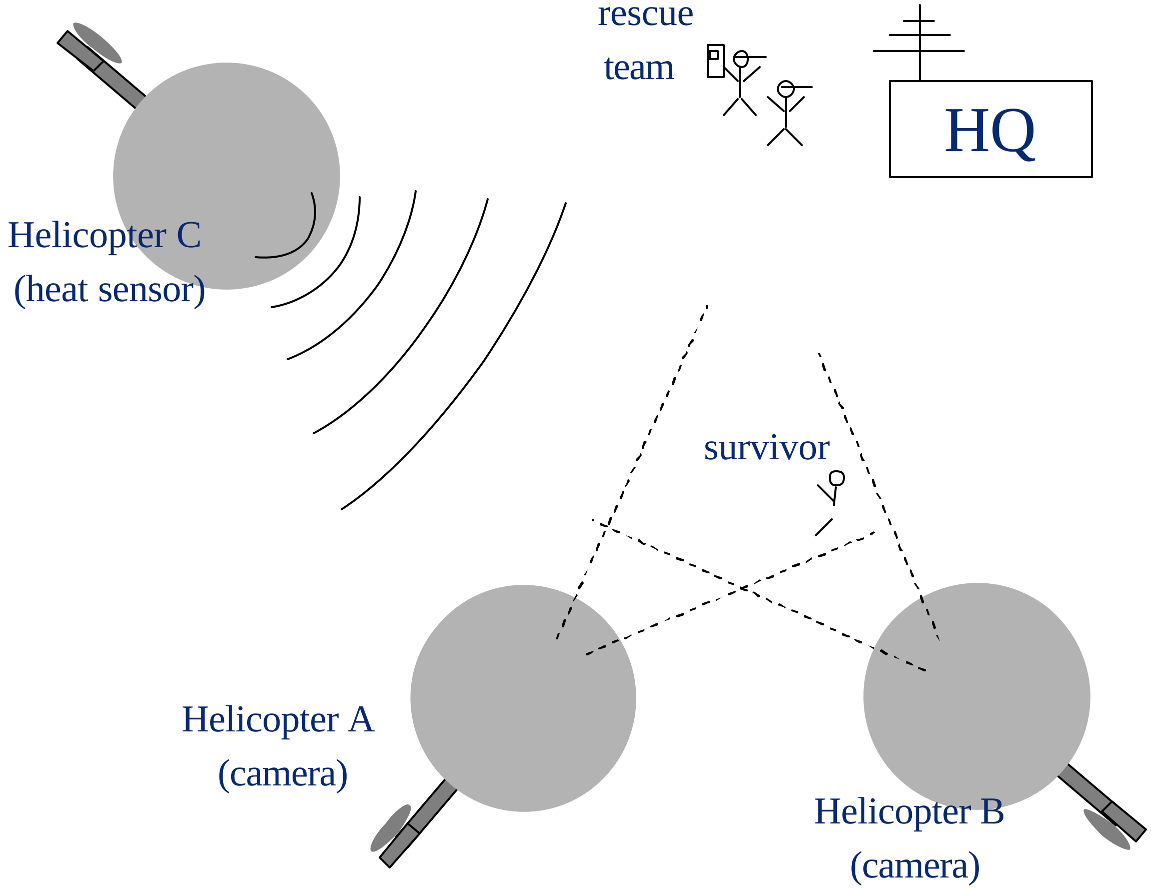
Sanny Gustafsson, Sten F. Andler

Distributed Real-Time Systems Group
DeeDS Project

Work overview

- Replication protocol for DRTDB
  - Prototype: DeeDS
  - Distributed whiteboard architecture
    - Virtual full replication
  - Keep up with real-time progress of environment
    - Local timeliness, eventual global consistency
  - Support application tolerance of inconsistencies
- Presentation focus: conflict management
  - Continuous convergence

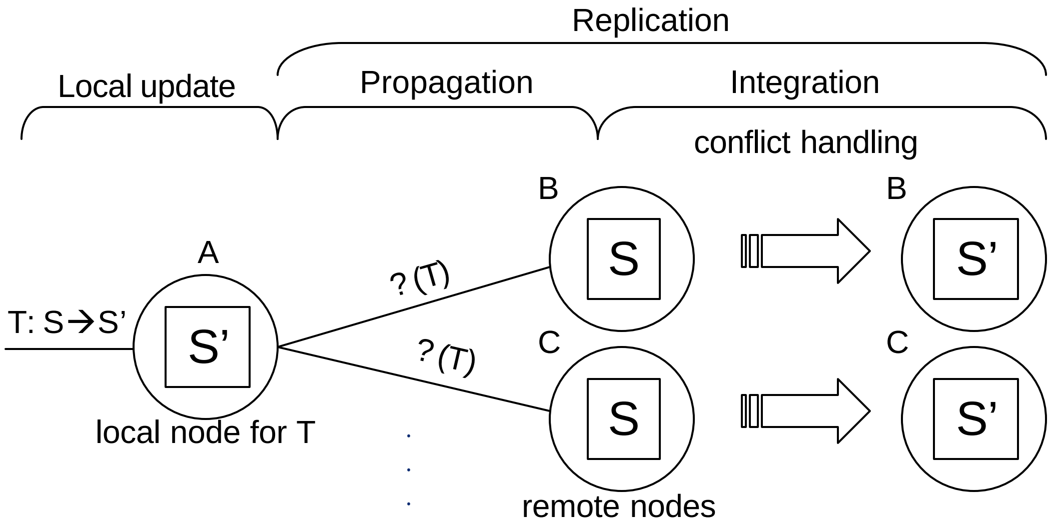
Example: introduction



Collaboration data:

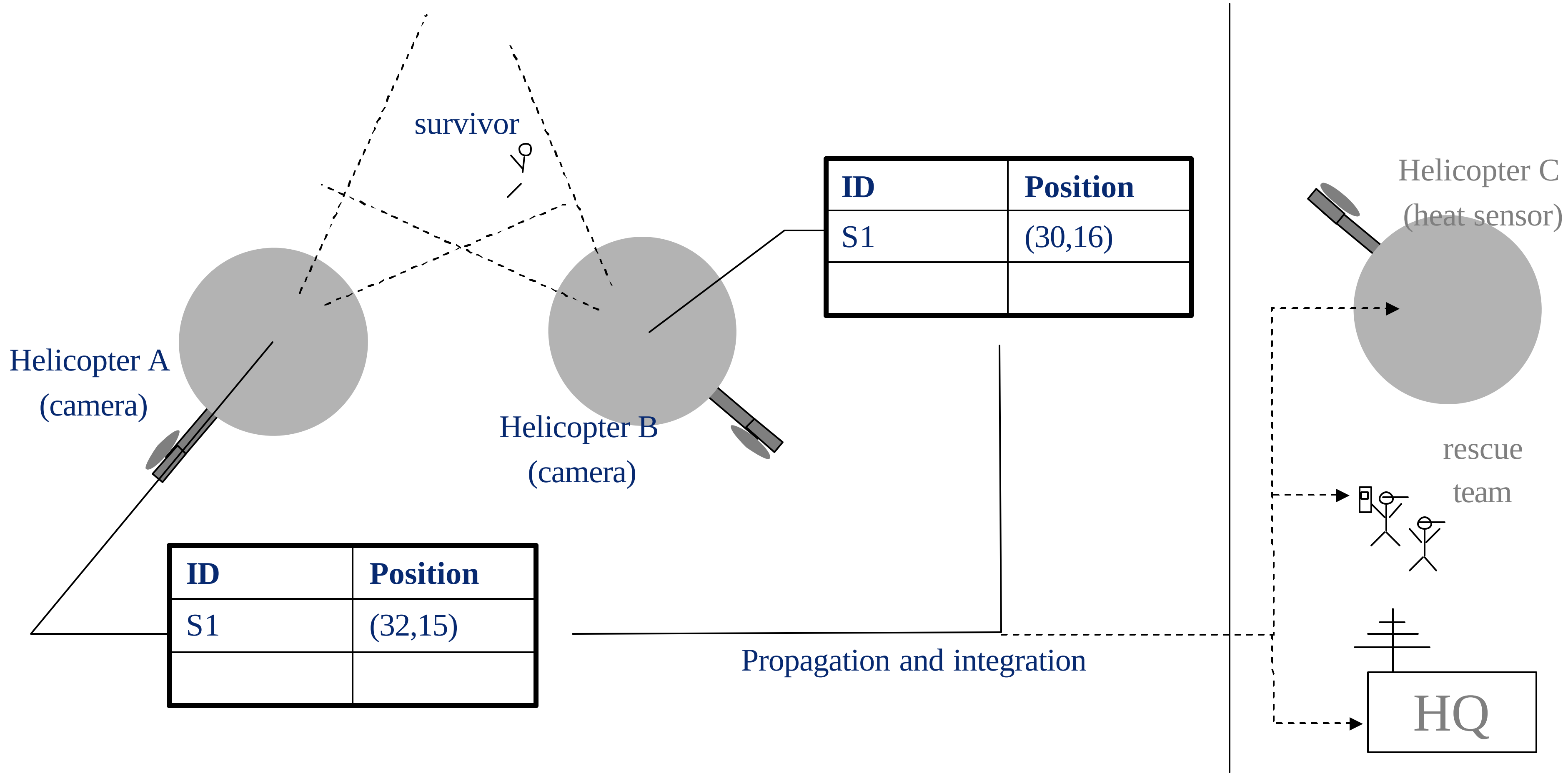
- Table of survivors
  - ID
  - Position
- Route table
  - Waypoints

PRiDe
Protocol for Replication in DeeDS



- Simplifying assumptions:
  - Predictable message delivery time
  - No local overloads
  - Single-update transactions
  - Static set of nodes

Example: concurrent updates

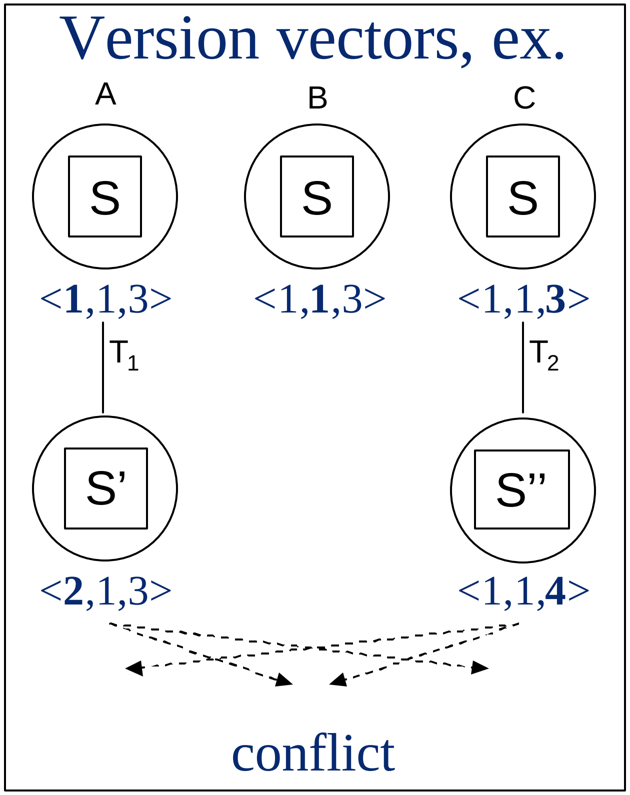


ID	Position
S1	(30,16)

ID	Position
S1	(32,15)

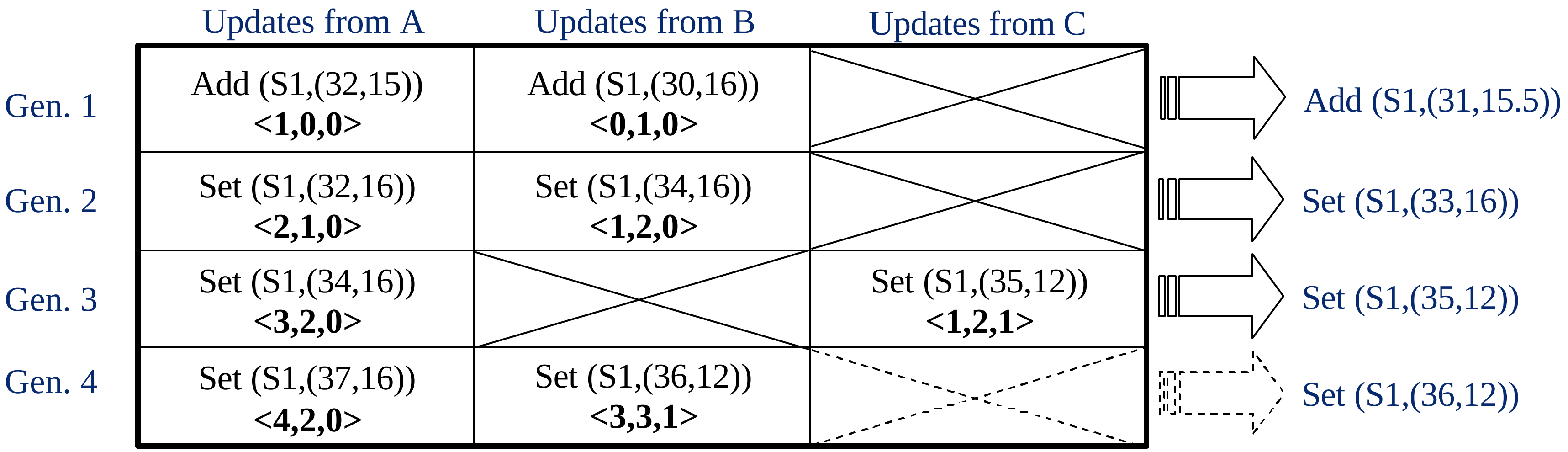
Conflict management

- Conflict detection
  - Version vectors
- Update qualification
  - Conflict sets, generations
- Conflict resolution
  - Forwards resolution only
  - Application-specific policies


Version vectors, ex.

Conflict sets & generations

- Conflict set: all *non-stable* updates to an object
- Generation: unit of conflict resolution
- Ensures deterministic resolution/update pruning
  - Assumes ordered messages between node pairs



	Updates from A	Updates from B	Updates from C	
Gen. 1	Add (S1,(32,15)) **<1,0,0>**	Add (S1,(30,16)) **<0,1,0>**		Add (S1,(31,15.5))
Gen. 2	Set (S1,(32,16)) **<2,1,0>**	Set (S1,(34,16)) **<1,2,0>**		Set (S1,(33,16))
Gen. 3	Set (S1,(34,16)) **<3,2,0>**		Set (S1,(35,12)) **<1,2,1>**	Set (S1,(35,12))
Gen. 4	Set (S1,(37,16)) **<4,2,0>**	Set (S1,(36,12)) **<3,3,1>**		Set (S1,(36,12))

Example: conflict management

- Conflict/resolution examples:
  - Conflicting add operations
    - Resolution: merge or allow
  - Conflicting position updates
    - Resolution: merge, possibly weighted by confidence
  - Conflicting routing orders (planner/HQ)
    - Resolution: use confidence; prioritize HQ orders
- Application tolerance
  - Can exploit *maximum-information* position/routing data
  - *Stable* values can be used to, e.g., log movement

Protocol properties

- Local predictability
  - No global locks or commit protocols
  - No transaction rollback or update undo/redo
- Eventual global consistency
  - Deterministic update ordering & conflict resolution
    - (Real-time network for bounded-time stabilization)
- Support for application tolerance
  - Maximum-information and stable values
  - Future work: bound on deviation, confidence metric

Extensions

- Unbounded replication time/partitions
  - Stabilization messages
  - Reconciliation protocol
- Multi-update transactions
  - Transaction-level conflict sets
- Overload management
  - Lower priority of integration/propagation transactions
  - Reconciliate as necessary

Conclusions

- Continuous convergence protocol
  - Local predictability, eventual global consistency
  - Forward conflict resolution using conflict sets, generations
- Application tolerance
  - Maximum-information values
  - Stable values
  - Support for application-specific conflict resolution
- Suitable for applications that can trade off strict consistency for predictability and progress