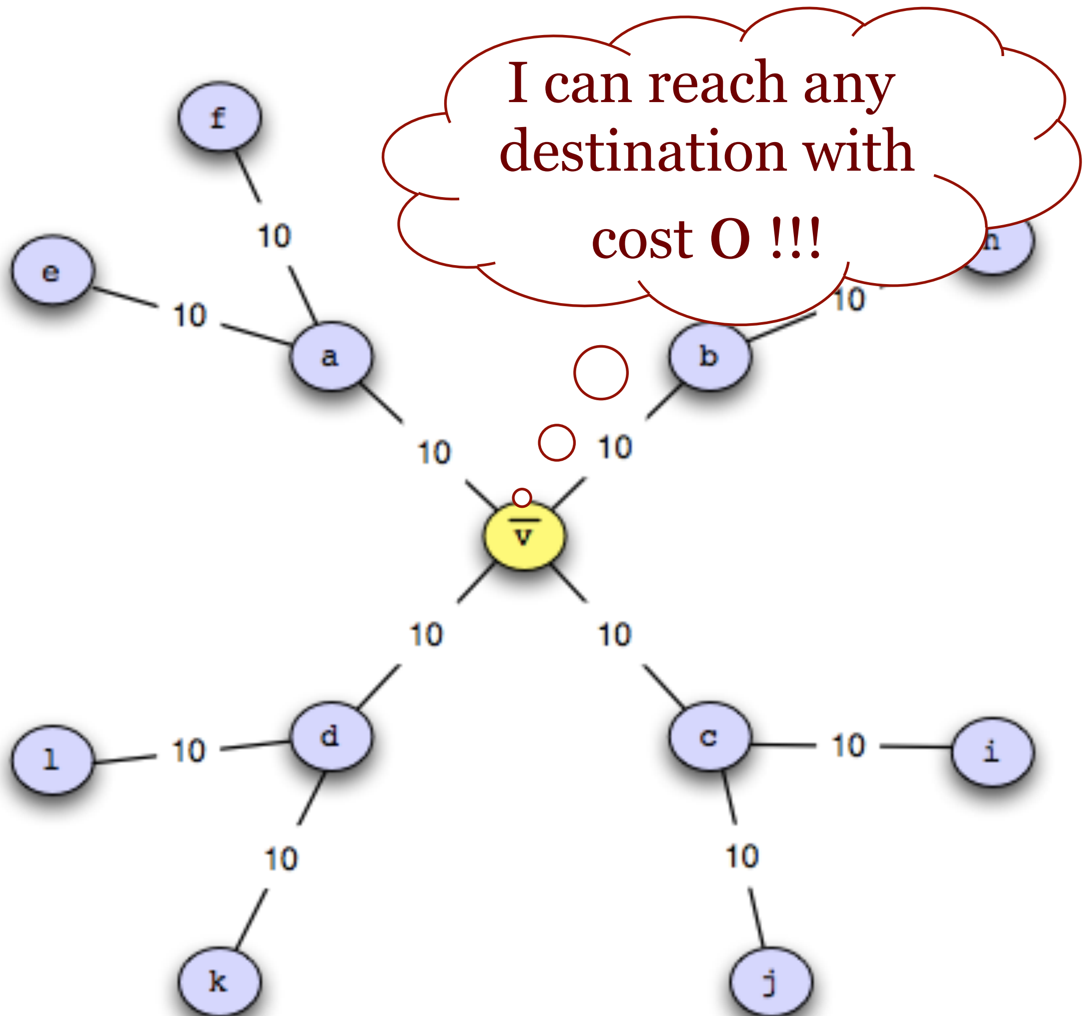
UMassAmherst

“Efficient Recovery from False State in Distributed Routing Algorithms”

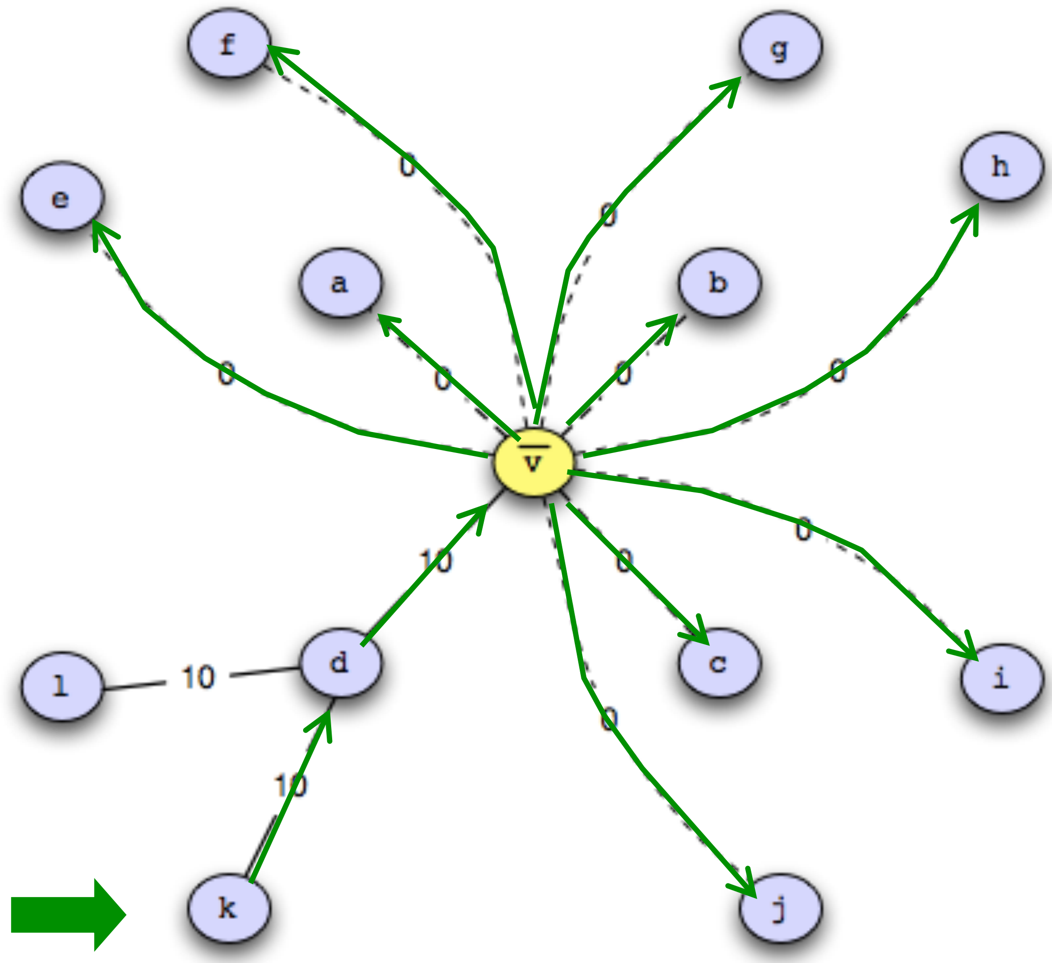
Daniel Gyllstrom, Sudarshan Vasudevan, Jim Kurose, and Gerome Miklau

University of Massachusetts, Amherst USA

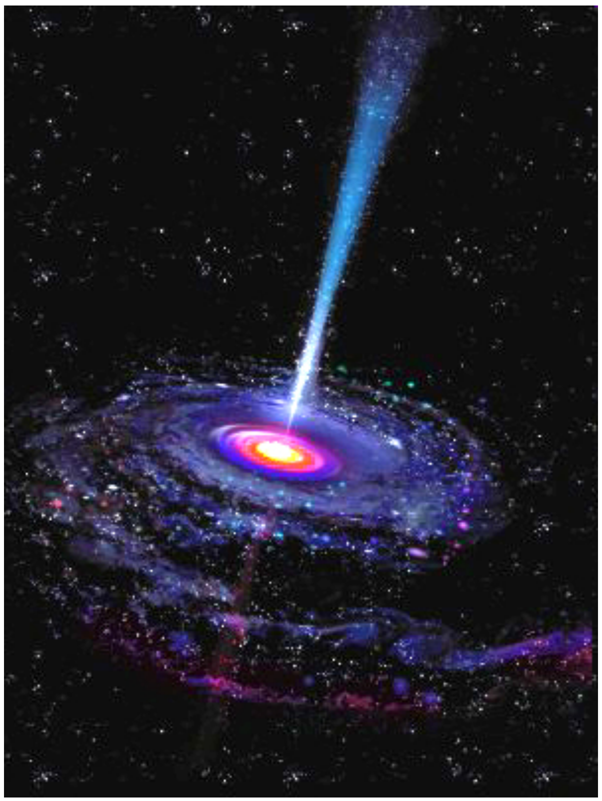
The Problem

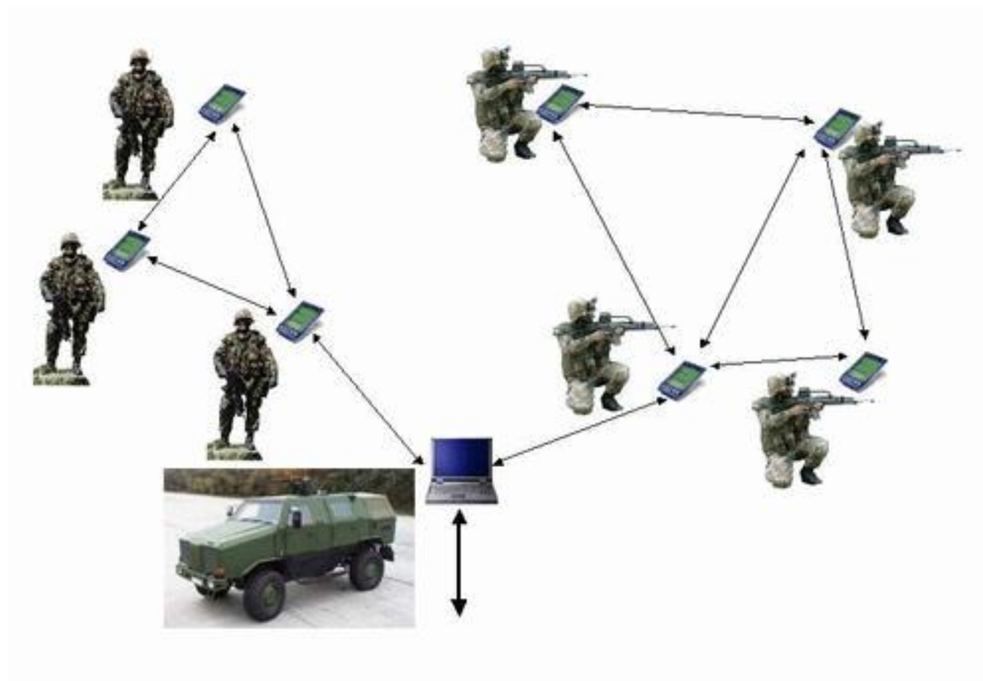


The Problem



The Problem





The Problem

Detecting compromised nodes *(see paper for references)*

Recovering after compromised nodes detected

Problem Statement

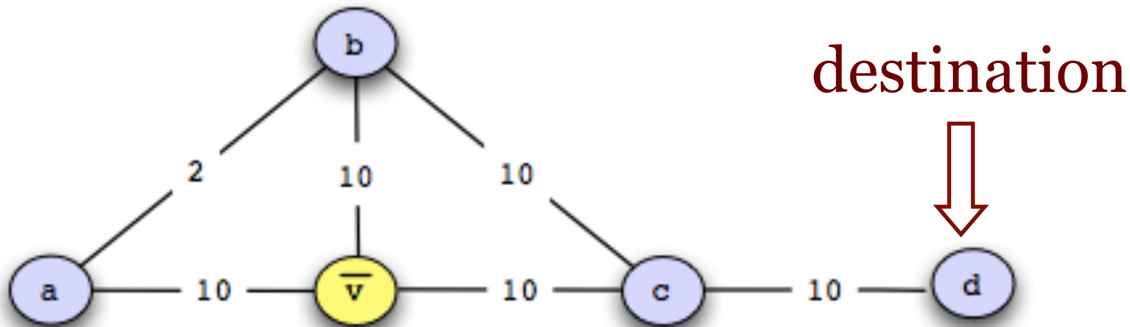
1. Nodes share false routing state
2. Outside algorithm identifies compromised nodes
3. Recover
   a. Remove compromised nodes from graph
   b. Compute least cost paths that route around compromised nodes

Assumptions

Synchronous communication model

Single compromised node

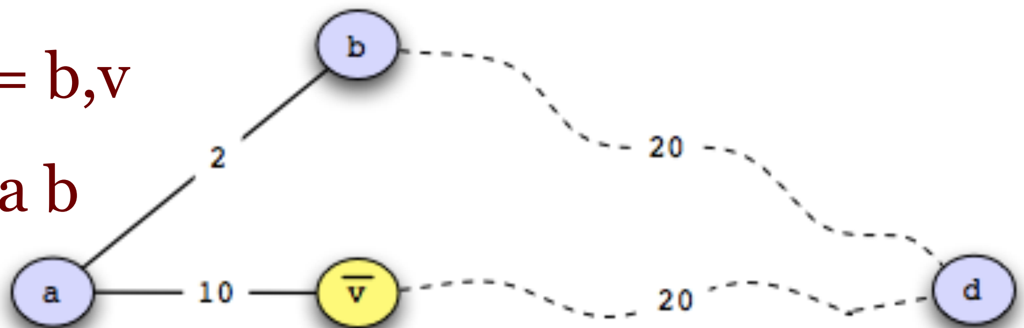
Distance Vector Routing Review



Node a’s Perspective

<u>Node a’s state</u>

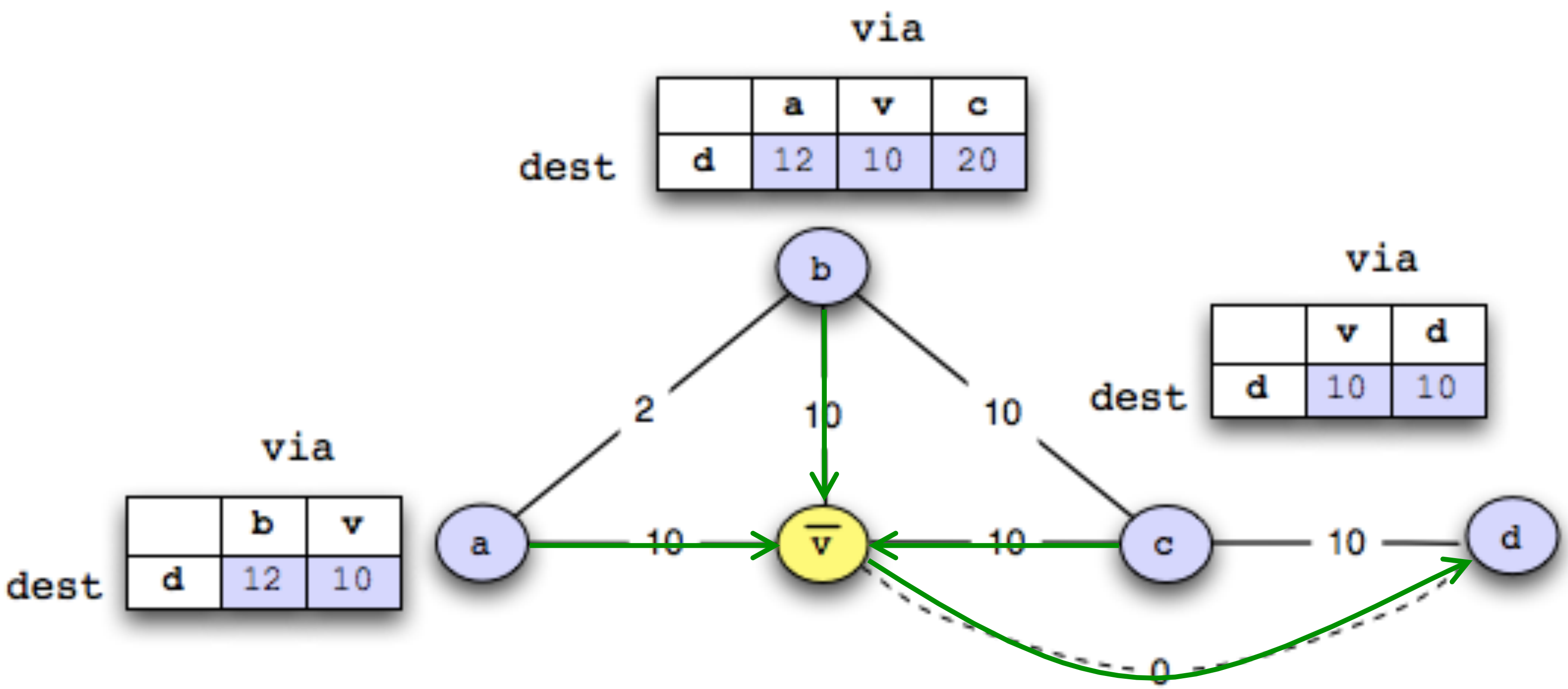
1. Neighbors = b,v
2. Cost to d via b and v



Our Recovery Algorithms

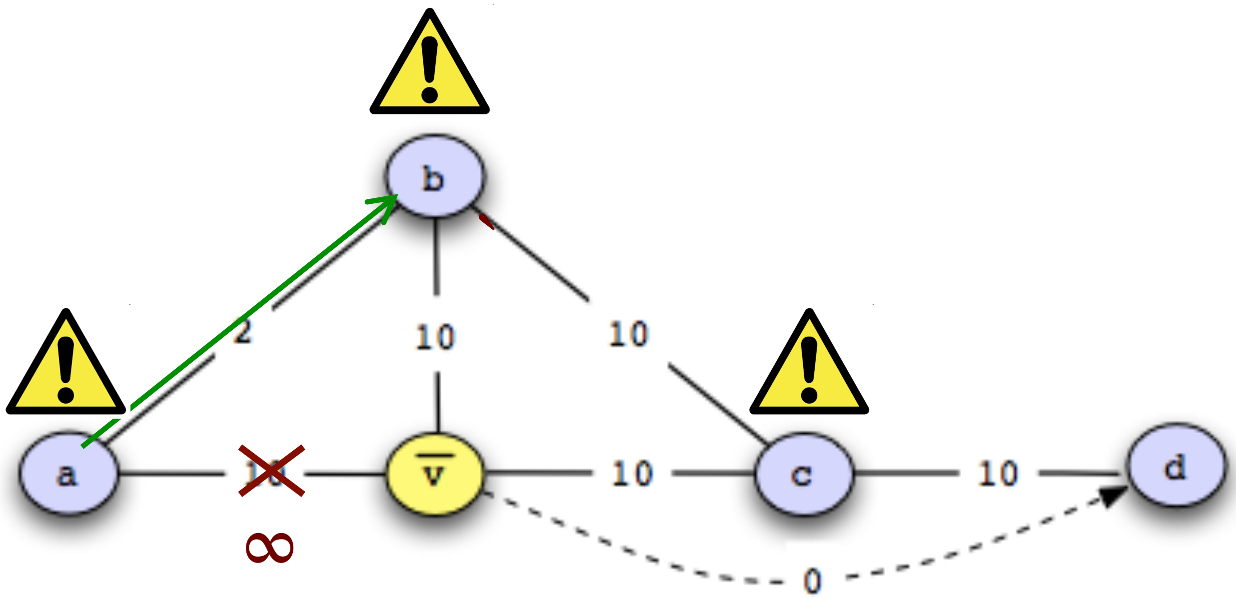
1. 2^{nd} *Best* Algorithm
2. *purge* Algorithm
3. *cpr* Algorithm

Running Example



2^{nd} *Best* algorithm

1. Remove $\overline{v}$ as a destination
2.

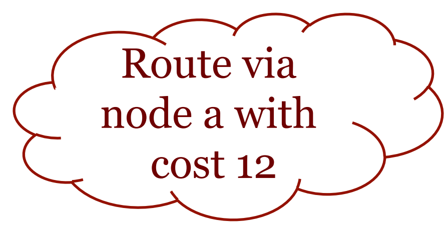


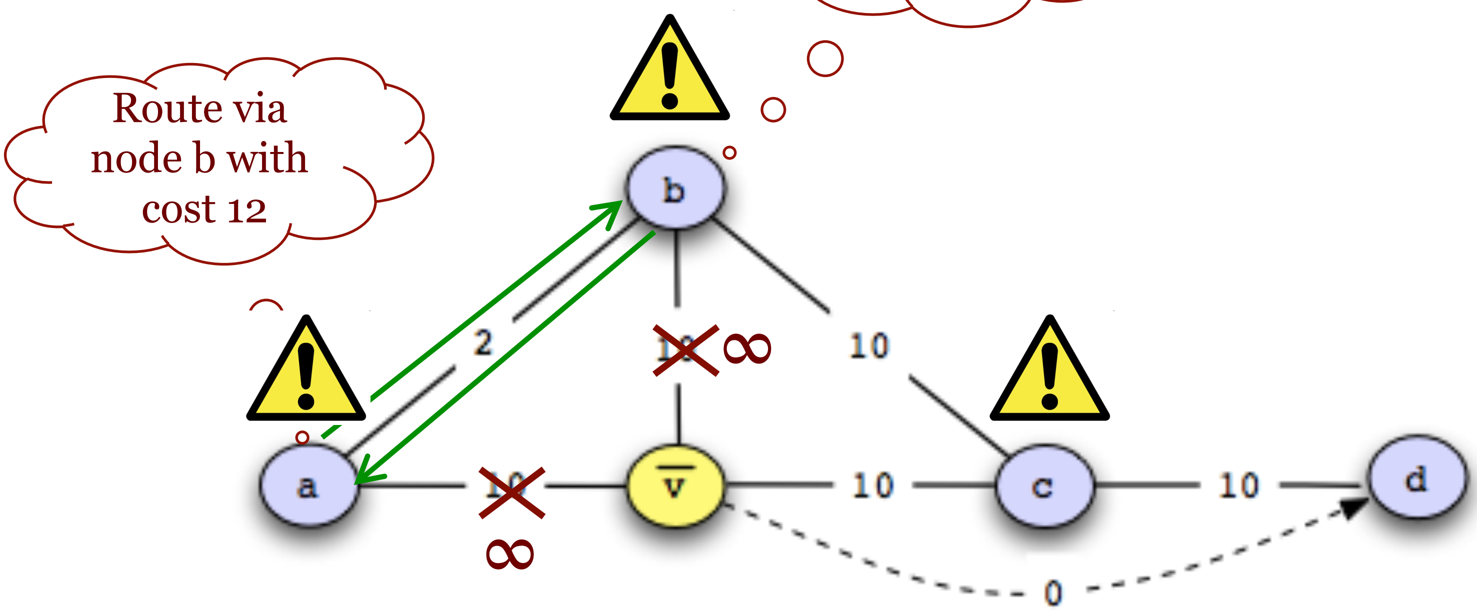
3. Run Distance Vector

2nd *Best* Algorithm

Strengths = Simple + no synchronization

Weakness = loops

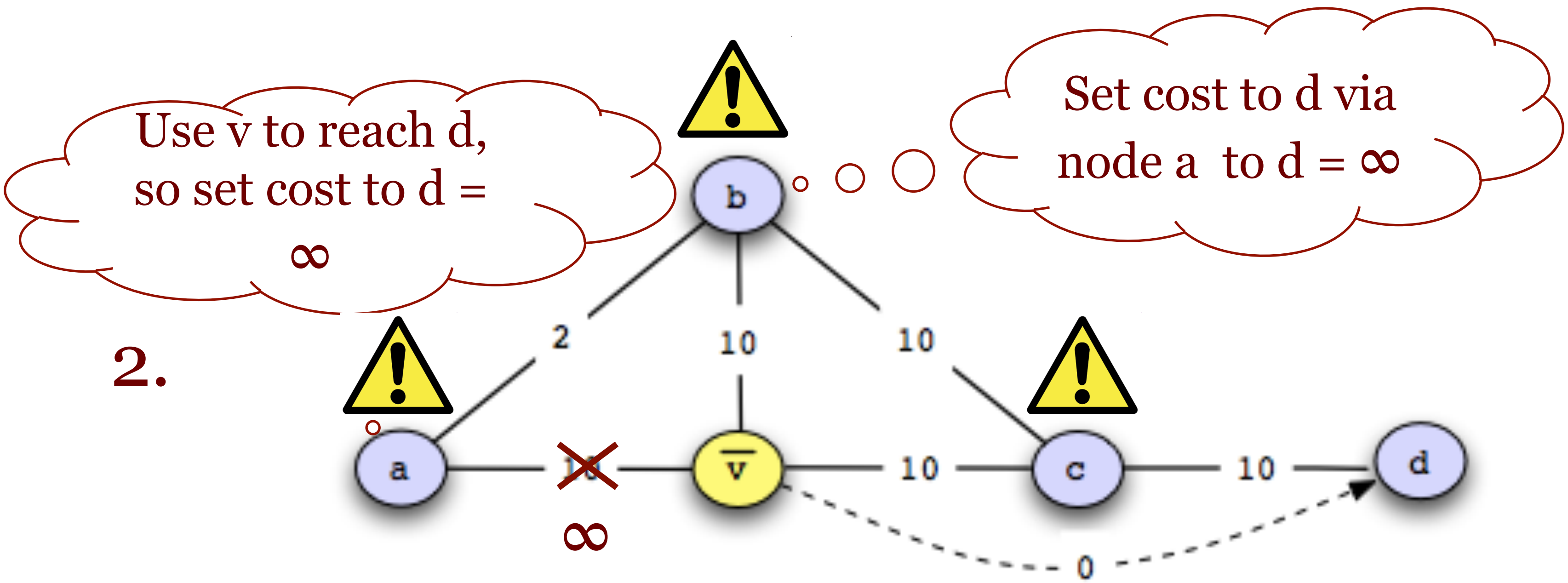




purge algorithm

1. Remove $\overline{v}$ as destination

2.



3. Run Distance Vector

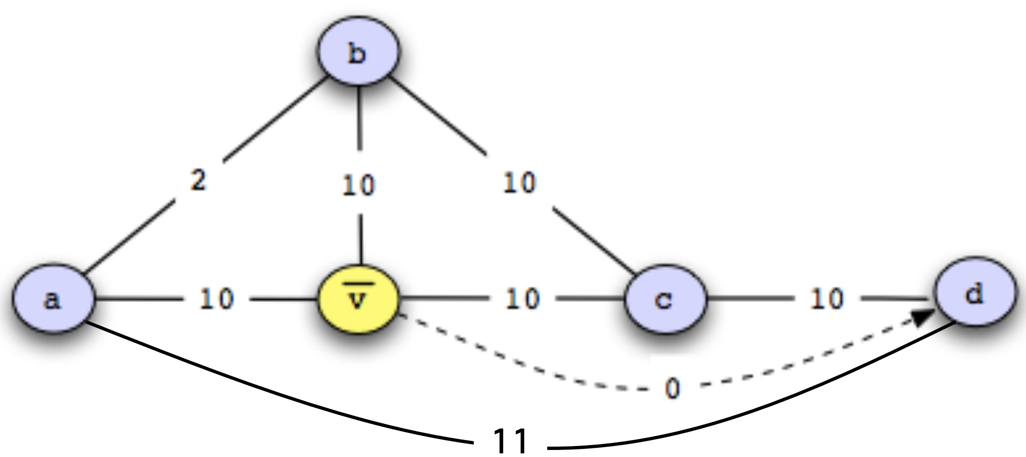
purge algorithm

Strengths

+ no synchronization needed

+ no routing loops

Weaknesses

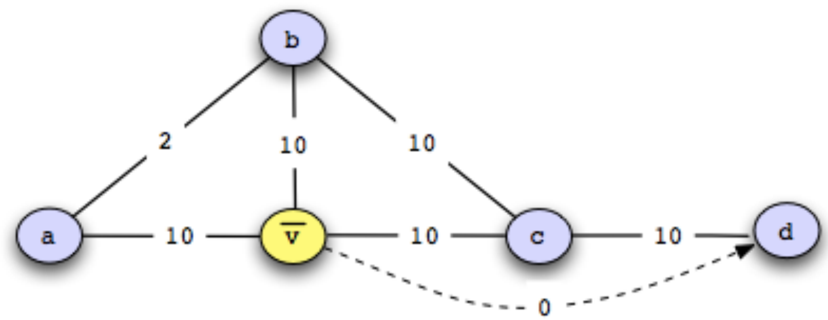


cpr algorithm

1. Periodically take snapshot of routing table

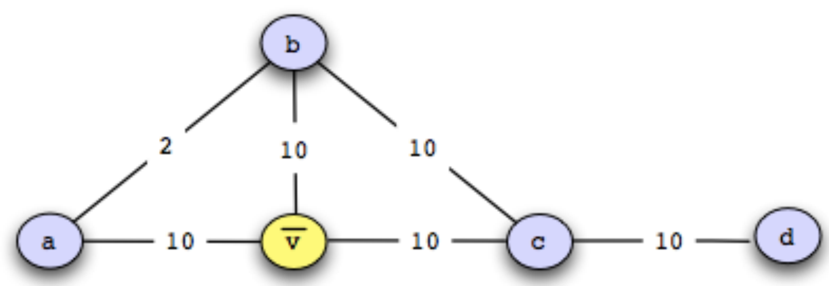
cpr algorithm

1. Periodically take snapshot of routing table
2. Rollback to a checkpoint taken before node is compromised



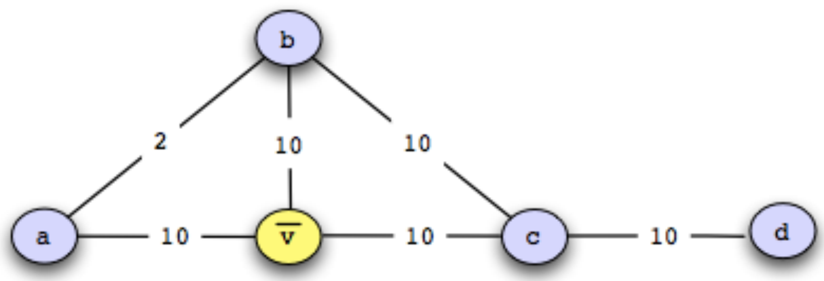
cpr algorithm

1. Periodically take snapshot of routing table
2. Rollback to a checkpoint taken before node is compromised



cpr algorithm

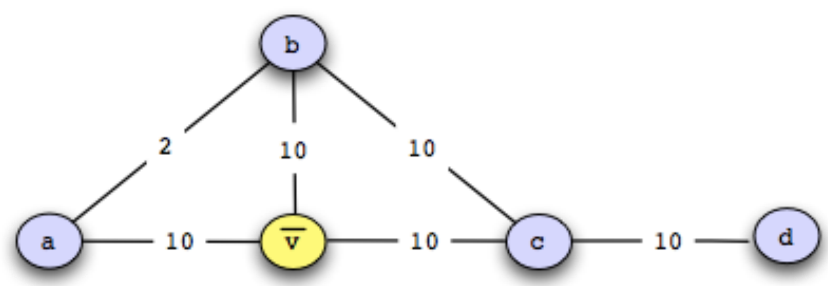
1. Periodically take snapshot of routing table
2. Rollback to a checkpoint taken before node is compromised



3. Remove compromised node

cpr algorithm

1. Periodically take snapshot of routing table
2. Rollback to a checkpoint taken before node is compromised



3. Remove compromised node
4. Run Distance Vector

cpr algorithm

Strengths

+ quickly remove false state w/ rollback

Weaknesses

- requires loosely synchronized clocks
- can have stale state after rolling back

Related Work

Garcia-Luna-Aceves’s DUAL algorithm for loop free routing

Database crash recovery

Malicious but committed transactions

Simulations

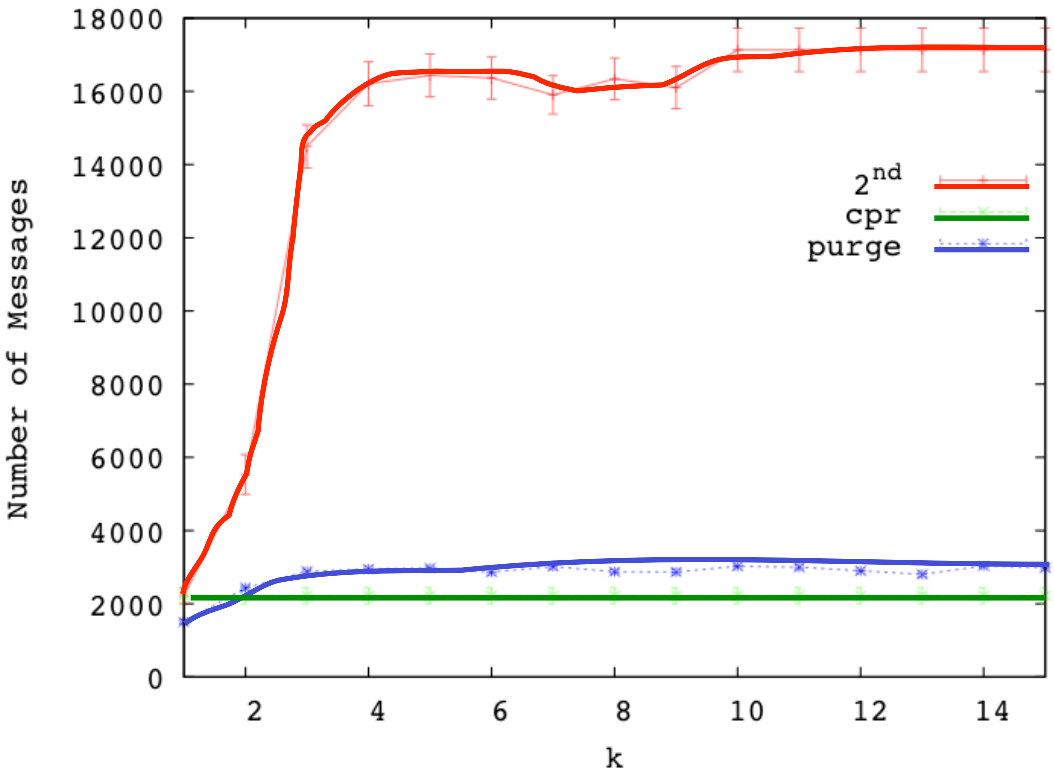
Synchronous communication model

Measure message and time overhead

Simulation Scenario

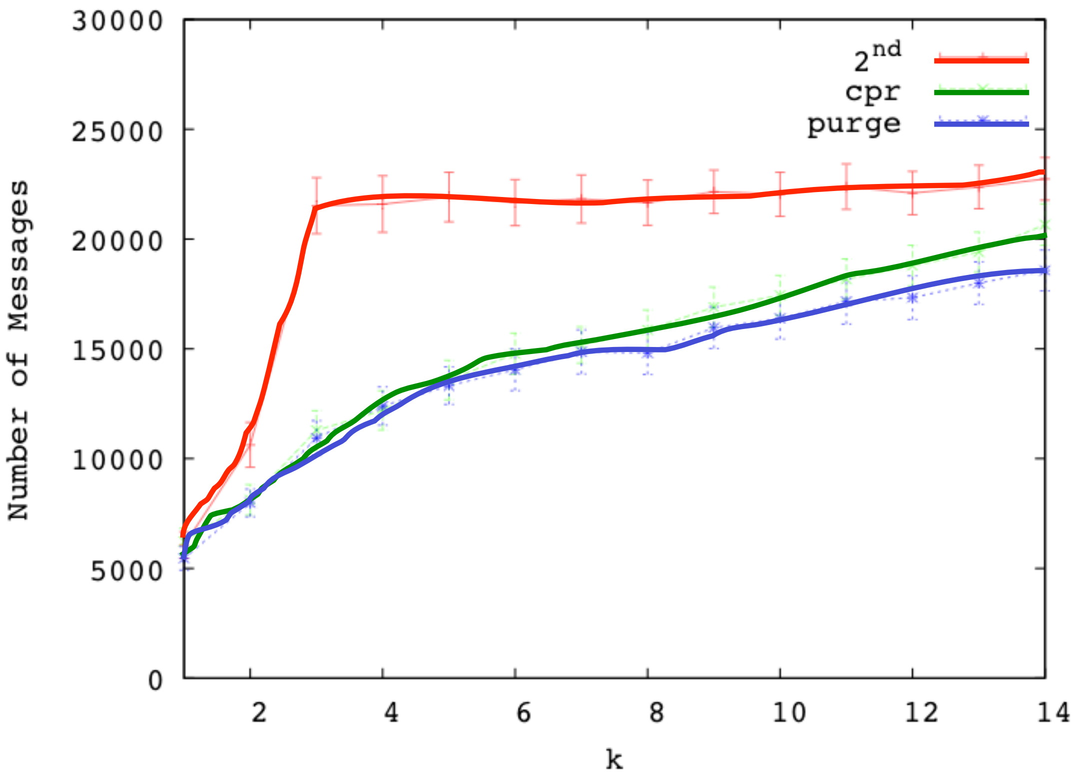
1. All nodes correctly compute least cost paths
2. A node is compromised and broadcasts (?) cost of *'1'* to every node
3. Nodes notified of compromised node
4. Run recovery algorithm
   1. (we count messages here)

Experiment 1 – Graphs with Fixed Link Costs



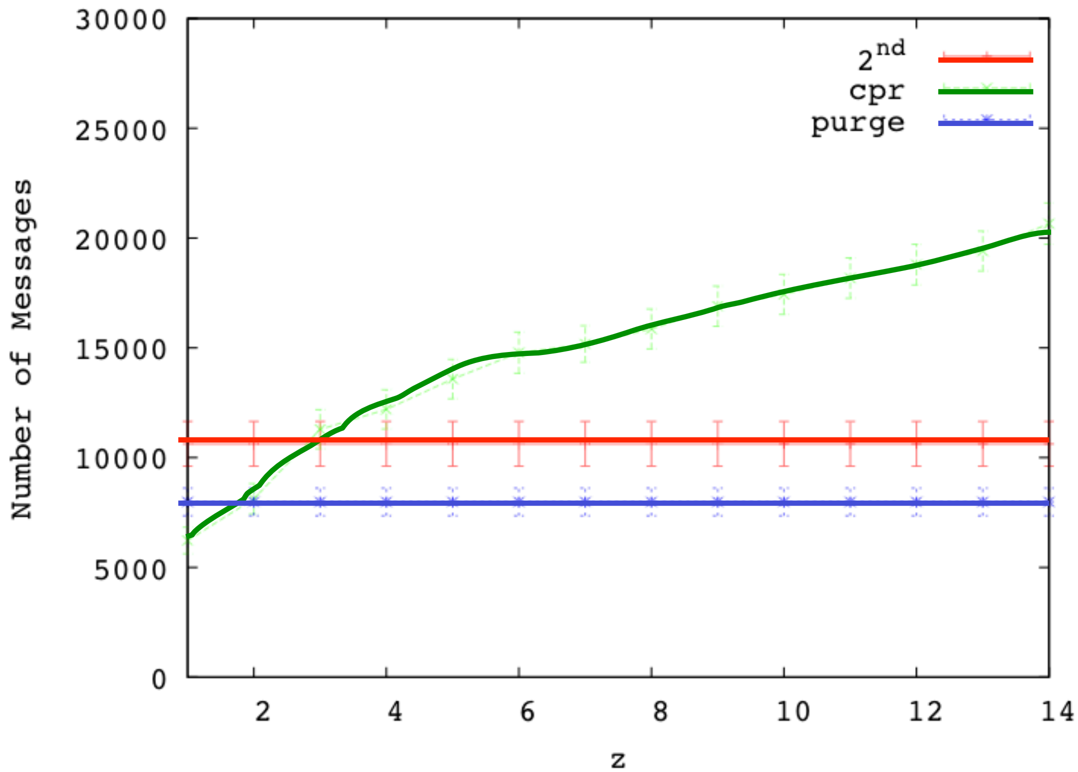
- 2^{nd} *Best* has many routing loops
- *cpr* removes state by rolling back
- 2^{nd} *Best* + *purge* use iterative distance vector

Experiment 2 – Graphs with Changing Link Costs



- 2^{nd} *Best* has many routing loops
- *cpr* has stale state after rolling back
- *purge* has no stale state

Experiment 3 – Vary Checkpoint Frequency



- *cpr* – less frequent checkpoints implies more overhead when rolling back
- 2nd *Best* and *purge* have constant overhead b/c don't checkpoint

Conclusions

- 2^{nd} Best suffers from routing loops

+ *cpr* is effective because rolling back quickly removes false state

Winner for fixed link costs

- *cpr* assume synchronized clocks

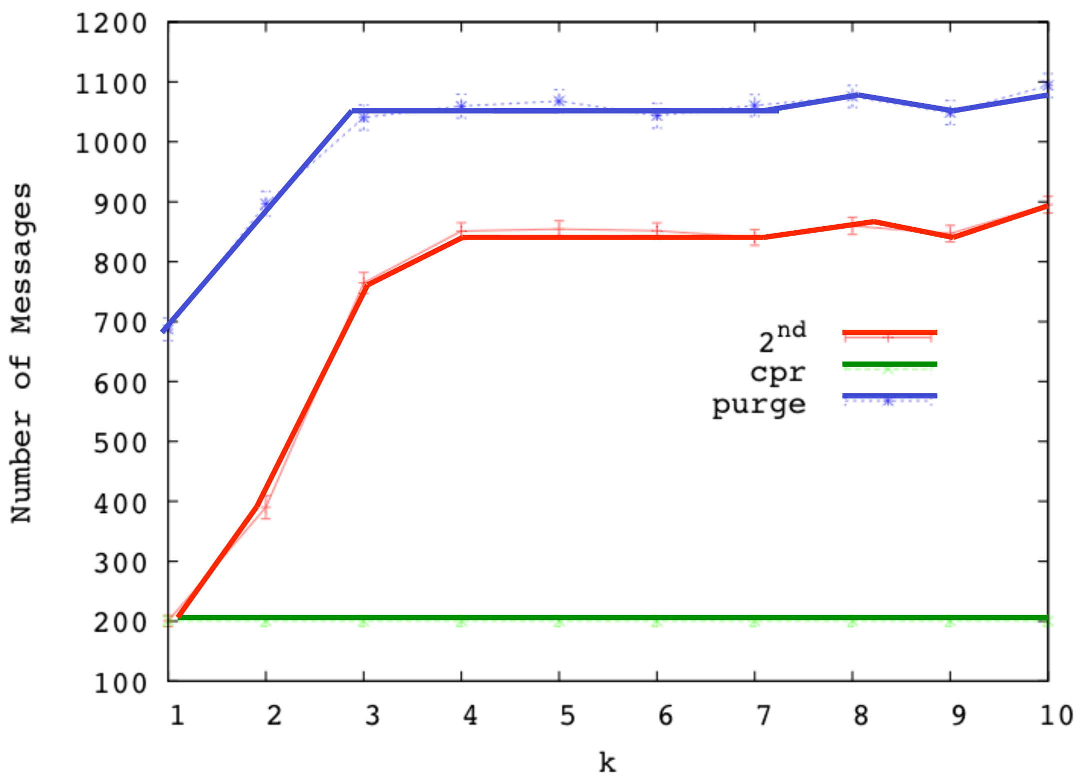
+ *purge* removes routing loops and no stale state

Winner for changing link costs

Thank You

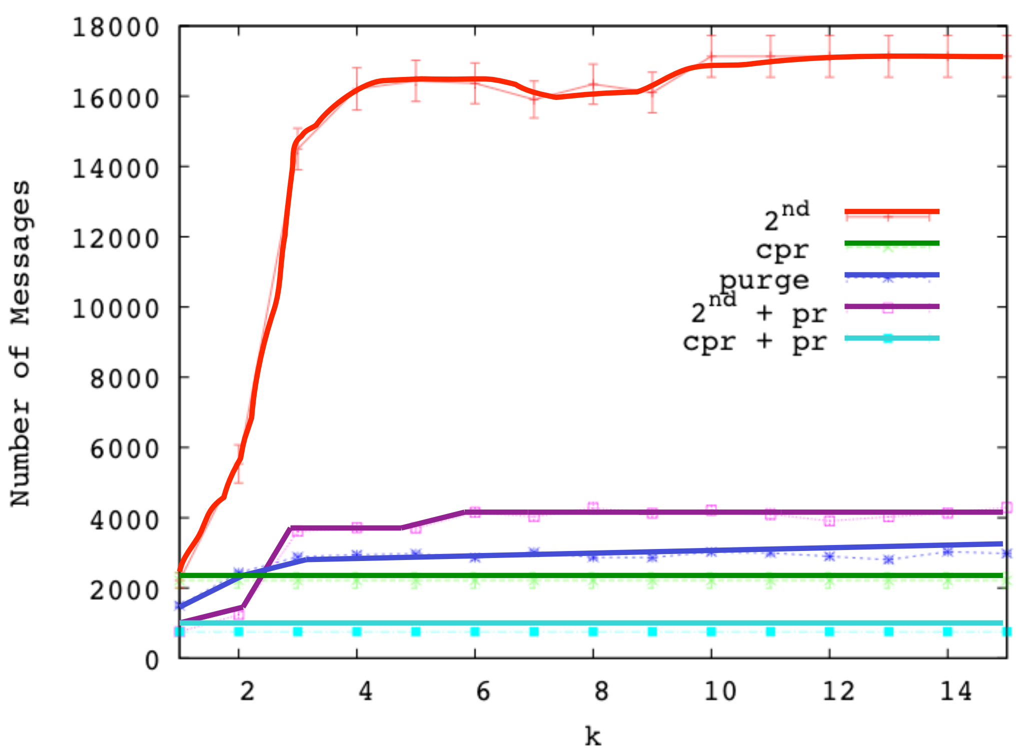
Questions + Comments

Experiment 0 – Graphs with Fixed Link Costs + Unit Link Weights



- *2nd Best* has few routing loops
- *purge* global state invalidation = wasteful
- *cpr* removes state by rolling back

Experiment 1b – Graphs with Fixed Link Costs + Poison Reverse

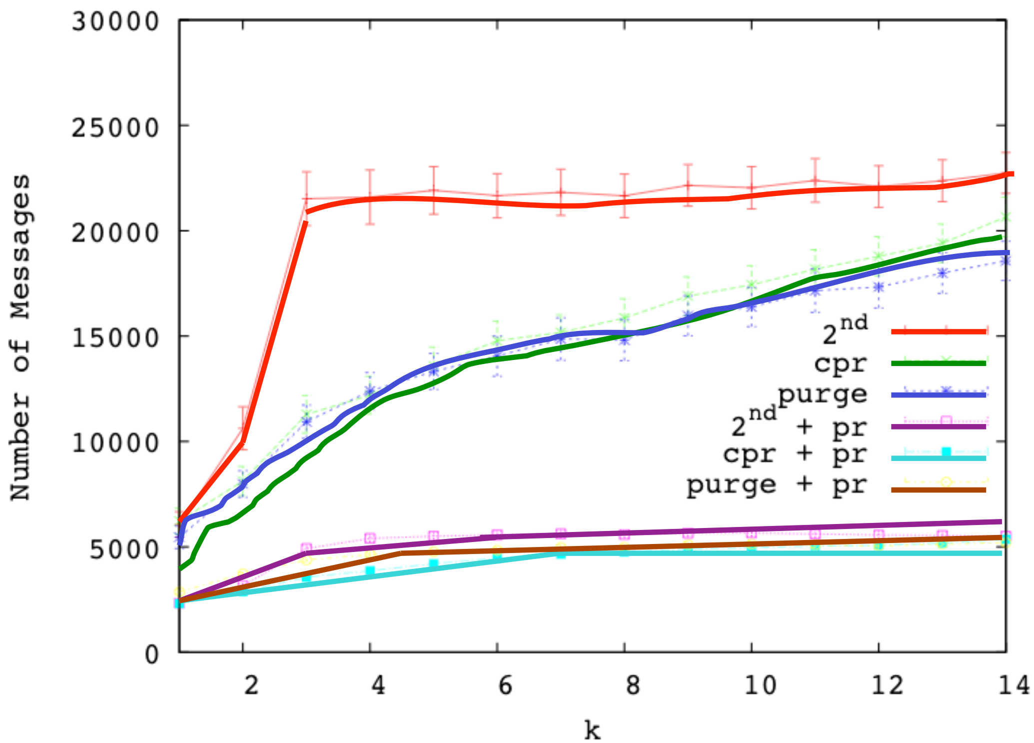


Poison Reverse effective

2nd Best + pr much better than *2nd Best*

cpr+pr modest improvements over *cpr*

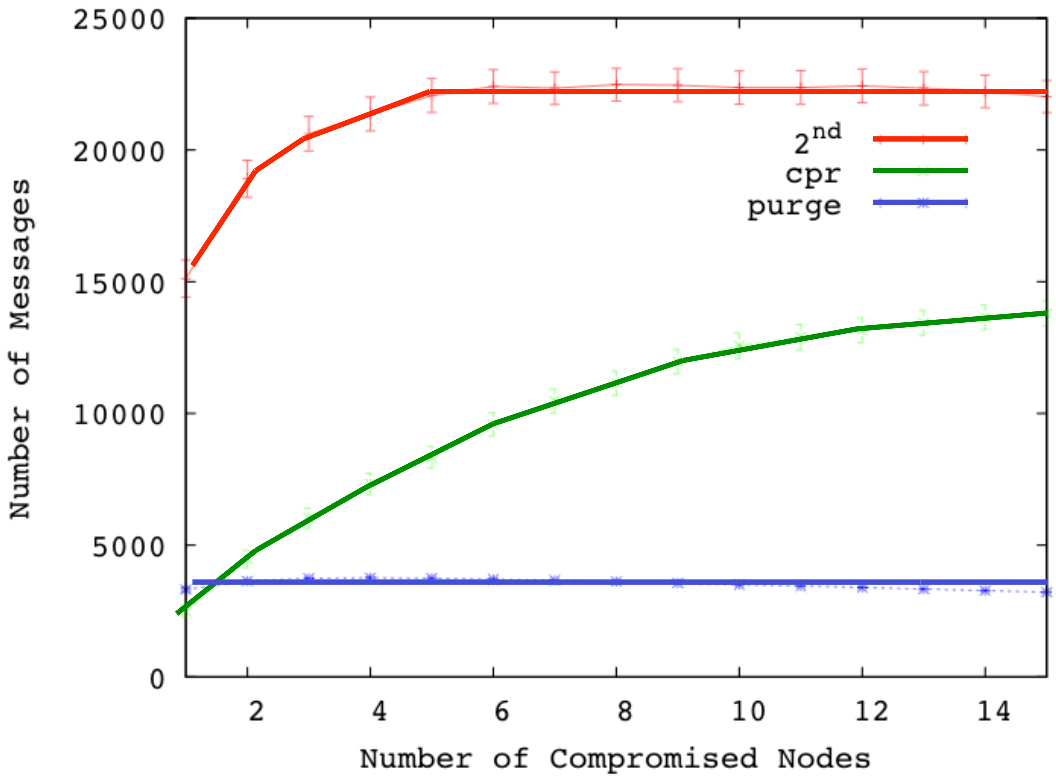
Experiment 2b – Graphs with Changing Link Costs+ Poison Reverse



Poison reverse makes removing stale state cheap, thus *cpr* + *pr* is best

purge + *pr* almost as good as *cpr*+*pr*

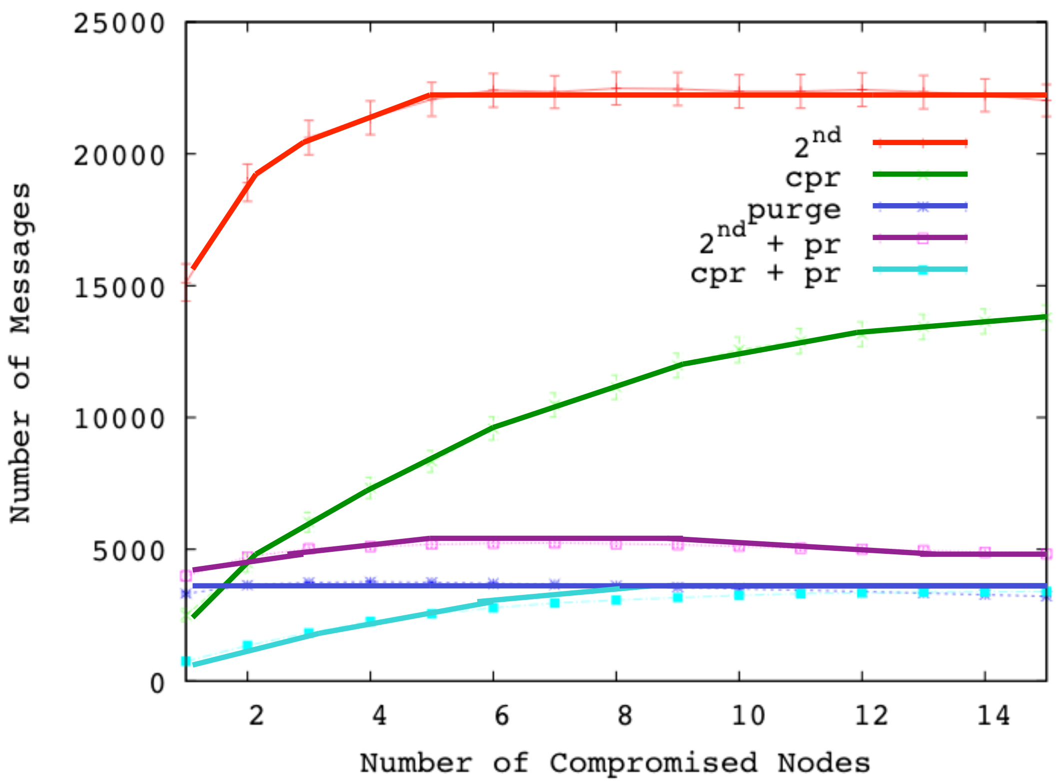
Experiment 4 – Multiple Compromised Nodes



- *purge* – no routing loops

- *2nd Best* and *cpr* have routing loops

Experiment 4 – Multiple Compromised Nodes + Poison Reverse



Poison reverse yields great improvements

purge and *cpr+pr* perform best

Possible Questions

How do nodes rollback?

How does this apply to BGP scenario?

Multiple compromised nodes?

Does 2nd best ever do well?

Theoretical results -