Deriving a Good Trade-off Between System Availability and Time Redundancy

Nils Müllner
nils.muellner@informatik.uni-oldenburg.de

System Software and Distributed Systems Group,
Universität Oldenburg, Germany

June 30, 2009

Table of contents

Orientation

- Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems
- How much ddelay is desired? $\Rightarrow$ as short as possible...
- How available should your system be? $\Rightarrow$ as high as possible...
- What is a good trade-off in between delay and availability?

Orientation

- Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems
- How much ddelay is desired? $\Rightarrow$ as short as possible...
- How available should your system be? $\Rightarrow$ as high as possible...
- What is a good trade-off in between delay and availability?

Orientation

- Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems
- How much ddelay is desired? $\Rightarrow$ as short as possible...
- How available should your system be? $\Rightarrow$ as high as possible...
- What is a good trade-off in between delay and availability?

Orientation

- Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems
- How much ddelay is desired? $\Rightarrow$ as short as possible...
- How available should your system be? $\Rightarrow$ as high as possible...
- What is a good trade-off in between delay and availability?

Orientation

- Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems
- How much ddelay is desired? $\Rightarrow$ as short as possible...
- How available should your system be? $\Rightarrow$ as high as possible...
- What is a good trade-off in between delay and availability?

Orientation

- Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems
- How much ddelay is desired? $\Rightarrow$ as short as possible...
- How available should your system be? $\Rightarrow$ as high as possible...
- What is a good trade-off in between delay and availability?

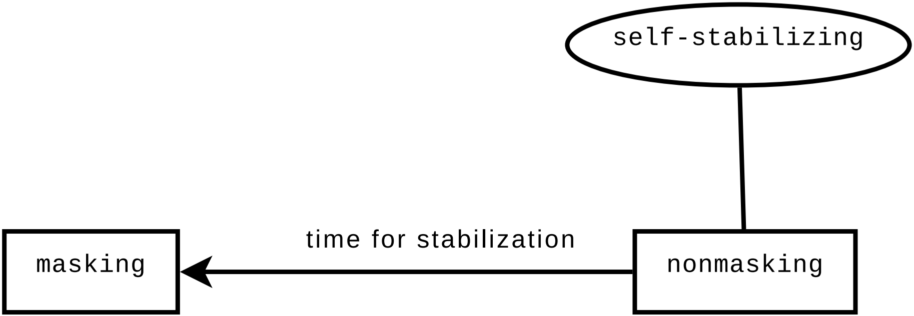
Orientation

- Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems
- How much ddelay is desired? $\Rightarrow$ as short as possible...
- How available should your system be? $\Rightarrow$ as high as possible...
- What is a good trade-off in between delay and availability?

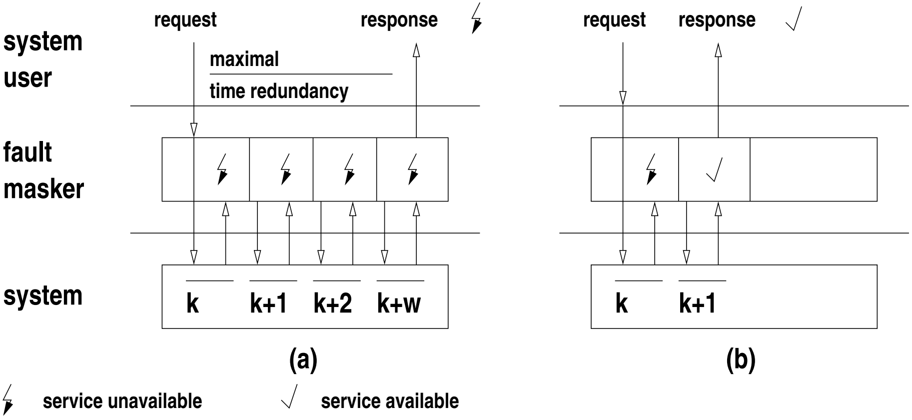
Orientation

- Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems
- How much ddelay is desired? $\Rightarrow$ as short as possible...
- How available should your system be? $\Rightarrow$ as high as possible...
- What is a good trade-off in between delay and availability?

Orientation



Instantaneous Window Availability



Availability

$$Availability : A = \frac{MTTF}{MTBF} \tag{1}$$

- **instantaneous availability**: at some arbitrary time k the system is available: $A(k)$
- **limiting availability**: the same, as k approaches ∞
- analysis determines limiting, but for simulation we can only choose high k
- at some arbitrary point k, what are the chances that we get the intended (correct) service?
- and what would happen if the system fails but we can wait for at most w timesteps for the system to recover?
- **Instantaneous Window Availability (IWA)**: given that a system is not available at k, what is the **availability increase** if we wait for at most w steps?
- How many steps must one wait to achieve a certain overall availability?

Availability

$$Availability : A = \frac{MTTF}{MTBF} \quad (1)$$

- **instantaneous availability**: at some arbitrary time k the system is available: $A(k)$
- **limiting availability**: the same, as k approaches ∞
- analysis determines limiting, but for simulation we can only choose high k
- at some arbitrary point k, what are the chances that we get the intended (correct) service?
- and what would happen if the system fails but we can wait for at most w timesteps for the system to recover?
- **Instantaneous Window Availability (IWA)**: given that a system is not available at k, what is the **availability increase** if we wait for at most w steps?
- How many steps must one wait to achieve a certain overall availability?

Availability

$$Availability : A = \frac{MTTF}{MTBF} \quad (1)$$

- **instantaneous availability**: at some arbitrary time k the system is available: $A(k)$
- **limiting availability**: the same, as k approaches ∞
- analysis determines limiting, but for simulation we can only choose high k
- at some arbitrary point k, what are the chances that we get the intended (correct) service?
- and what would happen if the system fails but we can wait for at most w timesteps for the system to recover?
- **Instantaneous Window Availability (IWA)**: given that a system is not available at k, what is the **availability increase** if we wait for at most w steps?
- How many steps must one wait to achieve a certain overall availability?

Availability

$$Availability : A = \frac{MTTF}{MTBF} \quad (1)$$

- **instantaneous availability**: at some arbitrary time k the system is available: $A(k)$
- **limiting availability**: the same, as k approaches ∞
- analysis determines limiting, but for simulation we can only choose high k
- at some arbitrary point k, what are the chances that we get the intended (correct) service?
- and what would happen if the system fails but we can wait for at most w timesteps for the system to recover?
- **Instantaneous Window Availability (IWA)**: given that a system is not available at k, what is the **availability increase** if we wait for at most w steps?
- How many steps must one wait to achieve a certain overall availability?

Availability

$$Availability : A = \frac{MTTF}{MTBF} \quad (1)$$

- **instantaneous availability**: at some arbitrary time k the system is available: $A(k)$
- **limiting availability**: the same, as k approaches ∞
- analysis determines limiting, but for simulation we can only choose high k
- at some arbitrary point k, what are the chances that we get the intended (correct) service?
- and what would happen if the system fails but we can wait for at most w timesteps for the system to recover?
- **Instantaneous Window Availability (IWA)**: given that a system is not available at k, what is the **availability increase** if we wait for at most w steps?
- How many steps must one wait to achieve a certain overall availability?

Availability

$$Availability : A = \frac{MTTF}{MTBF} \quad (1)$$

- **instantaneous availability**: at some arbitrary time k the system is available: $A(k)$
- **limiting availability**: the same, as k approaches ∞
- analysis determines limiting, but for simulation we can only choose high k
- at some arbitrary point k, what are the chances that we get the intended (correct) service?
- and what would happen if the system fails but we can wait for at most w timesteps for the system to recover?
- **Instantaneous Window Availability (IWA)**: given that a system is not available at k, what is the **availability increase** if we wait for at most w steps?
- How many steps must one wait to achieve a certain overall availability?

Availability

$$Availability : A = \frac{MTTF}{MTBF} \tag{1}$$

- **instantaneous availability**: at some arbitrary time k the system is available: $A(k)$
- **limiting availability**: the same, as k approaches ∞
- analysis determines limiting, but for simulation we can only choose high k
- at some arbitrary point k, what are the chances that we get the intended (correct) service?
- and what would happen if the system fails but we can wait for at most w timesteps for the system to recover?
- **Instantaneous Window Availability (IWA)**: given that a system is not available at k, what is the **availability increase** if we wait for at most w steps?
- How many steps must one wait to achieve a certain overall availability?

Availability

$$Availability : A = \frac{MTTF}{MTBF} \tag{1}$$

- **instantaneous availability**: at some arbitrary time k the system is available: $A(k)$
- **limiting availability**: the same, as k approaches ∞
- analysis determines limiting, but for simulation we can only choose high k
- at some arbitrary point k, what are the chances that we get the intended (correct) service?
- and what would happen if the system fails but we can wait for at most w timesteps for the system to recover?
- **Instantaneous Window Availability (IWA)**: given that a system is not available at k, what is the **availability increase** if we wait for at most w steps?
- How many steps must one wait to achieve a certain overall availability?

Availability

$$Availability : A = \frac{MTTF}{MTBF} \quad (1)$$

- **instantaneous availability**: at some arbitrary time k the system is available: $A(k)$
- **limiting availability**: the same, as k approaches ∞
- analysis determines limiting, but for simulation we can only choose high k
- at some arbitrary point k, what are the chances that we get the intended (correct) service?
- and what would happen if the system fails but we can wait for at most w timesteps for the system to recover?
- **Instantaneous Window Availability (IWA)**: given that a system is not available at k, what is the **availability increase** if we wait for at most w steps?
- How many steps must one wait to achieve a certain overall availability?

System Model

- serial execution semantics
- central demon/scheduler/monitor
- shared memory model
- perfect fault detection
- fault model: transient faults strike mem register with static probability
- distributed self-stabilizing algorithm
  - convergence: $\exists t \geq t_0 : c(t) \models P$
  - consistency: $\forall t_l > t_k : c(t_k) \models P \Rightarrow c(t_l) \models P$
- two systems:

- distributed self-stabilizing breadth first search (BFS)

System Model

- serial execution semantics
- central demon/scheduler/monitor
- shared memory model
- perfect fault detection
- fault model: transient faults strike mem register with static probability
- distributed self-stabilizing algorithm
  - convergence: $\exists t \geq t_0 : c(t) \models P$
  - consistency: $\forall t_l > t_k : c(t_k) \models P \Rightarrow c(t_l) \models P$
- two systems:

- distributed self-stabilizing breadth first search (BFS) [Dol00]

System Model

- serial execution semantics
- central demon/scheduler/monitor
- shared memory model
- perfect fault detection
- fault model: transient faults strike mem register with static probability
- distributed self-stabilizing algorithm
  - convergence: $\exists t \geq t_0 : c(t) \models P$
  - consistency: $\forall t_l > t_k : c(t_k) \models P \Rightarrow c(t_l) \models P$
- two systems:
- distributed self-stabilizing breadth first search (BFS) [Dol00]

System Model

- serial execution semantics
- central demon/scheduler/monitor
- shared memory model
- perfect fault detection
- fault model: transient faults strike mem register with static probability
- distributed self-stabilizing algorithm
  - convergence: $\exists t \geq t_0 : c(t) \models P$
  - consistency: $\forall t_l > t_k : c(t_k) \models P \Rightarrow c(t_l) \models P$
- two systems:

- distributed self-stabilizing breadth first search (BFS) [Dol00]

System Model

- serial execution semantics
- central demon/scheduler/monitor
- shared memory model
- perfect fault detection
- fault model: transient faults strike mem register with static probability
- distributed self-stabilizing algorithm
  - convergence: $\exists t \geq t_0 : c(t) \models P$
  - consistency: $\forall t_l > t_k : c(t_k) \models P \Rightarrow c(t_l) \models P$
- two systems:

- distributed self-stabilizing breadth first search (BFS) [Dol00]

System Model

- serial execution semantics
- central demon/scheduler/monitor
- shared memory model
- perfect fault detection
- fault model: transient faults strike mem register with static probability
- distributed self-stabilizing algorithm
  - convergence: $\exists t \geq t_0 : c(t) \models P$
  - consistency: $\forall t_l > t_k : c(t_k) \models P \Rightarrow c(t_l) \models P$
- two systems:

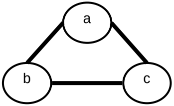
- distributed self-stabilizing breadth first search (BFS) [Dol00]

System Model

- serial execution semantics
- central demon/scheduler/monitor
- shared memory model
- perfect fault detection
- fault model: transient faults strike mem register with static probability
- distributed self-stabilizing algorithm
  - convergence: $\exists t \geq t_0 : c(t) \models P$
  - consistency: $\forall t_l > t_k : c(t_k) \models P \Rightarrow c(t_l) \models P$
- two systems:

System Model

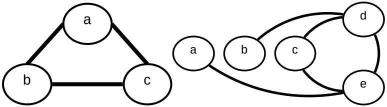
- serial execution semantics
- central demon/scheduler/monitor
- shared memory model
- perfect fault detection
- fault model: transient faults strike mem register with static probability
- distributed self-stabilizing algorithm
  - convergence: $\exists t \geq t_0 : c(t) \models P$
  - consistency: $\forall t_l > t_k : c(t_k) \models P \Rightarrow c(t_l) \models P$
- two systems:



- distributed self-stabilizing breadth first search (BFS) [Dol00]

System Model

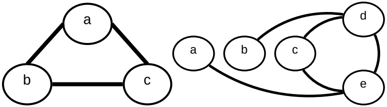
- serial execution semantics
- central demon/scheduler/monitor
- shared memory model
- perfect fault detection
- fault model: transient faults strike mem register with static probability
- distributed self-stabilizing algorithm
  - convergence: $\exists t \geq t_0 : c(t) \models P$
  - consistency: $\forall t_l > t_k : c(t_k) \models P \Rightarrow c(t_l) \models P$
- two systems:



- distributed self-stabilizing breadth first search (BFS) [Dol00]

System Model

- serial execution semantics
- central demon/scheduler/monitor
- shared memory model
- perfect fault detection
- fault model: transient faults strike mem register with static probability
- distributed self-stabilizing algorithm
  - convergence: $\exists t \geq t_0 : c(t) \models P$
  - consistency: $\forall t_l > t_k : c(t_k) \models P \Rightarrow c(t_l) \models P$
- two systems:



- distributed self-stabilizing breadth first search (BFS) [Dol00]

Methods

1. Analysis: build state space (3p: 6561, 5p: $\sim$7 Billion), form IWA in PCTL with *final* argument, calculate with PRISM [KNP07]

   $P =?[F <= 100 \text{"}state6523\text{"}\{true\}\{min\}]$

2. Simulation: build system, execute n steps, see, if $c(t) \models P$, if not, count i until $c(t+i) \models P$ [MDT08]

Methods

1. Analysis: build state space (3p: 6561, 5p: $\sim$7 Billion), form IWA in PCTL with *final* argument, calculate with PRISM [KNP07]

$$P =?[F <= 100\text{''}state6523\text{''}\{true\}\{min\}]$$

2. Simulation: build system, execute n steps, see, if $c(t) \models P$, if not, count i until $c(t+i) \models P$ [MDT08]

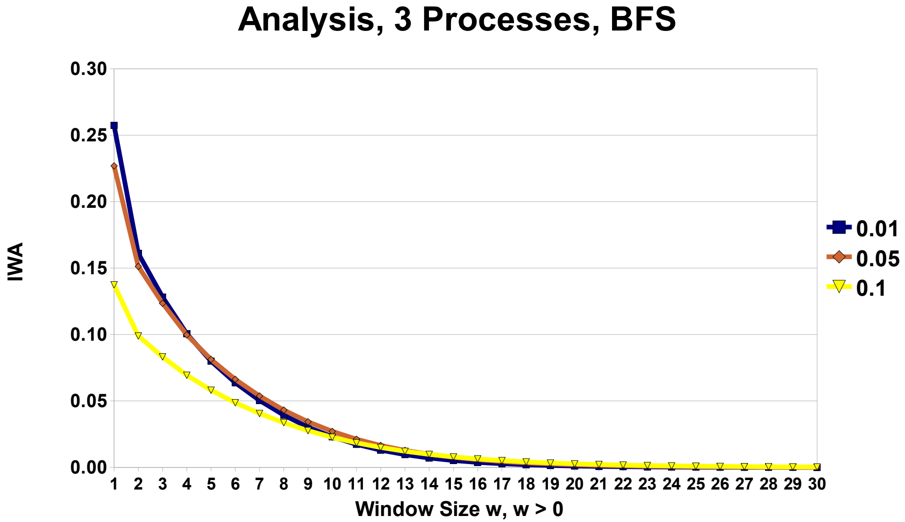
Methods

1. Analysis: build state space (3p: 6561, 5p: ∼7 Billion), form IWA in PCTL with *final* argument, calculate with PRISM [KNP07]

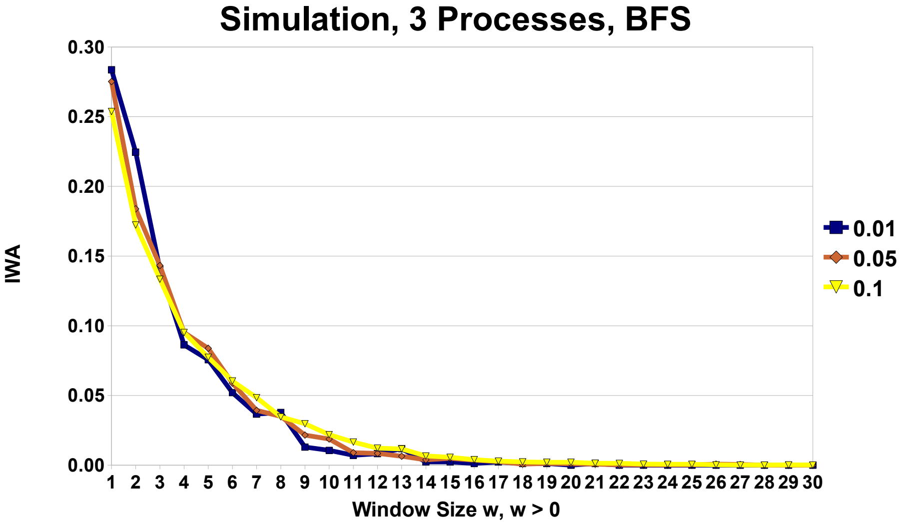
   $P = ?[F <= 100\text{''}state6523\text{''}\{true\}\{min\}]$

2. Simulation: build system, execute n steps, see, if $c(t) \models P$, if not, count i until $c(t+i) \models P$ [MDT08]

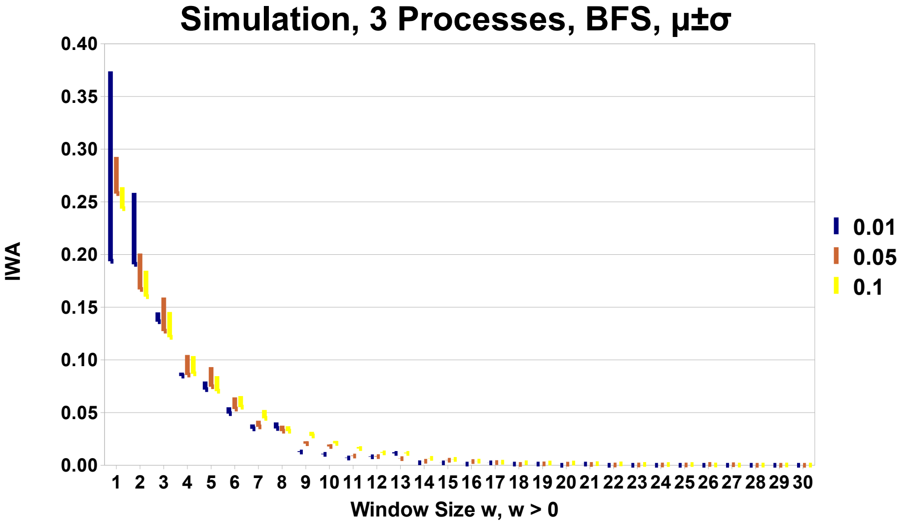
Results 3p System - Analysis



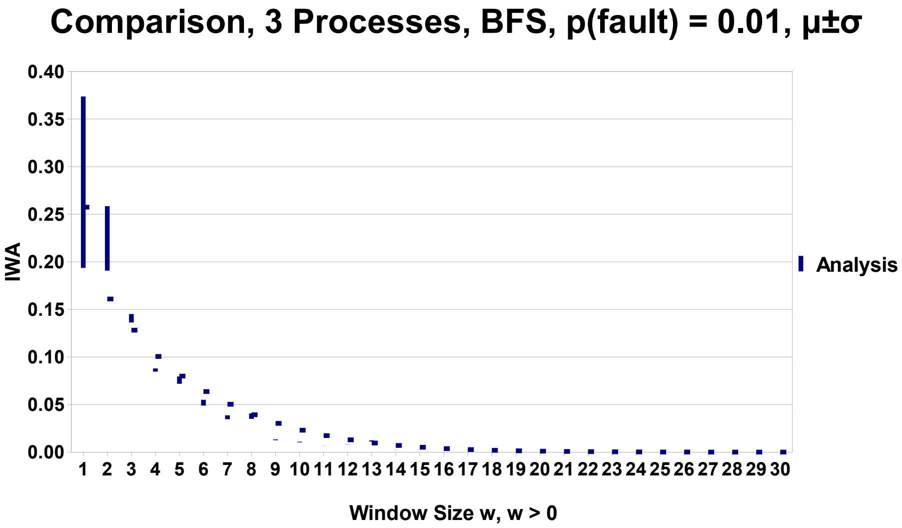
Results 3p System - Simulation



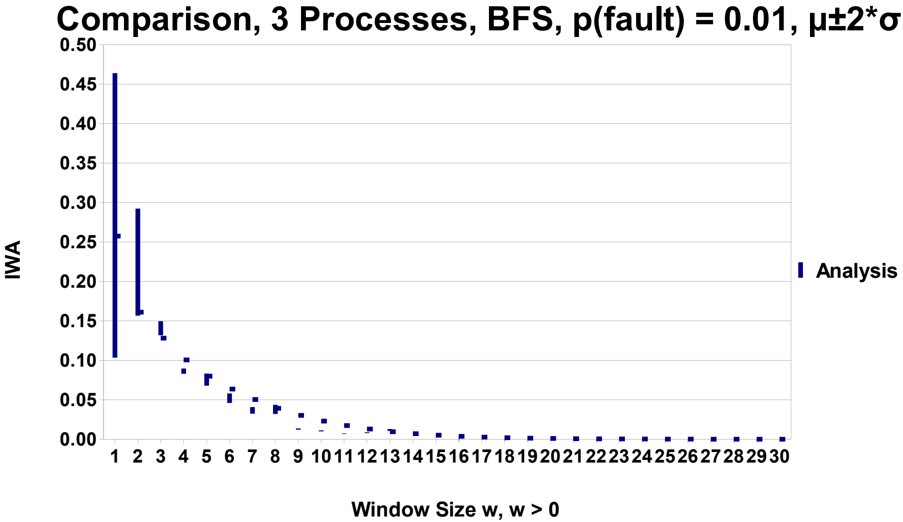
Results 3p System - Simulation - Standard Deviation



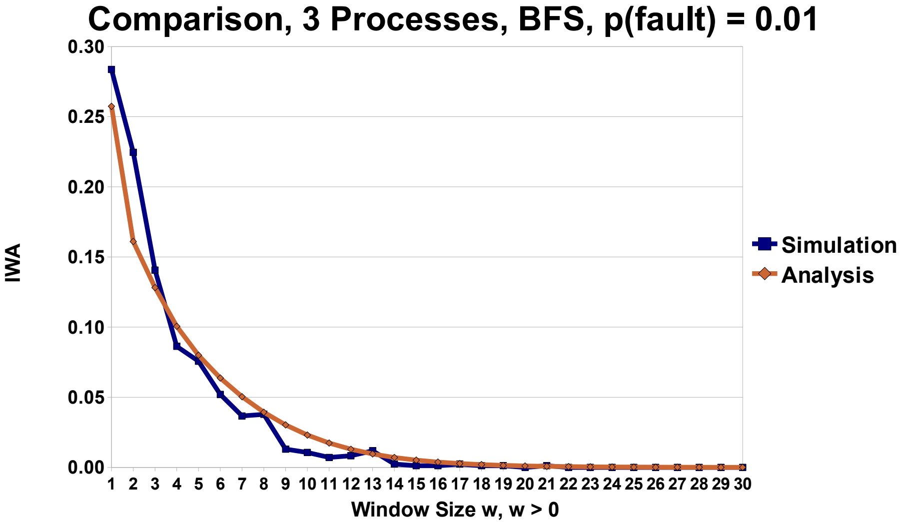
Results 3p System - Simulation - Standard Deviation



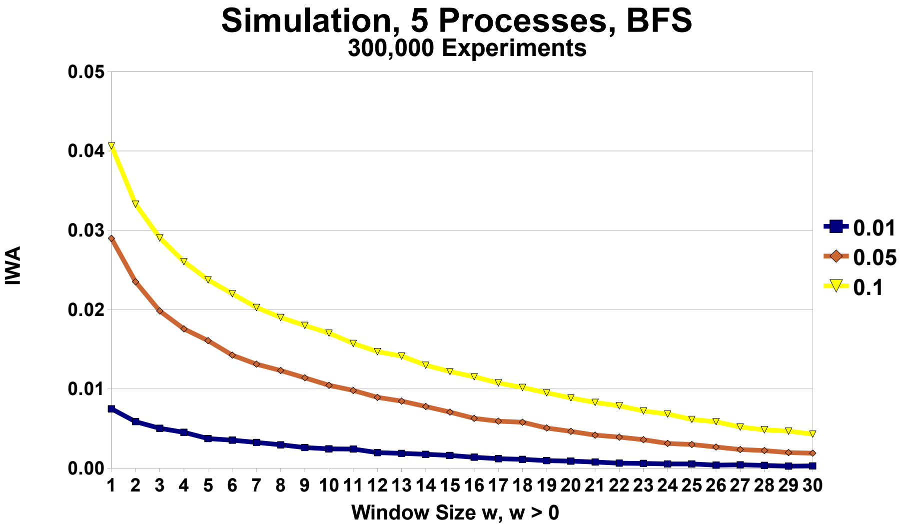
Results 3p System - Simulation - Standard Deviation



Comparison - Analysis & Simulation



Results 5p System - Simulation



Conclusion

- Relation: IWA vs. Delay
- Notion of IWA necessary to argue for trade-off.
- Analysis & Simulation coincide well.
- Limits of analysis (state space explosion) obvious, for simulation important for larger systems

Conclusion

- Relation: IWA vs. Delay
- Notion of IWA necessary to argue for trade-off.
- Analysis & Simulation coincide well.
- Limits of analysis (state space explosion) obvious, for simulation important for larger systems

Conclusion

- Relation: IWA vs. Delay
- Notion of IWA necessary to argue for trade-off.
- Analysis & Simulation coincide well.
- Limits of analysis (state space explosion) obvious, for simulation important for larger systems

Conclusion

- Relation: IWA vs. Delay
- Notion of IWA necessary to argue for trade-off.
- Analysis & Simulation coincide well.
- Limits of analysis (state space explosion) obvious, for simulation important for larger systems

Future Work

1. Framework for the Automatic Derivation of Trade-off solutions
    - find Pareto-optimal solutions
    - more dimensions (consistency, frequency, ...)
    - distributed analysis to cope with complex systems
2. Real world experiments with WSNs: constistency vs. availability vs. energy consumption vs. collisions vs. code strength vs. delay vs. ...

Future Work

1. Framework for the Automatic Derivation of Trade-off solutions
   - find Pareto-optimal solutions
   - more dimensions (consistency, frequency, ...)
   - distributed analysis to cope with complex systems
2. Real world experiments with WSNs: constistency vs. availability vs. energy consumption vs. collisions vs. code strength vs. delay vs. ...

Future Work

1. Framework for the Automatic Derivation of Trade-off solutions
   - find Pareto-optimal solutions
   - more dimensions (consistency, frequency, ...)
   - distributed analysis to cope with complex systems
2. Real world experiments with WSNs: constistency vs. availability vs. energy consumption vs. collisions vs. code strength vs. delay vs. ...

Future Work

1. Framework for the Automatic Derivation of Trade-off solutions
    - find Pareto-optimal solutions
    - more dimensions (consistency, frequency, ...)
    - distributed analysis to cope with complex systems
2. Real world experiments with WSNs: constistency vs. availability vs. energy consumption vs. collisions vs. code strength vs. delay vs. ...

Future Work

1. Framework for the Automatic Derivation of Trade-off solutions
   - find Pareto-optimal solutions
   - more dimensions (consistency, frequency, ...)
   - distributed analysis to cope with complex systems
2. Real world experiments with WSNs: constistency vs. availability vs. energy consumption vs. collisions vs. code strength vs. delay vs. ...

Future Work

1. Framework for the Automatic Derivation of Trade-off solutions
   - find Pareto-optimal solutions
   - more dimensions (consistency, frequency, ...)
   - distributed analysis to cope with complex systems
2. Real world experiments with WSNs: constistency vs. availability vs. energy consumption vs. collisions vs. code strength vs. delay vs. ...

Future Work

1. Framework for the Automatic Derivation of Trade-off solutions
   - find Pareto-optimal solutions
   - more dimensions (consistency, frequency, ...)
   - distributed analysis to cope with complex systems
2. Real world experiments with WSNs: constistency vs. availability vs. energy consumption vs. collisions vs. code strength vs. delay vs. ...

Future Work

1. Framework for the Automatic Derivation of Trade-off solutions
   - find Pareto-optimal solutions
   - more dimensions (consistency, frequency, ...)
   - distributed analysis to cope with complex systems
2. Real world experiments with WSNs: constistency vs. availability vs. energy consumption vs. collisions vs. code strength vs. delay vs. ...

Abhishek Dhama, Oliver E. Theel, and Timo Warns.
Reliability and Availability Analysis of Self-Stabilizing Systems.
In *SSS*, pages 244–261, 2006.

Shlomi Dolev.
Self-Stabilization.
MIT Press, Cambridge, MA, USA, 2000.

Nils Müllner, Abhishek Dhama, and Oliver Theel.
Derivation of Fault Tolerance Measures of Self-Stabilizing Algorithms by Simulation.
In *Proc. of AnSS'08*, pages 183–192. April 2008.

Edsger W. Dijkstra.
Self-Stabilizing Systems in Spite of Distributed Control.
Commun. ACM, 17(11):643–644, 1974.

M. Kwiatkowska, G. Norman, and D. Parker.
Stochastic Model Checking.
In*Proc. of SFM'07*, volume 4486 of *LNCS (Tutorial Volume)*, pages 220–270. Springer, 2007.

Thank your for your attention!

nils.muellner@informatik.uni-oldenburg.de

Questions?

Thank your for your attention!

nils.muellner@informatik.uni-oldenburg.de

Questions?

Thank your for your attention!

nils.muellner@informatik.uni-oldenburg.de

Questions?