MOP Framework For Actor Systems

Ian Cassar **Adrian Francalanza**

University of Malta

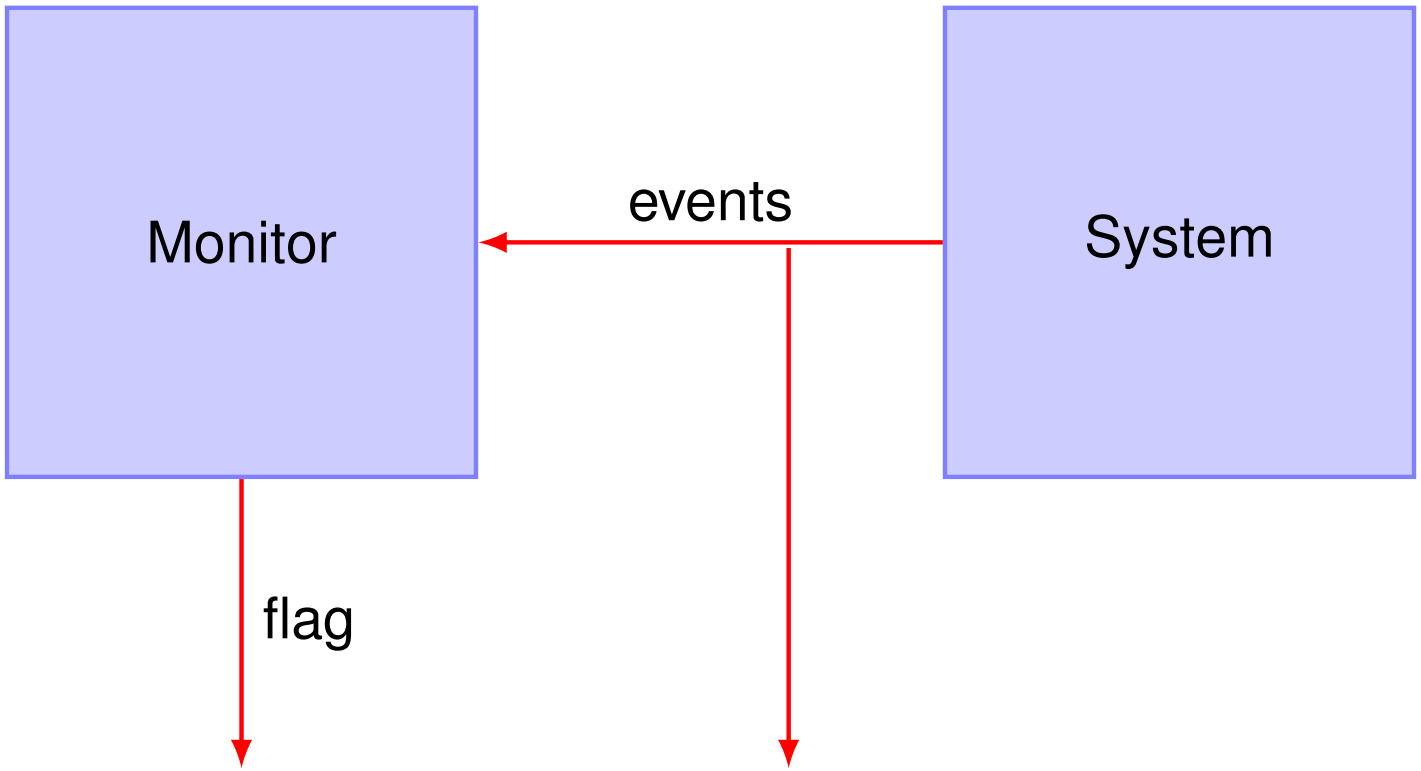
{*ian.cassar.10, adrian.francalanza*}*@um.edu.mt*

2 June 2016

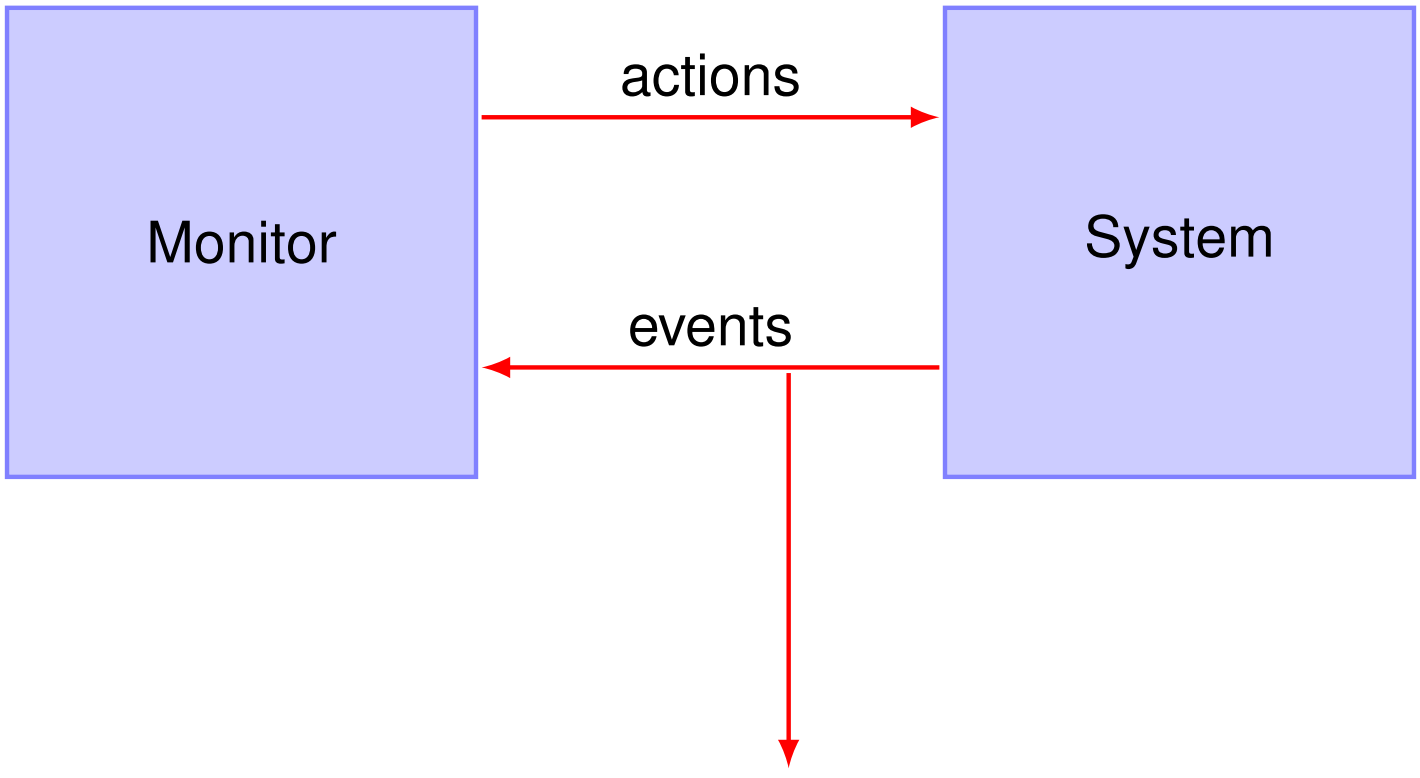
Monitor Oriented Programming (MOP)

- Incremetal (layered) architecture.
- Separation of concerns.
- Software customisation.
- Augmenting functionality.

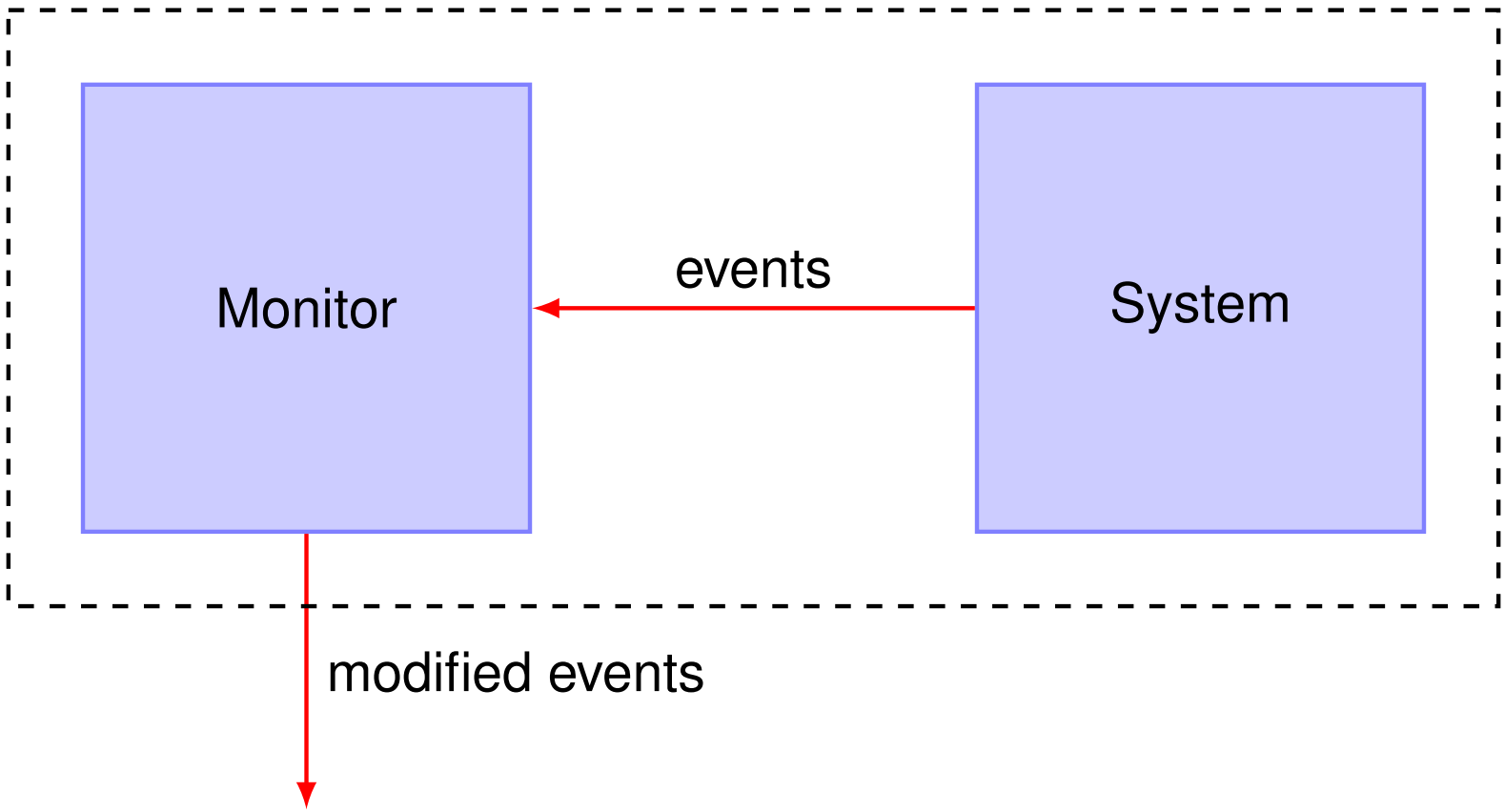
MOP: Verification, Adaptation and Enforcement



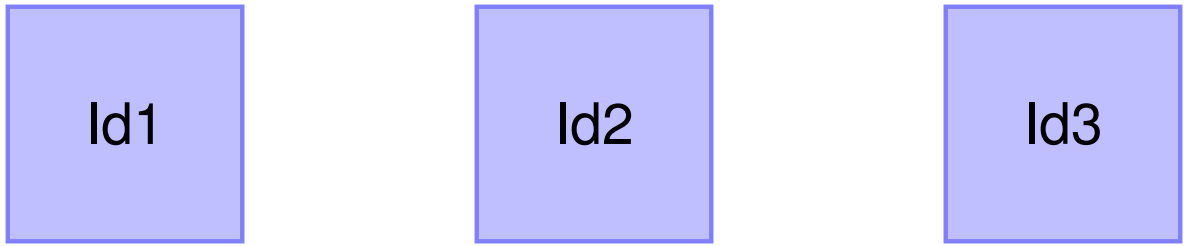
MOP: Verification, Adaptation and Enforcement



MOP: Verification, Adaptation and Enforcement

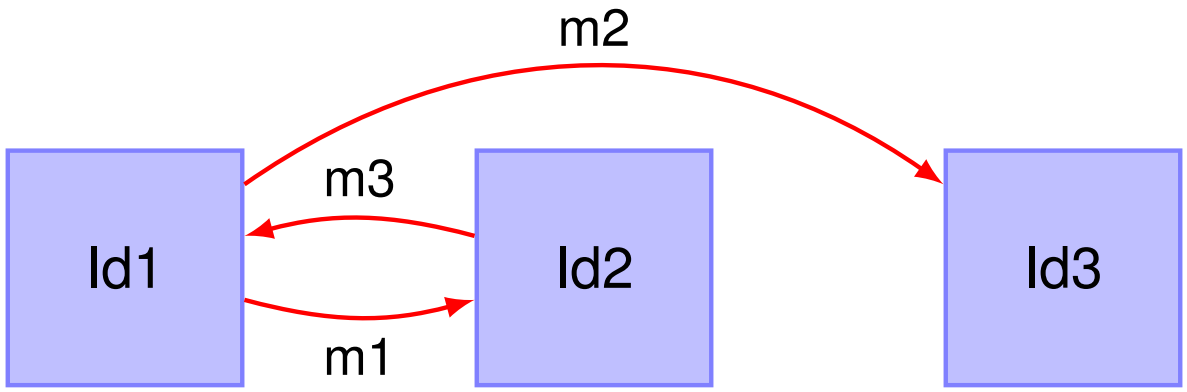


Actor Systems



Components

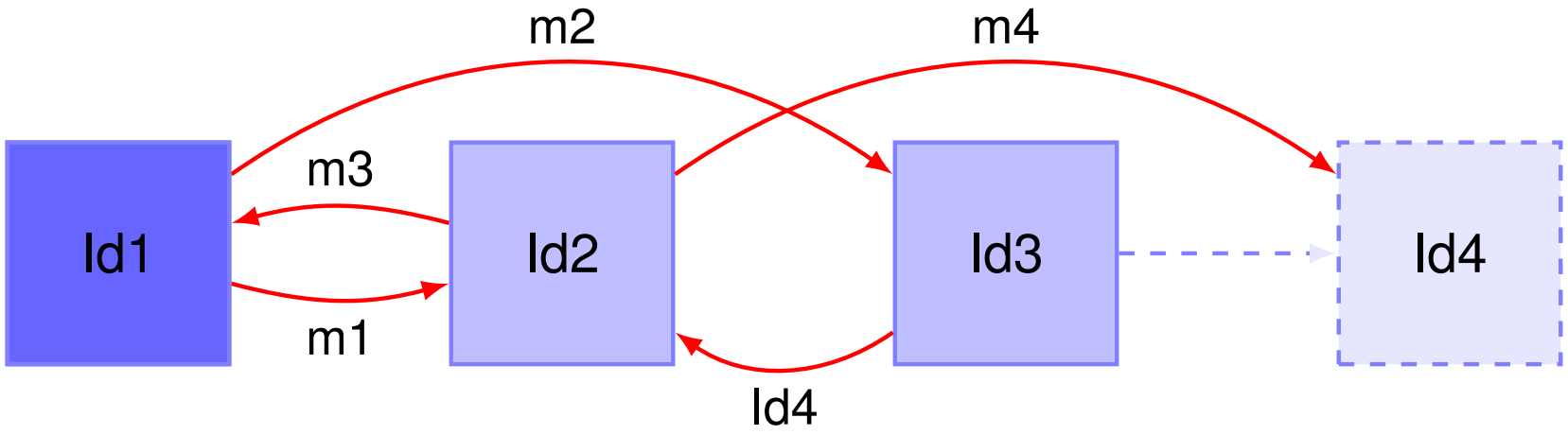
Actor Systems



Asynchronous Components

- Each component executes independently
- No shared data and non-blocking message passing.

Actor Systems



Asynchronous Dynamic Components

- Each component executes independently
- No shared data and non-blocking message passing.
- Spawn new processes
- Terminate independently

Technologies and Characteristics

Actor Languages/Frameworks

- Erlang
- Akka for Scala and Java
- Salsa (JVM)
- Stage (Python)

Technologies and Characteristics

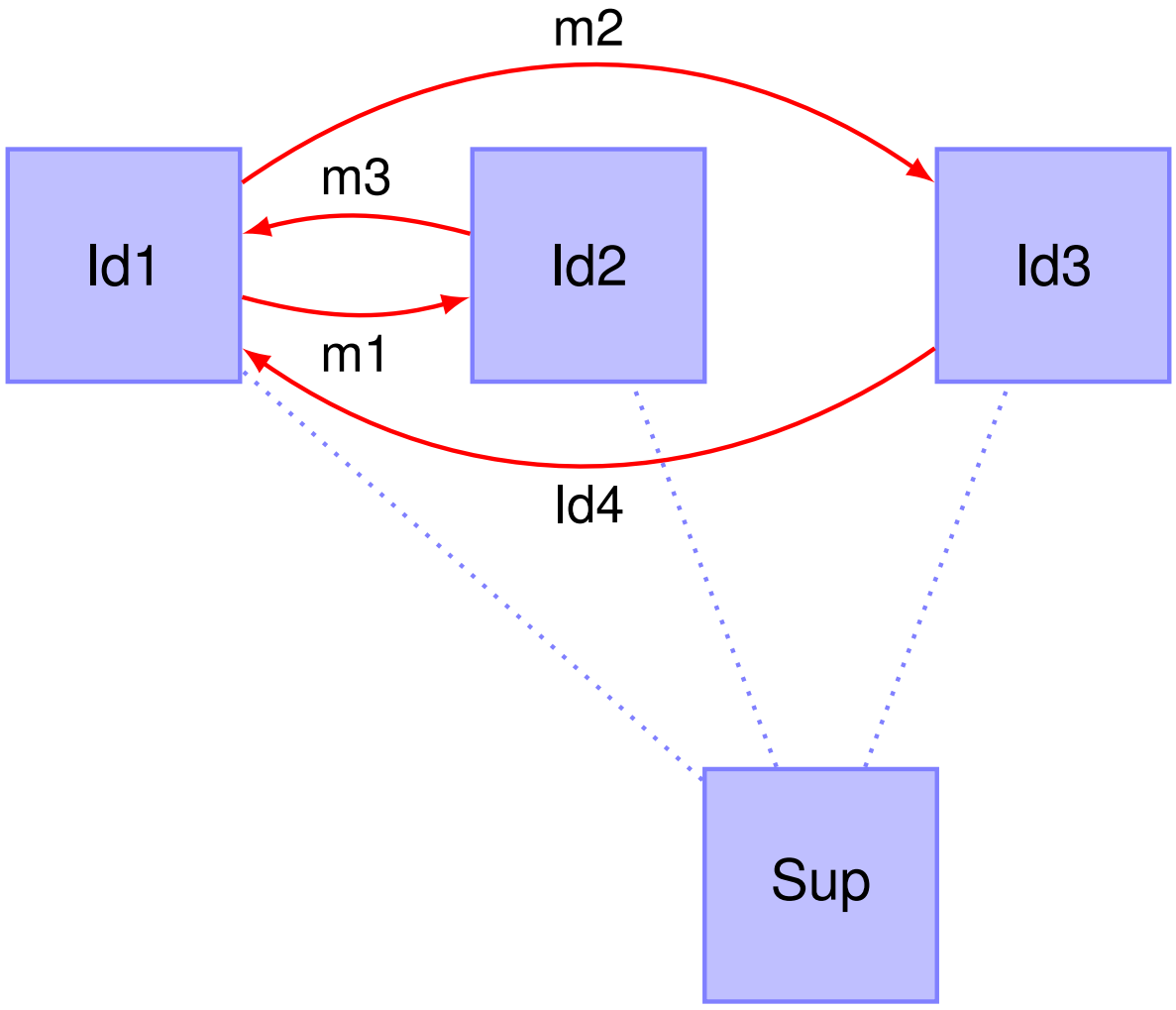
Actor Languages/Frameworks

- Erlang
- Akka for Scala and Java
- Salsa (JVM)
- Stage (Python)

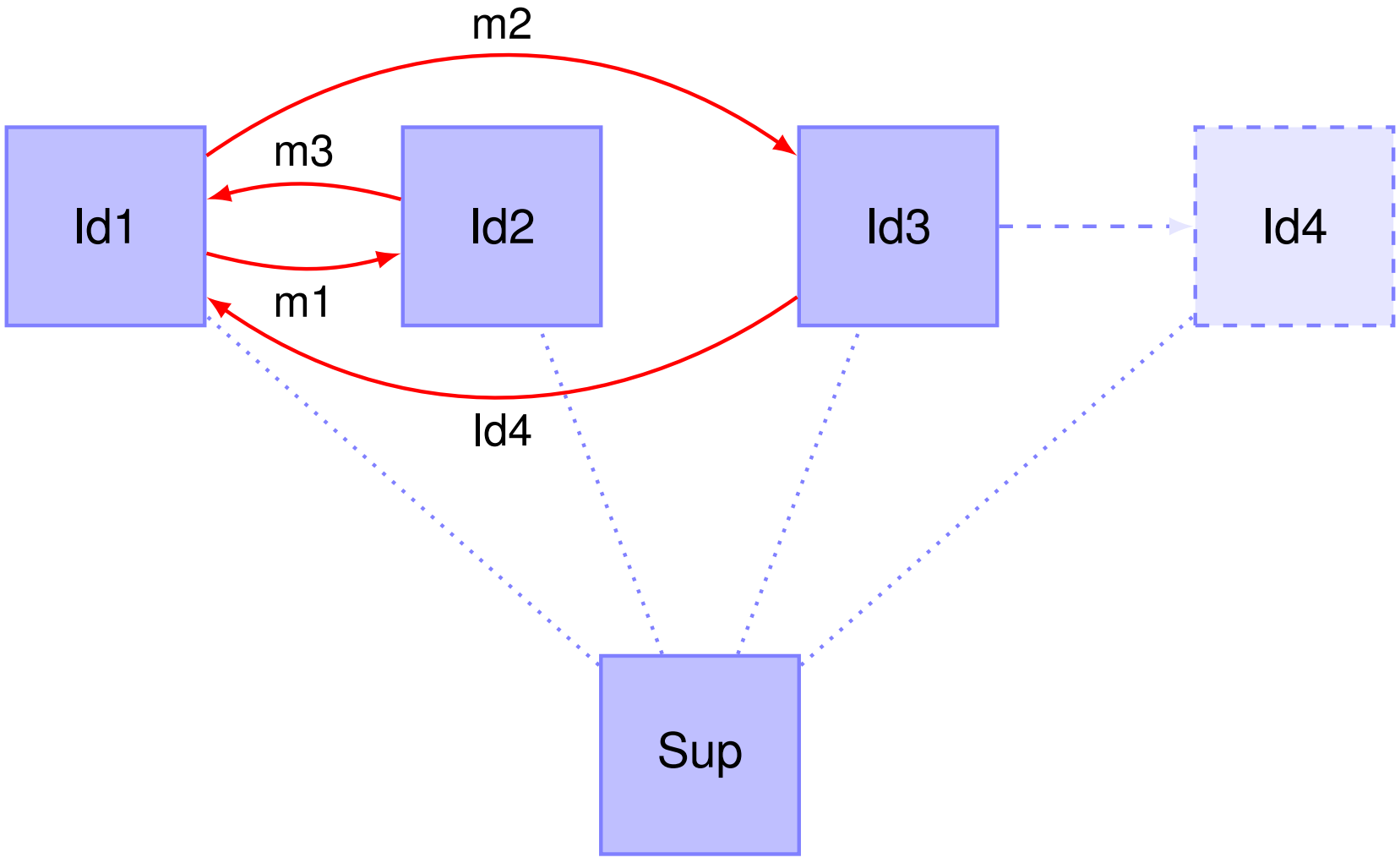
Characteristics

- Resilient
- Long-running
- Analysable
- Reconfigurable

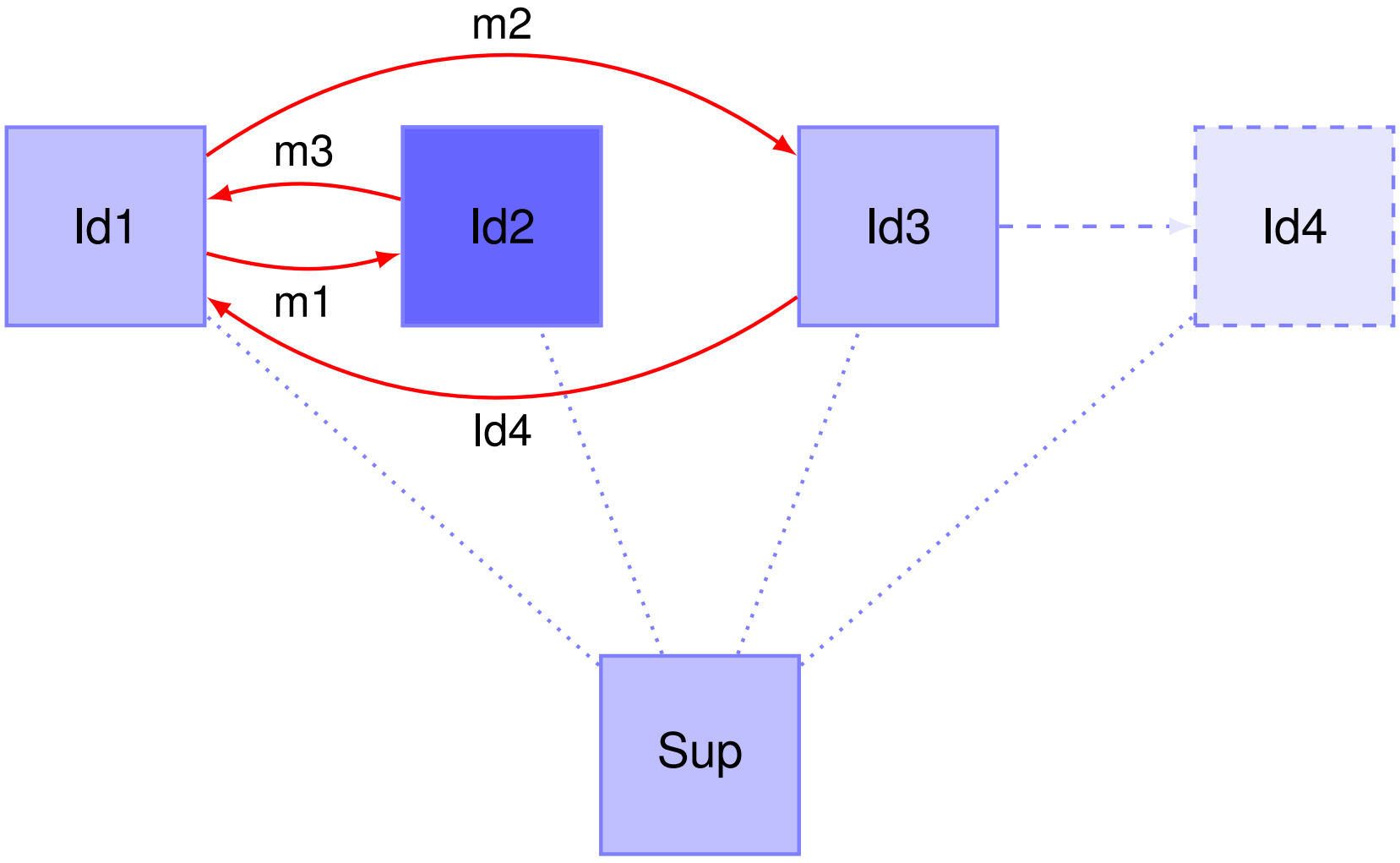
Runtime Adaptation in Actor Systems through Supervisors and Linking



Runtime Adaptation in Actor Systems through Supervisors and Linking



Runtime Adaptation in Actor Systems through Supervisors and Linking



Monitor-Oriented Programming

Adaptations

- Preemptive actions **before** something bad happens

Adaptations

- Preemptive actions **before** something bad happens
  - ✗ adapts *after* the bad event.

Adaptations

- Preemptive actions **before** something bad happens
  - ✗ adapts *after* the bad event.
- **Corrective** actions when something bad happens.

Adaptations

- Preemptive actions **before** something bad happens
  - ✗ adapts *after* the bad event.
- **Corrective** actions when something bad happens.
  - ✗ *Limited set* corrective actions
  - ✗ *Restrictive definition* of bad.

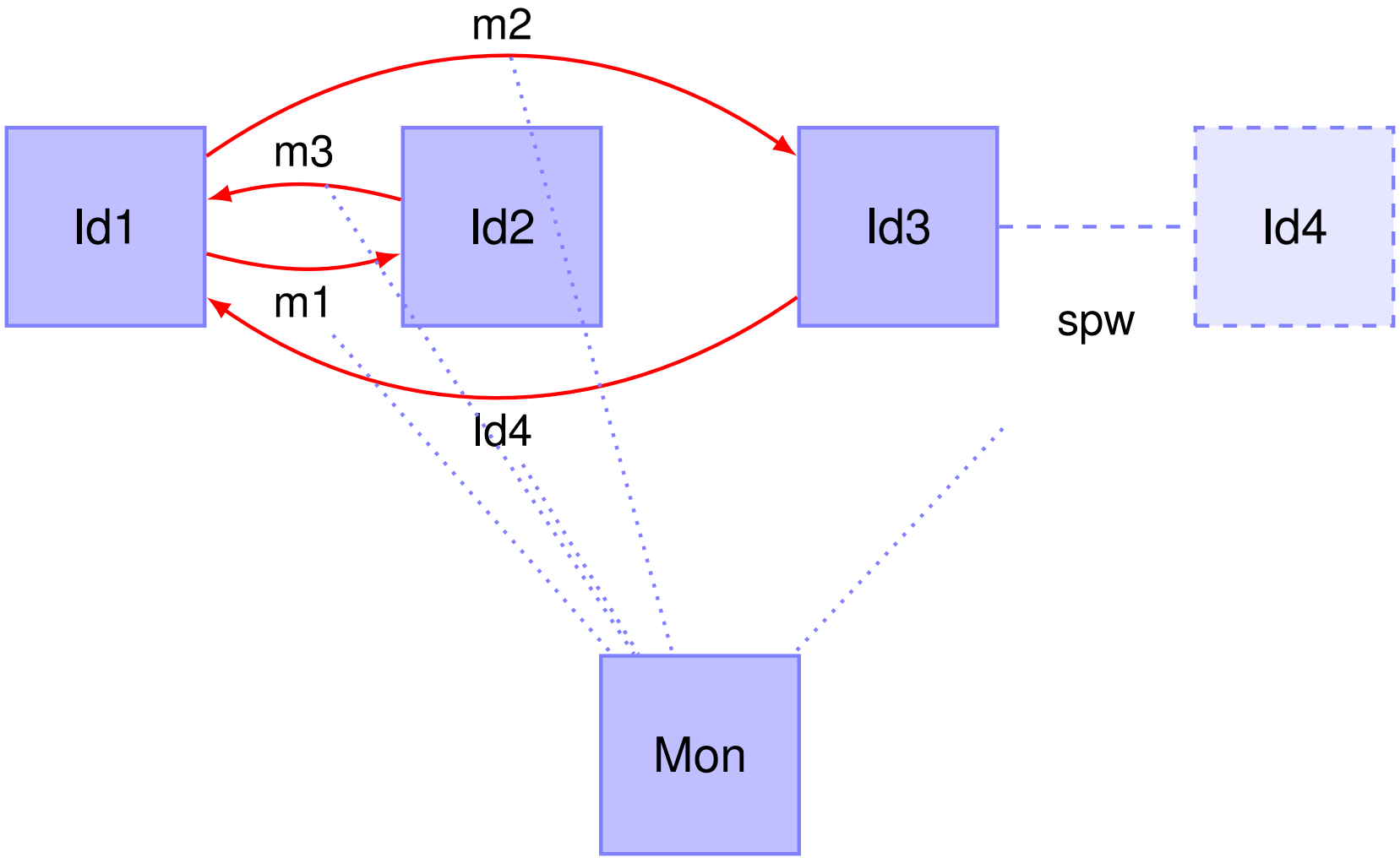
Adaptations

- Preemptive actions **before** something bad happens
  - ✗ adapts *after* the bad event.
- **Corrective** actions when something bad happens.
  - ✗ *Limited set* corrective actions
  - ✗ *Restrictive definition* of bad.
- Fine-tuning actions to **improve** resource usage.

Adaptations

- Preemptive actions **before** something bad happens
  - ✗ adapts *after* the bad event.
- **Corrective** actions when something bad happens.
  - ✗ *Limited set* corrective actions
  - ✗ *Restrictive definition* of bad.
- Fine-tuning actions to **improve** resource usage.
  - ✗ Has nothing to do with bad behaviour/failure.

General Monitors in Actor Systems



Aims and Challenges

1. **Specify** abstractly adaptation mechanisms
2. **Automate** (as much as possible) the monitor generation.
3. **Implement** efficiently the mechanism in a **minimally-intrusive** manner.

The Specification Language

$$
\begin{array}{lll}
c, d \in \mathtt{SPEC} ::= & \mathtt{flag} & \text{(detect)} \\
\mid & \mathtt{end} & \text{(terminate)} \\
\mid & c \,\&\, d & \text{(conjunction)} \\
\mid & \mathtt{if}\, b\, \mathtt{then}\, c\, \mathtt{else}\, d & \text{(branch)} \\
\mid & \mathtt{rec}\, X.c & \text{(recursion)} \\
\mid & X & \text{(recursive call)} \\
\mid & [p]\, \mathtt{rel}\, \vec{v}.\, c & \text{(guard)} \\
\mid & *[p]\, \mathtt{rel}\, \vec{v}.\, c & \text{(blocking guard)} \\
\mid & A(x)\, \mathtt{rel}\, \vec{v}.\, c & \text{(asyn. adaptation)} \\
\mid & S(x)\, \mathtt{rel}\, \vec{v}.\, c & \text{(sync. adaptation)}
\end{array}
$$

Specifying Adaptations

Example

$$\texttt{rec}\, X.[i{>}x\,!\,y].\,[j{>}x\,!\,z].\,\texttt{if}\, y{==}z\ \texttt{then}\ X\ \texttt{else}\ \texttt{flag}$$

Specifying Adaptations

Example

$$\texttt{rec}\, X\,.\,[i\!>\!x\,!\,y]\,.\,[j\!>\!x\,!\,z]\,.\,\texttt{if}\, y{=}{=}z\ \texttt{then}\ X\ \texttt{else}\ \texttt{flag}$$

$$\begin{aligned}&\texttt{rec}\, X\,.\,{*}[i\!>\!x\,!\,y]\ \texttt{rel}\ []\,.\,{*}[j\!>\!x\,!\,z]\ \texttt{rel}\ [i]\,.\\ &\qquad\texttt{if}\, y{=}{=}z\ \texttt{then}\ X\ \texttt{else}\\ &\qquad\ \mathit{kill}(x)\ \texttt{rel}\ []\,.\,\mathit{restart}(i)\ \texttt{rel}\ []\,.\,\mathit{restart}(j)\ \texttt{rel}\ [i,j]\,.\,X\end{aligned}$$

Specifying Adaptations

Example

$$\texttt{rec}\, X.[i{>}x\,!\,y].\,[j{>}x\,!\,z].\,\texttt{if}\, y{==}z\ \texttt{then}\ X\ \texttt{else}\ \texttt{flag}$$

$$\begin{aligned}&\texttt{rec}\, X.{*}[i{>}x\,!\,y]\,\texttt{rel}\,[].\,{*}[j{>}x\,!\,z]\,\texttt{rel}\,[i].\\ &\qquad \texttt{if}\, y{==}z\ \texttt{then}\ X\ \texttt{else}\\ &\qquad\ \mathit{kill}(x)\,\texttt{rel}\,[].\,\mathit{restart}(i)\,\texttt{rel}\,[].\,\mathit{restart}(j)\,\texttt{rel}\,[i,j].\,X\end{aligned}$$

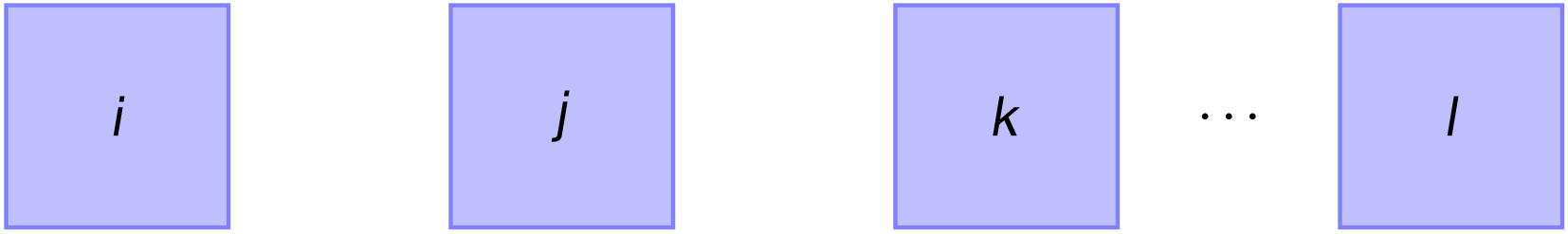
Specifying Adaptations

Example

$$\texttt{rec}\, X\,.\,[i{>}x\,!\,y]\,.\,[j{>}x\,!\,z]\,.\,\texttt{if}\, y{==}z\ \texttt{then}\ X\ \texttt{else}\ \texttt{flag}$$

$$\begin{aligned}&\texttt{rec}\, X\,.\,{*}[i{>}x\,!\,y]\ \texttt{rel}\ []\,.\,{*}[j{>}x\,!\,z]\ \texttt{rel}\ [i]\,.\\ &\qquad \texttt{if}\, y{==}z\ \texttt{then}\ X\ \texttt{else}\\ &\qquad\ \mathit{kill}(x)\ \texttt{rel}\ []\,.\,\mathit{restart}(i)\ \texttt{rel}\ []\,.\,\mathit{restart}(j)\ \texttt{rel}\ [i,j]\,.\,X\end{aligned}$$

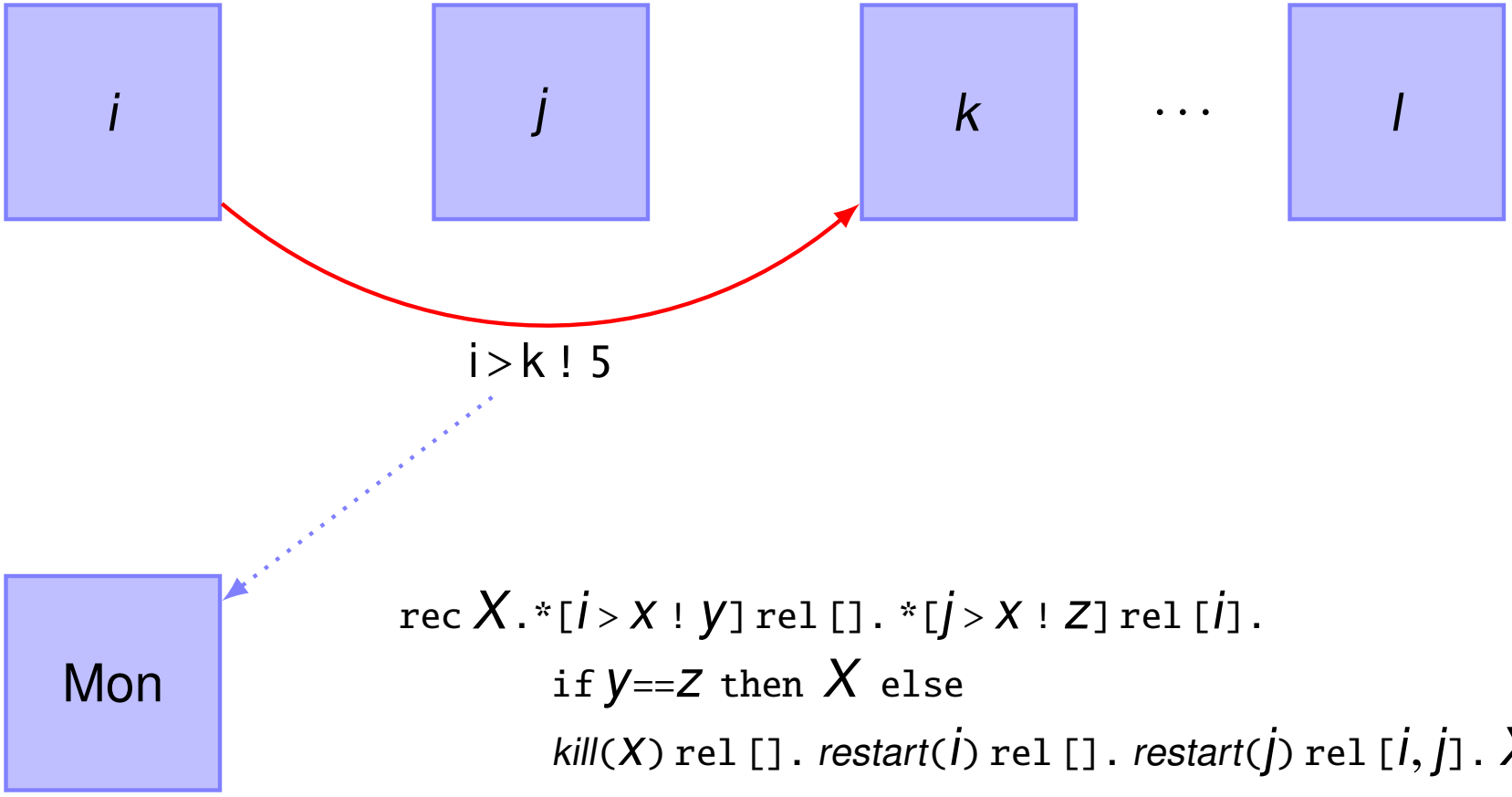
Adaptation Monitors in Actor Systems



Mon

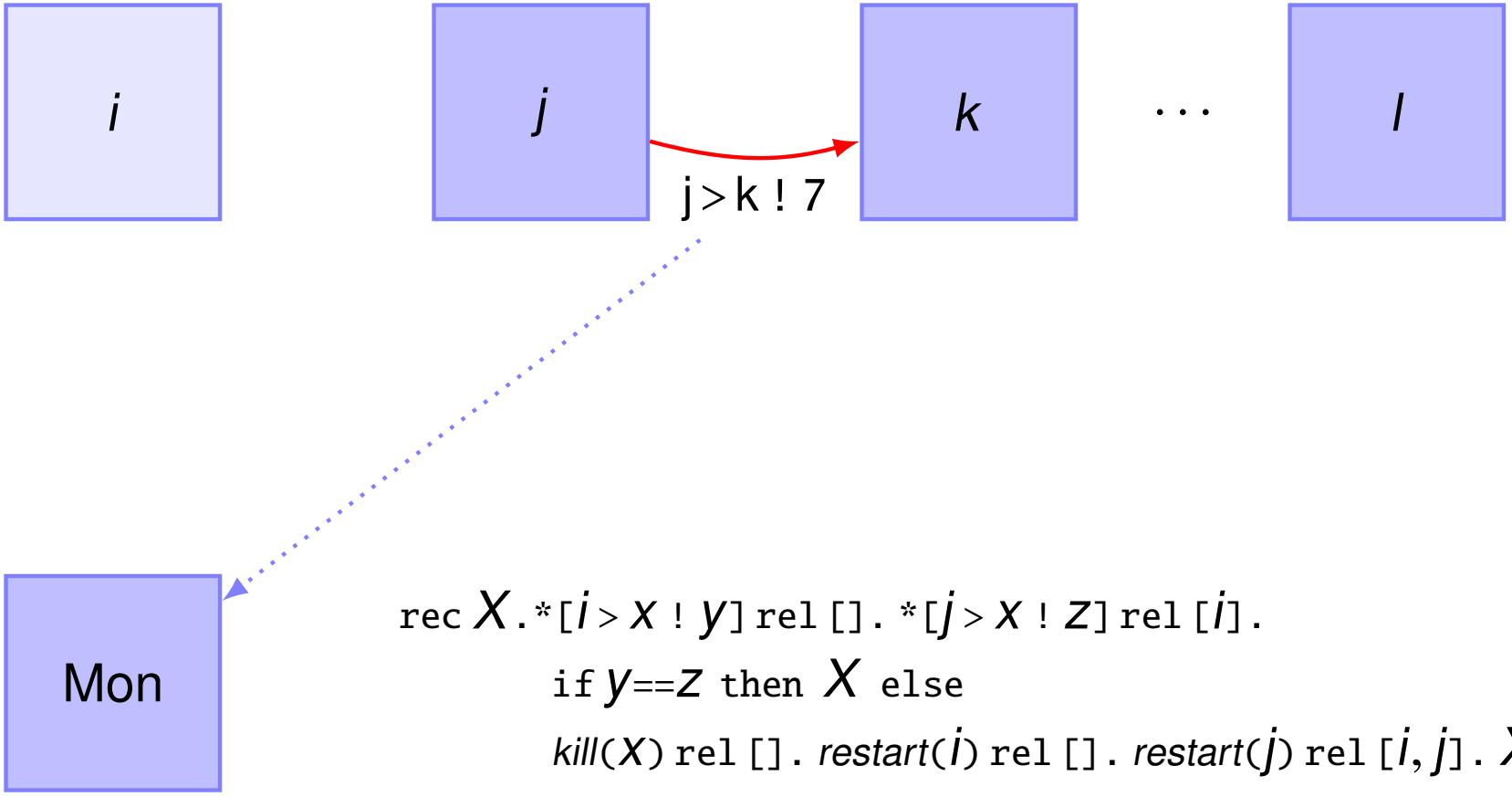
$$\texttt{rec}\ X.\texttt{*}[i > x\ \texttt{!}\ y]\ \texttt{rel}\ []\,.\ \texttt{*}[j > x\ \texttt{!}\ z]\ \texttt{rel}\ [i]\,.$$
$$\texttt{if}\ y == z\ \texttt{then}\ X\ \texttt{else}$$
$$\mathit{kill}(x)\ \texttt{rel}\ []\,.\ \mathit{restart}(i)\ \texttt{rel}\ []\,.\ \mathit{restart}(j)\ \texttt{rel}\ [i, j]\,.\ X$$

Adaptation Monitors in Actor Systems



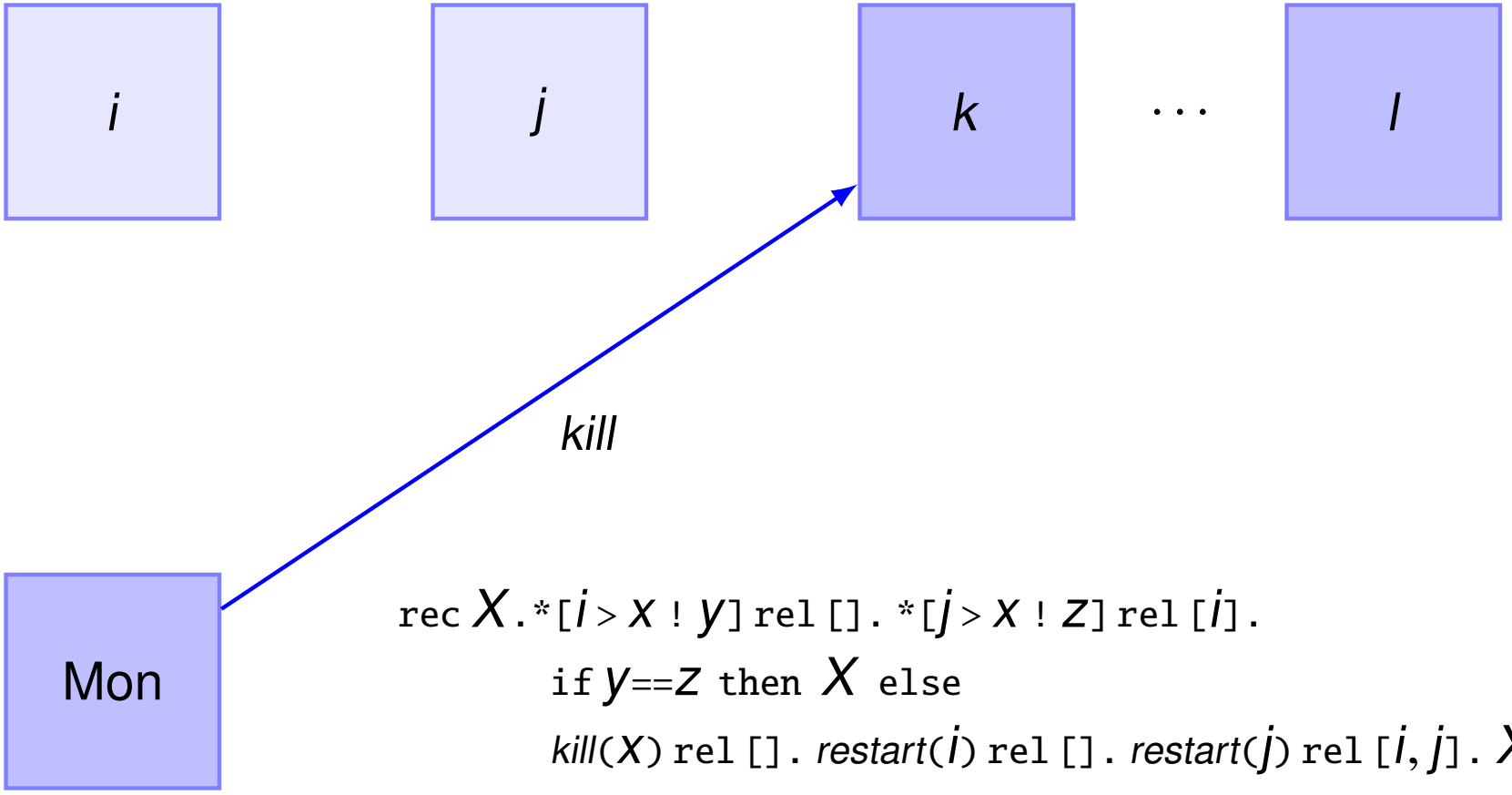
$$\texttt{rec}\ X.\ \texttt{*}[i > x\ \texttt{!}\ y]\ \texttt{rel}\ []\ .\ \texttt{*}[j > x\ \texttt{!}\ z]\ \texttt{rel}\ [i]\ .$$
$$\texttt{if}\ y == z\ \texttt{then}\ X\ \texttt{else}$$
$$\mathit{kill}(x)\ \texttt{rel}\ []\ .\ \mathit{restart}(i)\ \texttt{rel}\ []\ .\ \mathit{restart}(j)\ \texttt{rel}\ [i, j]\ .\ X$$

Adaptation Monitors in Actor Systems



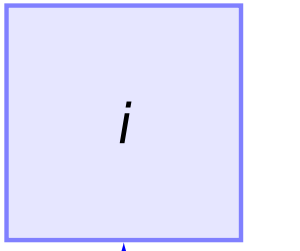
$$\texttt{rec}\ X.\ \texttt{*[}i > x\ \texttt{!}\ y\texttt{]}\ \texttt{rel []}.\ \texttt{*[}j > x\ \texttt{!}\ z\texttt{]}\ \texttt{rel [}i\texttt{]}.$$
$$\texttt{if}\ y == z\ \texttt{then}\ X\ \texttt{else}$$
$$\mathit{kill}(x)\ \texttt{rel []}.\ \mathit{restart}(i)\ \texttt{rel []}.\ \mathit{restart}(j)\ \texttt{rel [}i, j\texttt{]}.\ X$$

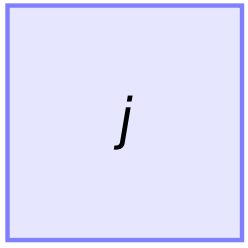
Adaptation Monitors in Actor Systems

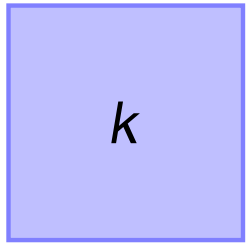


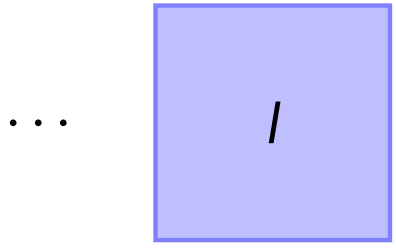
$$\texttt{rec}\ X.\texttt{*[}i\texttt{>}x\ \texttt{!}\ y\texttt{]}\ \texttt{rel}\ \texttt{[]}.\ \texttt{*[}j\texttt{>}x\ \texttt{!}\ z\texttt{]}\ \texttt{rel}\ \texttt{[}i\texttt{]}.$$
$$\texttt{if}\ y\texttt{==}z\ \texttt{then}\ X\ \texttt{else}$$
$$\mathit{kill}(x)\ \texttt{rel}\ \texttt{[]}.\ \mathit{restart}(i)\ \texttt{rel}\ \texttt{[]}.\ \mathit{restart}(j)\ \texttt{rel}\ \texttt{[}i,j\texttt{]}.\ X$$

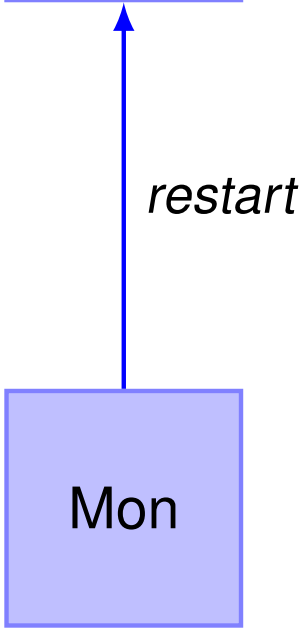
Adaptation Monitors in Actor Systems





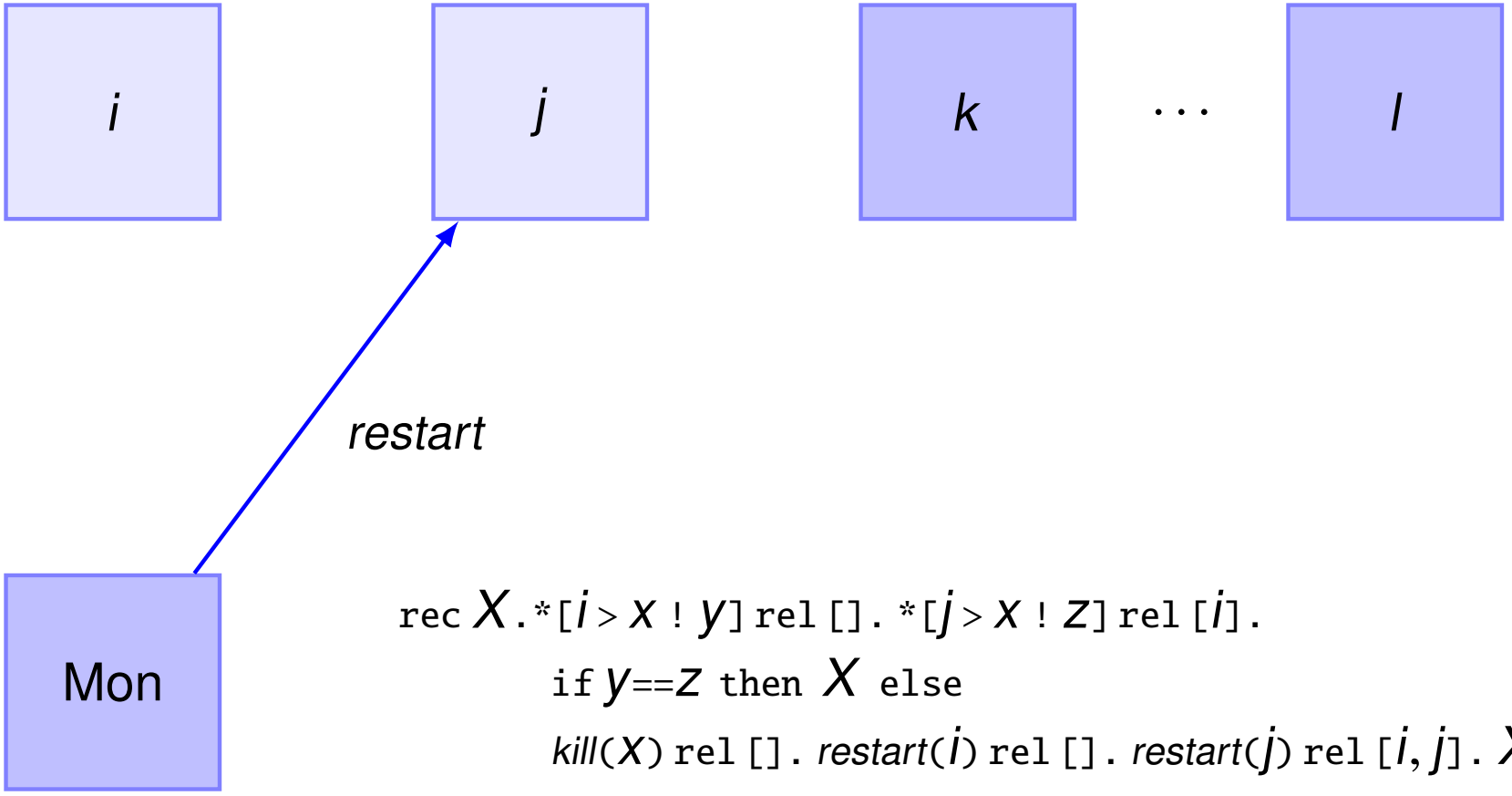






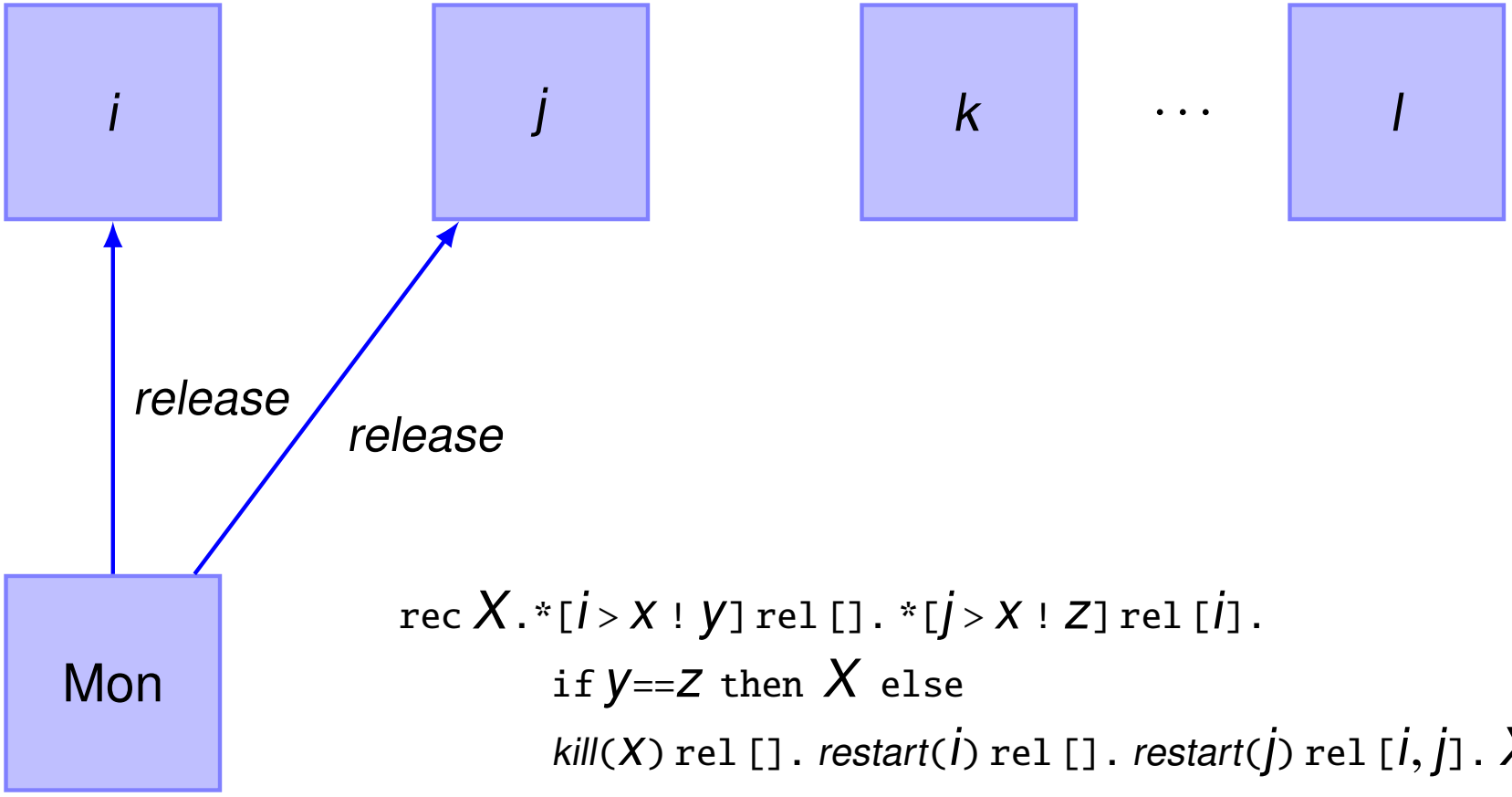
$$\texttt{rec}\, X.\texttt{*[}i > x\texttt{ ! }y\texttt{]}\,\texttt{rel}\,\texttt{[]}.\,\texttt{*[}j > x\texttt{ ! }z\texttt{]}\,\texttt{rel}\,\texttt{[}i\texttt{]}.$$
$$\texttt{if}\, y == z\, \texttt{then}\, X\, \texttt{else}$$
$$\mathit{kill}(x)\,\texttt{rel}\,\texttt{[]}.\,\mathit{restart}(i)\,\texttt{rel}\,\texttt{[]}.\,\mathit{restart}(j)\,\texttt{rel}\,\texttt{[}i, j\texttt{]}.\, X$$

Adaptation Monitors in Actor Systems



$$\texttt{rec}\ X.\texttt{*[}i\texttt{>}x\ \texttt{!}\ y\texttt{]}\ \texttt{rel}\ \texttt{[]}.\ \texttt{*[}j\texttt{>}x\ \texttt{!}\ z\texttt{]}\ \texttt{rel}\ \texttt{[}i\texttt{]}.$$
$$\texttt{if}\ y\texttt{==}z\ \texttt{then}\ X\ \texttt{else}$$
$$\mathit{kill}(x)\ \texttt{rel}\ \texttt{[]}.\ \mathit{restart}(i)\ \texttt{rel}\ \texttt{[]}.\ \mathit{restart}(j)\ \texttt{rel}\ \texttt{[}i,j\texttt{]}.\ X$$

Adaptation Monitors in Actor Systems

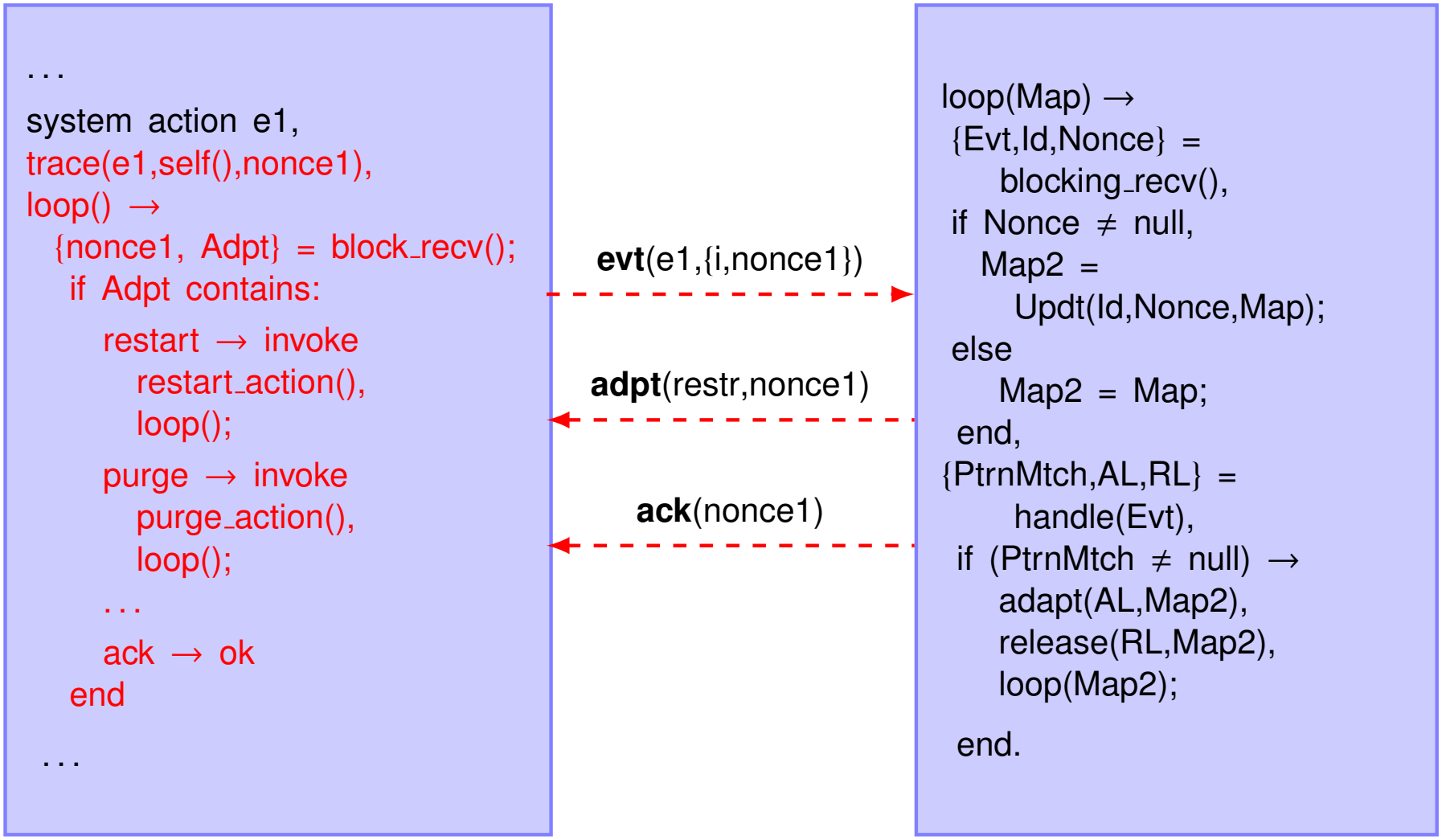


$$\texttt{rec}\ X.\texttt{*[}i\texttt{>}x\ \texttt{!}\ y\texttt{]}\ \texttt{rel}\ \texttt{[]}.\ \texttt{*[}j\texttt{>}x\ \texttt{!}\ z\texttt{]}\ \texttt{rel}\ \texttt{[}i\texttt{]}.$$
$$\texttt{if}\ y\texttt{==}z\ \texttt{then}\ X\ \texttt{else}$$
$$\mathit{kill}(x)\ \texttt{rel}\ \texttt{[]}.\ \mathit{restart}(i)\ \texttt{rel}\ \texttt{[]}.\ \mathit{restart}(j)\ \texttt{rel}\ \texttt{[}i,j\texttt{]}.\ X$$

Implementation Obstacles

	Actors	Adaptation
Execution	Asynchronous	Synchronous
Visibility/Accessibility	Encapsulated	Open

Actor Instrumention Protocol



Asynchronous Adaptations

- Actor killing via `exit()`
- Actor name (de)registering via `register()` and `deregister()`
- Memory optimisation via `garbage_collect()`
- Exit trapping via `process_flag()`
- Composite adaptations such as killing all actors linked to an actor using `process_info()`, `process_flag()` and `exit()`.

Synchronous Adaptations

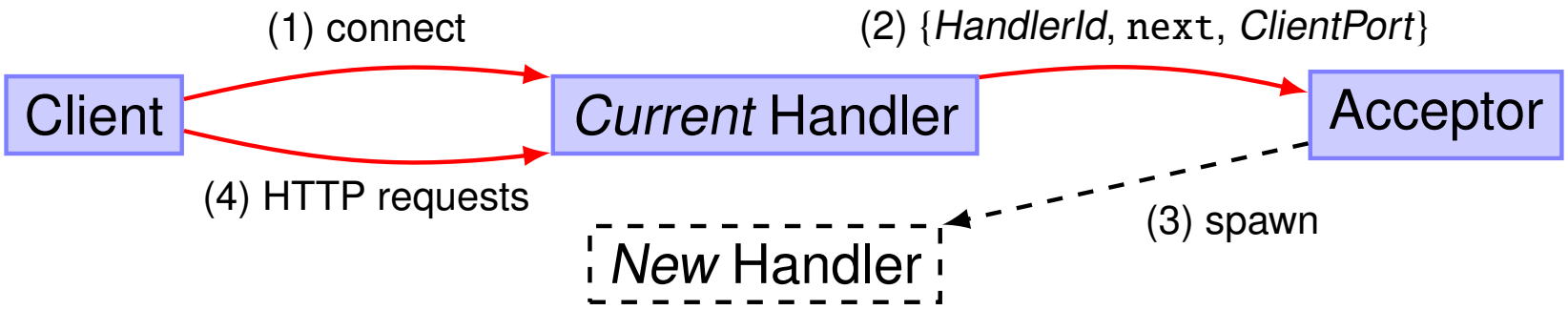
`purge()`: Empties an actor's mailbox.

`silent_kill()`: Composite adaptation made up of smaller adaptations.

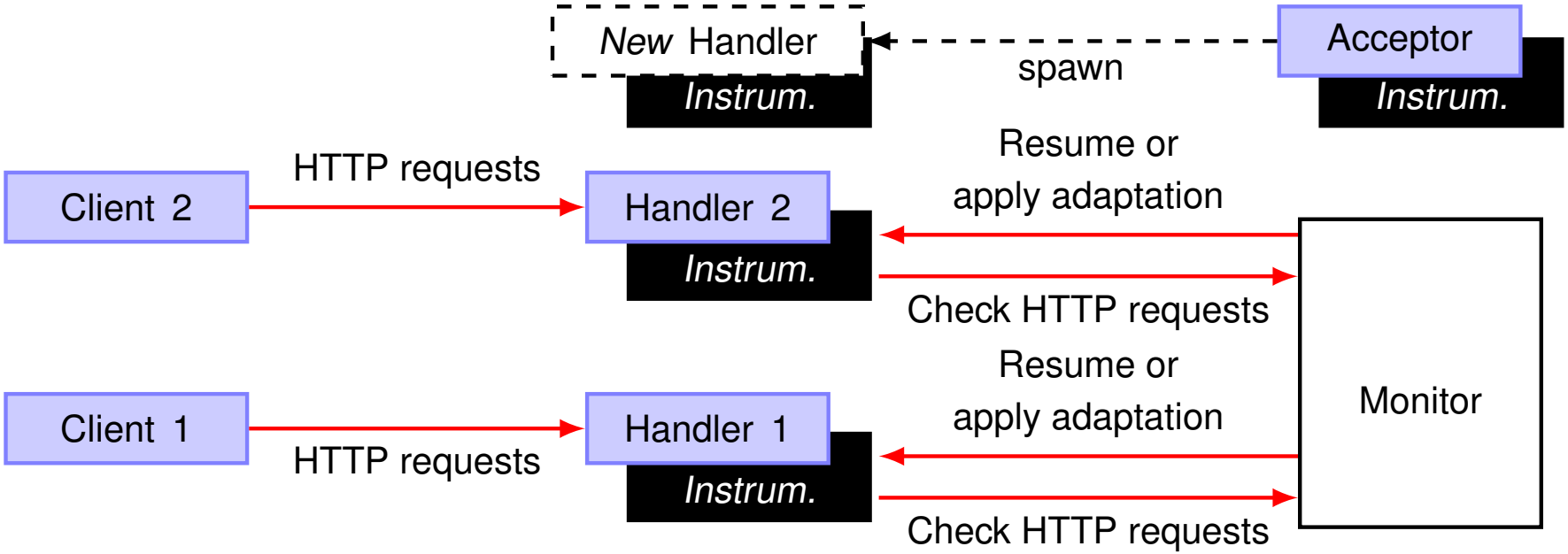
`restart()`: Restarts the execution of an actor with a reinitialised memory.

`untrace()`: Makes events of the actor unmonitorable.

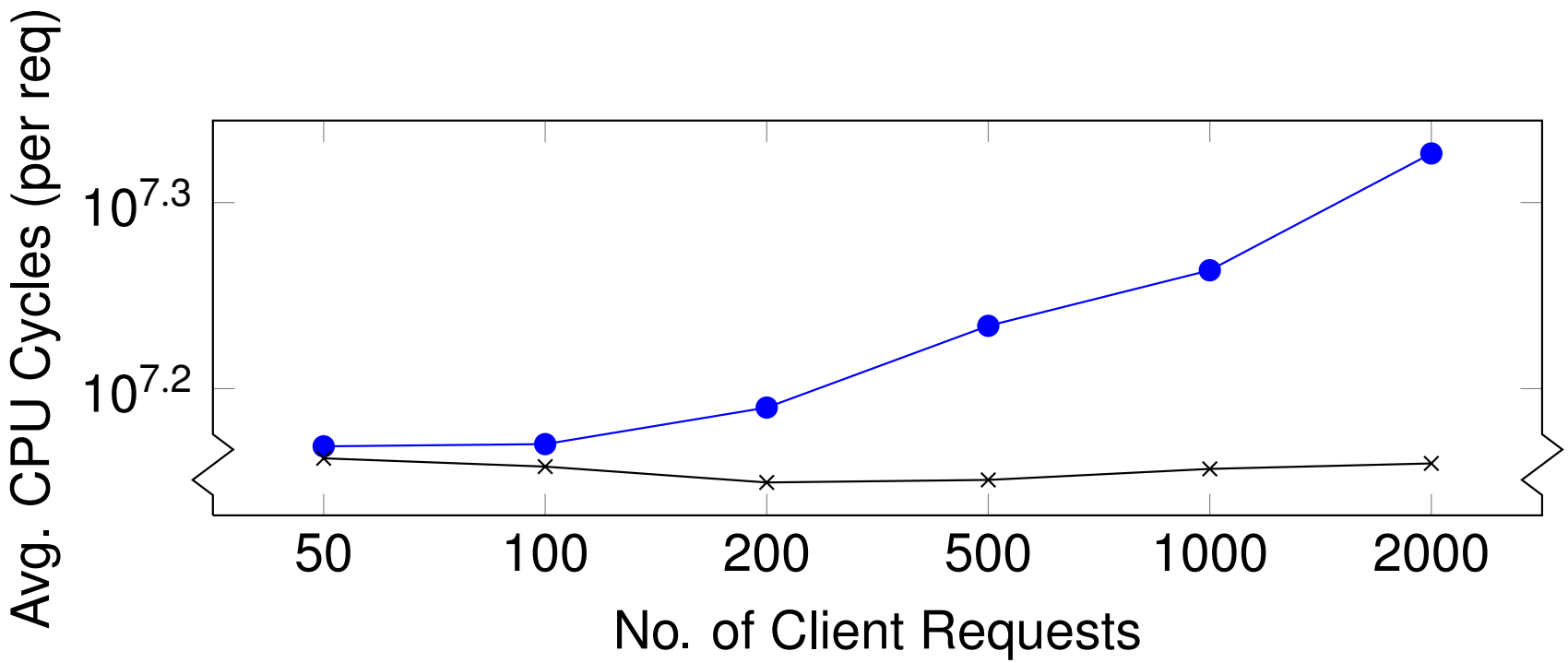
Case Study


(1) connect
(2) {*HandlerId*, `next`, *ClientPort*}
Client
Current Handler
Acceptor
(4) HTTP requests
(3) spawn
New Handler

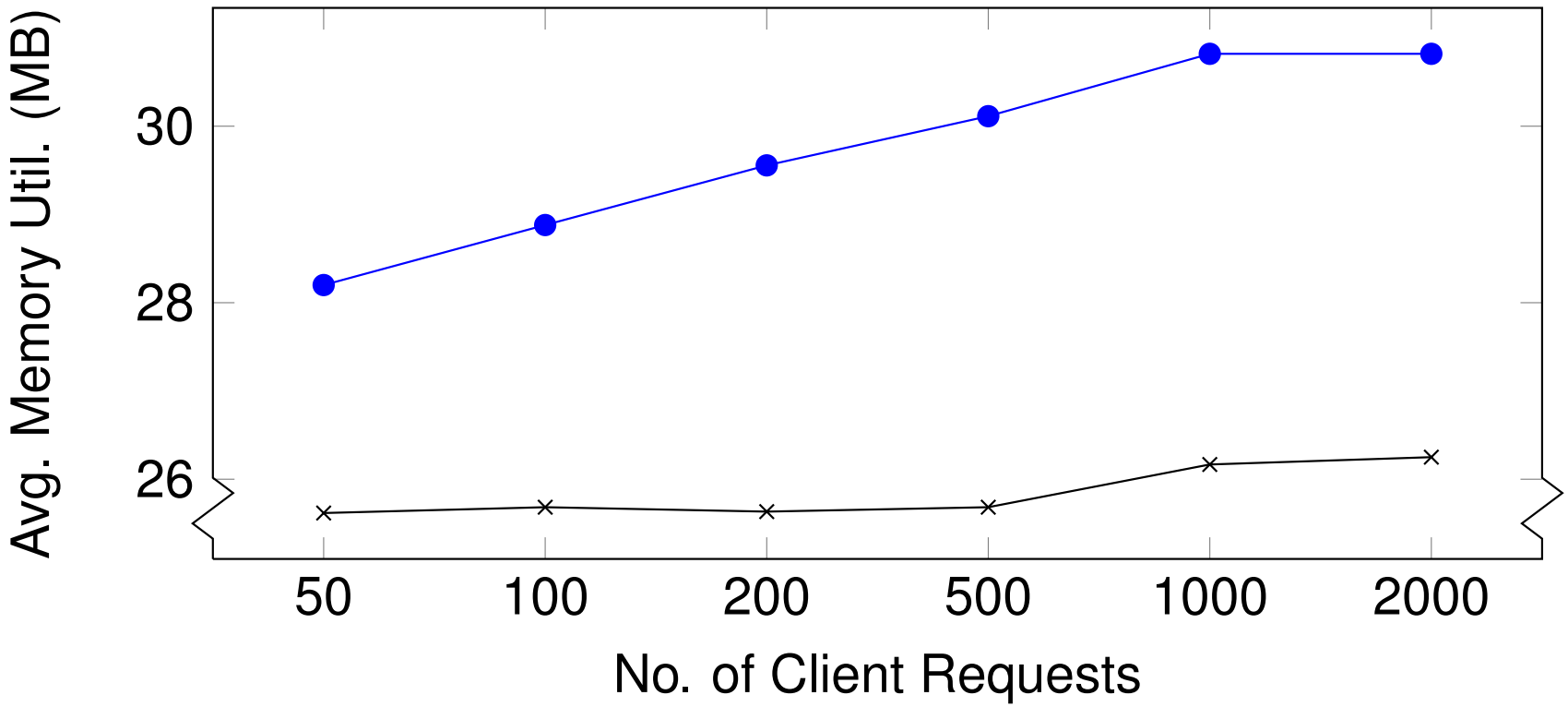
Case Study


New Handler
Instrum.
spawn
Acceptor
Instrum.
Resume or
apply adaptation
HTTP requests
Client 2
Handler 2
Instrum.
Check HTTP requests
Resume or
apply adaptation
Monitor
Client 1
Handler 1
Instrum.
HTTP requests
Check HTTP requests

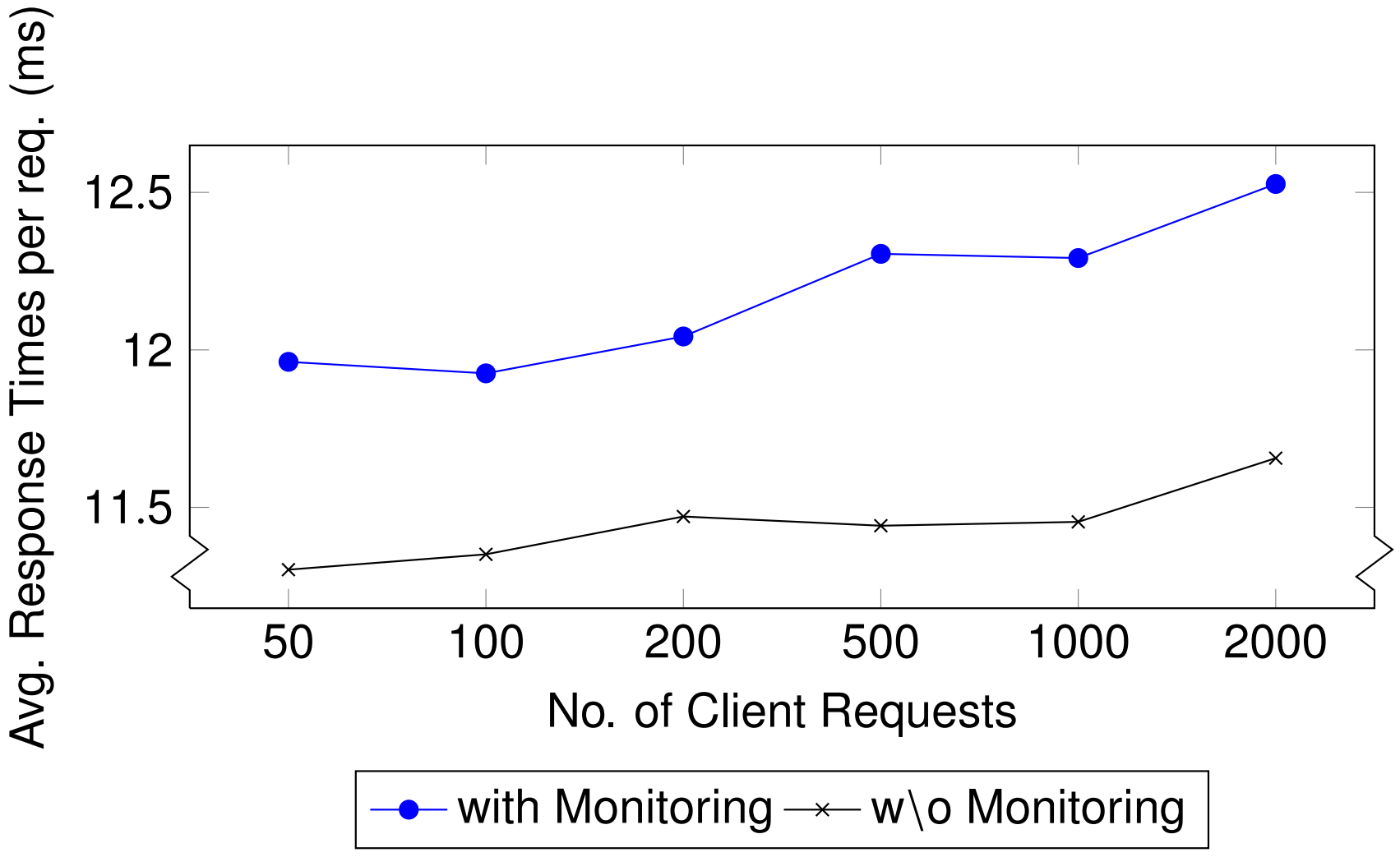
Adaptation Overheads



Adaptation Overheads



Adaptation Overheads



Conclusions

Contributions

1. POC extension of an adaptation framework for Actors that goes beyond failure events.
2. Mechanism for incremental synchronisations keeping overheads to a minimum.
3. Application of MOP to a third-party software.

References

Cassar, I., and Francalanza, A.

On Synchronous and Asynchronous Monitor Instrumentation for Actor Systems.
In *FOCLASA* (2014), vol. 175, pp. 54–68.

Cassar, I., and Francalanza, A.

Runtime Adaptation for Actor Systems.
In *RV* (2015), vol. 9333 of *LNCS*, Springer, pp. 38–54.

detectEr Project.

`http://www.cs.um.edu.mt/svrg/Tools/detectEr/`.