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

1 Motivation

2 New Availability Metric

3 Results from Analysis and Simulation

4 Conclusion

5 Future Work
Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems

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

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

- How much delay 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?
Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems.

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

How much delay is desired? ⇒ as short as possible...

How available should your system be? ⇒ as high as possible...

What is a good trade-off in between delay and availability?
Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems

How much delay is desired? ⇒ as short as possible...

How available should your system be? ⇒ as high as possible...

What is a good trade-off in between delay and availability?
Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems.

- How much delay is desired? ⇒ as short as possible...

- How available should your system be? ⇒ as high as possible...

- What is a good trade-off in between delay and availability?
Trade-off between time (steps to wait for intended service) and availability increase (to get intended service) in distributed systems

- How much delay is desired? ⇒ as short as possible...

- How available should your system be? ⇒ as high as possible...

- What is a good trade-off in between delay and availability?
Orientation

- Masking
- Nonmasking
- Self-stabilizing

Trade-off: System Avbl vs Time Redundancy
Instantaneous Window Availability

Trade-off: System Avbl vs Time Redundancy

Nils Müllner (Universität Oldenburg)
Availability

\[ Availability : A = \frac{MTTF}{MTBF} \]  \hspace{1cm} (1)

- **instantaneous availability**: at some arbitrary time \( k \) the system is available: \( A(k) \)
- **limiting availability**: the same, as \( k \) approaches \( \infty \)
- 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

\[ Availability : A = \frac{MTTF}{MTBF} \]  \hspace{1cm} (1)

- **instantaneous availability**: at some arbitrary time \( k \) the system is available: \( A(k) \)
- **limiting availability**: the same, as \( k \) approaches \( \infty \)
- 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

\[ Availability : A = \frac{MTTF}{MTBF} \] (1)

- **instantaneous availability**: at some arbitrary time \( k \) the system is available: \( A(k) \)
- **limiting availability**: the same, as \( k \) approaches \( \infty \)
- 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

\[ Availability : A = \frac{MTTF}{MTBF} \]  \hspace{1cm} (1)

- **instantaneous availability**: at some arbitrary time \( k \) the system is available: \( A(k) \)
- **limiting availability**: the same, as \( k \) approaches \( \infty \)
- 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

\[ \text{Availability} : A = \frac{MTTF}{MTBF} \quad (1) \]

- **instantaneous availability**: at some arbitrary time \( k \) the system is available: \( A(k) \)
- **limiting availability**: the same, as \( k \) approaches \( \infty \)
- 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

\[ \text{Availability} : A = \frac{MTTF}{MTBF} \]  

- **instantaneous availability**: at some arbitrary time \( k \) the system is available: \( A(k) \)
- **limiting availability**: the same, as \( k \) approaches \( \infty \)
- 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

\[ \text{Availability} : A = \frac{MTTF}{MTBF} \quad (1) \]

- **instantaneous availability**: at some arbitrary time \( k \) the system is available: \( A(k) \)
- **limiting availability**: the same, as \( k \) approaches \( \infty \)
- 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

\[ \text{Availability} : A = \frac{MTTF}{MTBF} \]  

- **instantaneous availability**: at some arbitrary time \( k \) the system is available: \( A(k) \)
- **limiting availability**: the same, as \( k \) approaches \( \infty \)
- 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

\[ \text{Availability} : A = \frac{MTTF}{MTBF} \]  \hspace{1cm} (1)

- **instantaneous availability**: at some arbitrary time \( k \) the system is available: \( A(k) \)
- **limiting availability**: the same, as \( k \) approaches \( \infty \)
- 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: $\forall t_i > t_k : c(t_k) \models P \Rightarrow c(t_i) \models P$
- 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: \( \exists t \geq t_0 : c(t) = P \)
  - consistency: \( \forall t_i > t_k : c(t_k) = P \Rightarrow c(t_i) = P \)
- 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) [Del00]
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) [Del00]
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: \( \forall t_I > t_K : c(t_K) \models P \Rightarrow c(t_I) \models P \)
- two systems:

- distributed self-stabilizing breadth first search (BFS) [Do10]
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) [Doi00]
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:

  ![Diagram](image_url)

- distributed self-stabilizing breadth first search (BFS) [Doi00]
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:

```
  a
 b c
  d
```

- 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:

  ![Diagram](attachment:image.png)

  - distributed self-stabilizing breadth first search (BFS) [Dol00]
Analysis: build state space (3p: 6561, 5p: ~7 Billion), form IWA in PCTL with final argument, calculate with PRISM [KNP07]

\[ P = \{ F \leq 100 \text{"state6523"} \{ true \}\{ min \} \] 

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: \(\sim7\) Billion), form IWA in PCTL with final argument, calculate with PRISM [KNP07]

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

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]

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

2. Simulation: build system, execute $n$ steps, see, if $c(t) \models P$, if not, count $i$ until $c(t+i) \models P$ [MDT08]
Analysis, 3 Processes, BFS

IWA

Window Size $w, w > 0$

- 0.01
- 0.05
- 0.1
Simulation, 3 Processes, BFS

Window Size $w$, $w > 0$

IWA

0.00
0.05
0.10
0.15
0.20
0.25
0.30

0.01
0.05
0.1
Comparison, 3 Processes, BFS, \( p(\text{fault}) = 0.01 \), \( \mu \pm \sigma \)
Comparison, 3 Processes, BFS, $p(\text{fault}) = 0.01, \mu \pm 2\sigma$

Analysis

Window Size $w, w > 0$

IWA
Comparison, 3 Processes, BFS, $p(\text{fault}) = 0.01$

- **Simulation**
- **Analysis**

Window Size $w$, $w > 0$

IWA

Comparison - Analysis & Simulation
Simulation, 5 Processes, BFS
300,000 Experiments

Window Size $w$, $w > 0$

IWA

Trade-off: System Avbl vs Time Redundancy
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: consistency 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: consistency 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: consistency 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: consistency 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: consistency 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: consistency 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: consistency 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: consistency vs. availability vs. energy consumption vs. collisions vs. code strength vs. delay vs. ...


Thank you for your attention!

nils.muellner@informatik.uni-oldenburg.de

Questions?
Thank you for your attention!

nils.muellner@informatik.uni-oldenburg.de

Questions?
Thank you for your attention!

nils.muellner@informatik.uni-oldenburg.de

Questions?