Derivation of Fault Tolerance Measures of Self-Stabilizing Algorithms by Simulation

Nils Müllner

February 11, 2009

Motivation

Analytical approach presented in "Reliability and Availability Analysis of Self-Stabilizing Systems" (SSS06)

- Fault tolerance measures are of high interest for system designers to choose the right system for a given fault environment.

Motivation

Analytical approach presented in "Reliability and Availability Analysis of Self-Stabilizing Systems" (SSS06)

- Fault tolerance measures are of high interest for system designers to choose the right system for a given fault environment.
- In [War06], Dhama, Warns and Theel presented an analytic method implemented in MatLab.

Motivation

Analytical approach presented in "Reliability and Availability Analysis of Self-Stabilizing Systems" (SSS06)

- Fault tolerance measures are of high interest for system designers to choose the right system for a given fault environment.
- In [War06], Dhama, Warns and Theel presented an analytic method implemented in MatLab.
  - Yet, assumptions about fault propagation and approximation hinder accuracy of measures obtained and

Motivation

Analytical approach presented in "Reliability and Availability Analysis of Self-Stabilizing Systems" (SSS06)

- Fault tolerance measures are of high interest for system designers to choose the right system for a given fault environment.
- In [War06], Dhama, Warns and Theel presented an analytic method implemented in MatLab.
  - Yet, assumptions about fault propagation and approximation hinder accuracy of measures obtained and
  - the method exhibits conservative assumptions only (worst case).

Motivation

Analytical approach presented in "Reliability and Availability Analysis of Self-Stabilizing Systems" (SSS06)

- Fault tolerance measures are of high interest for system designers to choose the right system for a given fault environment.
- In [War06], Dhama, Warns and Theel presented an analytic method implemented in MatLab.
  - Yet, assumptions about fault propagation and approximation hinder accuracy of measures obtained and
  - the method exhibits conservative assumptions only (worst case).
- In this work, we present a simulation framework that delivers average measures in an accurate, timely and extremely scalable fashion.

Simulation Framework 1/5

Simulation Framework called SiSSDA (Simulator for Self-Stabilizing Distributed Algorithms)

- implemented in Erlang

Simulation Framework 1/5

Simulation Framework called SiSSDA (Simulator for Self-Stabilizing Distributed Algorithms)

- implemented in Erlang
  - highly reliable (nine nines, χ^2-test)

Simulation Framework 1/5

Simulation Framework called SiSSDA (Simulator for Self-Stabilizing Distributed Algorithms)

- implemented in Erlang
  - highly reliable (nine nines, χ^2-test)
  - distributed (to cope with CPU demanding algorithms)

Simulation Framework 1/5

Simulation Framework called SiSSDA (Simulator for Self-Stabilizing Distributed Algorithms)

- implemented in Erlang
  - highly reliable (nine nines, χ^2-test)
  - distributed (to cope with CPU demanding algorithms)
  - platform independent (even heterogeneous Windows/Linux clusters)

Simulation Framework 1/5

Simulation Framework called SiSSDA (Simulator for Self-Stabilizing Distributed Algorithms)

- implemented in Erlang
  - highly reliable (nine nines, χ^2-test)
  - distributed (to cope with CPU demanding algorithms)
  - platform independent (even heterogeneous Windows/Linux clusters)
- support for large scale models (about 1,000 nodes per GB RAM; analytic approach limited to seven nodes!)

Simulation Framework 2/5

- monitoring facility (prints every n^{th} step)

Simulation Framework 2/5

- monitoring facility (prints every n^{th} step)
- runs until desired accuracy is reached (maximal acceptable deviation within last n turns)

Simulation Framework 2/5

- monitoring facility (prints every n^{th} step)
- runs until desired accuracy is reached (maximal acceptable deviation within last n turns)
- four distributed self-stabilizing algorithms provided

Simulation Framework 2/5

- monitoring facility (prints every n^{th} step)
- runs until desired accuracy is reached (maximal acceptable deviation within last n turns)
- four distributed self-stabilizing algorithms provided
  - Breadth First Search

Simulation Framework 2/5

- monitoring facility (prints every n^{th} step)
- runs until desired accuracy is reached (maximal acceptable deviation within last n turns)
- four distributed self-stabilizing algorithms provided
  - Breadth First Search
  - Depth First Search

Simulation Framework 2/5

- monitoring facility (prints every n^{th} step)
- runs until desired accuracy is reached (maximal acceptable deviation within last n turns)
- four distributed self-stabilizing algorithms provided
  - Breadth First Search
  - Depth First Search
  - Leader Election

Simulation Framework 2/5

- monitoring facility (prints every n^{th} step)
- runs until desired accuracy is reached (maximal acceptable deviation within last n turns)
- four distributed self-stabilizing algorithms provided
  - Breadth First Search
  - Depth First Search
  - Leader Election
  - Mutual Exclusion

Simulation Framework 2/5

- monitoring facility (prints every n^{th} step)
- runs until desired accuracy is reached (maximal acceptable deviation within last n turns)
- four distributed self-stabilizing algorithms provided
  - Breadth First Search
  - Depth First Search
  - Leader Election
  - Mutual Exclusion
- easy to extend

Simulation Framework 3/5

- exact fault environments (specify distinct values for each vertex and edge)

Simulation Framework 3/5

- exact fault environments (specify distinct values for each vertex and edge)
- dynamic fault environments

Simulation Framework 3/5

- exact fault environments (specify distinct values for each vertex and edge)
- dynamic fault environments
- dynamic execution semantics possible (number of nodes executing per step in parallel).

Simulation Framework 3/5

- exact fault environments (specify distinct values for each vertex and edge)
- dynamic fault environments
- dynamic execution semantics possible (number of nodes executing per step in parallel).
- external fault injection and monitoring facilities

Simulation Framework 3/5

- exact fault environments (specify distinct values for each vertex and edge)
- dynamic fault environments
- dynamic execution semantics possible (number of nodes executing per step in parallel).
- external fault injection and monitoring facilities
- event logging (if needed)

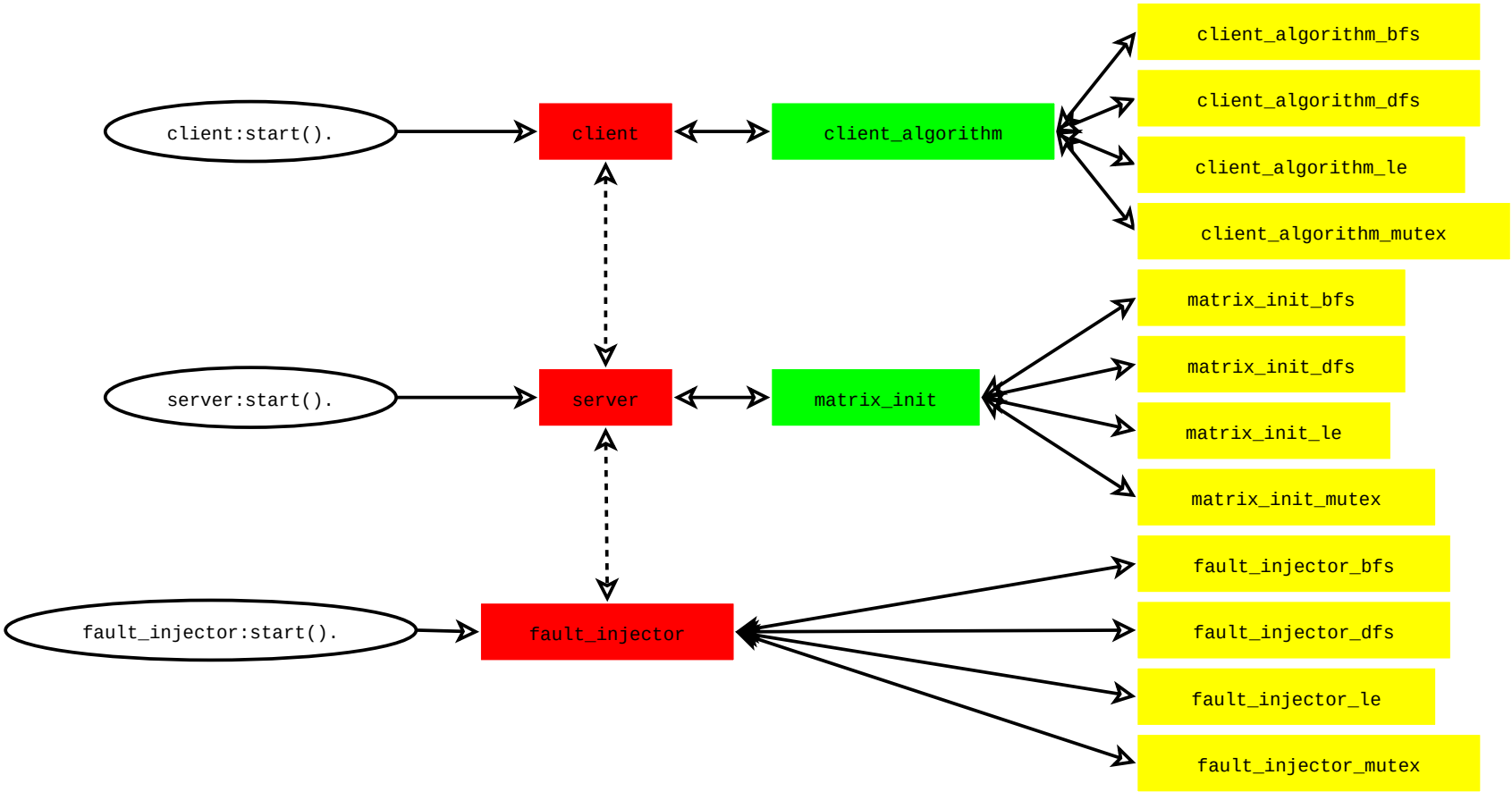
Simulation Framework 3/5

- exact fault environments (specify distinct values for each vertex and edge)
- dynamic fault environments
- dynamic execution semantics possible (number of nodes executing per step in parallel).
- external fault injection and monitoring facilities
- event logging (if needed)
- choice of schedulers (three provided)
- uses *discrete time Markov chains* as system model

Simulation Framework 4/5

```
7> server:start().
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                                                                     %%%%%
%%%%%                               SiSSDA                                %%%%%
%%%%%                                                                     %%%%%
%%%%%                               v "1.0"                               %%%%%
%%%%%                                                                     %%%%%
%%%%%                                                                     %%%%%
%%%%%                  Welcome to the Simulator for                       %%%%%
%%%%%              Self-Stabilizing Distributed Algorithms                %%%%%
%%%%%                                                                     %%%%%
%%%%%                               SERVER                                %%%%%
%%%%%                                                                     %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% INITIALIZATION-PHASE 1: CHOOSE ALGORITHM                            %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
true
%%%%% The following algorithms are available:                             %%%%%
%%%%%    [1]bfs
%%%%%    [2]dfs
%%%%%    [3]le
%%%%%    [4]mutex
%%%%% Please enter the appropriate number [n.]>
```

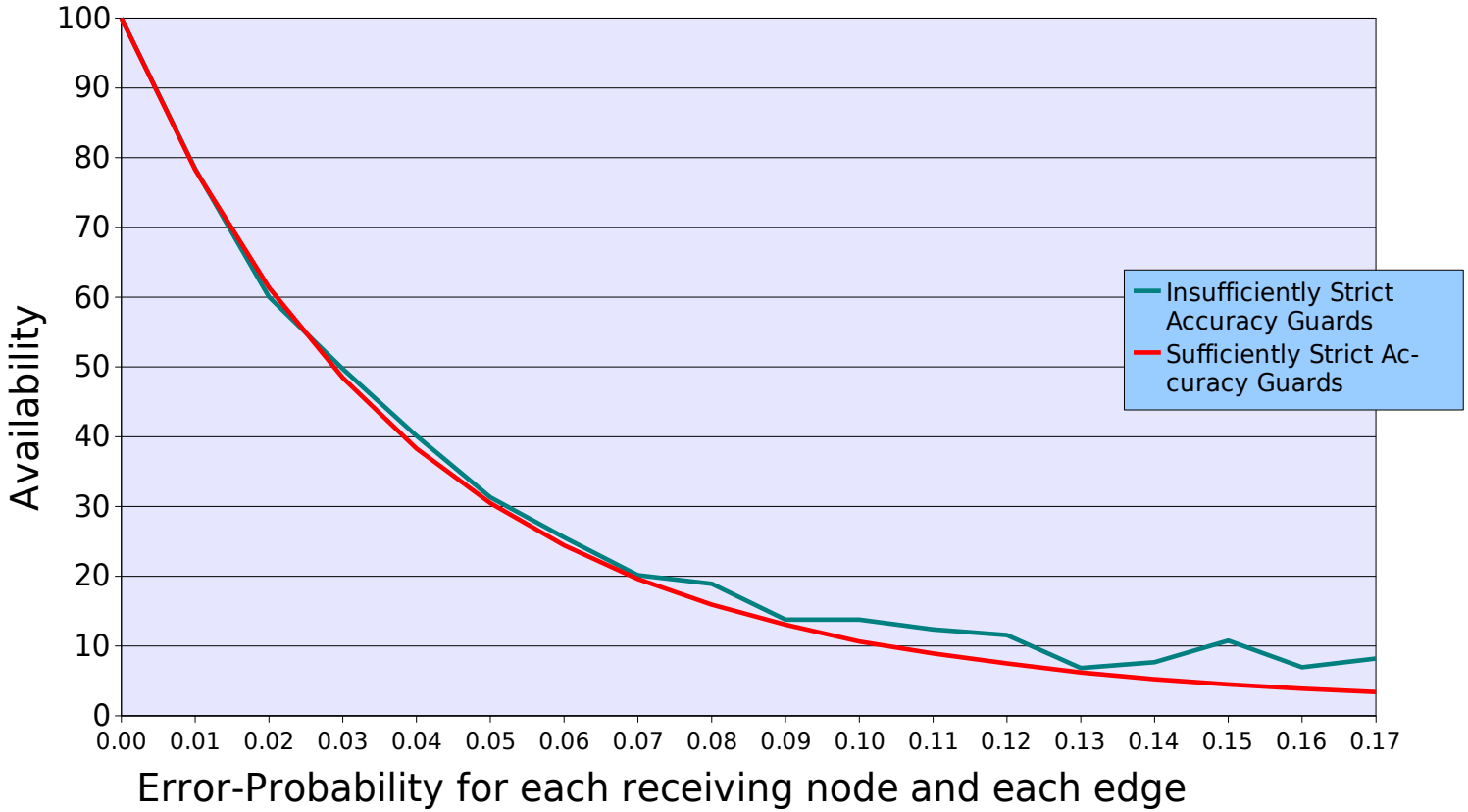
Simulation Framework 5/5



Accuracy 1/2

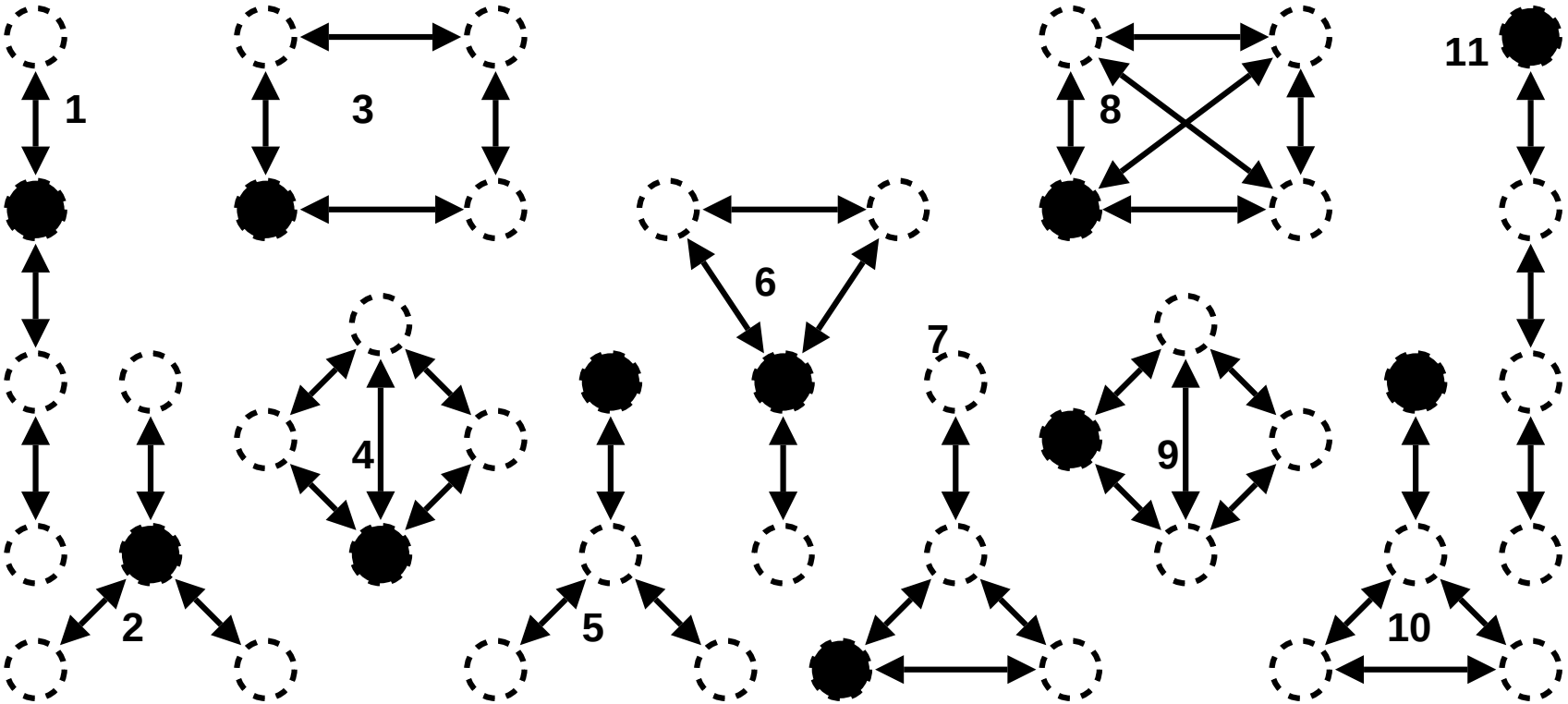
This figure exemplifies availability for first $20,000$ steps of an eight-processor system. The desired accuracy is reached if maximum the deviation within last n steps is lower than a certain barrier. The Results presented in the following feature about $1,000,000$ steps per system node.

Accuracy 2/2



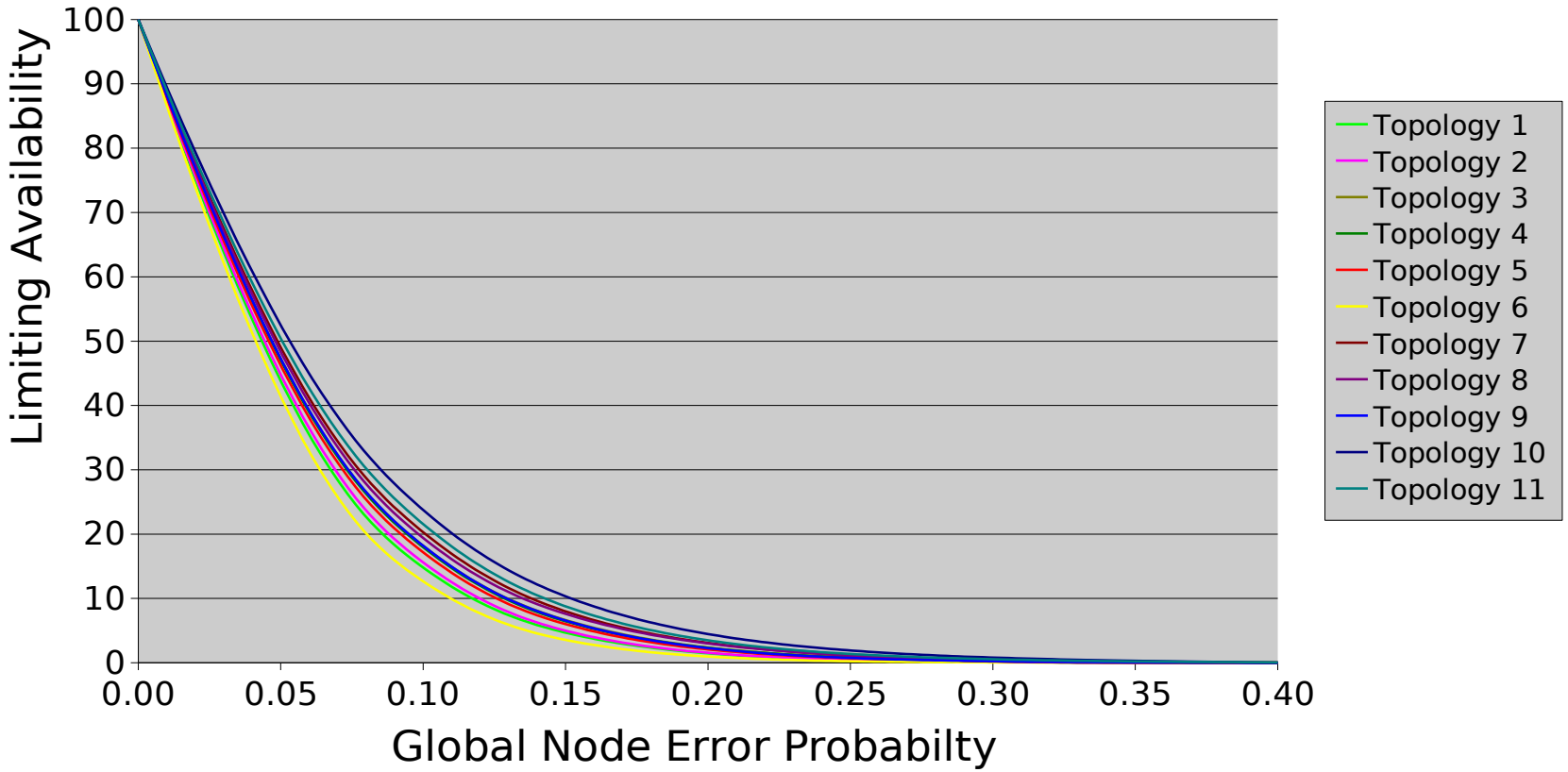
Strictness of accuracy guards is crucial for reliability of results!

Test Case: All Possible 4-node Graphs

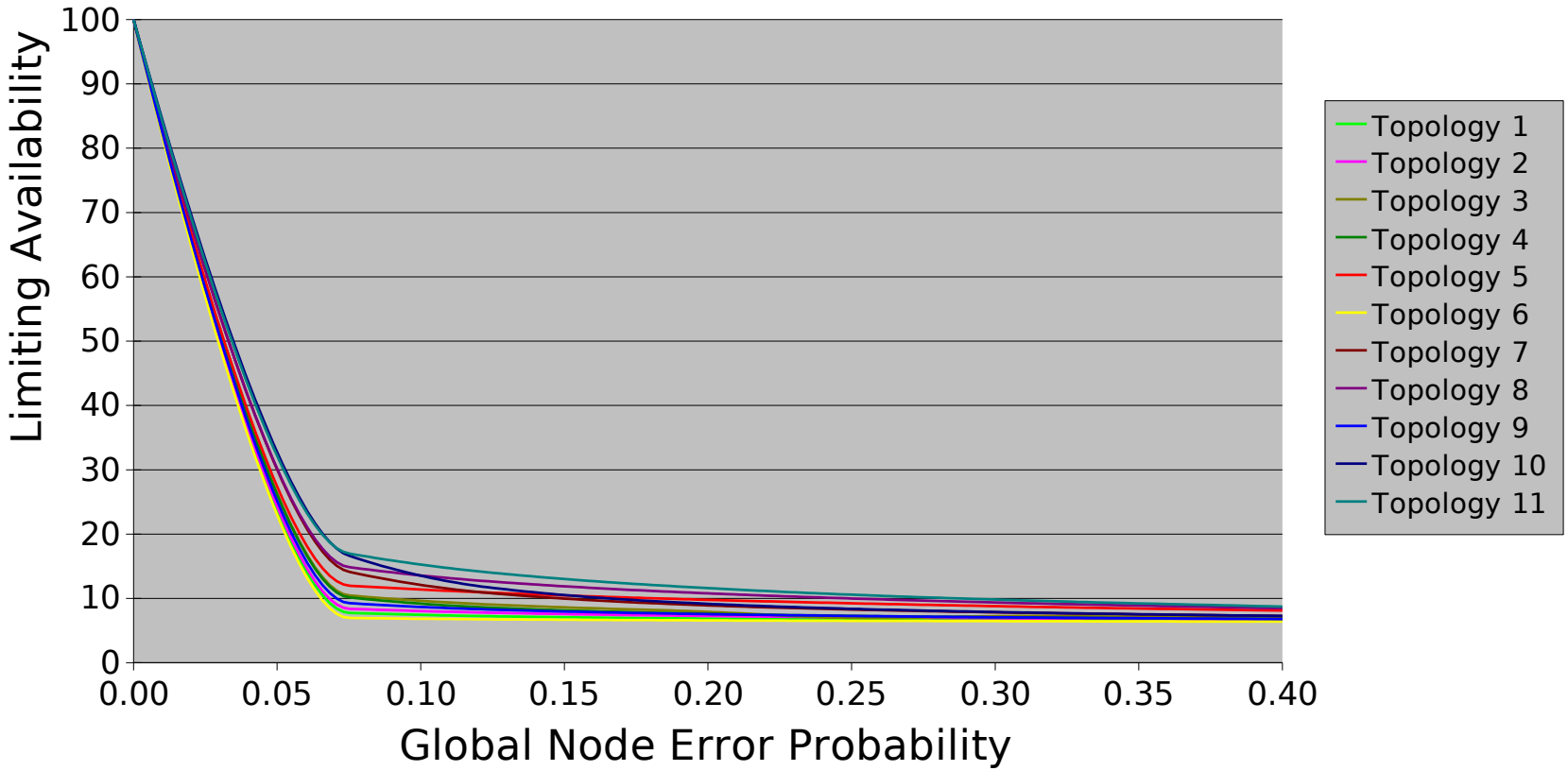


We chose *depth first search* (DFS) and *breadth first search* (BFS) algorithms for comparison with the analytic approach, executed on all possible 4-node graphs.

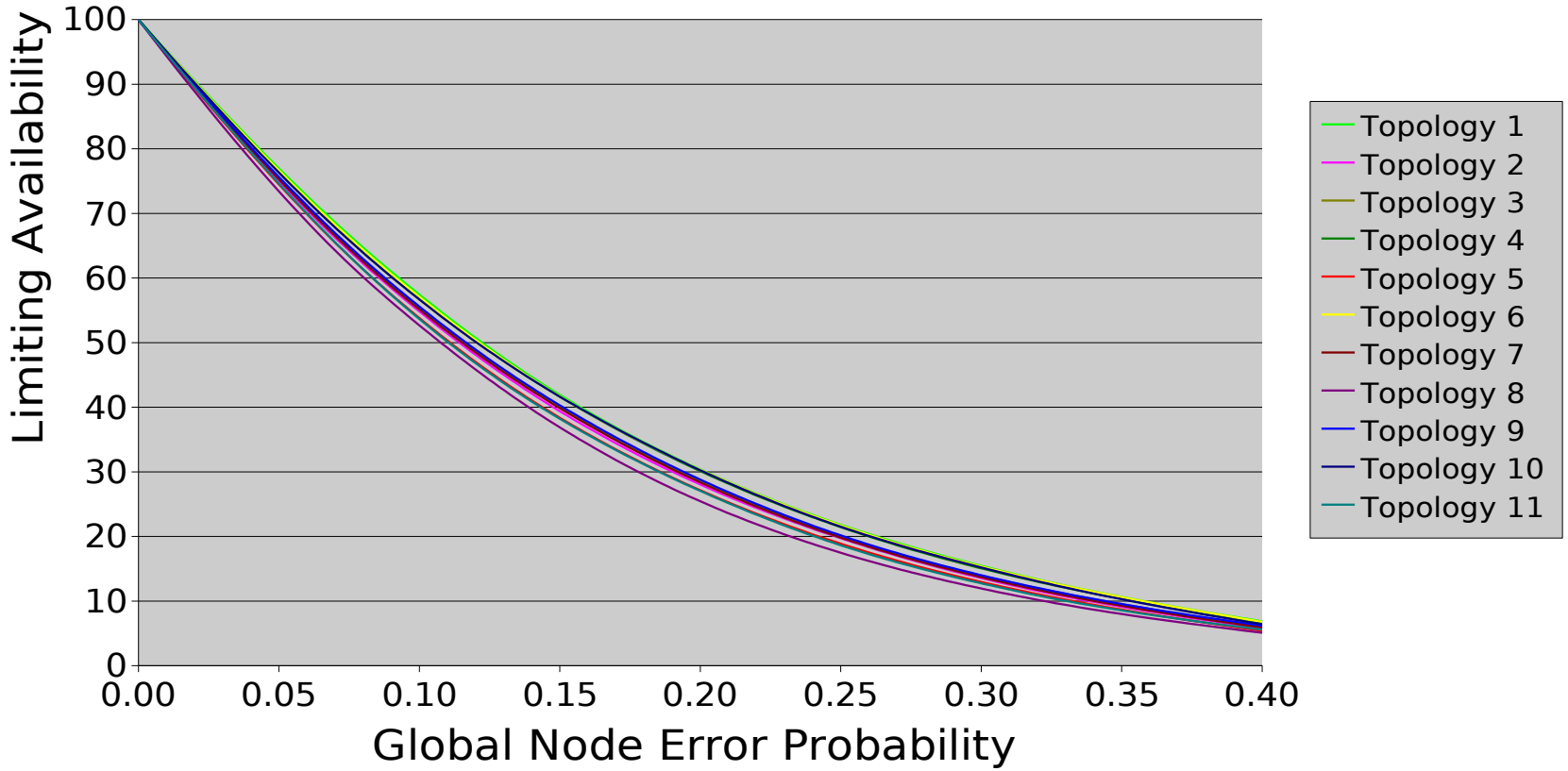
Breadth First Search - Simulation



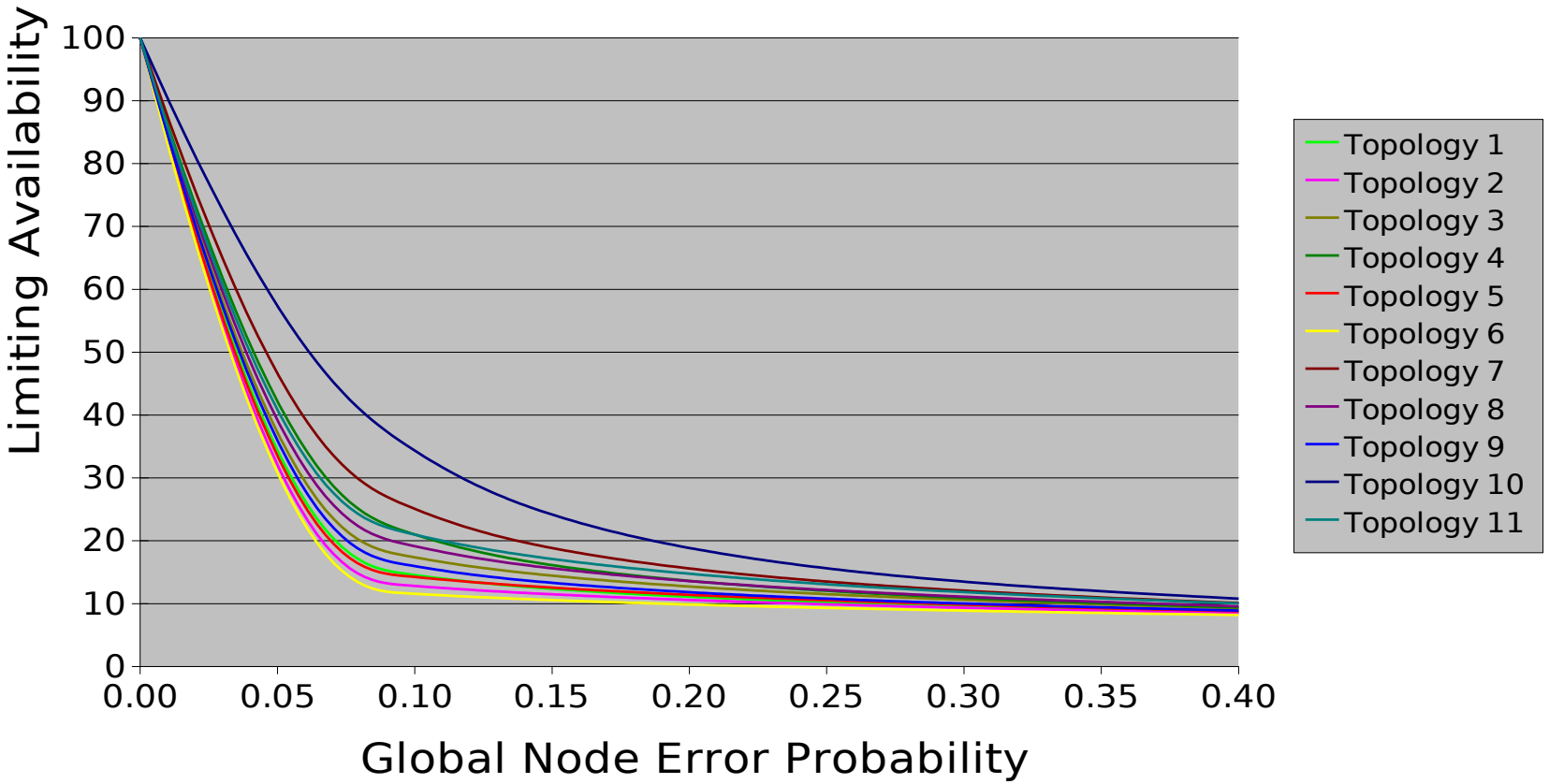
Breadth First Search - Analysis



Depth First Search - Simulation



Depth First Search - Analysis



Conclusions

Derivation of fault tolerance measures by simulation

- reason: analytic method is insufficient

Conclusions

Derivation of fault tolerance measures by simulation

- reason: analytic method is insufficient
- method: simulation of self-stabilizing distributed algorithms

Conclusions

Derivation of fault tolerance measures by simulation

- reason: analytic method is insufficient
- method: simulation of self-stabilizing distributed algorithms
- features: modular design, scalability, performance, reliability of results


Shlomi Dolev.
Self-Stabilization.
MIT Press, March 2000.


Marco Schneider.
Self-stabilization.
ACM Comput. Surv., 25(1):45–67, 1993.


Kishor Shridharbhai Trivedi.
Probability and Statistics with Reliability, Queuing and Computer Science Applications.
Prentice Hall PTR, Upper Saddle River, NJ, USA, 1982.


Abhishek Dhama; Oliver Theel; Timo Warns.
Reliability and Availability Analysis of Self-Stabilizing Systems.
In *8th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, page 17. Springer, 2006.