



CXI API Code Snippets

Topic: Load Balancing

Utimaco IS GmbH

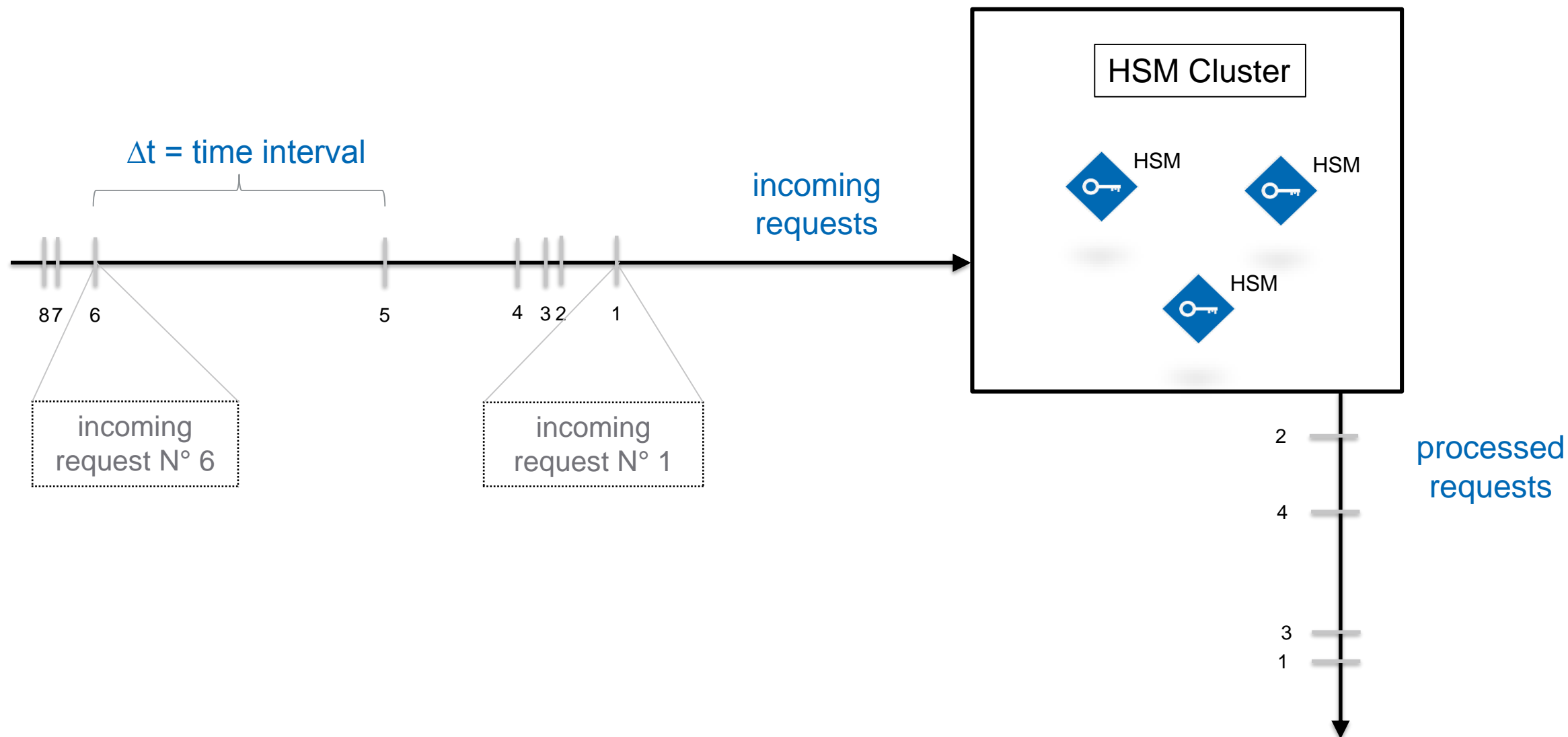


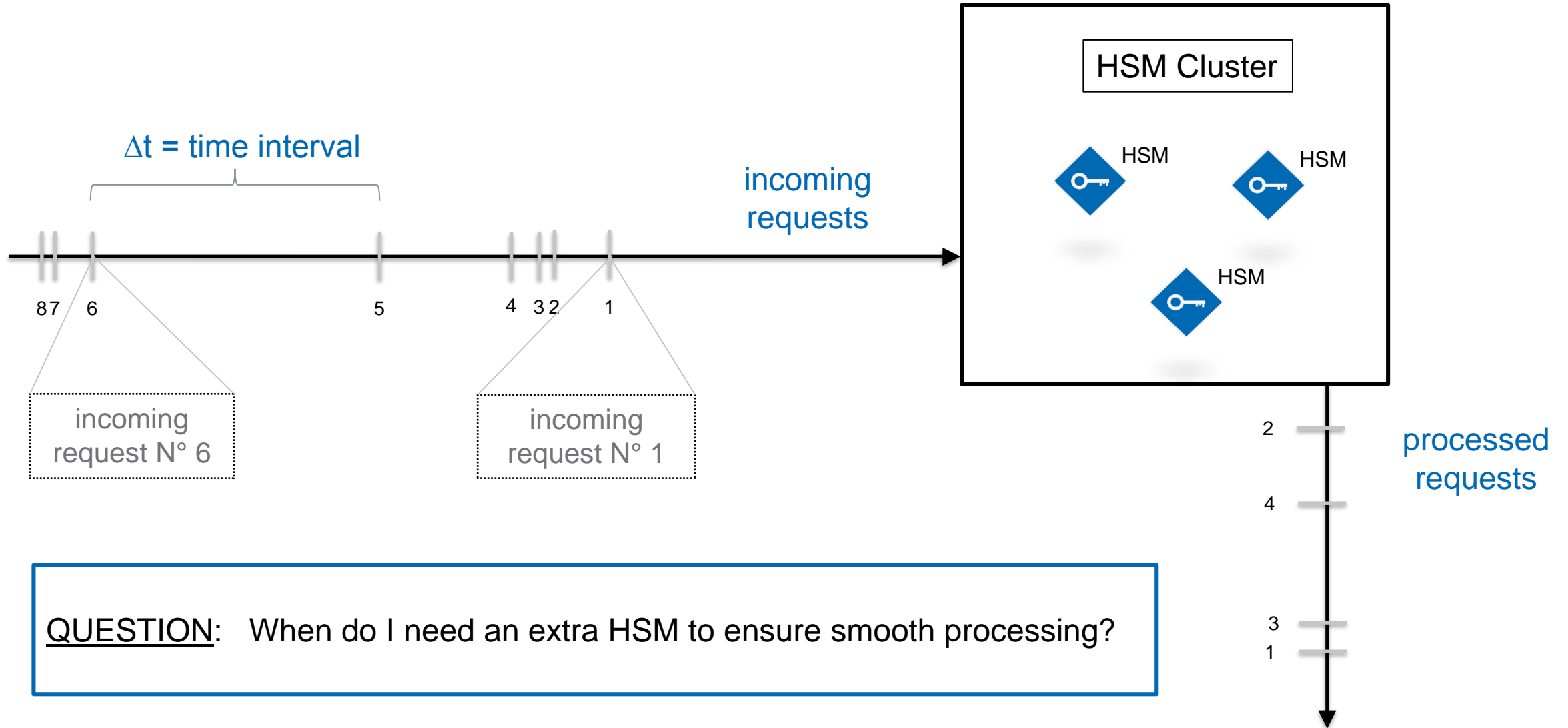
- [Where](#) do I find the example on the product CD ?

<Product CD>/Software/<Operating System>/<Platform>/Crypto_APIs/CXI/sample/cxi_example_LoadBalancing/

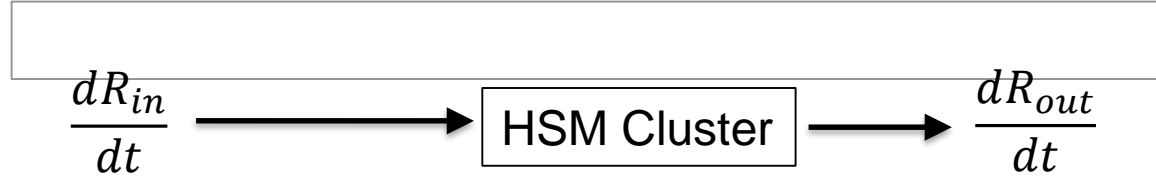
- The example [does](#)
 - show usage of Utimaco's CXI API and its cluster feature
 - serve as a “toy” example which you can use to play with, change, improve, adapt and extend
 - provide analysis ideas and could be used for very rough resource estimates
- The example [does NOT](#)
 - serve as a one-size-fits-all solution to a specific infrastructure
 - provide realistic estimates when used with Utimaco's HSM simulators
 - provide an optimized and universally applicable solution code

- 1. Essentials
- 2. Motivation – Use Case
- 3. Basic Concepts & Analysis
- 4. Details
 - Utimaco's HSM and Feature Characteristics
 - The Example
- 5. In Short





- 1. Essentials
- 2. Motivation – Use Case
- 3. Basic Concepts & Analysis
- 4. Details
 - Utimaco's HSM and Feature Characteristics
 - The Example
- 5. In Short



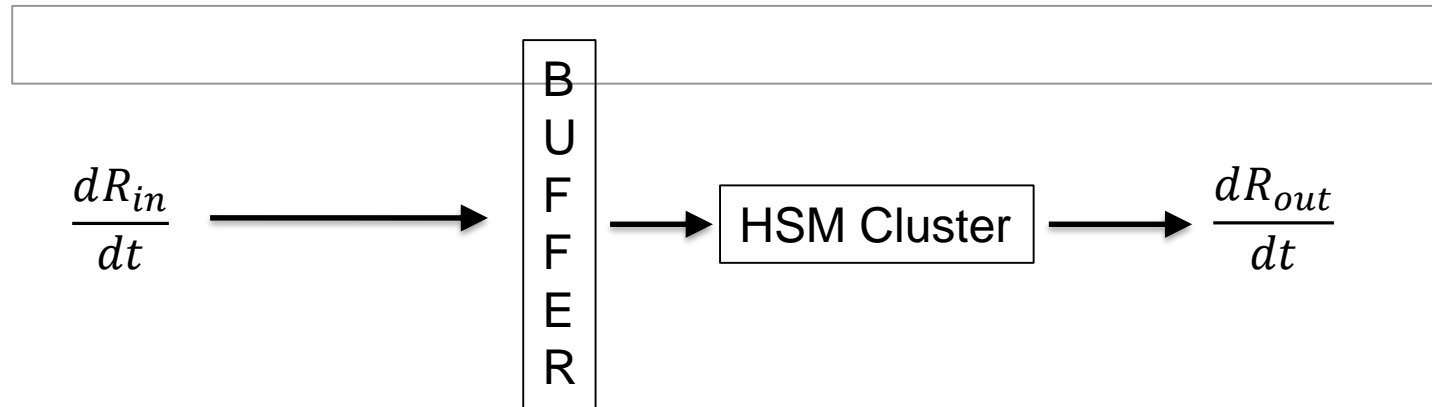
dR_{in}/dt = incoming request rate
 dR_{out}/dt = outgoing, processed request rate
 t = time

No traffic jam if : $\frac{dR_{in}}{dt} \leq \frac{dR_{out}}{dt}$

Difficulty: $\frac{dR_{in}}{dt} \neq \text{constant}$

incoming and processed request
rates are time-varying

time dependence of	determined by, caused by
incoming request rate	<ol style="list-style-type: none">1. the clients' frequency of requests2. infrastructure<ul style="list-style-type: none">- concurrency (programs / algorithms)- transmission times (network)- algorithm designs-
outgoing, processed request rate	<ol style="list-style-type: none">1. The clients' types of request2. HSM cluster<ul style="list-style-type: none">- cluster size- single HSM performance- algorithm designs- transmission times (for distant cluster members)-



buffer's purpose: „absorb“ peaks in incoming request rate,
temporary short term „traffic jams“ are allowed

no „clogging“: buffer size stays finite at very long times

QUESTION: when do I need an extra HSM to ensure smooth processing ?
ANSWER : on impending buffer overflow

Analyse your infrastructure, quantify your current and estimate your expected future needs

What ?	Quantified by (for example)								
<p>analyse the incoming request frequency</p> <p>for example: measure time interval (= waiting time Δt) between consecutive incoming requests over a long period of time.</p>	<p>waiting time distribution</p> <p>$P(\Delta t)$</p>								
<p>analyse the request type distribution</p> <p>for example: record over a long period of time the arriving types of requests</p>	<p>discrete request type distribution</p> <table> <tr> <th>request type</th><th>probability</th></tr> <tr> <td>1</td><td>p_1</td></tr> <tr> <td>..</td><td>...</td></tr> <tr> <td>n</td><td>p_n</td></tr> </table>	request type	probability	1	p_1	n	p_n
request type	probability								
1	p_1								
..	...								
n	p_n								

There are a lot of ways to get an estimate of when an extra HSM is needed

Here is one possible way:

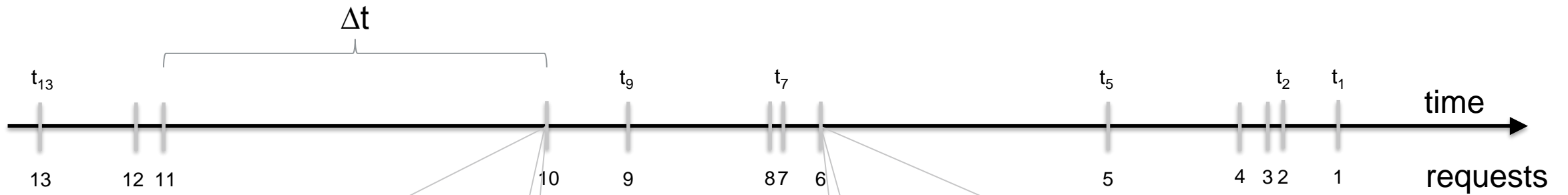
simulate your future needs on your current HSM cluster

The main purpose of this hands-on example is to demonstrate Utimaco's load-balancing and CXI-API usage. In addition it addresses the use-case question of when an extra HSM is needed.

Configure the load balancing example:

1. open etc/cxi.cfg: set your HSM cluster by specifying the IP addresses
2. open etc/lb.cfg: set your waiting time distribution
(For the sake of simplicity only extremely simple distributions are given.
[Please adapt the example code and implement the incoming request rate you need !!](#))
3. open etc/lb.cfg: set the request type distribution
(For the sake of simplicity only a sign and verify operation is implemented.
You can vary the number of times it is repeated to „emulate“ different request types.
[Please adapt the example code and implement the request types you need !!](#))

- 1. Essentials
- 2. Motivation – Use Case
- 3. Basic Concepts & Analysis
- **4. Details**
 - Utimaco's HSM and Feature Characteristics
 - The Example
- 5. In Short



Transaction Request N° 10	
operation	: „sign“
repeated	: N_{10} times
data	: hashed data
key	: key template
signature	: NULL

Transaction Request N° 6	
operation	: „verify“
repeated	: N_6 times
data	: hashed data
key	: key template
signature	: signature

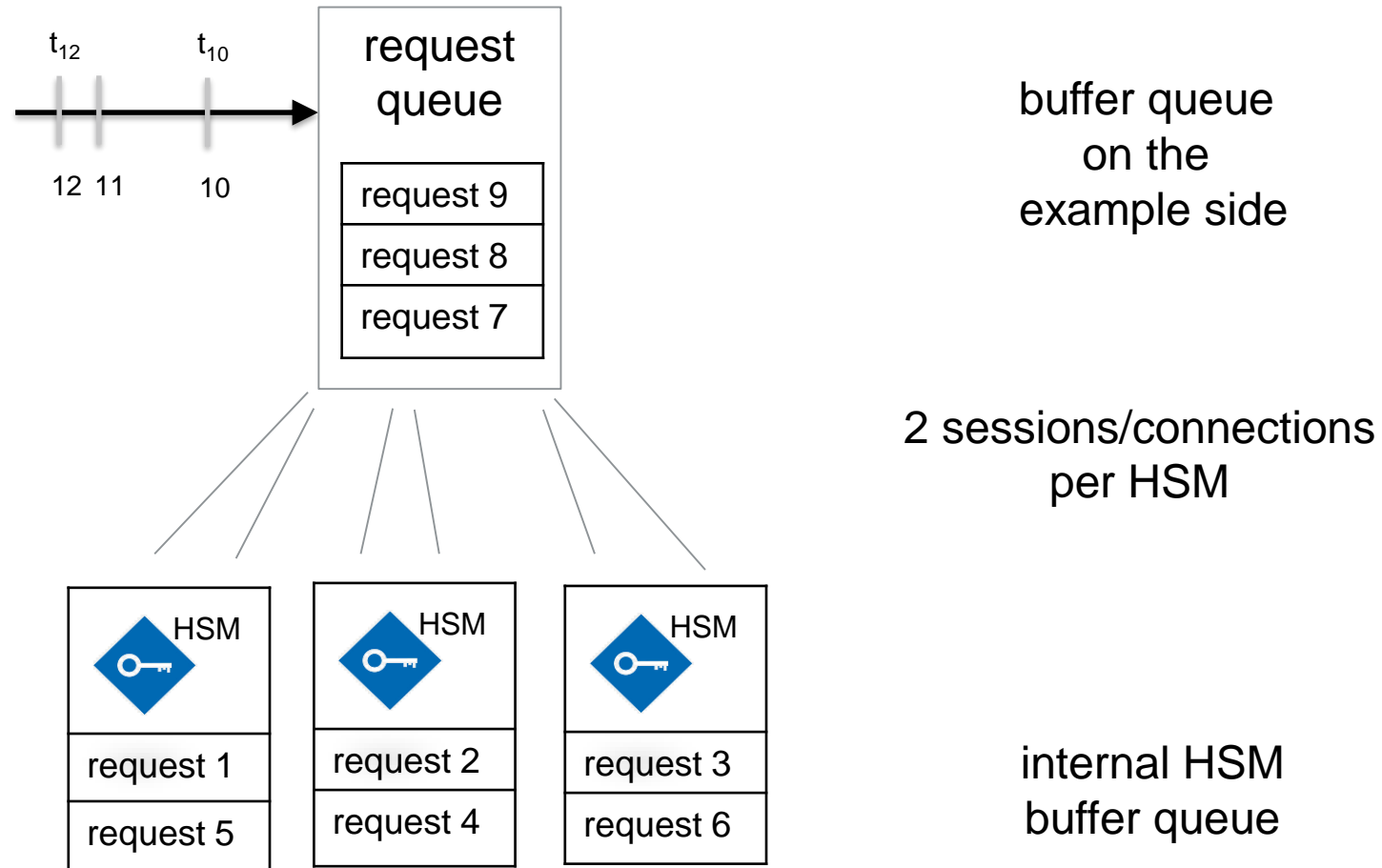
Characteristics

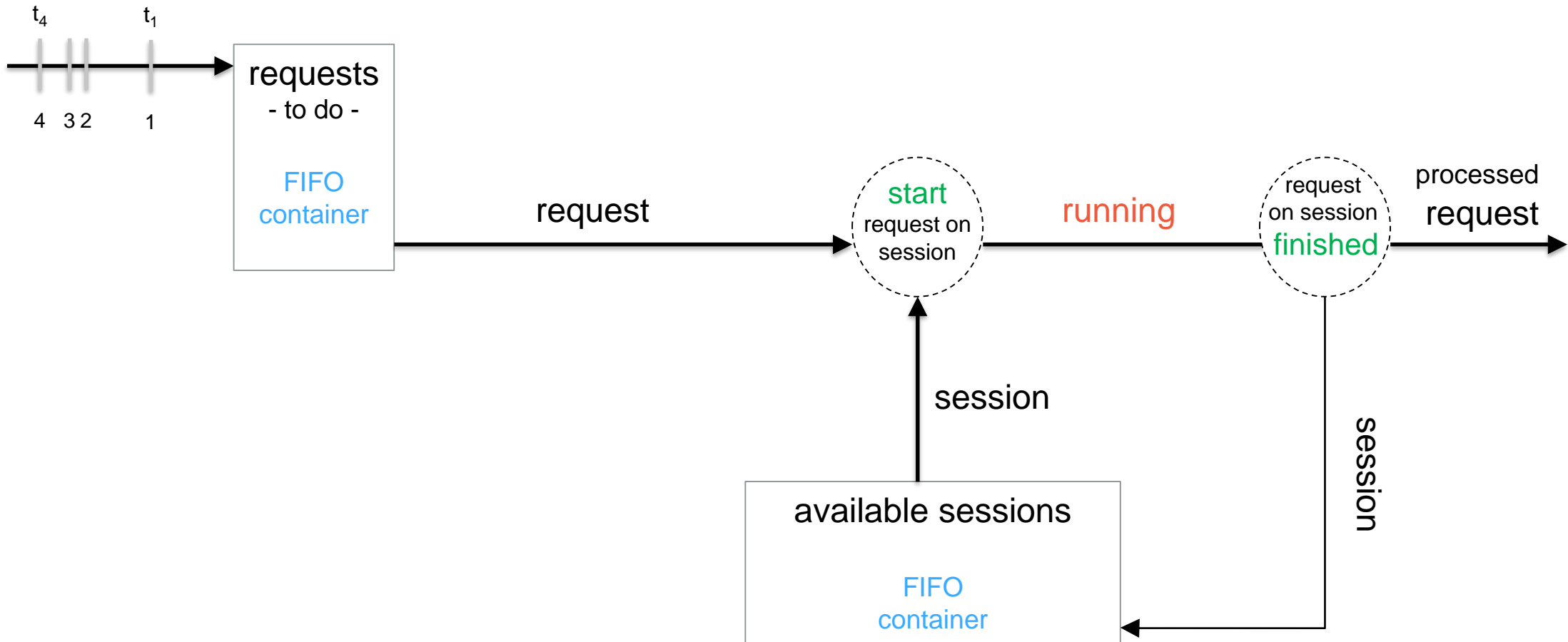
- Up to 4096 connections/secure messaging sessions can be opened per HSM
- BUT: Utimaco's HSM executes commands in order
- Therefore: increasing the number of sessions DOES NOT increase performance. It is just the question of where you want the requests to get buffered: inside or outside of the HSM.
- Only for distant CryptoServer LANs a few more sessions might increase performance (because of network transmission times)
- Therefore Utimaco recommends not much more than 2 or 3 sessions per HSM.
- An increase in performance is achieved by increasing the cluster size.

Utimaco's load balancing feature ensures **equally distributed connections** to HSMs inside a cluster:



- Frequent opening and closing of sessions introduces a large overhead
- Recommended usage :
Open the connection once at the beginning
with the flag „keep alive“





Output File	Content of File				
(1) data_requests_in.txt	incoming requests over time <table><tr><td><u>column 1</u></td><td><u>column 2</u></td></tr><tr><td>t</td><td>$R_{in}(t)$</td></tr></table>	<u>column 1</u>	<u>column 2</u>	t	$R_{in}(t)$
<u>column 1</u>	<u>column 2</u>				
t	$R_{in}(t)$				
(2) data_requests_out.txt	outgoing, processed requests over time <table><tr><td><u>column 1</u></td><td><u>column 2</u></td></tr><tr><td>t</td><td>$R_{out}(t)$</td></tr></table>	<u>column 1</u>	<u>column 2</u>	t	$R_{out}(t)$
<u>column 1</u>	<u>column 2</u>				
t	$R_{out}(t)$				
(3) data_requests_queueing.txt	requests in queue over time <table><tr><td><u>column 1</u></td><td><u>column 2</u></td></tr><tr><td>t</td><td>size of requests-to-do container</td></tr></table>	<u>column 1</u>	<u>column 2</u>	t	size of requests-to-do container
<u>column 1</u>	<u>column 2</u>				
t	size of requests-to-do container				

Parameter		Impact on
(1)	Number of requests	overall runtime
(2)	Number of sessions per HSM	length of HSM's waiting queue
(3)	Number of HSMs	
<div><div></div><div>(2) x (3) = max. number of sessions</div></div>		performance
(4)	Type of request	computation time in HSM
(5)	Request arrival frequency	filling rate of request container

- 1. Essentials
- 2. Motivation – Use Case
- 3. Basic Concepts & Analysis
- 4. Details
 - Utimaco's HSM and Feature Characteristics
 - The Example
- 5. In Short

EXAMPLE USE CASE:

At what load do I need an extra HSM ?

- (1) Estimate your future request frequency and future request type distribution
- (2) Take the example and modify it. Adjust it according to your system architecture. Use Utimaco's HSM simulators to validate your code's functionality.
- (3) Use a realistic setup with actual HSMs and network traffic to analyze performance.
 - Does the request-to-do container constantly increase in size and reach its predefined limit ?
Then buy another HSM.
 - Does the request-to-do container size stay bounded ?
Then your current setup will meet your estimated future needs.