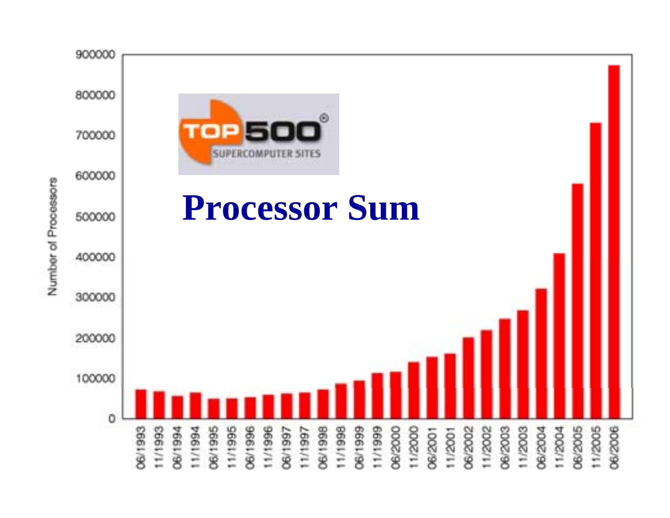
Scalable Fault Tolerant Protocol for Parallel Runtime Environments

Thara Angskun, Graham E. Fagg, George Bosilca,
Jelena Pjesivac-Grbovic, and Jack J. Dongarra

Euro PVM/MPI 2006
(09/19/06)

INNOVATIVE COMPUTING LABORATORY
COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF TENNESSEE

HPC Trend



- Increase number of processors
- Decrease MTTF
  - Dynamic Environment

- Parallel Runtime Environment
  - Extension of OS services for message passing library or application development
  - **SCALABLE** and **FAULT-TOLERANT**

Parallel Runtime Environments

- MPI runtime environments
  - Start / terminate jobs
  - Transfer signals (e.g. Ctrl-C)
  - Redirect STDIN, collect stdout / stderr
  - Collect exit status
  - Monitoring job status
  - (Optional) Interface with debugger, scheduler etc.
- Communication Protocol
  - Handle multiple types of message transmissions
    - Broadcast, Multicast, Unicast
  - **SCALABLE** and **FAULT-TOLERANT**

MPI Runtime Environments

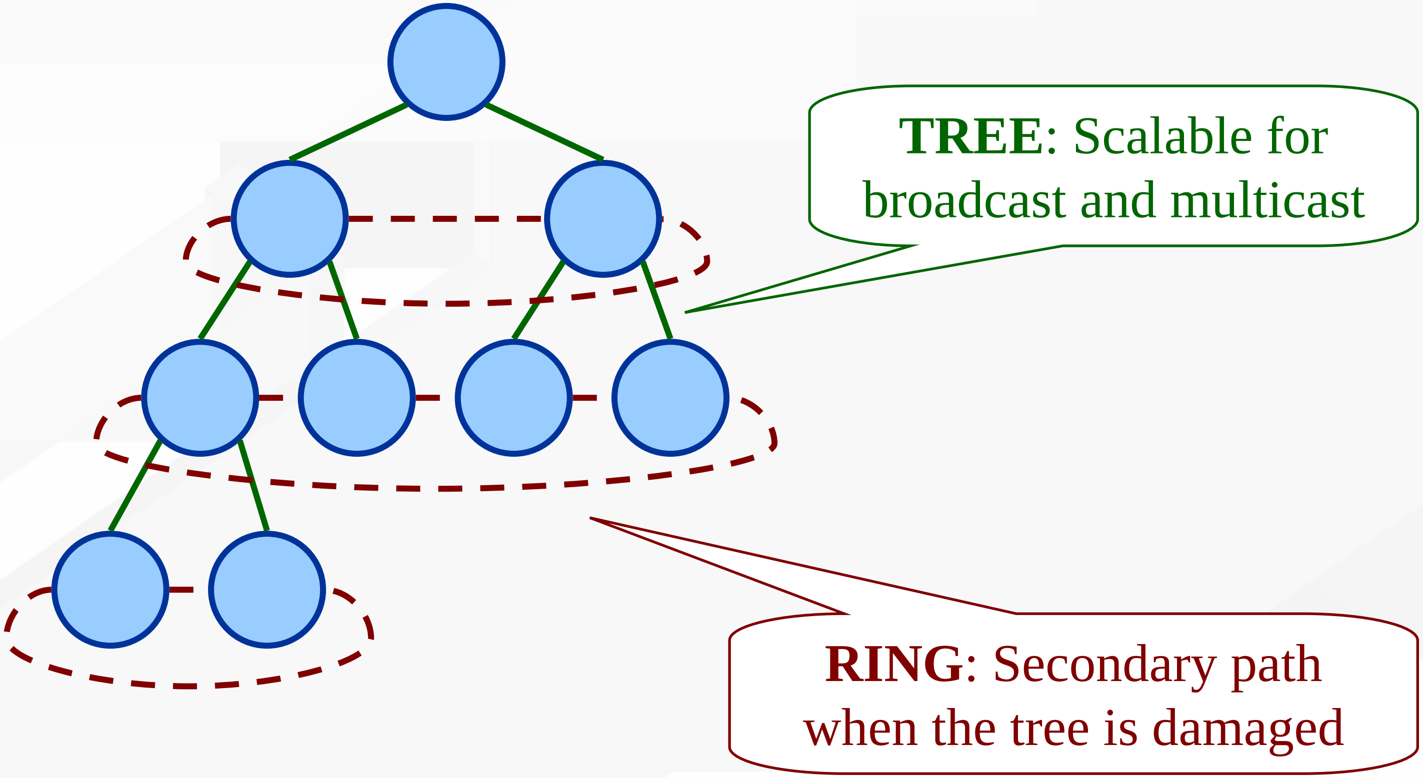
- MPICH2 - MPD (Multi-Purpose Daemon)
  - Ring or Tree topology
- Open MPI – Open RTE
  - Linear
- LAM/MPI – LAM
  - Linear
- FT-MPI – HARNESS
  - Linear

Scalable and Fault-Tolerant Issues

- Structured peer-to-peer networking
  - Based on distributed hash tables
    - CAN, Chord, Pastry, Tapestry
  - Focus on resource discovery (Unicast)
- Sensor or large scale ad-hoc networking
  - Based on gossiping (epidemic algorithm)
  - Focus on information aggregation.

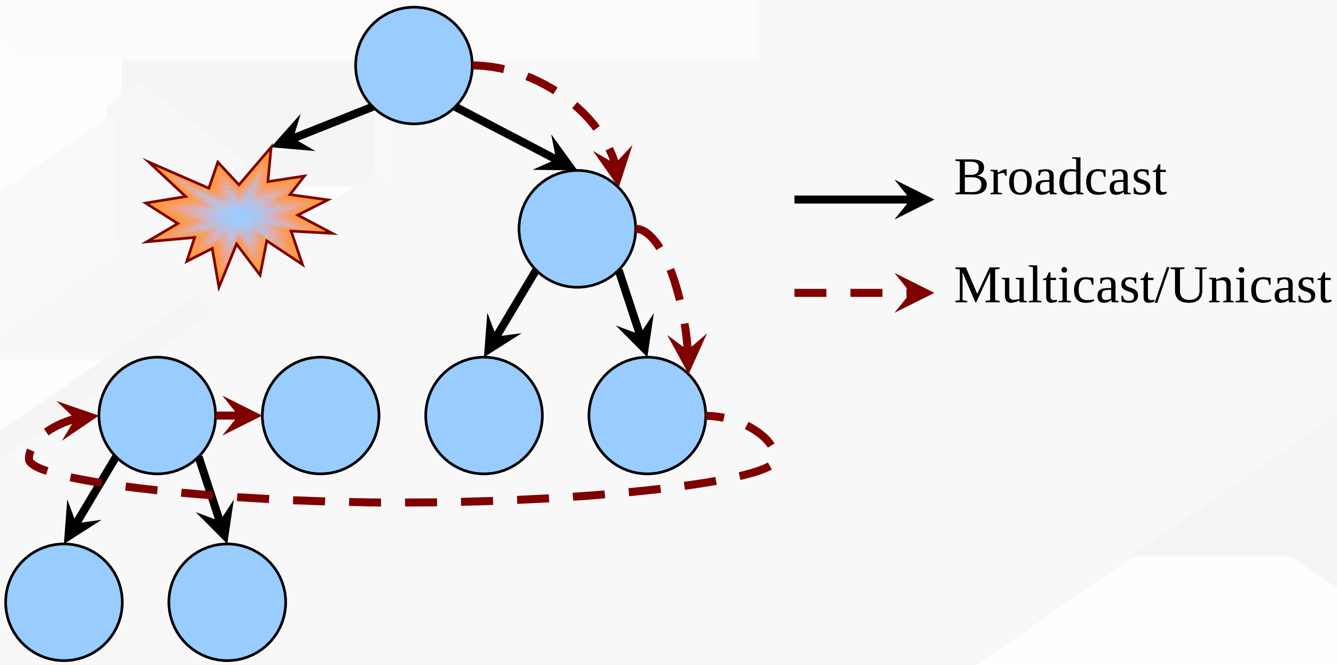
Scalable and Fault-Tolerant Protocol

- Based on k-ary sibling tree
  - K is number of fan-out (k ≥ 2)



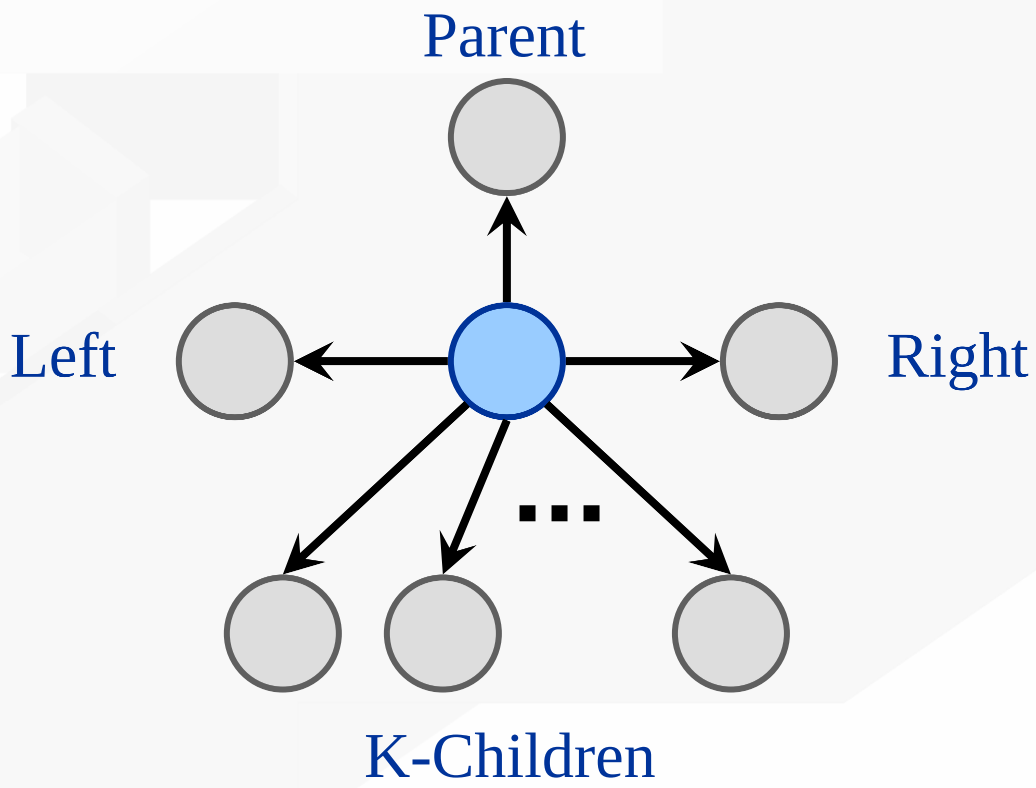
Scalable and Fault-Tolerant Protocol

- Example : survives a failure
  - A broadcast message is encapsulated in a multicast message sent from parent to children of a dead node.



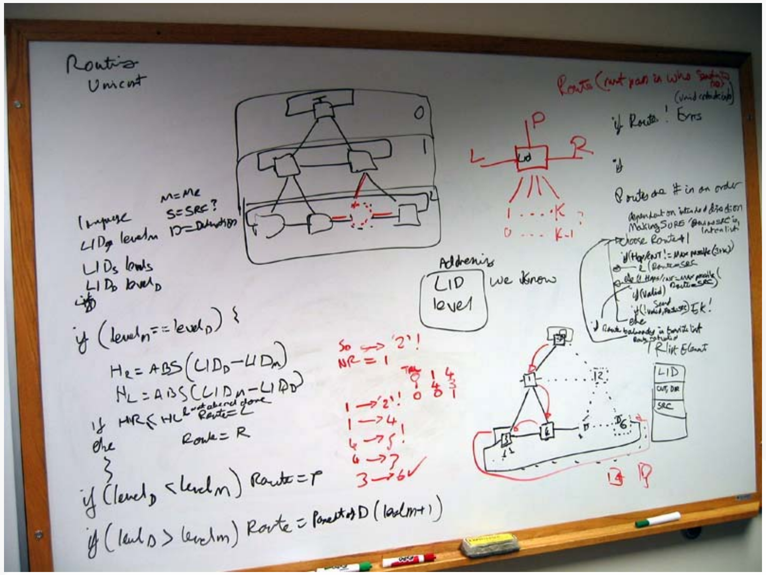
Scalable and Fault-Tolerant Protocol

- Low storage cost
  - Each node needs to know
    - the contact information of at most k+3 neighbors
    - State of the link to its neighbors



Scalable and Fault-Tolerant Protocol

- Protocol Specification
  - Service Specification
  - Environment Assumption
  - Protocol Vocabulary
  - Message Format
  - Procedure Rules



Protocol Specification

- Service Specification
  - Deliver broadcast, multicast, unicast
  - Normal circumstance
    - Uses the k-ary tree to send messages
  - Failure cases:
    - Uses the neighbor to reroute messages
  - Best effort routing

Protocol Specification

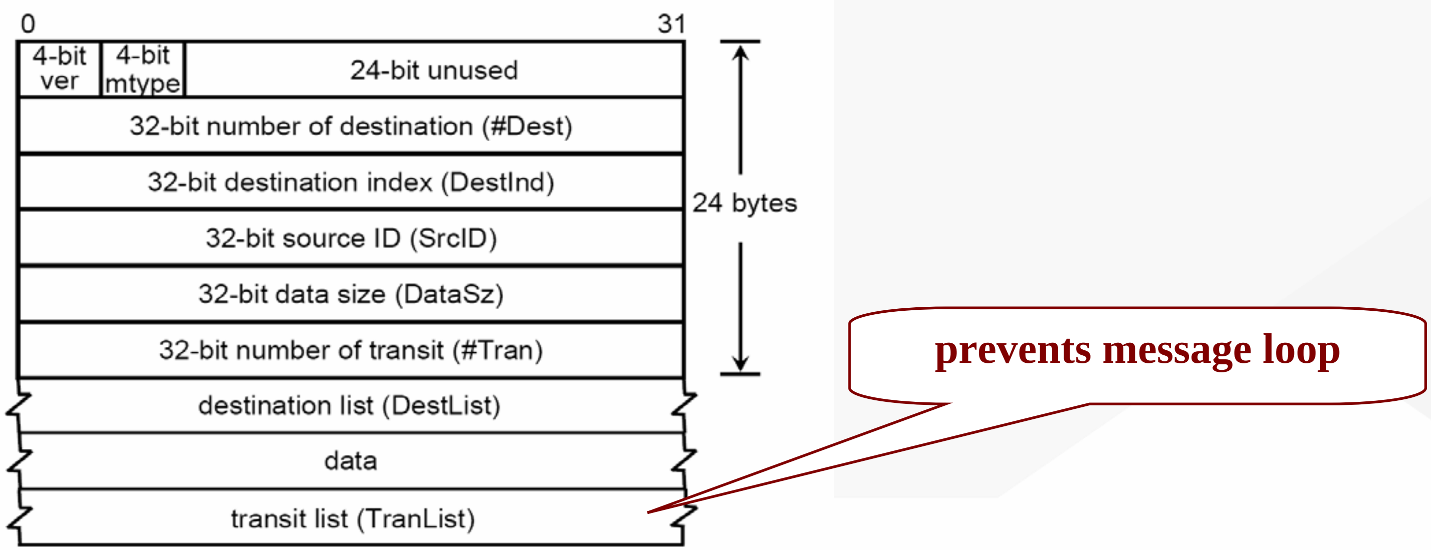
- Environment Assumption
  - Failures
    - Assumes Fail-stop (rather than Byzantine)
    - At least one neighbor of each node should be alive
      - Unless allow each node to contact a directory service
  - Transmission channel
    - Can detect and recover from transmission error
      - E.g. TCP, Reliable UDP
    - Consequence: never lose a message
      - Unless message is destroyed with a node before being pass on

Protocol Specification

Introduction
Background
Design
Verification
Results
Conclusion

- Protocol Vocabulary
  - Hello – Initialize messages (construct k-ary tree)
  - Mcast – Multicast messages (including Unicast)
  - Bcast – Broadcast messages

- Message Format

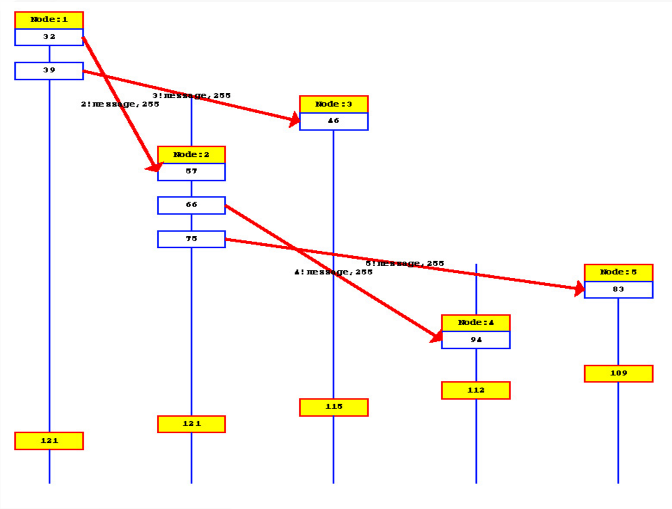


Protocol Specification

- Procedure Rules:
  - Initialization
    - Register itself to the directory service
    - Get its logical ID
    - Send hello to Parent, Left
    - (and to Right if the right most in each level)
  - Routing (best effort)
    - Bcast: send to all of its children
      - If a child died: encapsulate in Mcast and reroute to its grand children
    - Mcast: send to a valid neighbor (highest priority)
      - Otherwise backtrack to sender
    - ETC...

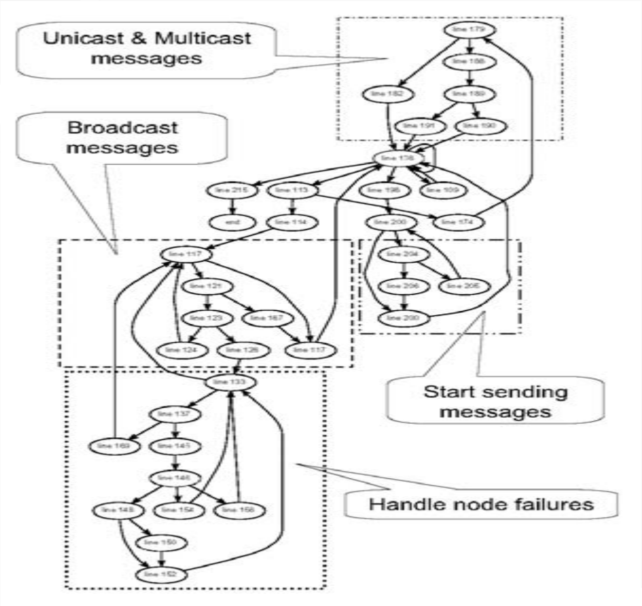
Protocol Verification

- SPIN verification (and simulation) tool
  - Model checker using automata-theoretical.
  - Deadlocks, non-progress cycle, non-reachable state, etc.
  - Provide counterexample in error cases.
  - PROMELA (Process Meta Language)



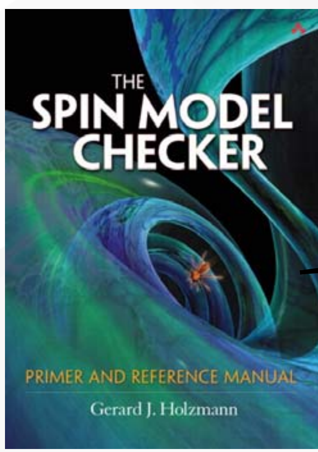
Protocol Verification

- Specifying the Protocol in PROMELA
  - Model broadcast with exclusive channels
  - Failures is simulated with non-deterministic selection (‘if’ selection construct)
  - Speedup with ‘atomic’ construct



Protocol Verification

- Verification Results
  - No deadlock, livelock, invalid end state
  - No unreachable codes and assertion violation



"I am SPIN and I approve this protocol"

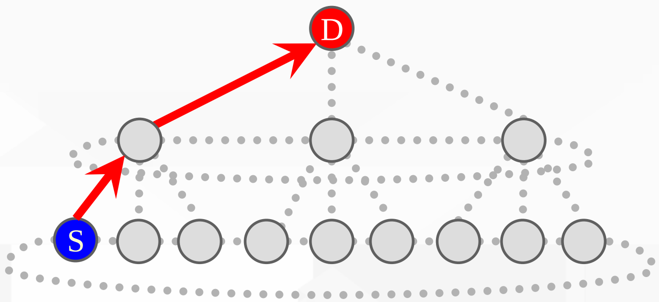
Experimental Results

- Routing Algorithms
  - 1) Basic
    - Fixed first hop based on static topology
    - Rule based method to estimate cost
    - Known locally failed links

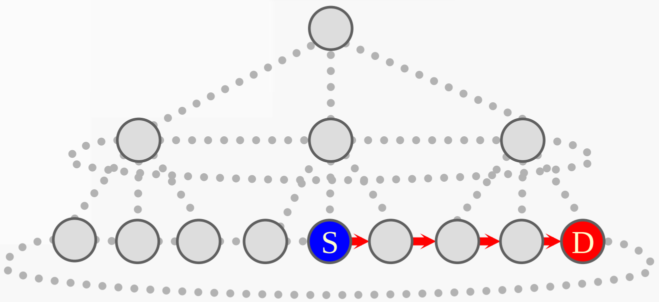
```
if my level = destination level then
        Send to left/right
else if my level > destination level then
        Send to my parent
else
        if my child is an ancestor of destination then
                Send to the child
        else
                Send to left/right who is closer to
                an ancestor of the destination in my level
```

Experimental Results

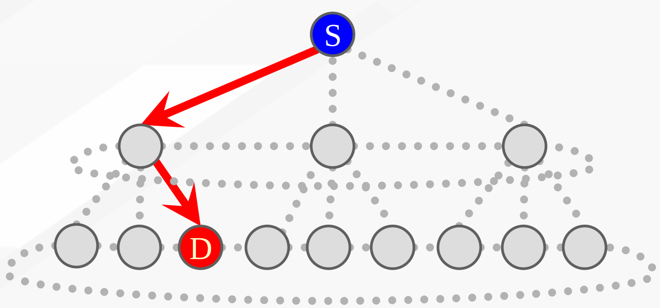
» Basic routing examples



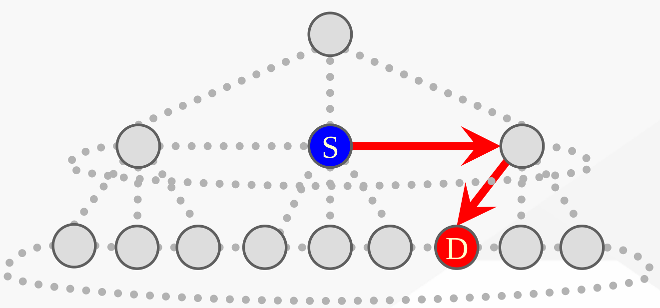
Destination Level above Sender Level



Destination Level = Sender Level



Case 1



Case 2

Destination Level below Sender Level

Experimental Results

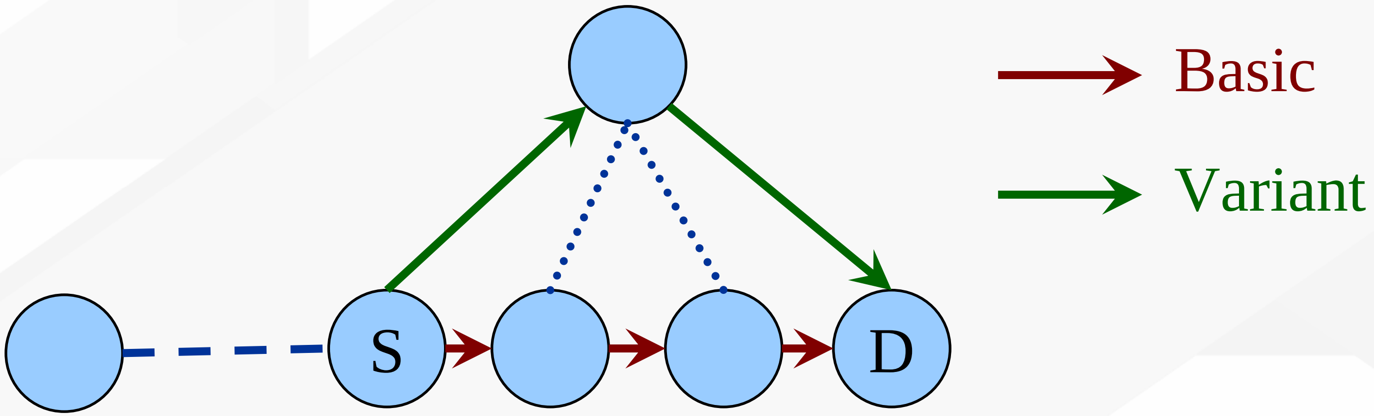
- Introduction
- Background
- Design
- Verification
- **Results**
- Conclusion

» Routing Algorithms

  » 2) Variant (of 1)

    » Based on ordering of current possible hops to shorten distance

    » (i.e. allows to go in a direction that does not toward destination)



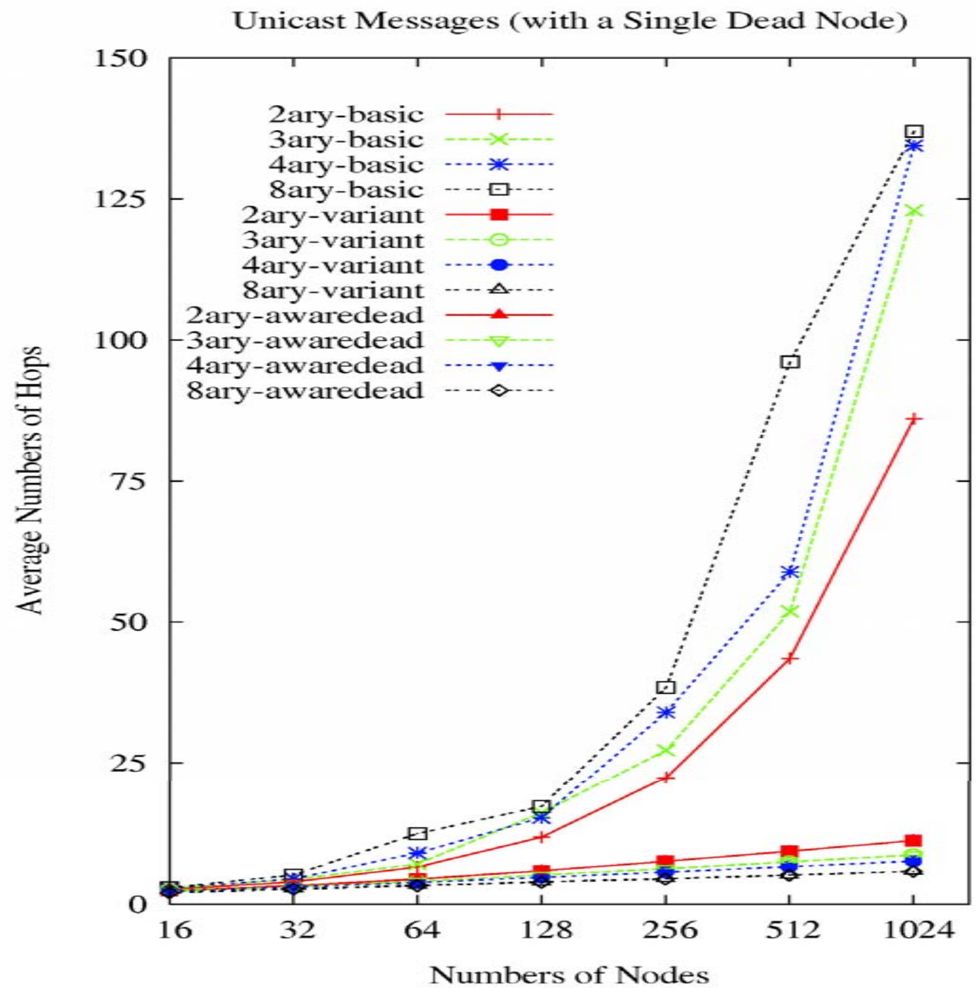
Experimental Results

- Routing Algorithms
  - 3) Breadth first search
    - Graph-coloring which explore only alive nodes
    - Use knowledge of Previously detected dead nodes

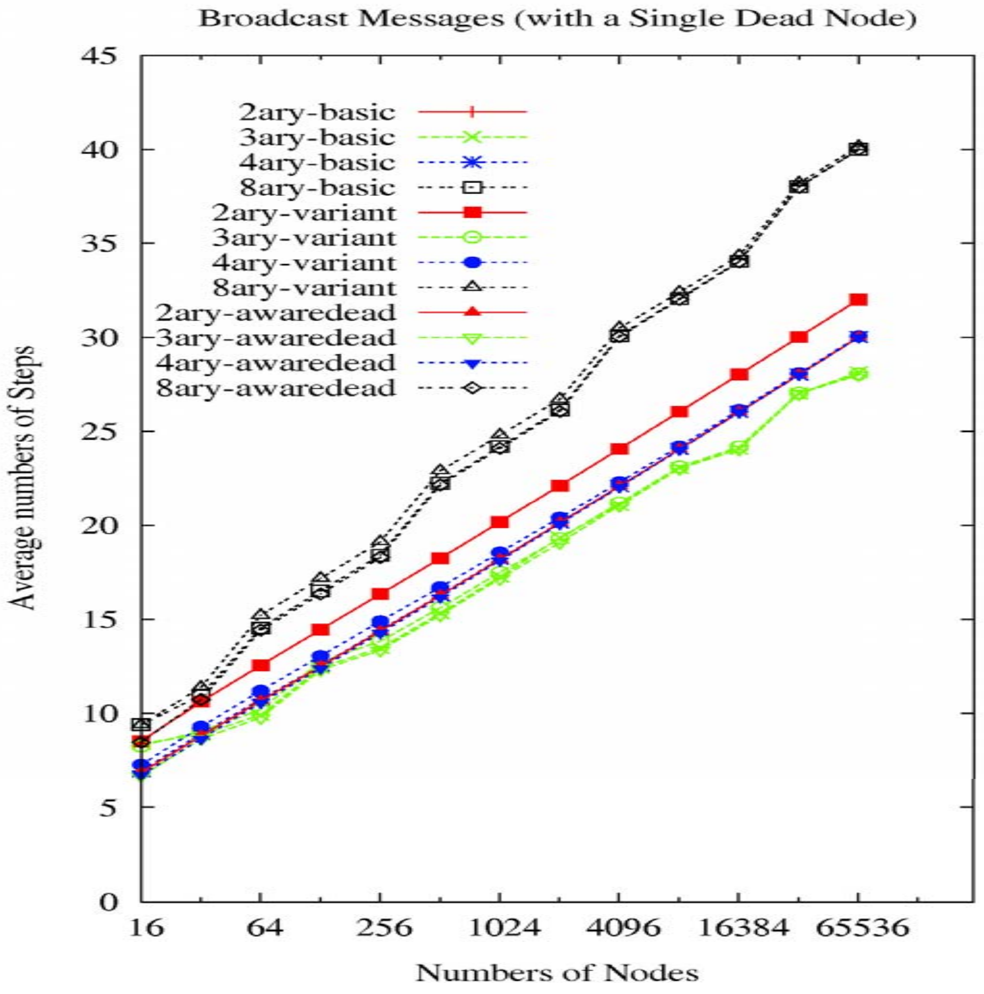
    - Note: more accurate, but time consumption

Experimental Results

- Introduction
- Background
- Design
- Verification
- **Results**
- Conclusion



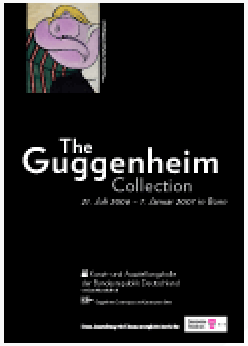
Experimental Results



Conclusion and Future Work

- Scalable and Fault-Tolerant Protocol
  - Designed for parallel runtime environments
  - Formally proven to work (normal and failure)
- Future Work
  - Protocol aware underlying network topology
    - Add a function cost on each path
  - Faster and more accurate re-routing algorithm
  - Basic message distribution of Harness/Open RTE ala. FT-MPI/Open MPI

Please don't forget the excursion at 4 PM ☺



Thank You / Danke schön

For more information:
angskun@cs.utk.edu