		Status and Outlook

# Unmasking Fault Tolerance: Masking vs. Non-masking Fault-tolerant Systems

#### Nils Müllner

nils.muellner@informatik.uni-oldenburg.de Abteilung Systemsoftware und verteilte Systeme Department für Informatik Carl von Ossietzky Universität Oldenburg

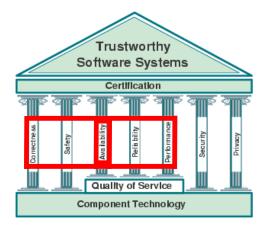
February 22, 2011





			Status and Outlook

#### Orientation



#### Nils Müllner

			Status and Outlook

# Outline



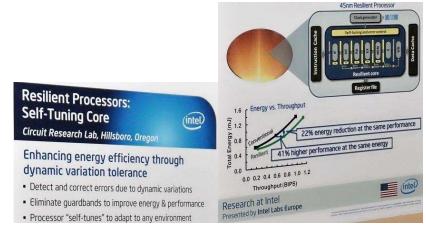
2 Basics

- **3** Computation of *LWAV*
- **4** Lumping
- **5** Decomposition

#### 6 Status and Outlook

Motivation			Status and Outlook
00			

#### Intel: Palisades



#### [BTL+10]

#### Nils Müllner



#### Focus: Basic Research

 fault tolerance in distributed systems is important for a variety of systems like CPU, WSN, ...

#### Focus: Basic Research

- fault tolerance in distributed systems is important for a variety of systems like CPU, WSN, ...
- focus: not **system specific** fault tolerance methods, but fundamental principles.

#### Focus: Basic Research

- fault tolerance in distributed systems is important for a variety of systems like CPU, WSN, ...
- focus: not **system specific** fault tolerance methods, but fundamental principles.

 $\Rightarrow$ : relation between quality (degree of masking) and cost.

	Basics		Status and Outlook





- **3** Computation of *LWAV*
- **4** Lumping
- **5** Decomposition
- 6 Status and Outlook

Basics		Status and Outlook
00000000		

## Outline

- 1 fault tolerance demands redundancy
- 2 fault tolerance classification
- 3 the fault masker concept
- 4 unmasking fault tolerance
- 6 redundancy classification
- 6 self-stabilization

## Fault Tolerance Demands Redundancy

- to tolerate faults, they must be detected and/or corrected
- detection and correction are functions that require resources
- typically either space (functional or information redundancy) or time (but commonly both)
- sometimes convertible (e.g., TMR)

# Fault Tolerance Demands Redundancy

- to tolerate faults, they must be detected and/or corrected
- detection and correction are functions that require resources
- typically either space (functional or information redundancy) or time (but commonly both)
- sometimes convertible (e.g., TMR)

Example: **Cyclic Redundancy Checks (CRC)** requires space (extends the package, information redundancy), and more space (code for the computation of CRC, functional redundancy), and time (for the computation, and transmission, temporal

redundancy)

Basics		Status and Outlook
000000000		

	safe	not safe
live	masking	non-masking
not live	failsafe	intolerant

Table: Fault Tolerance Classes [KA97, Gär99]

		not safe	$\leftarrow detectors$		
live	masking	non-masking	-		
not live	failsafe	non-masking intolerant			
↑ correctors Table: Fault Tolerance Classes [KA97, Gär99]					

		not safe	$\leftarrow detectors$		
live	masking	non-masking	-		
not live	failsafe	non-masking intolerant			
↑ correctors					
Table: Fault Tolerance Classes [KA97, Gär99]					

#### non-masking fault tolerance

- requires correction
- relatively cheap

- safe
   not safe
   ← detectors

   live
   masking
   non-masking
   ←

   not live
   failsafe
   intolerant

   ↑ correctors
   Table:
   Fault Tolerance Classes [KA97, Gär99]
- non-masking fault tolerance
  - requires correction
  - relatively cheap

#### masking fault tolerance

- requires detection and correction
- most desirable

		not safe	$\leftarrow detectors$		
live	masking	non-masking	-		
not live	failsafe	non-masking intolerant			
↑ correctors					
Table: Fault Tolerance Classes [KA97, Gär99]					

non-masking fault tolerance

- requires correction
- relatively cheap

masking fault tolerance

- requires detection and correction
- most desirable

non-/masking fault tolerant with regards to a distinct fault class

Nils Müllner

Basics 000●00000		Status and Outlook

### Example: CRC

- intolerant: corrupted packet contained matching checksum
- non-masking fault tolerant: faults were detected, but could not be corrected / re-request violates temporal boundaries
- masking: correct transmission / or faults could be corrected on the spot

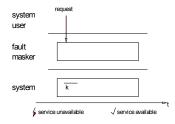
Basics 000●00000		Status and Outlook

#### Example: CRC

- intolerant: corrupted packet contained matching checksum
- non-masking fault tolerant: faults were detected, but could not be corrected / re-request violates temporal boundaries
- masking: correct transmission / or faults could be corrected on the spot

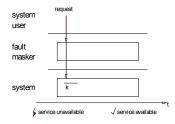
non-/masking fault tolerant with regards to a distinct fault class

	Basics				Status and Outlook
00	000000000	0000000	00000	000000	000



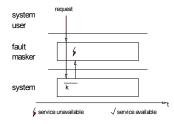
#### [MDT09]

	Basics				Status and Outlook
00	000000000	0000000	00000	000000	000



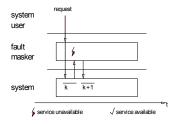
#### [MDT09]

	Basics				Status and Outlook
00	000000000	0000000	00000	000000	000



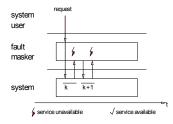
#### [MDT09]

	Basics				Status and Outlook
00	000000000	0000000	00000	000000	000



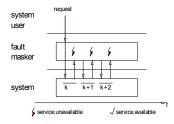
#### [MDT09]

	Basics				Status and Outlook
00	000000000	0000000	00000	000000	000



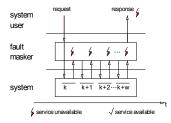
#### [MDT09]

	Basics				Status and Outlook
00	000000000	0000000	00000	000000	000

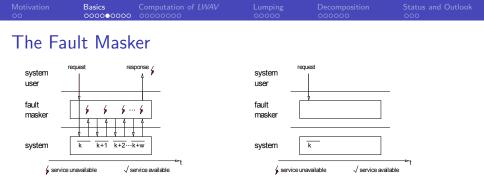


#### [MDT09]

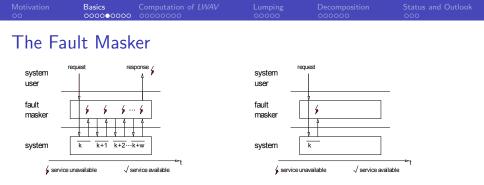
Motivation Basics Computation of LWAV Lumping Decomposition	Status and Outlook	
00 0000000 0000000 000000 000000	000	l,

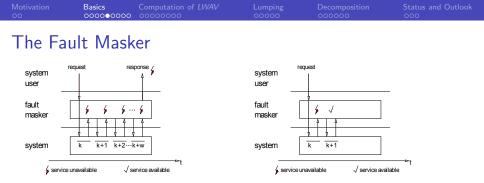


#### [MDT09]



11/47





Motivation 00	Basics 0000●0000	Computation of <i>LWAV</i>	Lumping 00000	Decomposition 000000	Status and Outlook
The Fau	ult Mask	ker			
system re user	rquest res	sponse /	system reque user	st response √	
fault masker	↓	· • •	fault masker	<i>¥</i> √	

system

k+1

√ service available

k

🖌 service unavailable

[MDT09]

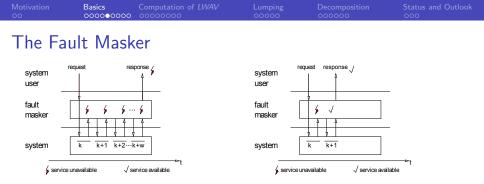
k

🖌 service unavailable

k+1 k+2...k+w

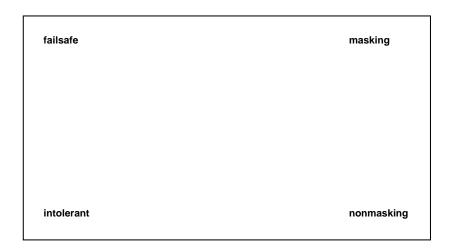
√ service available

system



#### the fault masker detects all faults

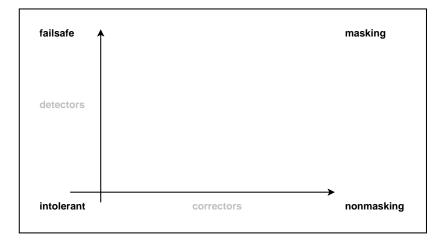
#### $\mathsf{Detection} \Rightarrow \mathsf{Safety}, \ \mathsf{Correction} \Rightarrow \mathsf{Liveness}$



Nils Müllner



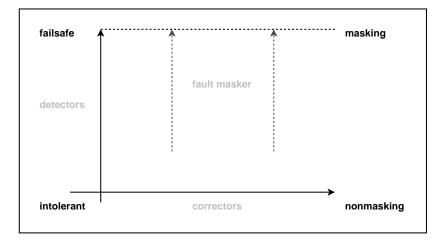
#### Detection $\Rightarrow$ Safety, Correction $\Rightarrow$ Liveness



Nils Müllner



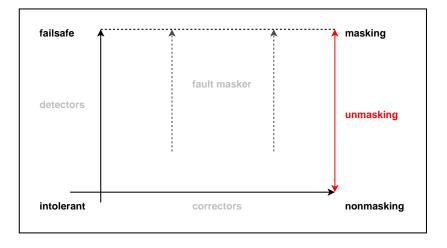
#### Detection $\Rightarrow$ Safety, Correction $\Rightarrow$ Liveness



Nils Müllner



#### Detection $\Rightarrow$ Safety, Correction $\Rightarrow$ Liveness



Nils Müllner

### Redundancy Establishes Detection and Correction

- information redundancy
  - error correcting or detecting codes
  - N-Modular Redundancy

- temporal Redundancy
  - self-stabilization
  - re-requests
  - N-Modular Redundancy

# Focus: Correction Based on Temporal Redundancy (e.g., Self-Stabilization)

information redundancy: thoroughly discussed

- we can compute the quality of spatial redundancy (i.e., number and severity of faults covered, either in a masking or non-masking fashion)
- spatial redundancy commonly used to ensure data integrity

# Focus: Correction Based on Temporal Redundancy (e.g., Self-Stabilization)

information redundancy: thoroughly discussed

- we can compute the quality of spatial redundancy (i.e., number and severity of faults covered, either in a masking or non-masking fashion)
- spatial redundancy commonly used to ensure data integrity

temporal redundancy (assuming detection as given):

- commonly used for system integrity
- how good can time heal/cure the system from faults?
- what is a proper metric?
- how can we calculate this metric?

# Self-Stabilization

## Definition (Self-Stabilization [Dol00, Dij74])

A system is self-stabilizing wrt. a safety predicate  $\mathcal{P}$  iff:

- Starting from any state, it is guaranteed that the system will eventually reach a state that satisfies the safety predicate P (convergence property), provided that no fault happens.
- Q Given that the system satisfies the safety predicate, it is guaranteed to stay in a state that satisfies the safety predicate P (closure property), provided that no fault happens.

Motivation Basics Computation of LWAV Lumping Decomposition		
00 00000000 0000000 000000 000000 000000	000	





## **3** Computation of *LWAV*

## **4** Lumping

**6** Decomposition

## 6 Status and Outlook

#### Nils Müllner

# A Suitable Metric 1/2

## Definition (Limiting Window Availability (LWA))

Assume that at time t = 0, an initial system state distribution holds that corresponds to the steady state distribution of a system. Then, Limiting Window Availability of window size w (of this system), denoted by LWA<sub>w</sub>,  $w \ge 0$ , is the probability that the system has at least once reached a state satisfying  $\mathcal{P}$  within the following w time steps:

$$LWA_w = prob\{\exists k, 0 \le k \le w : c_k \models \mathcal{P}\}$$

w is called window size.

Nils Müllner

	Computation of <i>LWAV</i> ○●○○○○○○		Status and Outlook

# A Suitable Metric 2/2

Definition (Limiting Window Availability Vector (LWAV))

The limiting window availability vector of size *i* (of a system), denoted by LWAV<sub>i</sub>, is an *i*-dimensional vector of probabilities. The element in the *i*<sup>th</sup> position is the limiting window availability of window size *i* - 1 of that system:  $LWAV_i := \langle LWA_0, LWA_1, \dots, LWA_{i-1} \rangle$ .

# A Suitable Metric 2/2

Definition (Limiting Window Availability Vector (LWAV))

The limiting window availability vector of size *i* (of a system), denoted by LWAV<sub>i</sub>, is an *i*-dimensional vector of probabilities. The element in the *i*<sup>th</sup> position is the limiting window availability of window size *i* - 1 of that system:  $LWAV_i := \langle LWA_0, LWA_1, \dots, LWA_{i-1} \rangle$ .

# Definition (Limiting Window Availability Vector Gradient (*LWAVG*))

The limiting window availability vector gradient of size *i* (of a system), denoted by LWAVG<sub>i</sub>, is an *i*-dimensional vector of probabilities. The element in the *i*<sup>th</sup> position is the limiting window availability of window size *i* minus the limiting window availability of window size *i* – 1 of that system:

 $LWAVG_i := \langle LWA_1 - LWA_0, LWA_2 - LWA_1, \dots, LWA_i - LWA_{i-1} \rangle.$ 

#### Nils Müllner

		Computation of LWAV			Status and Outlook
<u>const</u> ic <u>var</u> reg repeat{ reg : Figure: the Root	$\mathbf{I} := 0,$	Algorithm and Sub-Algorithm for	$\frac{const}{const}$ id $\frac{const}{const}$ id $\frac{var}{reg},$ $\frac{var}{reg},$ repeat{ $\neg((\exists reg),$ $\rightarrow$ $\Box \exists reg_i,$ $\rightarrow$ Figure:	eighbors $:= \{$ I $:= \min\{id(\pi)\}$	$i_{i}, \dots \} + 1,$ $e_{i}, e_{j} \in \{1, 2\}$ $t_{i} = id-1\}$ $t_{i} = 2$ $t_{i} = 0$ $reg_{i} = 0$ $reg_{i} = 2$ Algorithm for
NULL NATILITY					10/47

Motivation 00	Basics 00000000	Computation of <i>LWAV</i>	Lumping 00000	Decomposition 000000	Status and Outlook
const id var reg, repeat { reg : Figure: the Root	I := 0, = 0} Broadcast t Process 	Sub-Algorithm for	$\frac{const}{const} id$ $\frac{const}{var} id$ $\frac{var}{reg},$ $\frac{var}{reg},$ $repeat{}$ $\neg((\exists reg$ $xor(\exists reg);$ $\rightarrow$ $\Box \exists reg_i$	$eighbors := \{$ $I := min\{id(\pi)$	$\{i_i\}, \dots\} + 1,$ $\{eg_i\} \in$ $\{i_i\} = id-1\}$ $\{et \land reg_i = 2\}$ $\{et \land reg_i = 0\}$ $\{reg_i = 0\}$
			Figure: E	Broadcast Sub-	Algorithm for

Motivation 00	Basics 000000000	Computation of <i>LWAV</i>	Lumping 00000	Decomposition 000000	Status and Outlook
Test Set <u>const</u> id <u>var</u> reg, repeat{ reg :=	= 0; $= 0;$ Broadcast S Process $= 0$	Sub-Algorithm for	Topology <u>const</u> id <u>var</u> reg, <u>var</u> set neig repeat{ ¬((∃reg xor(∃re →I □∃reg;: →		$\{\pi_i, \ldots\},\ i), \ldots\} + 1,$ $\{\mathbf{e}_{i}\} \in \{\mathbf{f}_i\} = \mathrm{id} - 1\}$ $\mathbf{t} \wedge \mathbf{reg}_i = 2$ ) $\{\mathbf{e}_i \wedge \mathbf{reg}_i = 0\}$ )
			Eleven I	Draadaaat Cub	Algorithms for

Figure: Broadcast Sub-Algorithm for Non-Root Processes

#### Nils Müllner

Motivation 00	Basics 00000000	Computation of <i>LWAV</i>	Lumping 00000	Decomposition 000000	Status and Outlook
const id var reg, repeat { reg : Figure: the Root	I := 0, = 0} Broadcast t Process 	Sub-Algorithm for	$\begin{array}{c} \hline const \ ni} \\ \hline const \ id \\ \hline var \ reg, \\ \hline var \ set \\ \hline neig \\ repeat \{ \\ \neg((\exists reg \\ xor(\exists re \\ \rightarrow) \\ \Box \exists reg_i \\ \rightarrow \\ \Box \exists reg_i \\ \end{array} \right)$	eighbors $:= \{$ I $:= min \{ id(\pi) \}$	$i_i), \dots \} + 1,$ $eg_i) \in$ $i_i = id - 1$ $et \land reg_i = 2$ $et \land reg_i = 0$ )) $\land reg_i = 0$
			Figure:	Broadcast Sub-	Algorithm for

#### Nils Müllner

Motivation 00	Basics 00000000	Computation of <i>LWAV</i>	Lumping 00000	Decomposition 000000	Status and Outlook
const id var reg, repeat { reg : Figure: the Root	I := 0, = 0} Broadcast t Process	Sub-Algorithm for	$\frac{const}{const} id$ $\frac{const}{var} id$ $\frac{var}{reg},$ $\frac{var}{reg},$ $repeat{}$ $\neg((\exists reg),$ $xor(\exists reg),$ $\rightarrow$ $\Box \exists reg_i;$	$eighbors := \{$ $I := min\{id(\pi)$	$\{i_i\}, \dots\} + 1,$ $\{eg_i\} \in$ $\{i_i\} = id-1\}$ $\{et \land reg_i = 2\}$ $\{et \land reg_i = 0\}$ $\{reg_i = 0\}$
			Figure: I	Broadcast Sub-	Algorithm for

Motivation 00	Basics 00000000	Computation of <i>LWAV</i>	Lumping 00000	Decomposition 000000	Status and Outlook
<u>const</u> id <u>var</u> reg; repeat{ reg :: Figure: I the Root	I := 0, = 0} Broadcast t Process 	Sub-Algorithm for	$\begin{array}{c} \hline const \ neig\\ \hline const \ id\\ \hline var \ reg,\\ \hline var \ set\\ \hline neig\\ repeat \{\\ \neg((\exists reg\\ xor(\exists re\\ \rightarrow i\\ \Box \exists reg_i;\\ \rightarrow\\ \hline \Box \exists reg_i;\\ \end{array}$	eighbors := { I := min{id( $\pi$	$i_i), \dots \} + 1,$ $eg_i) \in$ $i_i) = id - 1 \}$ $et \land reg_i = 2$ ) $et \land reg_i = 0$ )) $\land reg_i = 0$
			Figure:	Broadcast Sub-	Algorithm for

#### Nils Müllner

		Computation of <i>LWAV</i>			Status and Outlook
<u>const</u> id <u>var</u> reg, repeat{ reg :: the Root	I := 0, = 0}	Sub-Algorithm and $\pi$	$\frac{const}{const}$ id $\frac{const}{const}$ id $\frac{var}{reg},$ $\frac{var}{reg},$ repeat{ $\neg((\exists reg),$ $xor(\exists red),$ $\neg \exists reg;$ $\rightarrow$ $\exists reg;$ $\rightarrow$ Figure: find the set of the	eighbors := { I := min{id $(\pi_i)$	$i_{i}, \dots \} + 1,$ $e_{i}, e_{i} \in 0$ $i_{i} = id - 1$ $t \land reg_{i} = 2$ $e_{i} \land reg_{i} = 0$ $\land reg_{i} = 2$ Algorithm for
NUL MULLER					10/47

	Computation of <i>LWAV</i>			Status and Outlook
	Sub-Algorithm for $\pi_3$ $\pi_4$	Topology <u>const</u> id <u>const</u> id <u>var</u> reg, <u>var</u> set neig repeat{ $\neg((\exists reg)$ $xor(\exists reg)$ ; $\rightarrow$	000000	$\pi_i, \dots\},$ $\pi_i, \dots\} + 1,$ $\mathbf{eg}_i) \in$ $\mathbf{i}_i) = \mathrm{id} - 1\}$ $\mathbf{t} \wedge \mathbf{reg}_i = 2)$ $\mathbf{et} \wedge \mathbf{reg}_i = 0))$
$reg_2 = 0/$ $reg_3 = 0/$		-	reg := $2$ }	0.
01		Figure: I	Broadcast Sub-/	Algorithm for
		Non-Roc	ot Processes	

#### Nils Müllner

	Computation of <i>LWAV</i>		Status and Outlook

# The Resulting Markov Chain

$\langle 0, 0, 0, 0 \rangle$	$\langle 0, 0, 0, 2 \rangle$	$\langle 0, 0, 2, 0 \rangle$	$\langle 0, 2, 0, 0 \rangle$	$\langle 2, 0, 0, 0 \rangle$
$p(e_1 + e_2 + e_3 + e_4)$	$qe_4$	qe3	qe <sub>2</sub>	qe <sub>1</sub>
pe <sub>4</sub>	$p(e_1 + e_2 + e_3) + qe_4$			
pe <sub>3</sub>		$p(e_1 + e_2) + qe_3$		
pe <sub>2</sub>			$p(e_1 + e_4) + qe_2$	
pe <sub>1</sub>				$p(e_3 + e_4) + qe_1$
	pe3			
	pe <sub>2</sub>		pe <sub>4</sub>	
		pe <sub>2</sub>		
	pe <sub>1</sub>			pe <sub>4</sub>
		pe <sub>1</sub>		pe <sub>3</sub>
			pe <sub>1</sub>	
	$\frac{p(e_1 + e_2 + e_3 + e_4)}{pe_4}$ $\frac{pe_4}{pe_3}$ $pe_2$	$\begin{array}{c c} p(e_1 + e_2 + e_3 + e_4) & qe_4 \\ \hline pe_4 & pe_3 \\ pe_2 & pe_2 \\ pe_1 & pe_3 \\ pe_2 & pe_2 \\ pe_2 & pe_2 \end{array}$	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	$\begin{array}{c c c c c c c c c c c c c c c c c c c $

Table: Transitions Grouped by Number of Operational Processes

- $e_i$ : probability, that  $\pi_i$  is elected for execution
- q: probability, that a fault occurs
- p = 1 q
- $e_1 = e_2 = e_3 = e_4 = 0.25, \ q = 0.01$

#### Nils Müllner

$\begin{array}{ c c c c c c } \hline Compound & State & Steady State Probability \\ \hline 0 & \langle 0,0,0,0\rangle & 0.936254913358677 \\ \hline 1 & \langle 0,0,0,2\rangle & 0.020767040703947 \\ \hline 1 & \langle 0,0,2,0\rangle & 0.006443085000445 \\ \hline 1 & \langle 0,2,0,0\rangle & 0.005896554367512 \\ \hline 1 & \langle 2,0,0,0\rangle & 0.004721801275921 \\ \hline 2 & \langle 0,0,2,2\rangle & 0.011734460936930 \\ \hline 2 & \langle 0,2,0,2\rangle & 0.000103249069863 \\ \hline 2 & \langle 0,2,2,0\rangle & 0.003596242185866 \\ \hline 2 & \langle 2,0,0,2\rangle & 0.000101514623954 \\ \hline 2 & \langle 2,0,2,0\rangle & 0.00028052478081 \\ \hline 2 & \langle 2,2,0,0\rangle & 0.002411422793886 \\ \hline 3 & \langle 0,2,2,2\rangle & 0.000049131622044 \\ \hline 3 & \langle 2,2,0,2\rangle & 0.000042503806239 \\ \hline 3 & \langle 2,2,2,0\rangle & 0.001243938539611 \\ \hline 4 & \langle 2,2,2,2\rangle & 0.001401634860264 \\ \hline \end{array}$	vation	Basics Compu	tation of <i>LWAV</i>	Lumping 00000	Decomposition 000000	Status and Outlook
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		Compound	State	Steady St	ate Probability	
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		0	$\langle 0,0,0,0  angle$	0.9362549	913358677	
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		1	$\langle 0,0,0,2 \rangle$	0.0207670	040703947	
$\begin{array}{c c c c c c c c c c c c c c c c c c c $		1	$\langle 0,0,2,0  angle$	0.0064430	085000445	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		1	$\langle 0,2,0,0  angle$	0.0058965	554367512	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		1	$\langle 2,0,0,0 angle$	0.0047218	301275921	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		2	$\langle 0, 0, 2, 2 \rangle$	0.0117344	460936930	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		2	$\langle 0,2,0,2  angle$	0.0001032	249069863	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		2	$\langle 0,2,2,0  angle$	0.0035962	242185866	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		2	$\langle 2,0,0,2  angle$	0.000101	514623954	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		2	$\langle 2,0,2,0  angle$	0.0000280	052478081	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		2	$\langle 2,2,0,0 angle$	0.0024114	422793886	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		3	$\langle 0, 2, 2, 2 \rangle$	0.0052044	454376759	
<u>3</u> ⟨2,2,2,0⟩ 0.001243938539611		3	$\langle 2, 0, 2, 2 \rangle$	0.0000493	131622044	
		3	$\langle 2,2,0,2 \rangle$	0.0000425	503806239	
4 (2,2,2,2) 0.001401634860264		3	$\langle 2,2,2,0 \rangle$	0.0012439	938539611	
		4	$\langle 2,2,2,2 \rangle$	0.0014016	534860264	

Table: Steady State Probability Distribution

Nils Müllner

# How to Get There: State Space Analysis

- 1 compute transition probabilities between each pair of states
  - $\Rightarrow$  (ergodic) Markov chain
- 2 compute steady state probability distribution
- **3** use steady state distribution as initial probability distribution for modified chain
- 4 transform set of legal states into sink
- probability mass in set of legal states after *i* computation steps is *LWA<sub>i</sub>*

	Computation of <i>LWAV</i>		Status and Outlook

## The Markov Chain Yielding the LWA

$\downarrow$ from/to $\rightarrow$	$\langle 0,0,0,0  angle$	$\langle 0, 0, 0, 2 \rangle$	$\langle 0, 0, 2, 0 \rangle$	$\langle 0, 2, 0, 0 \rangle$	$\langle 2,0,0,0 angle$
<del>(0,0,0,0)</del>	$\frac{p(c_1 + c_2 + c_3 + c_4)}{c_1 + c_2 + c_3 + c_4}$	<del>dca</del>	<del>dc3</del>	<del>902</del>	<del>de</del> l
$\langle 0,0,0,0 \rangle$	1	0	0	0	0
$\langle 0, 0, 0, 2 \rangle$	pe <sub>4</sub>	$p(e_1 + e_2 + e_3) + qe_4$			
$\langle 0, 0, 2, 0 \rangle$	pe <sub>3</sub>		$p(e_1 + e_2) + qe_3$		
$\langle 0, 2, 0, 0 \rangle$	pe <sub>2</sub>			$p(e_1 + e_4) + qe_2$	
$\langle 2, 0, 0, 0 \rangle$	pe <sub>1</sub>				$p(e_3 + e_4) + qe_1$
$\langle 0, 0, 2, 2 \rangle$		pe <sub>3</sub>			
$\langle 0, 2, 0, 2 \rangle$		pe <sub>2</sub>		pe <sub>4</sub>	
$\langle 0, 2, 2, 0 \rangle$			pe <sub>2</sub>		
$\langle 2, 0, 0, 2 \rangle$		pe <sub>1</sub>			pe <sub>4</sub>
$\langle 2, 0, 2, 0 \rangle$			pe <sub>1</sub>		pe <sub>3</sub>
$\langle 2, 2, 0, 0 \rangle$				pe <sub>1</sub>	

Table: Transitions Grouped by Number of Operational Processes

Nils Müllner

	Computation of LWAV		Status and Outlook
	0000000		

# Limitations

- computation works for example
- but what about larger systems?
  - state space explosion is obvious
- solution: two ways
  - lumping
  - decomposition

	Lumping	Status and Outlook

## 1 Motivation



## **3** Computation of *LWAV*

## **4** Lumping

**5** Decomposition

## 6 Status and Outlook

#### Nils Müllner

• goal: smaller Markov chains

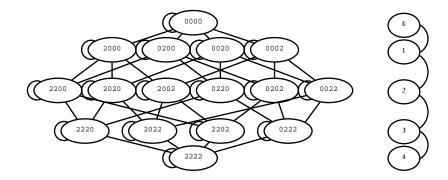
- goal: smaller Markov chains
- lumping aggregates states and transitions

- goal: smaller Markov chains
- lumping aggregates states and transitions
- question: what states (and transitions) can be lumped still being the *LWAV*?

- goal: smaller Markov chains
- lumping aggregates states and transitions
- question: what states (and transitions) can be lumped still being the *LWAV*?
- answer (for this example): all states that have the same amount of incorrect processes

		Computation of LWAV	Lumping		Status and Outlook
00 0	000000000	0000000	00000	000000	000

# Lumping Example 1/3



	Lumping	Status and Outlook
	00000	

Lumping Example 2/3

Lumping aggregates states and transitions.

	Lumping	Status and Outlook
	00000	

Lumping Example 2/3

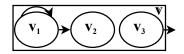
Lumping aggregates states and transitions.

$$prob(\overrightarrow{v,w}) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} p(\overrightarrow{v_i,w_j}) \cdot p(v_i)}{\sum_{i=0}^{n} p(v_i)}$$

Nils Müllner

	Lumping 000●0	Status and Outlook

Lumping Example 3/3



	Lumping 000●0	Status and Outlook

Lumping Example 3/3

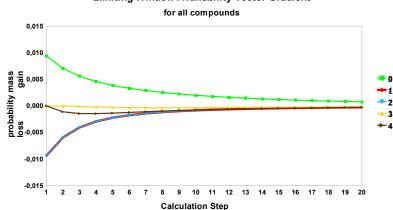
$$(v_1)$$
  $(v_2)$   $(v_3)$ 

$$p(\overrightarrow{\mathbf{v},\mathbf{v}}) = \frac{p(v1,v1) \cdot p(v1) + p(v1,v2) \cdot p(v1)}{p(v1) + p(v2) + p(v3)}$$

Nils Müllner



## Small Example: Result



## Limiting Window Availability Vector Gradient

#### Nils Müllner

00 0000000 0000000 000000 000000 000000	000

## 1 Motivation

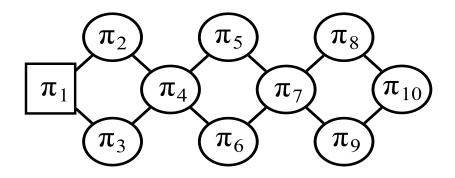


- **3** Computation of *LWAV*
- **4** Lumping
- **5** Decomposition

## 6 Status and Outlook

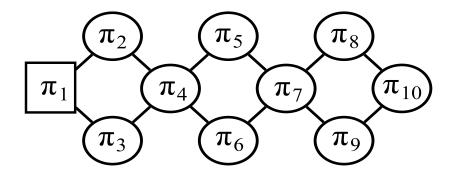
		Decomposition •00000	Status and Outlook

LWA at Large



		Decomposition ●00000	Status and Outlook

LWA at Large

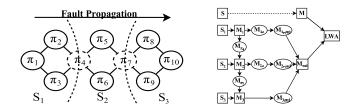


#### 17496 state Markov chain

Nils Müllner

# Decomposing and Lumping

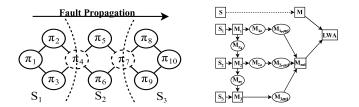
lumping aggregates states that have something in common



#### Nils Müllner

# Decomposing and Lumping

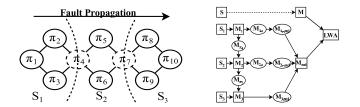
- lumping aggregates states that have something in common
- here: lumping of states that have the same amount of defective processes in common



#### Nils Müllner

# Decomposing and Lumping

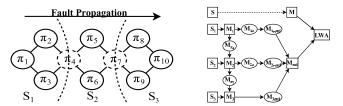
- lumping aggregates states that have something in common
- here: lumping of states that have the same amount of defective processes in common
- decomposition allows the construction of (much) smaller sub-Markov chains



#### Nils Müllner

## Decomposing and Lumping

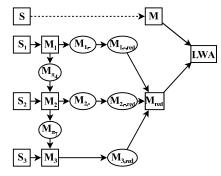
- lumping aggregates states that have something in common
- here: lumping of states that have the same amount of defective processes in common
- decomposition allows the construction of (much) smaller sub-Markov chains
- recomposition of smaller lumped Markov chains yields the exact result (80 instead of 17496 states)



#### Nils Müllner

				Decomposition	Status and Outlook
00	000000000	0000000	00000	00000	000

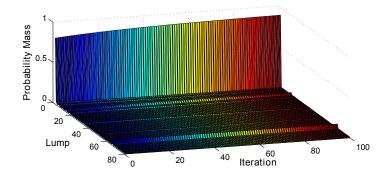
### **Decomposition Scheme**



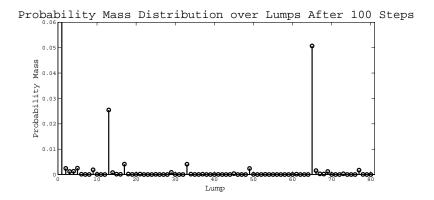
- Μ comprises 17496 states  $M_{1,-}$  $M_{2,-}$  $M_1$ comprises 24 states Mэ comprises 81 states  $M_{1,-,red}$  $M_{2,-,red}$ Мз comprises 81 states  $M_{3,red}$  $M_{\pi_A}$ comprises 3 states  $M_{\pi_7}$ comprises 3 states  $M_{red}$
- comprises 8 states comprises 27 states comprises 3 states comprises 3 states comprises 4 states comprises 80 states

#### Nils Müllner

		Decomposition	Status and Outlook
		000000	

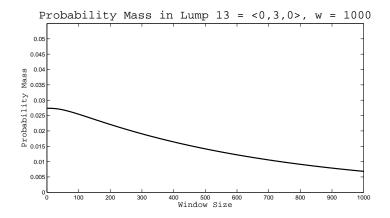






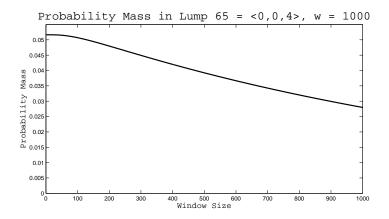
#### Nils Müllner





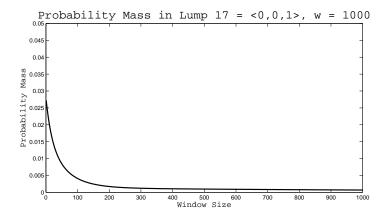
35/47





#### Nils Müllner





#### Nils Müllner



### Hierarchical Towards Heterarchical Systems 1/2

• fault propagation unidirectional

### Hierarchical Towards Heterarchical Systems 1/2

- fault propagation unidirectional
- decomposition easy: no cyclic dependencies

## Hierarchical Towards Heterarchical Systems 1/2

- fault propagation unidirectional
- decomposition easy: no cyclic dependencies
- what about any-way propagation

## Hierarchical Towards Heterarchical Systems 2/2

 <u>hierarchical</u> self-stabilizing systems demand a hierarchy (order) among the processes. Fault propagation strictly occurs from root towards leafs.

## Hierarchical Towards Heterarchical Systems 2/2

- <u>hierarchical</u> self-stabilizing systems demand a hierarchy (order) among the processes. Fault propagation strictly occurs from root towards leafs.
- <u>semi-hierarchical</u> self-stabilizing systems possess the ability to dynamically reassign the role of the root. Switching the root is called an epoch. Fault propagation during an epoch is unidirectional.

## Hierarchical Towards Heterarchical Systems 2/2

- <u>hierarchical</u> self-stabilizing systems demand a hierarchy (order) among the processes. Fault propagation strictly occurs from root towards leafs.
- <u>semi-hierarchical</u> self-stabilizing systems possess the ability to dynamically reassign the role of the root. Switching the root is called an epoch. Fault propagation during an epoch is unidirectional.
- <u>heterarchical</u> self-stabilizing systems achieve their goal in the absence of any order among the processes. Fault propagation can occur in any direction at any time.

			Status and Outlook

#### **1** Motivation



- **3** Computation of *LWAV*
- **4** Lumping
- **6** Decomposition

#### 6 Status and Outlook

			Status and Outlook ●○○

# Timeline

2006		20	07			20	08			20	09			20	10			20	11			20	12	
4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
<i></i>																								
		7777	7111	<i></i>	7772																			
					1	1111	1111	////	7777	////	1111	1111	7111	<i></i>	<i></i>	<i></i>		1						
																		////	1111	<i></i>	7777			
					1	////																		
																	////	1						
																				////	1			
																					1			
					1	////	(11)			////		<i>ann</i>												
										////			////	////						7777	1			
	4		4 1 2	4 1 2 3				4 1 2 3 4 1 2 3	4 1 2 3 4 1 2 3 4 11111 1111111111111111 11111111111	4     1     2     3     4     1     2     3     4     1       1111113     111111111111111111111111111111111111		4     1     2     3     4     1     2     3       1     1     2     3     4     1     2     3		4       1       2       3       4       1       2       3       4       1         1       1       2       3       4       1       2       3       4       1         1       1       1       2       3       4       1       2       3       4       1         1       1       1       2       3       4       1       2       3       4       1         1       1       1       1       1       1       1       2       3       4       1         1 <td< th=""><th></th><th>4       1       2       3       4       1       2       3       4       1       2       3         1       1       2       3       4       1       2       3       4       1       2       3         1       1       1       1       2       3       4       1       2       3         1       1       1       1       1       1       1       1       1       2       3         1       1       1       1       1       1       1       1       1       2       3         1       <td< th=""><th></th><th></th><th>4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       3       4       1       1       3       4       1       1       3       4</th><th>4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       1       1       1       1</th><th>4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1</th><th>4       1       2       3       4       1       1       2       3       4       1</th><th>4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       1       1       1       1</th><th>4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1</th></td<></th></td<>		4       1       2       3       4       1       2       3       4       1       2       3         1       1       2       3       4       1       2       3       4       1       2       3         1       1       1       1       2       3       4       1       2       3         1       1       1       1       1       1       1       1       1       2       3         1       1       1       1       1       1       1       1       1       2       3         1 <td< th=""><th></th><th></th><th>4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       3       4       1       1       3       4       1       1       3       4</th><th>4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       1       1       1       1</th><th>4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1</th><th>4       1       2       3       4       1       1       2       3       4       1</th><th>4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       1       1       1       1</th><th>4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1</th></td<>			4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       3       4       1       1       3       4       1       1       3       4	4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       1       1       1       1	4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1	4       1       2       3       4       1       1       2       3       4       1	4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       1       1       1       1	4       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1       1       2       3       4       1

### Current Focus

- FINA 22.-25. March
  - LWA, LWAV, and LWAVG
  - the computation thereof,
  - basics of lumping

 $\Rightarrow$  will be presented next month at 7<sup>th</sup> Int'l Symposium on Frontiers of Systems and Network Applications

- SSS 22. April: system decomposition of hierarchical self-stabilizing systems
- ICPADS 24. June: system decomposition of heterarchical self-stabilizing systems either by iterations, or maybe flow equations...
- writing it up

Motivation			Status and Outlook
00			000

goal: determination of the sweet spot

- be as masking as possible
- with as little effort as possible

goal: determination of the sweet spot

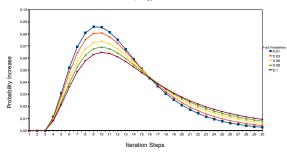
- be as masking as possible = maximize degree of masking fault tolerance
- with as little effort as possible = minimize time and space redundancy

goal: determination of the sweet spot

- be as masking as possible = maximize degree of masking fault tolerance
- with as little effort as possible = minimize time and space redundancy
- $\Rightarrow$  determination of the optimal trade-off thereof

goal: determination of the sweet spot

- be as masking as possible = maximize degree of masking fault tolerance
- with as little effort as possible = minimize time and space redundancy
- $\Rightarrow$  determination of the optimal trade-off thereof



WAVG, 4 Process Topology, Breadth First Search

#### Nils Müllner

Motivation 00	Basics 000000000	Computation of <i>LWAV</i>	Lumping 00000	Decomposition	Status and Outlook
		Dijkstra. zing Systems in Sp ACM, 17(11):643–		ributed Contro	ı.
	Shlomi Dole Self-Stabiliz MIT Press,		USA, 2000.		
	Yamashita. Weak vs. So In ICDCS'0 Distributed	Devismes, Sébastie elf vs. Probabilisti 8: Proc. of the 28 Computing Syste 2008. IEEE Compu	c Stabilizat 8th Internat ms, pages 6	ion. ional Conferen 581–688, Wasł	

#### Felix C. Gärtner.

Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments.

ACM Computing Surveys, 31(1):1–26, 1999.

Sandeep S. Kulkarni and Anish Arora. Compositional Design of Multitolerant Repetitive Byzantine Agreement.

Lecture Notes in Computer Science, 1346:169-182, 1997.

 A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y-R. Choi, T. Herman, S. S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita.
 A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking. Computer Networks, pages 605–634, 2004. Motivatio

 Keith A. Bowman, James W. Tschanz, Shih-Lien L. Lu, Paolo A. Aseron, Muhammad M. Khellah, Arijit Raychowdhury, Bibiche M. Geuskens, Carlos Tokunaga, Chris B. Wilkerson, Tanay Karnik, and Vivek K. De. Resilient Microprocessor Design for High Performance & Energy Efficiency.
 In ISLPED'10: Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design, pages 355–356, New York, NY, USA, 2010. ACM.

 Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber.
 DRAM Errors in the Wild: A Large-Scale Field Study.
 In SIGMETRICS'09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems, pages 193–204, New York, NY, USA, 2009. ACM.

			Status and Outlook

Nils Müllner, Abhishek Dhama, and Oliver Theel. Derivation of Fault Tolerance Measures of Self-Stabilizing Algorithms by Simulation.

In AnSS'08: Proceedings of the 41st Annual Symposium on Simulation, pages 183–192. IEEE Computer Society Press, April 2008.

Nils Müllner, Abhishek Dhama, and Oliver Theel. Deriving a Good Trade-off Between System Availability and Time Redundancy.

In Proceedings of the Symposia and Workshops on Ubiquitious, Automatic and Trusted Computing, number E3737, pages 61–67. IEEE Computer Society Press, July 2009.

		Status and Outlook 000

#### Nils Müllner and Oliver Theel.

The Degree of Masking Fault Tolerance vs. Temporal Redundancy.

To appear, In Proceedings of the 2011 IEEE 25th International Conference on Advanced Information Networking and Applications Workshops, FINA'11, Singapore, 2011. IEEE Computer Society.

		Status and Outlook

# Unmasking Fault Tolerance: Masking vs. Non-masking Fault-tolerant Systems

#### Nils Müllner

nils.muellner@informatik.uni-oldenburg.de Abteilung Systemsoftware und verteilte Systeme Department für Informatik Carl von Ossietzky Universität Oldenburg

February 22, 2011



