

# Fault –Tolerant Resource Discovery in Computational Grids

Ian Chang-Yen  
University of Louisiana at Lafayette



*The Center for Advanced Computer Studies*



## Agenda

- Project Scope
- Background & Existing Systems
- System Components & Design
- System Layout & Implementation
- System Integration & Operation
- Tests & Results
- Summary & Further Work

## [ Project Scope ]

- Suited for small groups with loose broadcast requirements (e.g. LAN)
- Combination with other architectural techniques – hybrid systems
- Currently implemented as a proof-of-concept test bed
- Possible integration with existing resource discovery systems in the future (MDS)

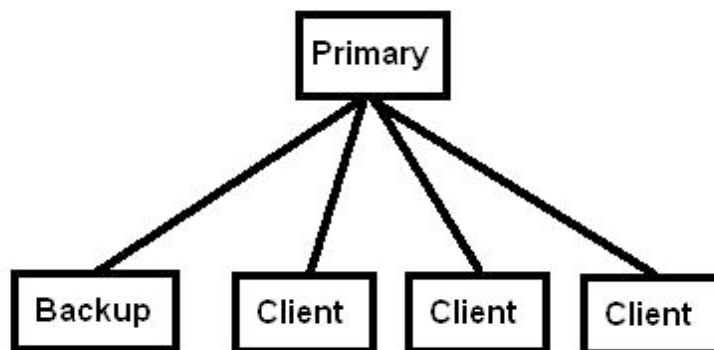
## [ Background & Existing Systems ]

- Architecture types:
  - Fully Centralized
  - Fully Decentralized
  - Hybrid
- Agent-based – “ants”, Organic Grid
- Peer-to-peer approaches –Gnutella
- MDS, Condor-G, AppLeS

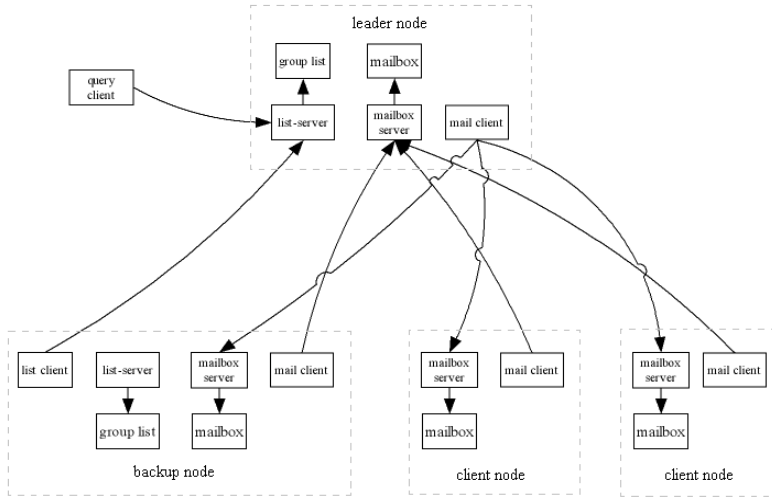
## [ System Components & Design ]

- Node types:
  - Leader
  - Backup
  - Client
- Principal inter-node communication and system queries via UDP datagrams & mailboxes
- Large data transfers via TCP connections
- Group lists maintained as compressed objects – facilitates transfer between hosts

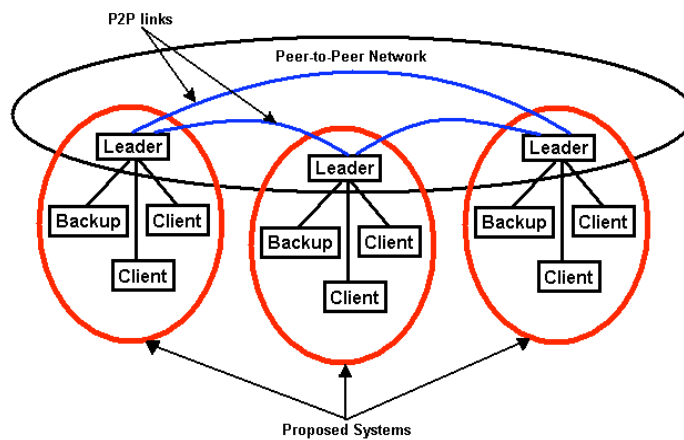
## [ System Layout ]



# System Implementation



# System Integration



Possible System Integration Scenario

## [ System Operation (1) ]

- Operating requirements:
  - Broadcast (suited for LAN environments)
- Node roles:
  - Leader – maintains master group list
  - Backup – synchronized copy of list
  - Client – register with leader
- Basic membership - soft-state registration
- Dynamic recovery – hot standby

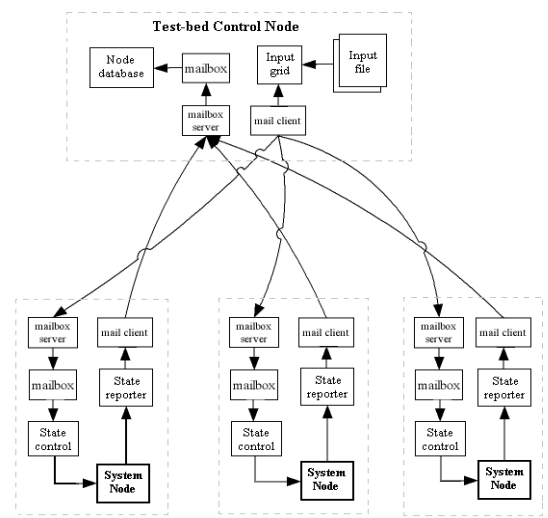
## [ System Operation (2) ]

- Dynamic recovery:
  - Automatic backup determination
  - Leader-backup synchronization
  - Backup can sense leader failure, assume leadership
  - Advantage: Self-assembling structure

# Test Environment (1)

- Proof-of-Concept Test-bed built in Java©
- Centralized reporting system
- Simulated with 10, 15 & 20 node group sizes
- Measured traffic and settle time
- Failure scenarios tested:
  - Cascade failure test
  - Random failure test
  - Batch failure test
- Static system tested for comparison

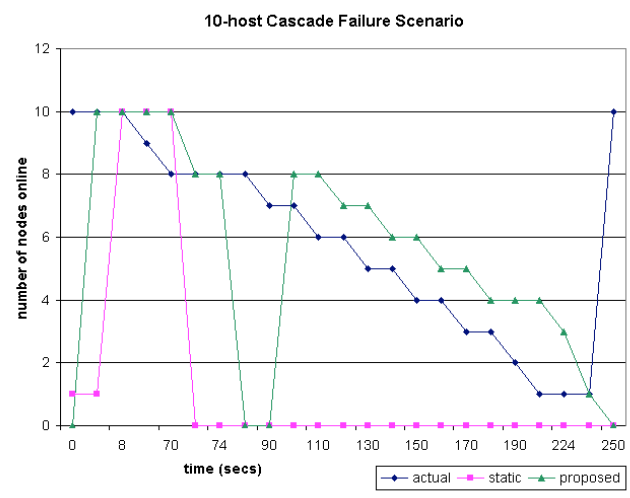
# Test Environment (2) - Layout



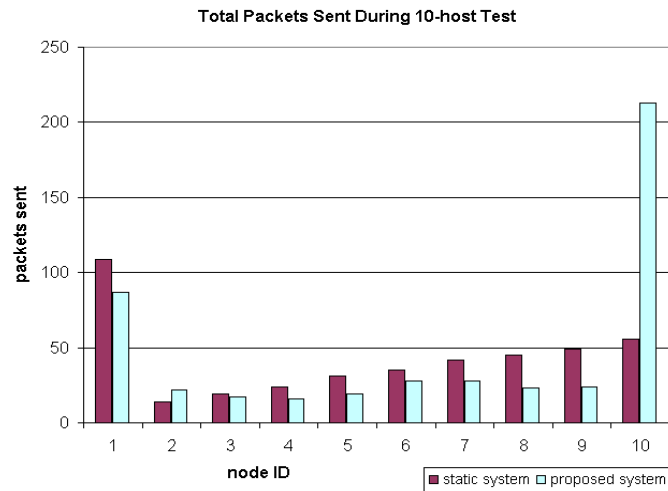
# Test Results (1)

- Results drawn primarily from cascade tests
- Observations of system:
  - Reported resources matched true resources more closely than static system
  - Controllable settle time
  - Reduced volume of traffic compared to static system
  - Leader contention issues in larger group sizes

# Test Results (2) - Output



## Test Results (3) - Traffic



## Summary & Further Work

- Proposed system:
  - More resistant to failure
  - Less traffic produced
- Planned improvements:
  - Improve leader contention
  - Optimize settle time of nodes
  - More failure scenarios

## [ Acknowledgements ]

- Dr. N. F. Tzeng – research guidance
- Computer Architecture and Network (CAN) Lab, Center for Advanced Computer Studies – testing facilities
- NSF Grant
- US Department of Energy Grant
- Louisiana Board of Regents Contract