GLOBE

Global Object Exchange

A dynamically fault-tolerant and dynamically scalable distributed tuplespace for heterogenous, loosely coupled networks

Jesper Honig Spring

http://www.diku.dk/students/eglarsen/GLOBE

Agenda

- The Tuplespace Paradigm
- Tuplespace Semantics
- Achieving Fault-tolerance
- Achieving Scalability
- Measurements of GLOBE
- Related projects
- Conclusion
- Demonstration and Questions

The Tuplespace Paradigm

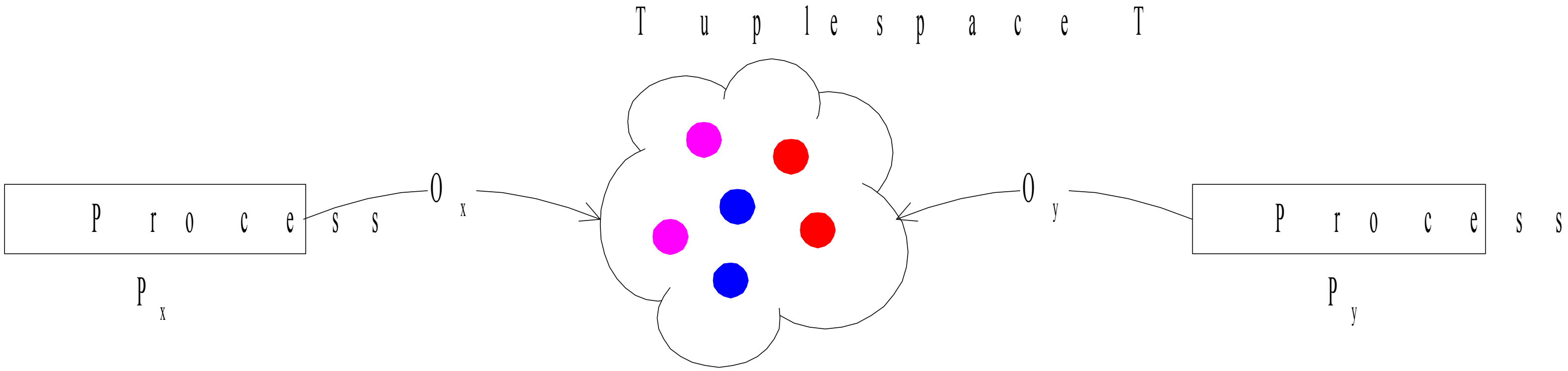
- A communication paradigm (from ’85)
- Communication unit:
  Tuple (≈ Object)
- Tuplespace – an intermediate container
- Tuples immutable in the tuplespace
- Implementations:
  Linda, JavaSpaces, TSpaces etc.

```
Class Person extends Tuple {
  field String name;
  field Integer age;

  method String getName()
  method void setName(newName)

  method Integer getAge()
  …
}
```

The Tuplespace Paradigm



- Atomic Operations:
  - Insertion (out)
  - Withdrawal (in, inp*)
  - Inspection (rd, rdp*)
  - Additional operations

*rdp/inp – predecate

- Matching:
  - Templates (anti-tuple)
  - Exact/Wildcard matching:
    - Tuple Type (null tuple)
    - Tuple Field Values (wildcard fields)

The Tuplespace Paradigm

- Groupware
  - Chat server
  - Shared Blackboard
- High Performance Computing
  - SETI@Home-like calculations
- Intelligent Connectionware
  - Internet Services (internally in Jini LUS)
  - Intelligent Home (TSpaces at IBM)

The Tuplespace Paradigm

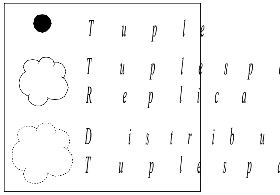
The Classical Problems with Centralized Systems:

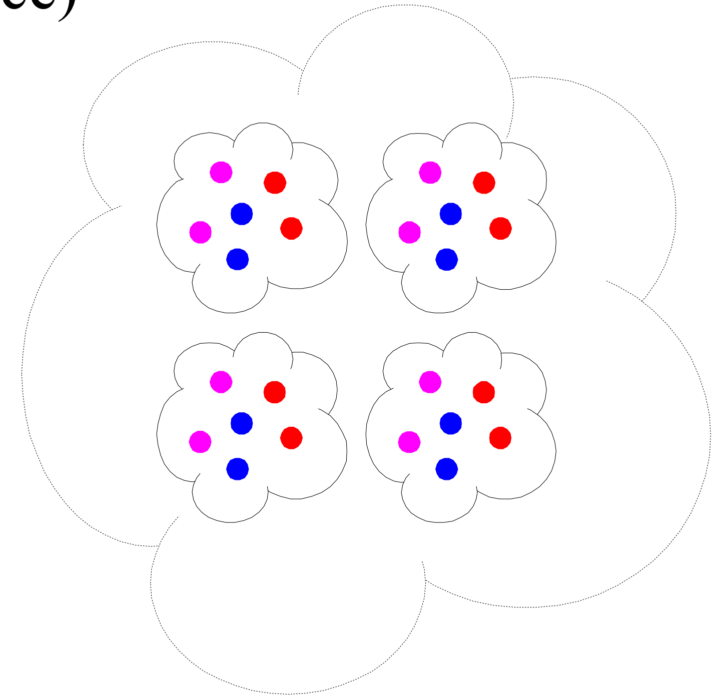
- Availability (Level of Fault-Tolerance)
  - Single Point of Failure
- Scalability
  - Cannot Scale beyond the Single Entity

The Tuplespace Paradigm

- Purpose of GLOBE:
  - Increase Availability (Fault-tolerance)
  - Increase Scalability
  - Dynamic Adjustment

- Distributed Tuplespace Abstraction





Operations Semantics

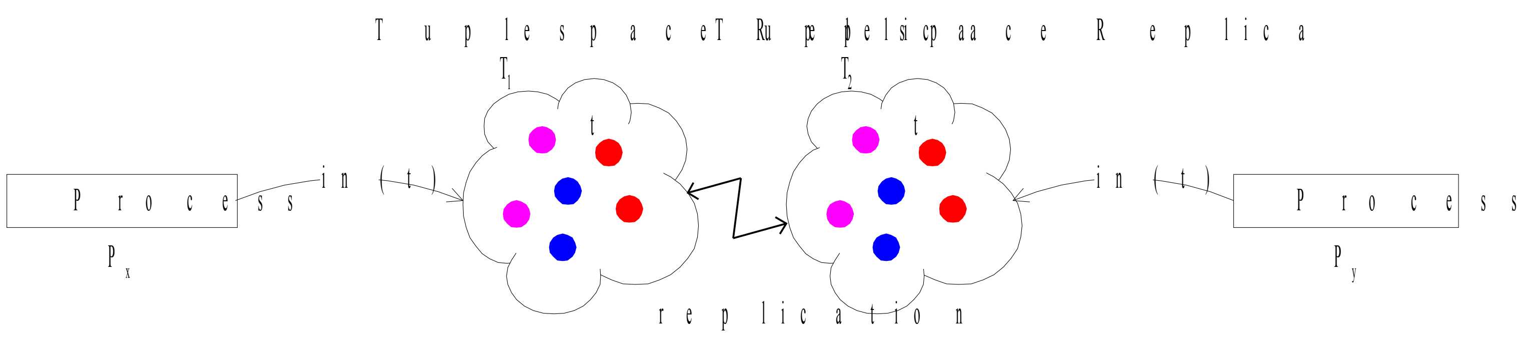
- Tuplespace Semantics vaguely defined:
  - Selection of matching tuple
    - Arbitrary, FIFO, LIFO etc.
  - Selection of process to withdraw tuple
    - Concurrent withdrawals for same tuple
    - Specified as “fair”

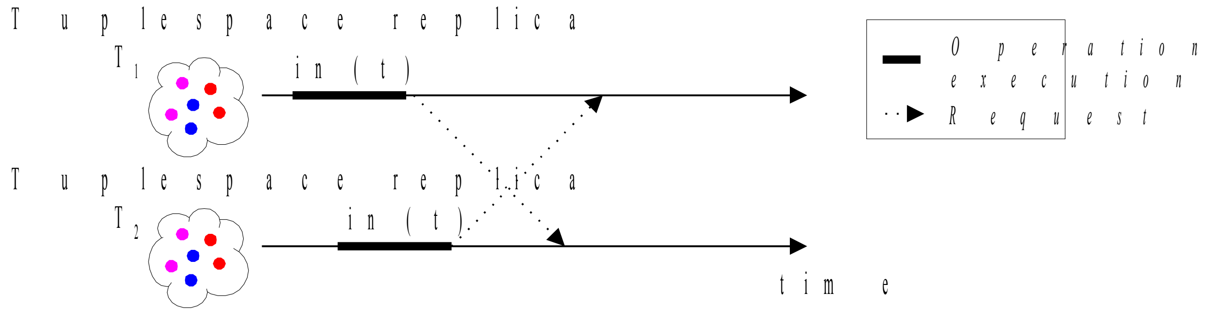
In addition: Issue related to distribution:

- Predicate operations semantically unclear
  - Inspect “present” state
  - What is present state in a distributed environment?

Operations Semantics

Concurrent and Distributed Tuple Withdrawal





- Withdrawal operations must be performed atomically!
- Global ordering across replicas

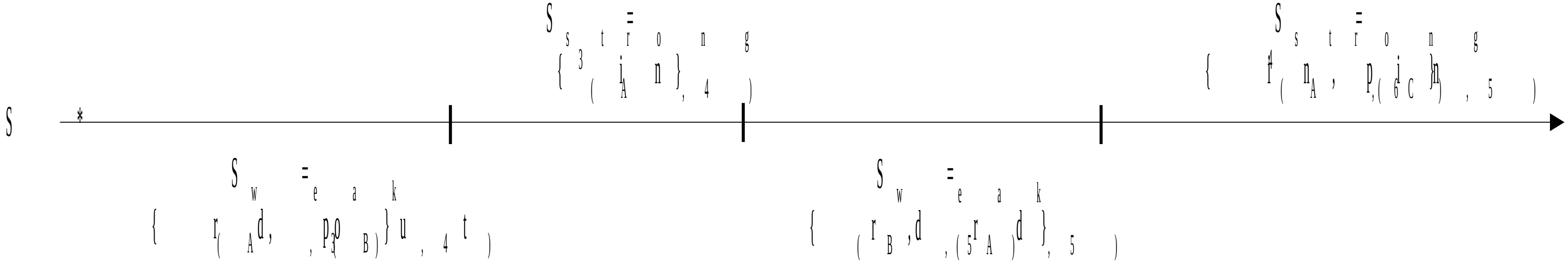
Semantics of GLOBE

- Two Categories of Tuplespace Operations:
  - Strong Operations (in, inp)
  - Weak Operations (out, rd, rdp)

- GLOBE adapts *loose inp/rdp* semantics
  - Weak Operations are performed “locally” and any changes (insertions) propagated later (depending on the synchronization tightness).
  - inp/rdp may show “false” results!

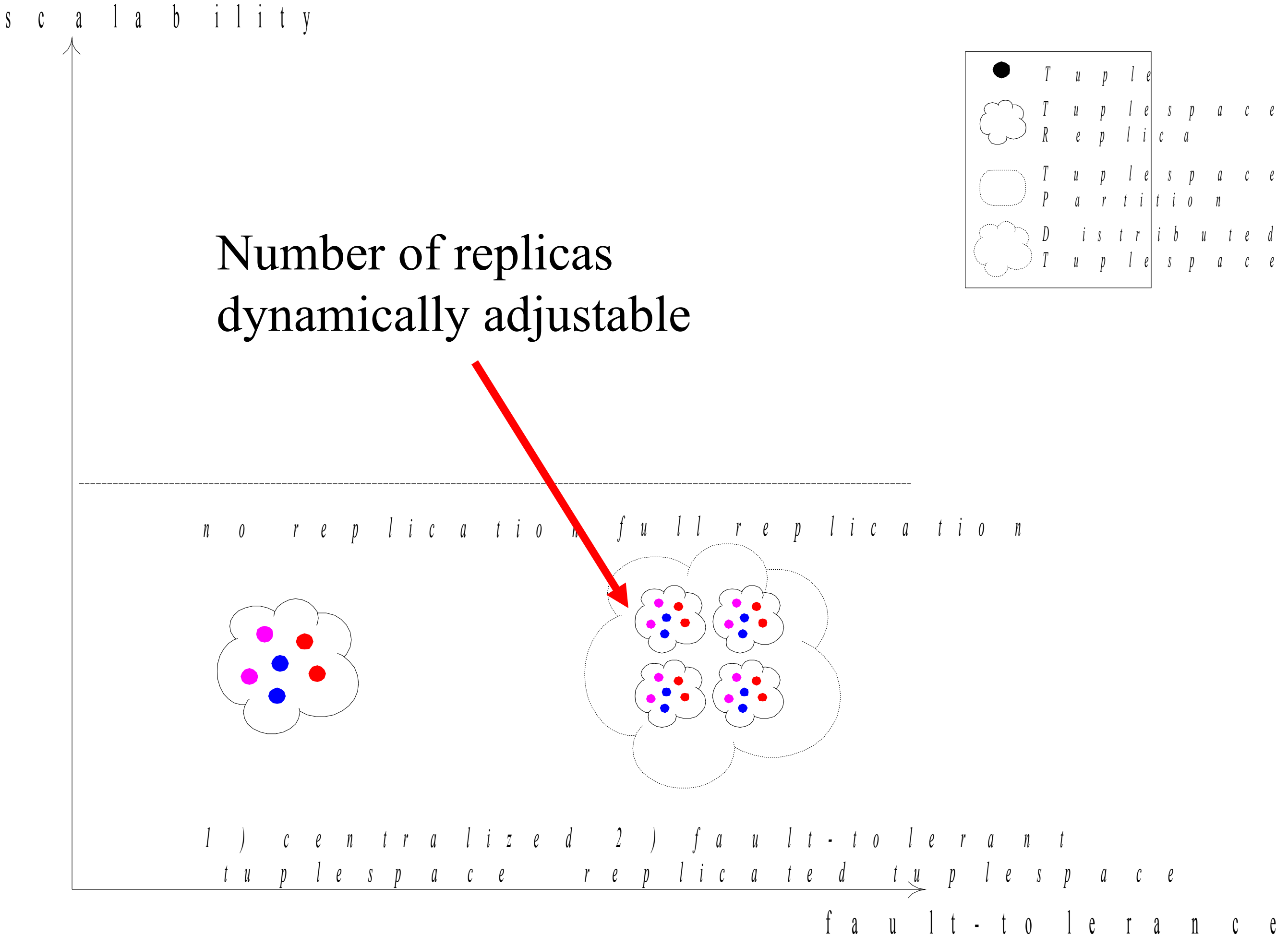
Semantics of GLOBE

- Operation Ordering:
  - Strong Operations are Globally Ordered
  - Weak Operations are Globally Unordered
  - All Operations satisfy Partial Ordering



S* -- a sequence of tuplespace operations performed on a replica

Achieving Availability (Fault-tolerance) and limited Scalability by Replication



Number of replicas dynamically adjustable

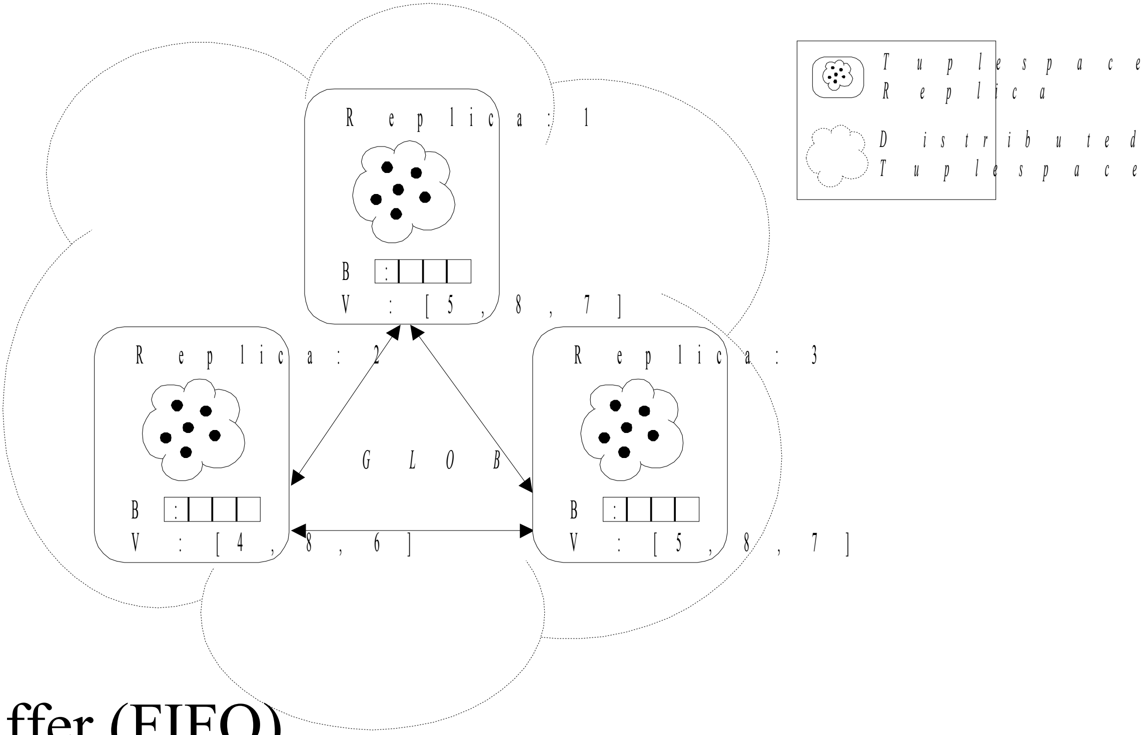
Replica Update Protocol

- Active propagation
  - Operations are persisted before propagation
  - Replicas responsible for propagation
  - Fast propagation
  - Majority voting for all atomic operations (tuplespace operations and configuration operations)
- Problem: Not 100% reliable!
  - Adjustability problem (removal of replica)
  - Inconsistency in case of failure

Replica Update Protocol

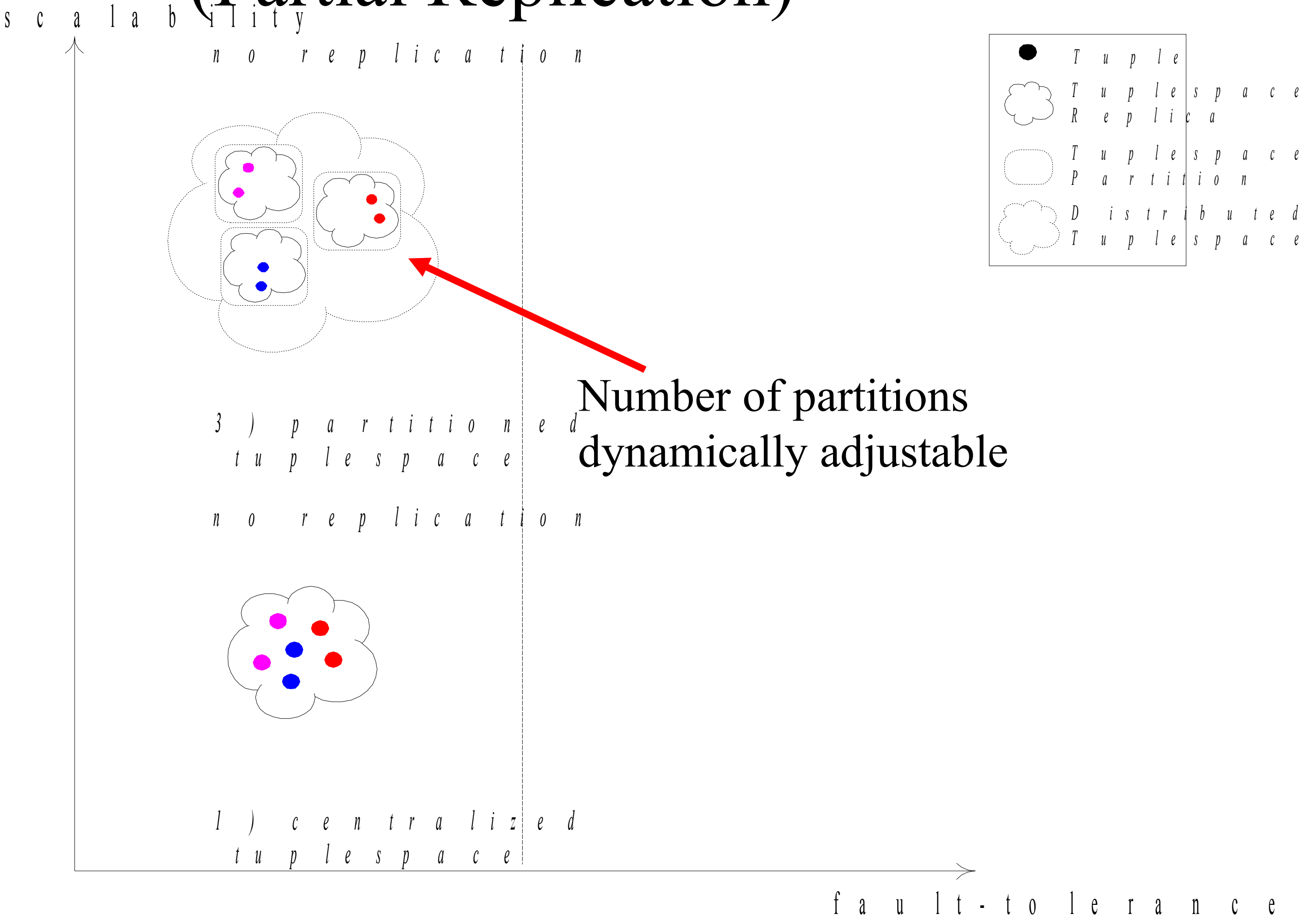
- Anti-entropy
  - Epidemic algorithm (Bayou), pair-wise reconciliation - slow!
  - Replicas responsible for updating themselves (pull-based to avoid duplicates)
  - Synchronization in case of failure
- Hybrid replica update protocol
  - Combines Active Propagation and Anti-Entropy
  - Ensures consistency convergence

Update Propagation



- Operation Buffer (FIFO)
- Operation Vector
- Logical Operation Numbers

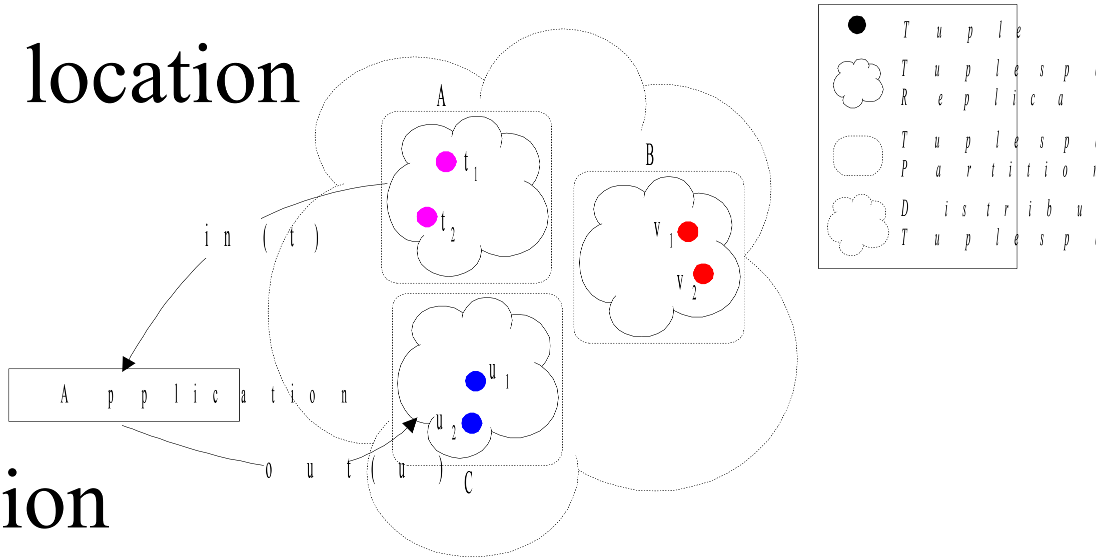
Achieving Scalability by Partitioning (Partial Replication)



Number of partitions dynamically adjustable

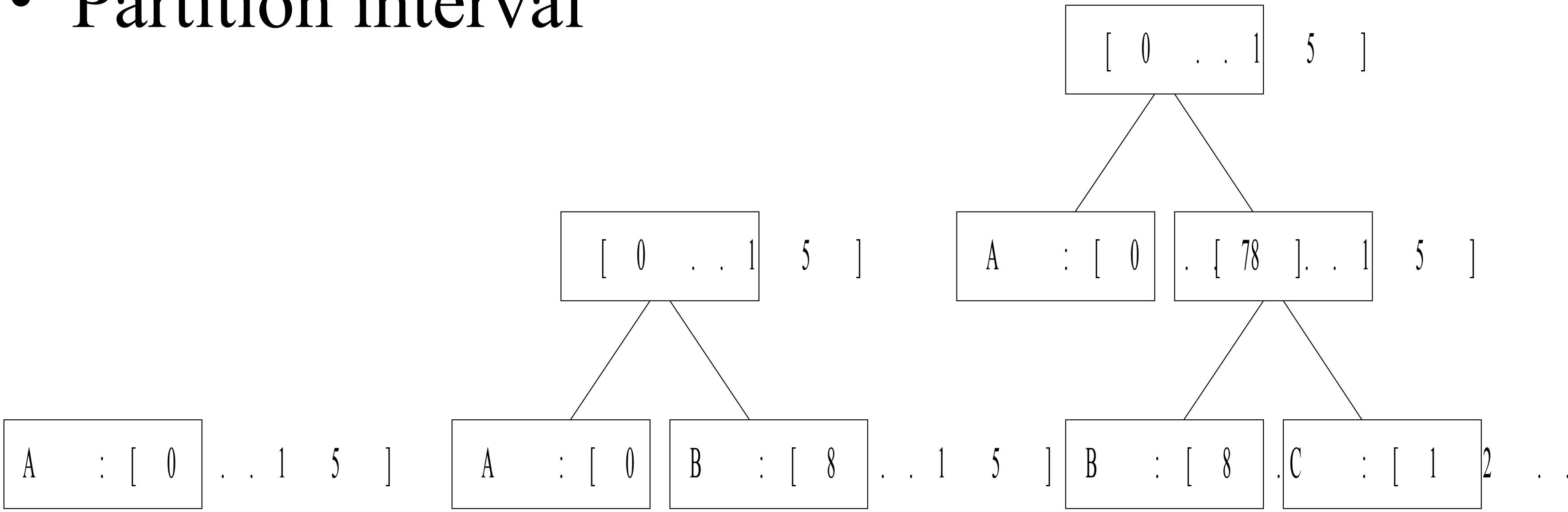
Tuplespace Partitioning

- Load balancing
- Resolution of tuple location
  - Non-deterministic
  - Deterministic
- Hash code
- Operation Redirection
- Problem: Dynamic adjustment
  - Complete rehashing of tuples

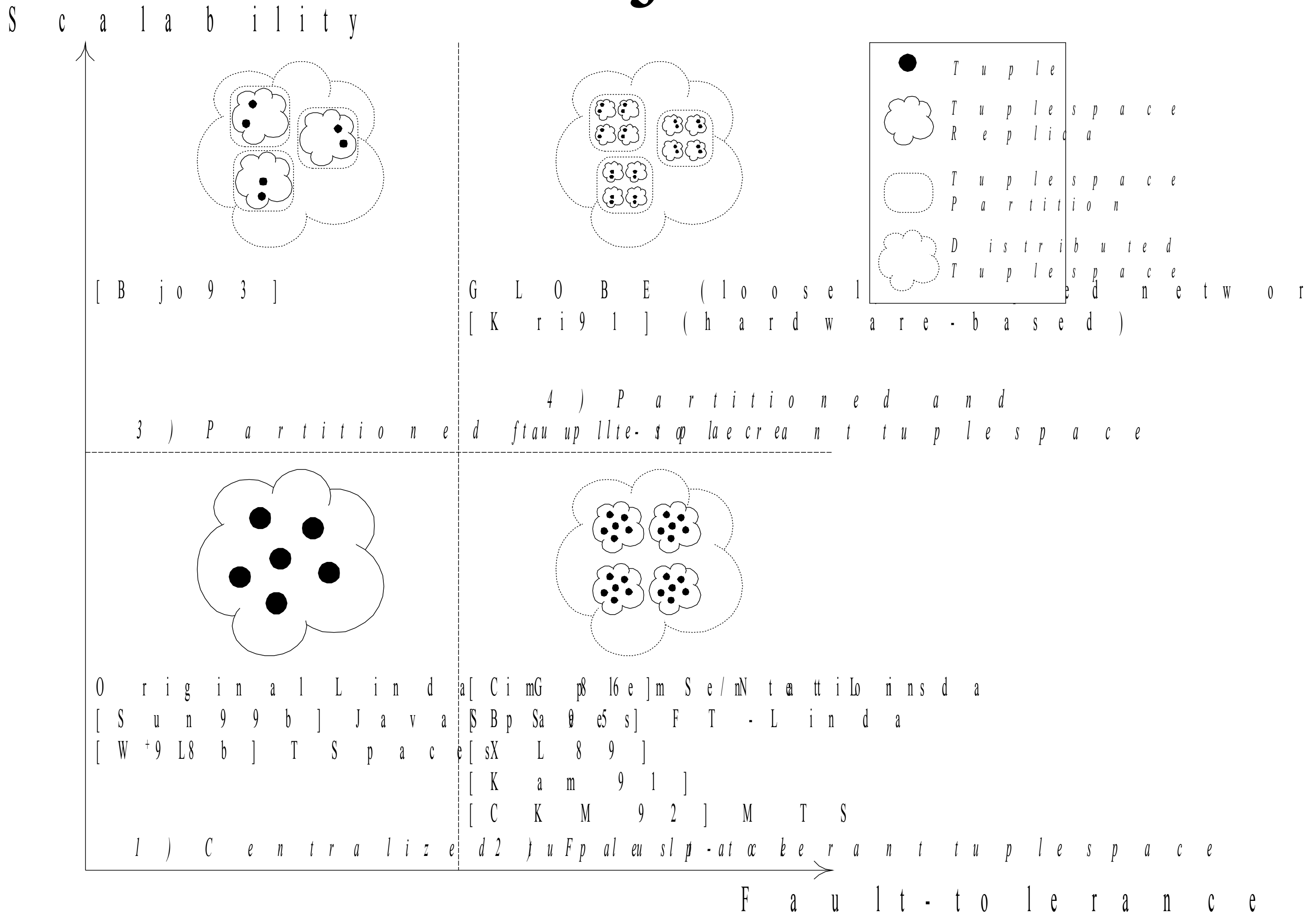


Dynamic Partitioning

- Partitioning by hash code
- Partition interval



Related Projects



[Bjo93]

GLOBE (loosel ed networ
[Kri91] (hardware-based)

3) Partitioned

4) Partitioned and fault-tolerant tuplespace

Original Linda implementations
[Sun99b] JavaSpaces
[W+98b] TSpaces

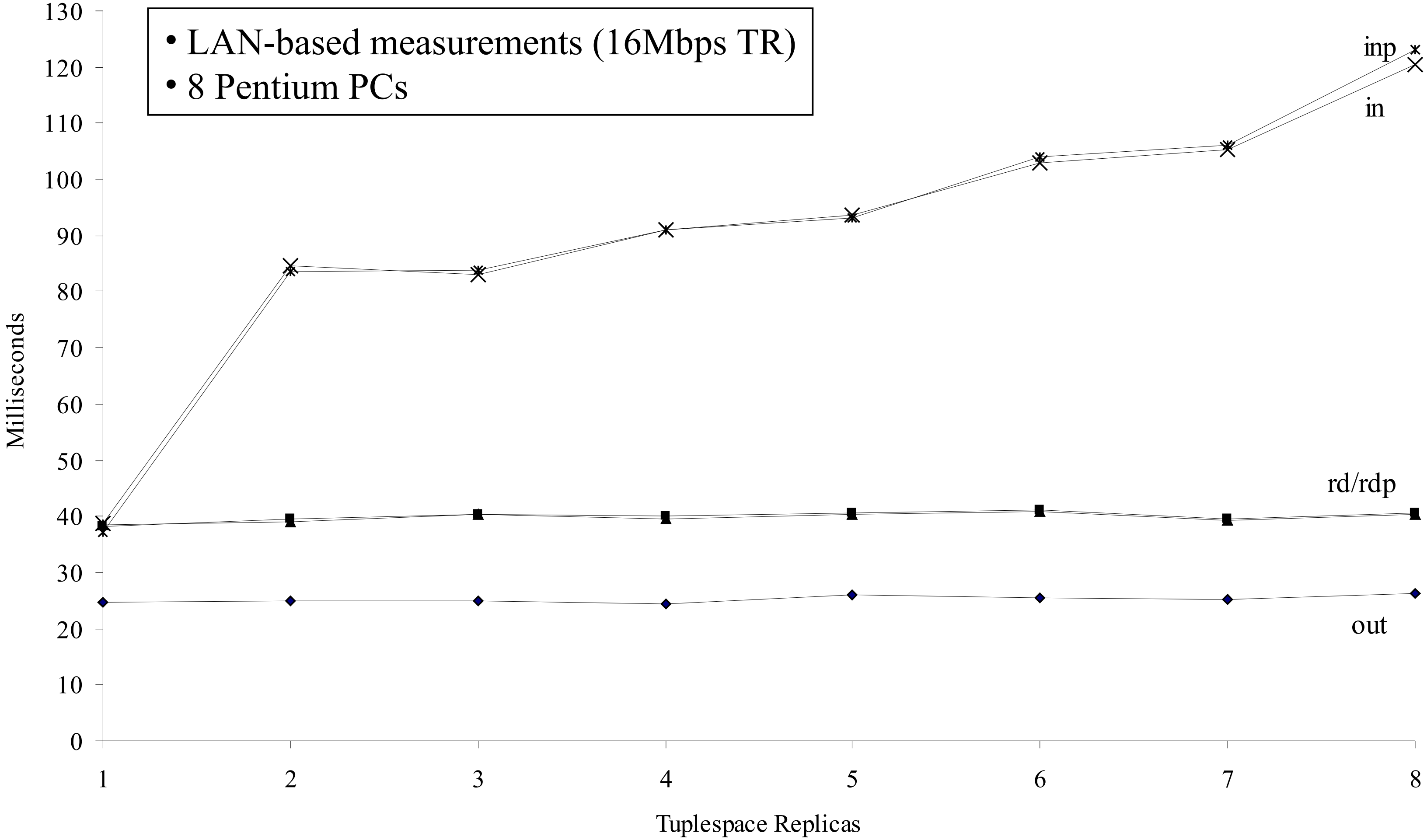
[CG86] S/Net Linda
[BS95] FT-Linda
[XL89]
[Kam91]
[CKM92] MTS

1) Centralized

2) Fault-tolerant tuplespace

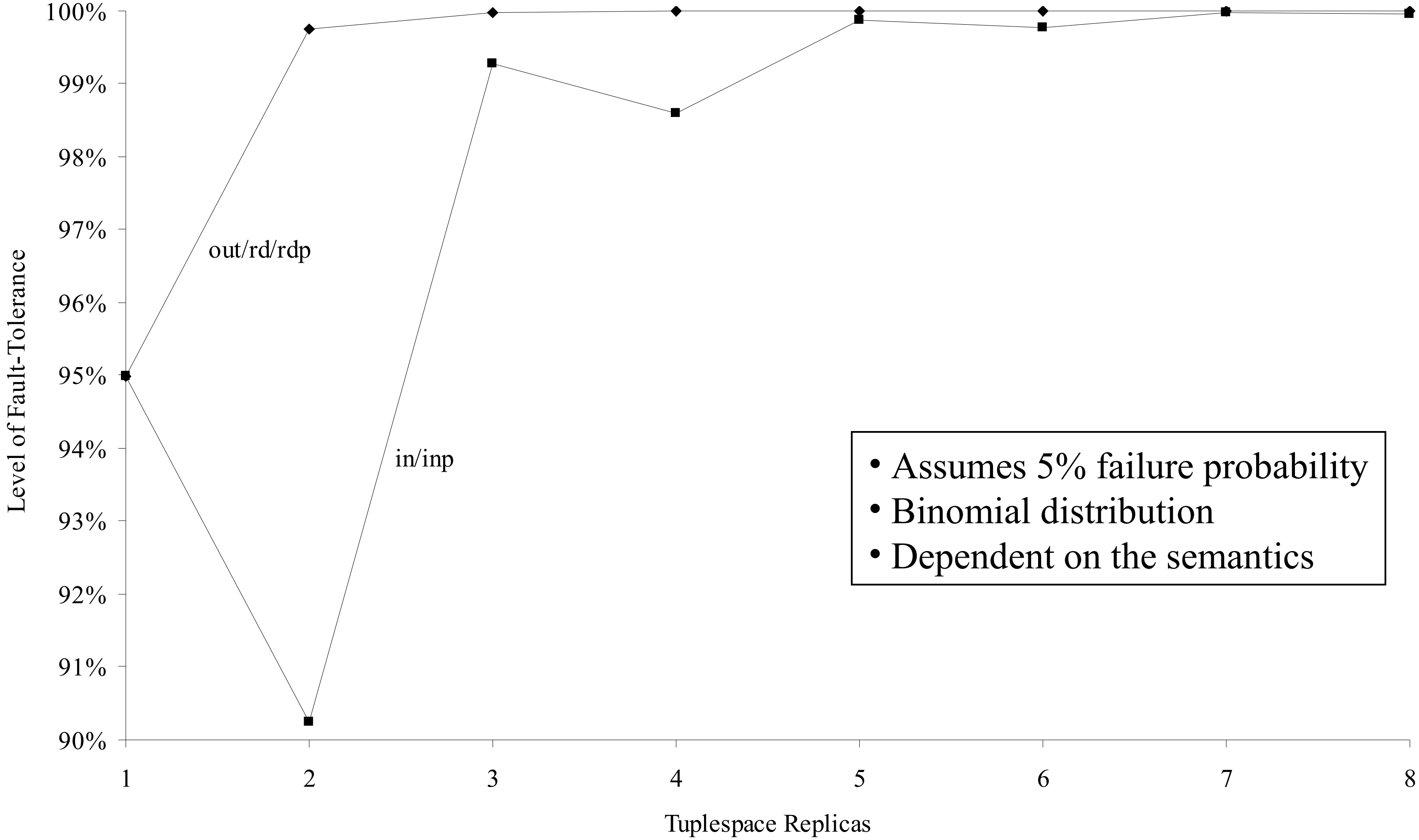
Cost of Fault-Tolerance

(Empirical)


- LAN-based measurements (16Mbps TR)
- 8 Pentium PCs

Gains of Fault-Tolerance

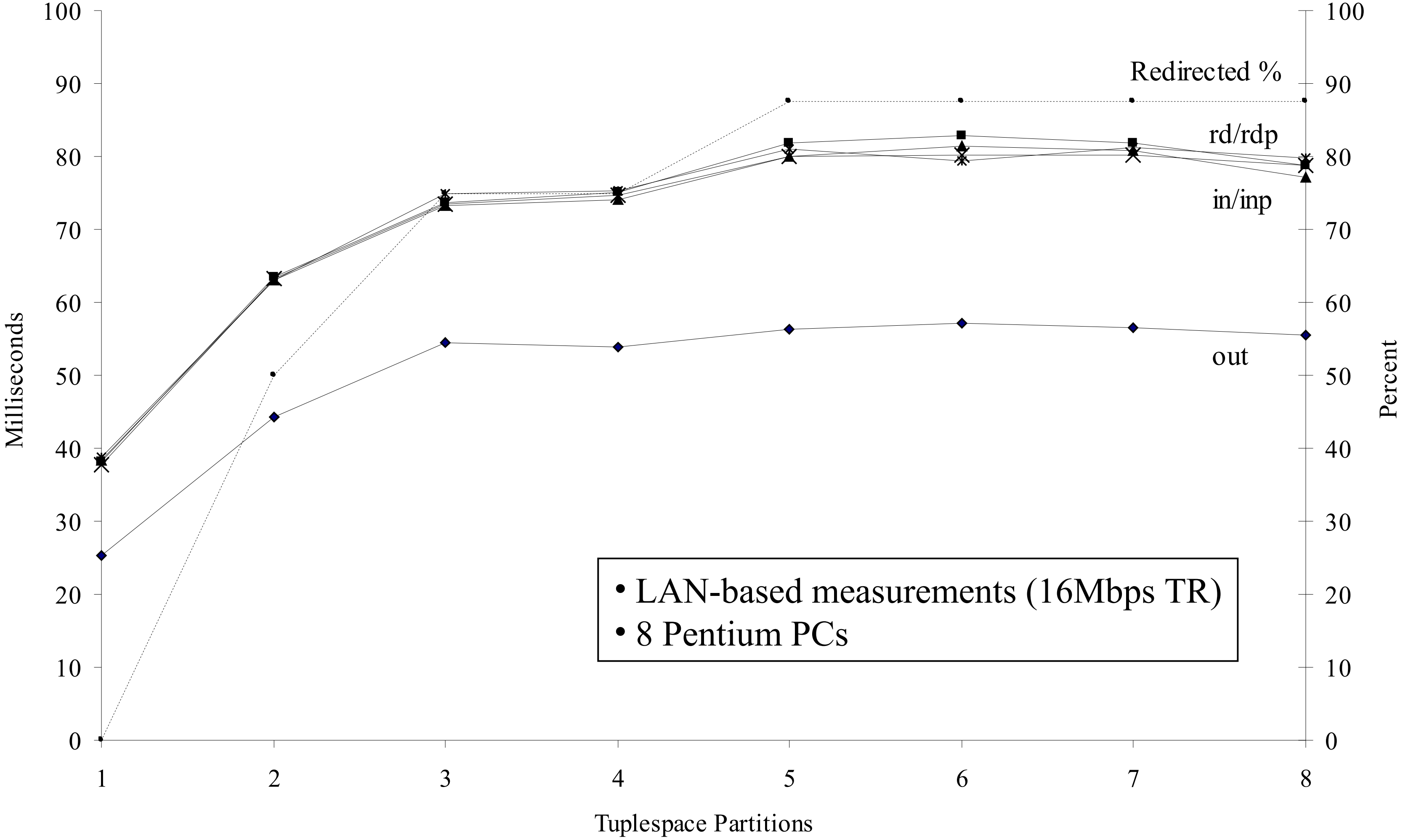
(Theoretical)



- Assumes 5% failure probability
- Binomial distribution
- Dependent on the semantics

Cost of Scalability

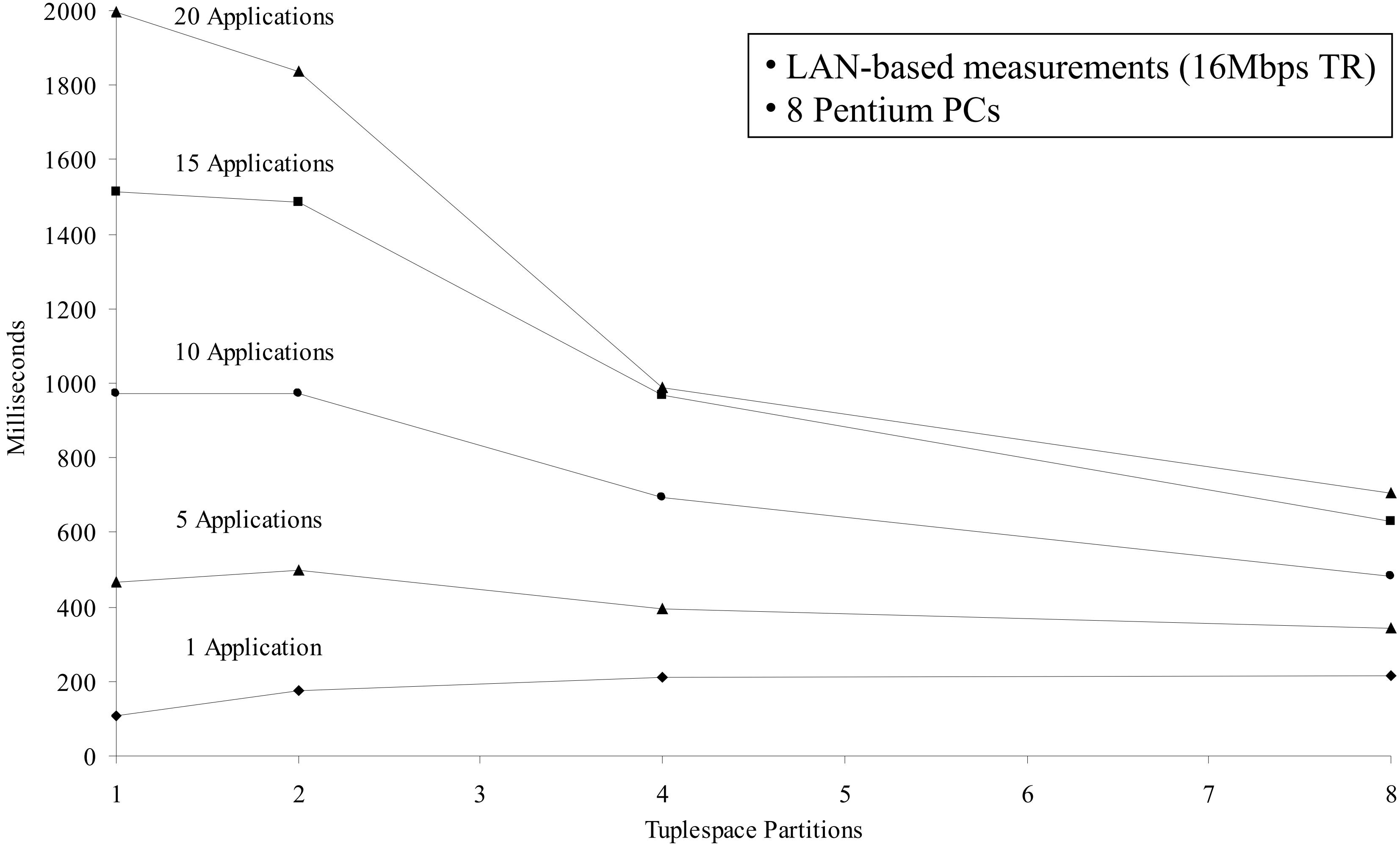
(Empirical)



- LAN-based measurements (16Mbps TR)
- 8 Pentium PCs

Gains of Scalability

(Empirical)



- LAN-based measurements (16Mbps TR)
- 8 Pentium PCs

Conclusion

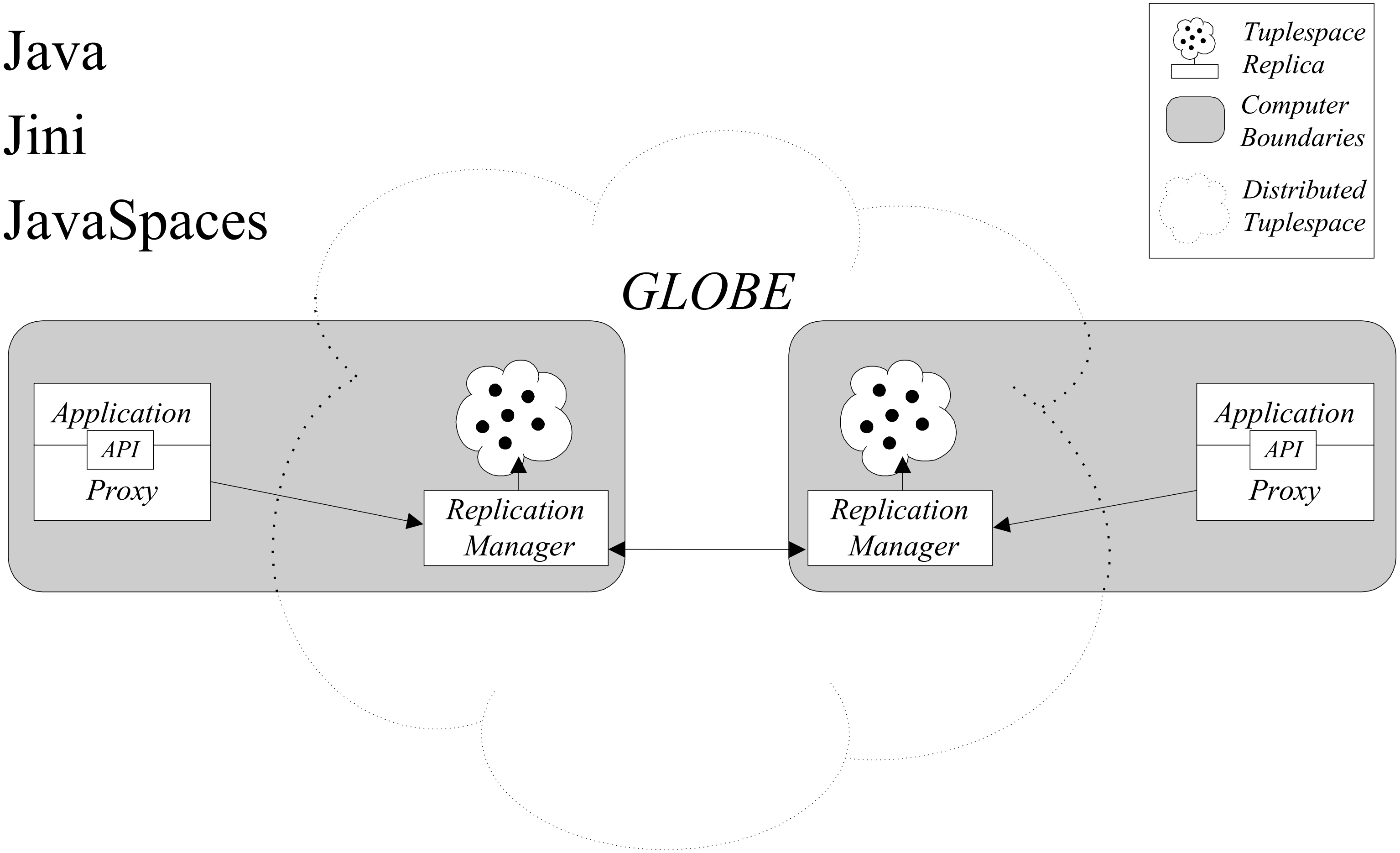
- Distributed Tuplespace Semantics is suitable
- Hybrid Replica Update Protocol is fast and ensures consistency
- Higher level of Fault-Tolerance
- Higher level of Scalability
- Fault-Tolerance and Scalability dynamically Adjustable
- Outperforms a highly loaded centralized Tuplespace

Future Work

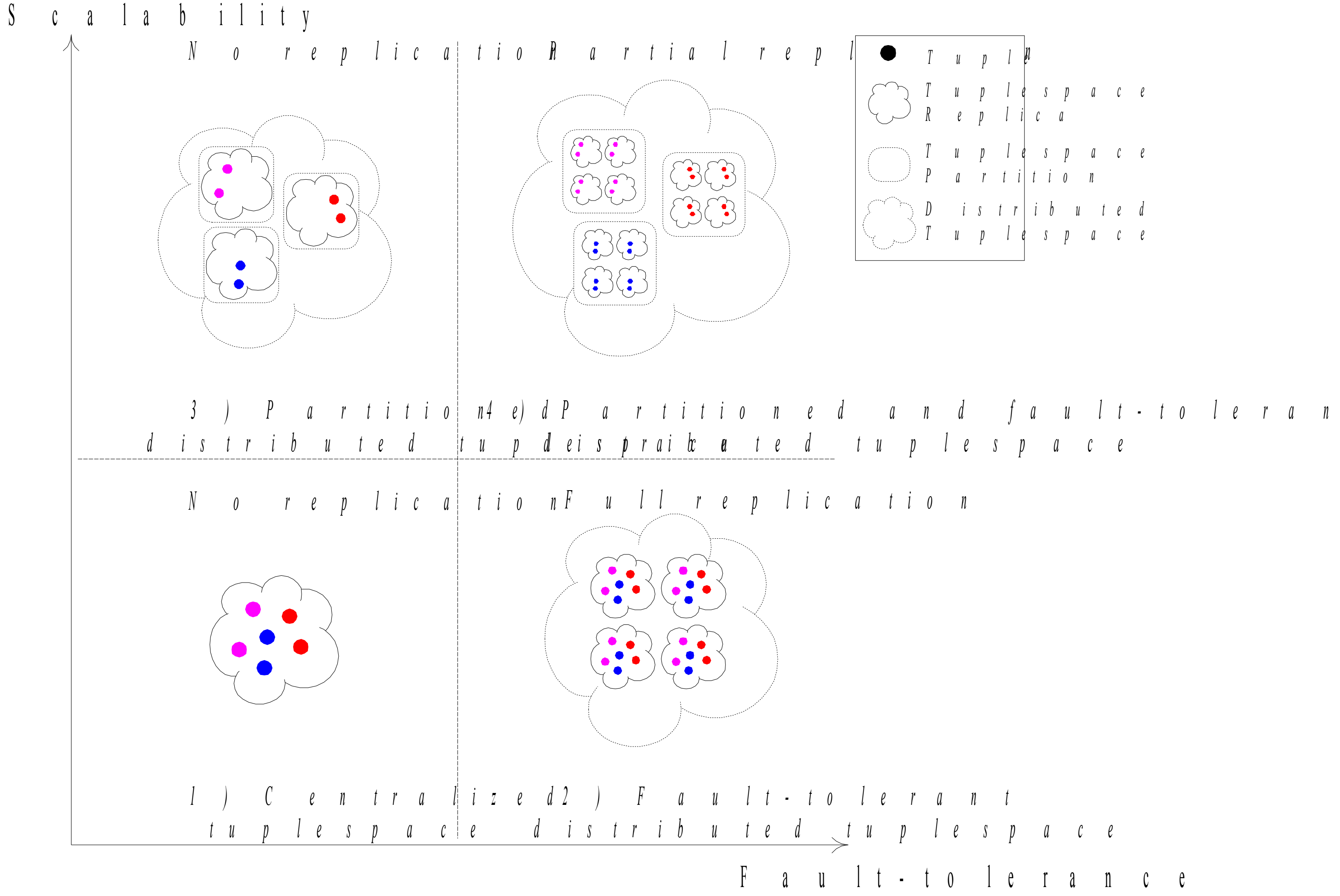
- GLOBE enhancements
  - Elimination/reduction of Redirection
  - Load-balancing of Applications
  - Implementation Optimizations
  - Additional Tuplespace Features

Demonstration

- Implementation:
  - Java
  - Jini
  - JavaSpaces



Fault-Tolerance & Scalability



Scalability

No replication

Partial replication

● Tuple

Tuplespace Replica

Tuplespace Partition

Distributed Tuplespace

3) Partitioned distributed tuplespace

4) Partitioned and fault-toleran

distributed tuplespace

No replication

Full replication

1) Centralized tuplespace

2) Fault-tolerant distributed tuplespace

Fault-tolerance