



Prof. George Spanoudakis

SERENITY: Monitoring Support



EuroTrustAmI, September 2008

- Overview of SERENITY: aims and motivation
- Need for monitoring in SERENITY
- The SERENITY monitoring approach
- An example
- Monitoring lifecycle
- Monitoring infrastructure
- Specification of monitoring rules
- Examples of monitoring rules
- Monitoring process
- Reaction to monitoring results
- Conclusions
- Further readings

Overview of SERENITY

Aims:

Dynamic

- selection
- (re-) configuration
- integration, and
- deployment

of components that can realise **Security** and **Dependability** (S&D) **properties** to applications driven by **S&D patterns**

Motivation:

Applications

- Have continually changing S&D requirements
- Often need to operate in changing operational environments and contents
- Interact with dynamically assembled distributed components

Need for monitoring in SERENITY

Runtime monitoring of S&D solutions is required in order to

- Check **preconditions** and **invariants** required for the correct operation of the solutions
- **Verify dynamically** that an S&D solution operates according to its specification in all circumstances (static verification and testing cannot provide full guarantees for this)
- **Predict possible violations** of conditions and take (if possible) **pre-emptive actions**

3 alternatives

- The application performs the checks itself
- The checks are performed by an external entity
- The checks are performed by both the application and an external entity

3 alternatives

- The application performs the checks itself

Requires **extra programming** effort, **expensive** to change when the system is in operation and needs to deploy a new S&D solution, some checks need to be applied on the deployed S&D solution which the application has no control of

- The checks are performed by an external entity

- The checks are performed by both the application and an external entity

3 alternatives

- The application performs the checks itself

Requires **extra programming** effort, **expensive** to change when the system is in operation and needs to deploy a new S&D solution, some checks need to be applied on the deployed S&D solution which the application has no control of

- The checks are performed by an external entity

Requires **monitoring specifications**, **more flexible** when operational environments change and S&D solutions change, **can be applied to external components**, **less efficient** than application based testing

- The checks are performed by both the application and an external entity

3 alternatives

- The application performs the checks itself

Requires **extra programming** effort, **expensive** to change when the system is in operation and needs to deploy a new S&D solution, some checks need to be applied on the deployed S&D solution which the application has no control of

- The checks are performed by an external entity

Requires **monitoring specifications**, **more flexible** when operational environments change and S&D solutions change, **can be applied to external collaborators**, **less efficient** than application based testing

- The checks are performed by both the application and an external entity


Increased fault tolerance (two independent implementations of the same check), more **expensive** and **less flexible** option, **necessary** in certain circumstances

3 alternatives

- The application performs the checks itself

Requires **extra programming** effort, **expensive** to change when the system is in operation and needs to deploy a new S&D solution, some checks need to be applied on the deployed S&D solution which the application has no control of

- The checks are performed by an external entity



Requires **monitoring specifications**, **more flexible** when operational environments change and S&D solutions change, **can be applied to external collaborators**, **less efficient** than application based testing

- The checks are performed by both the application and an external entity

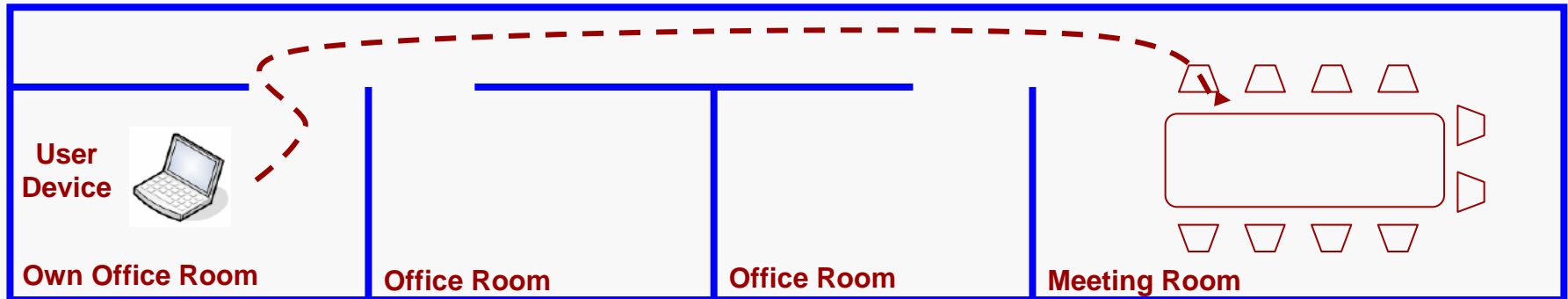
Increased fault tolerance (two independent implementations of the same check), more **expensive** and **less flexible** option, **necessary** in certain circumstances

Example: overview

- **Access control system** providing access to **enterprise resources** (e.g. printers, Internet access etc) from **mobile user devices** (PDAs, laptops etc)
- When a user requests access to a resource, the system may provide it, depending on:
 - the credentials of the user,
 - the ability to authenticate the device from which access is requested, and
 - the location of the device

Example: overview

- **Access control system** providing access to **enterprise resources** (e.g. printers, Internet access etc) from **mobile user devices** (PDAs, laptops) (based on [11])
- When a user requests access to a resource, the system may provide it depending on:
 - the credentials of the user,
 - the ability to authenticate the device from which access is requested, and
 - the location of the device



Access to

- Intranet, Internet
- Room's printer
- Printers in common areas

No access to

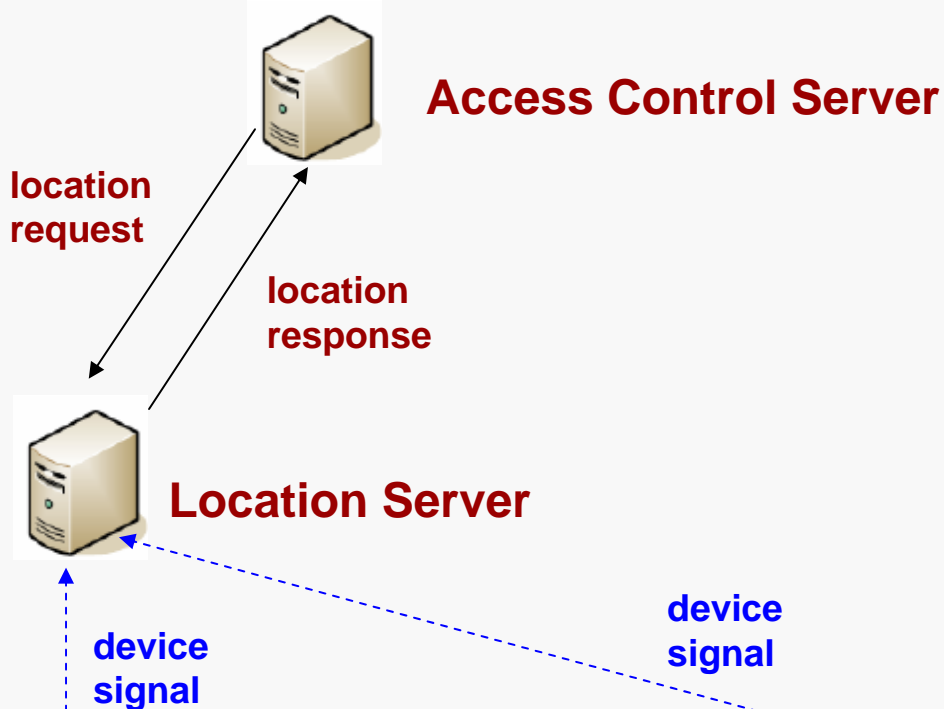
- printers in other rooms

Provided that both the mobile device and its user have been authenticated

Access to

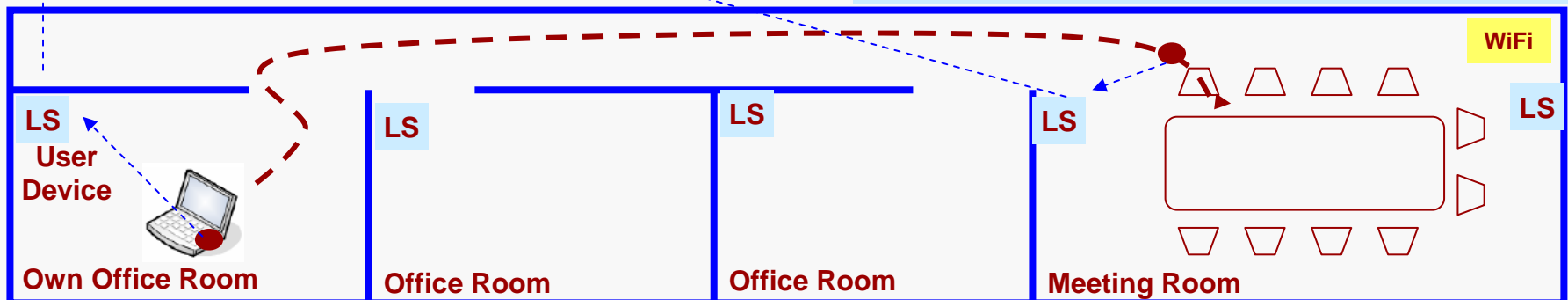
- Room's printer
 - Internet
- No access to**
- printers in other rooms
 - Intranet

Example: device position calculation



Zone based Security assessment pattern

- A daemon in mobile devices sends signals to location server (via location sensors)
- Based on the signals received from different sensors, the location server can calculate the position of a device with some accuracy measure
- The access control server requests the location server to calculate the position of devices



Example: some monitoring conditions

- **Availability of the location server:**

Whenever the access control server makes a request for the location of a device to the location server it must receive a response (or otherwise no access decisions can be made or access will be continually over-restricted)

- **Liveness of signal daemons in mobile devices:**

Every device that is known to the control server should be sending signals to the location server periodically and the maximum period of not receiving a signal should not be less than m time units (or otherwise it won't be possible to calculate the position of the device)

- **Accuracy of location information:**

The accuracy of the device location information that is provided by the location server must always (on average) exceed a certain accuracy threshold

Monitoring life cycle

Monitoring life cycle

Development of S&D solutions

Monitoring life cycle

Development of S&D solutions

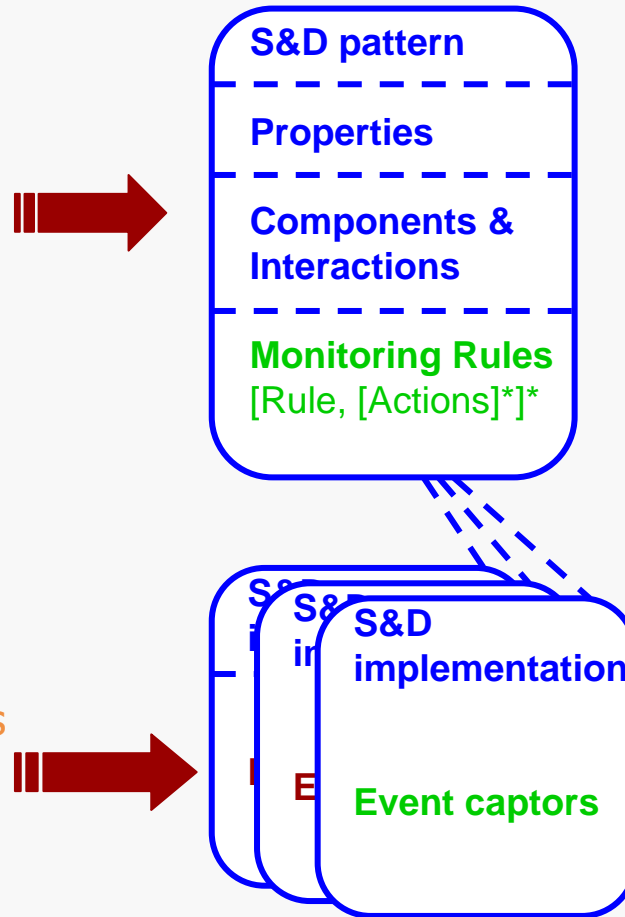
- Specify the **conditions** that need to be monitored at runtime and the **actions** that need to be taken when the conditions are violated within **S&D patterns**



Monitoring life cycle

Development of S&D solutions

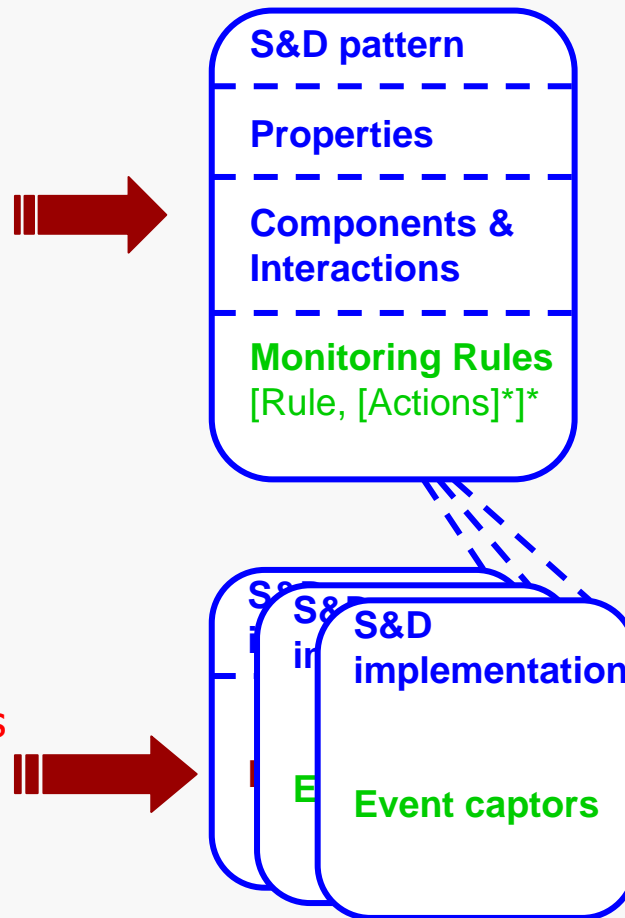
- Specify the **conditions** that need to be monitored at runtime and the **actions** that need to be taken when the conditions are violated within **S&D patterns**
- Provide implementations of patterns (aka S&D solutions) incorporating **captors** that can provide the events required to monitor the conditions of the pattern



Monitoring life cycle

Development of S&D solutions

- Specify the **conditions** that need to be monitored at runtime and the **actions** that need to be taken when the conditions are violated within **S&D patterns**
- Provide implementations of patterns (aka S&D solutions) incorporating **captors** that can provide the events required to monitor the conditions of the pattern



At runtime

When an S&D pattern is selected:

- Start the process of checking its monitoring rules
- Activate the relevant S&D implementation and its captors

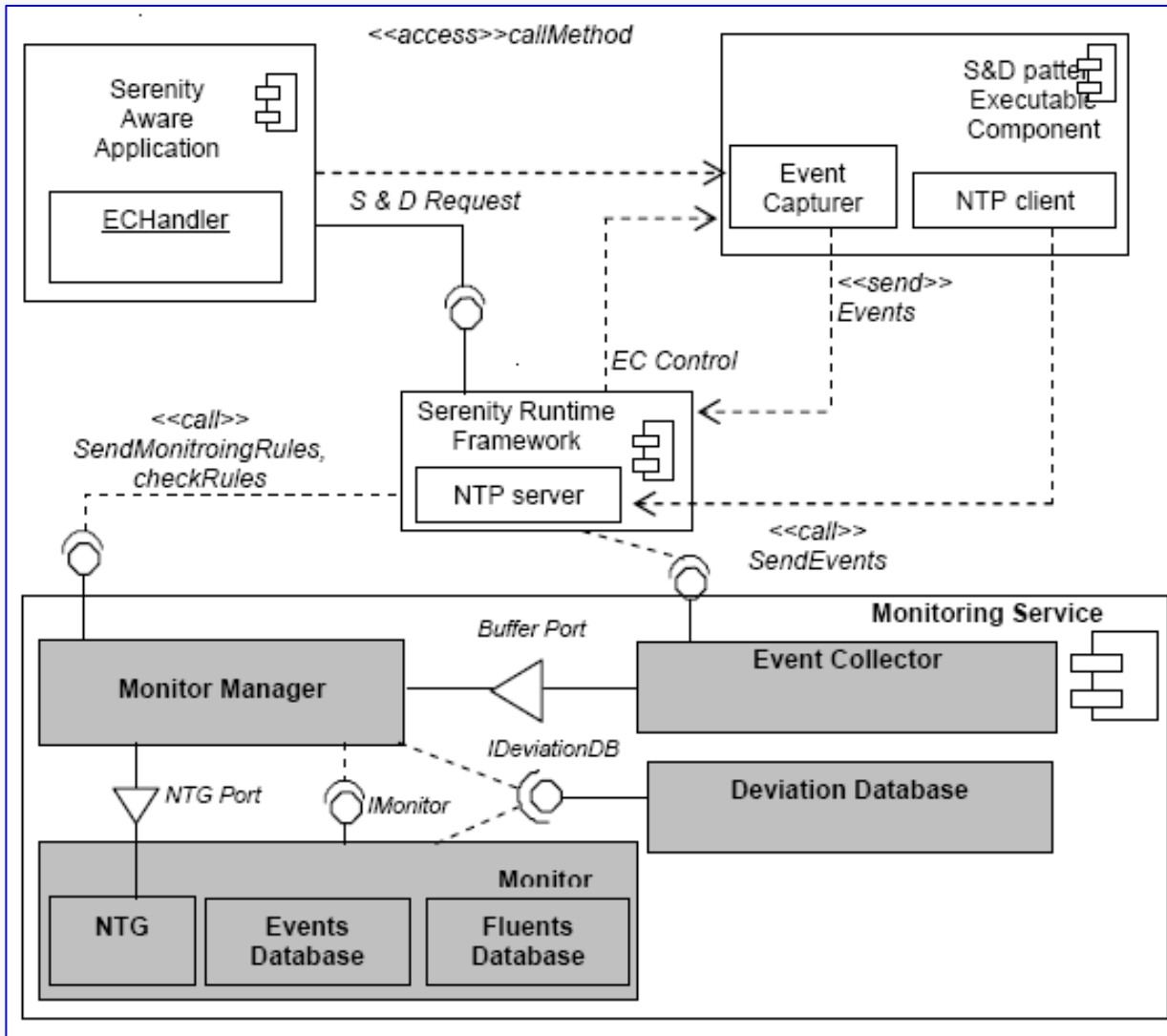
When a monitoring rule is violated:

- Execute the action(s) specified for it (if any)

When an S&D pattern is deactivated:

- Stop the process of checking its monitoring rules
- Deactivate the relevant S&D implementation and its captors

Monitoring Infrastructure



Monitoring service

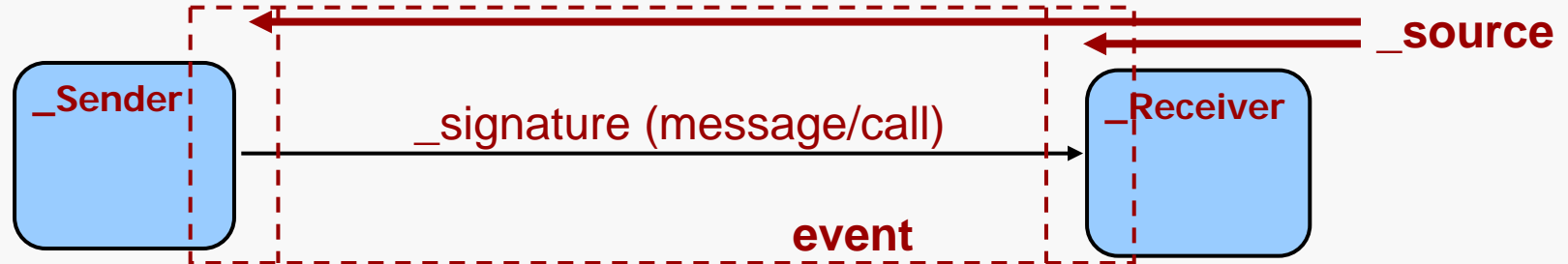
- Is available as a service to the SERENITY runtime framework (SRF)
- Receives specifications of the rules to be monitored and runtime events from the SRF
- Performs the checking
- Can be polled for monitoring results

SRF

- Activates patterns and their executable implementations
- Sends monitoring rules to the monitoring services
- Receives events from captors of pattern implementations and forwards them to the monitoring service
- Polls the monitoring service for results and executes actions according to them

- **Monitoring rules:** express the properties/requirements that need to be monitored
- General form
$$B_{t_1} \Rightarrow H_{t_2} \text{ (if } B_{t_1} \text{ is true then } H_{t_2} \text{ must be true)}$$
- B_{t_1} :
 - rule's body (a conjunction of conditions, e.g. occurrences of events, conditions regarding the state of the system)
 - It is typically expressed as a conjunction of **Happens**, **HoldsAt**, **relational** or **time** predicates
- H_{t_2} :
 - rule's head (a number of consequences)
 - It is typically expressed as a conjunction of **Happens**, **HoldsAt**, **relational** or **time** predicates

- Rules and assumptions are specified in **Event Calculus** – a **first order temporal logic language** – in terms of
 - **Events**: things that happen within a system of instantaneous duration (e.g. receipt of component messages, execution of internal or system operations)
 - **Fluents**: conditions about the state of a system
 $\text{relation}(\text{obj}_1, \dots, \text{obj}_N)$
- **Predefined predicates**:
 - **Happens**($e, t, \mathcal{R}(t_1, t_2)$) – occurrence of an event e of instantaneous duration at some time t within the time range $\mathcal{R}(t_1, t_2)$
 - **Initiates**(e, f, t) – fluent f starts to hold after the event e at time t .
 - **Terminates**(e, f, t) – fluent f ceases to hold after the event e occurs at time t
 - **HoldsAt**(f, t) – fluent f holds at time t .
 - **Relational predicates**: $x \text{ REL } y$ (e.g. EqualTo, NotEqualTo, ...)
 - **Time predicates**: $t_1 \text{ TREL } t_2$ (e.g. TEqualTo, TLessThan ...)



Events: General form

`e(_id, _senderRole, _senderID, _receiverRole, _receiverID, _status, _signature _sourceRole, _sourceID))`

- `_signature`: the type of a message sent by the component/system
- `_status`: indicates whether the message is incoming or outgoing
- `_senderRole`: the role of the component that sends the message
- `_senderID`: the id of the component that sends the message
- `_receiverRole`: the role of the component that receives the message
- `_receiverID`: the id of the component that receives the message
- `_sourceRole`: the role of the component at which the message is captured
- `_sourceID`: the id of the component at which the message is captured

Events typically correspond to operations defined in the interfaces of the components of the S&D pattern

- Other features
 - Calls to **built-in functions** implementing complex computations (e.g. statistical functions)

Happens(call(o), t_1 , R(t_1 , t_1)) \wedge Happens(response(o), t_2 , R(t_1 , t_2)) \wedge
HoldsAt(o_response_times(RT[]), t_2) \Rightarrow **m:append**(RT[], $t_2 - t_1$), t_2)

HoldsAt(o_response_times(RT[]), t_1) \Rightarrow **m:avg**(RT[]) < k

Examples of monitoring rules: Rule for location server availability



Condition: when the access control server sends a location request to the location server it should receive a response from it within 3 seconds

Examples of monitoring rules:

Rule for location server availability



Condition: when the access control server sends a location request to the location server it should receive a response from it within 3 seconds

Rule 1

Happens(e(_eID1, _controlServerRole, _controlServerID, _locationServerRole, _locationServerID, REQ, locationRequest(_dev, _loc, _prob), _controlServerRole, _controlServerID), t1, R(t1, t1))

⇒

Happens(e(_eID2, _locationServerRole, _locationServerID, _controlServerRole, _controlServerID, RES, locationRequest(_dev, _loc, _prob), _controlServerRole, _controlServerID), t2, R(t1+1, t1+3000))

Examples of monitoring rules:

Rules for liveness of device daemons

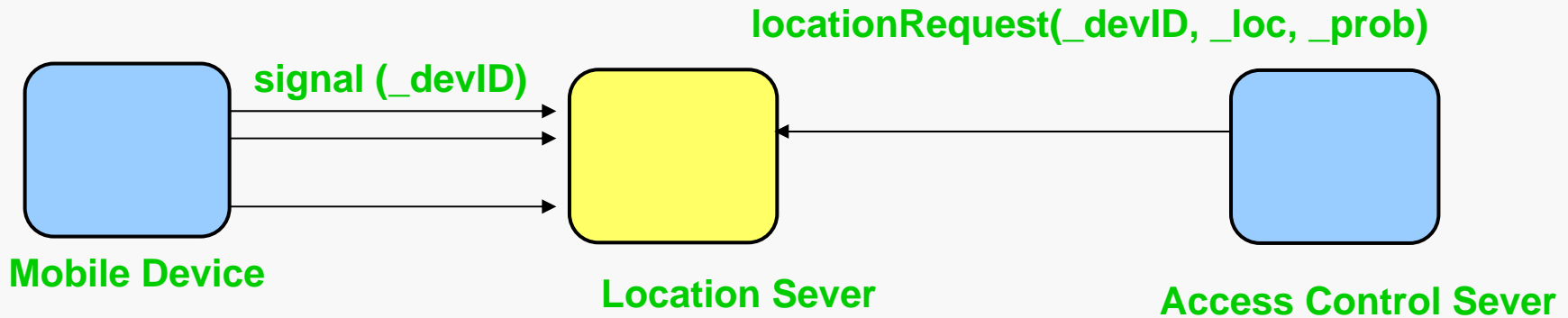
Condition: Every mobile device that is known to the control server should be sending signals to the location server periodically and the maximum period of not receiving a signal should not be less than m time units

Can be specified by 2 rules:

- A rule for checking when the first signal from a mobile device should be received
- A rule for checking the continuous receipt of signals

Examples of monitoring rules:

Rules for liveness of device daemons



Rule 2:

Happens($e_eID1, _cServerRole, _cServerID, _lServerRole, _lServerID, REQ, locationRequest(_devID, _loc, _prob), _lServerRole, _lServerID), t1, R(t1, t1) \wedge \neg \exists t2. \mathbf{Happens}(e_eID2, _cServerRole, _cServerID, _lServerRole, _lServerID, REQ, locationRequest(_devID, _loc, _prob), _lServerRole, _lServerID), t2, R(0, t1-1)) \Rightarrow \exists t3. \mathbf{Happens}(e_eID3, _deviceRole, _devID, _lServerRole, _lServerID, RES, signal(_devID), _lServerRole, _lServerID), t3, R(t1-m, t1))$

Rule 3:

Happens($e_eID1, _deviceRole, _devID, _lServerRole, _lServerID, REQ, signal(_devID), _lServerRole, _lServerID), t1, R(t1, t1) \Rightarrow$

Happens($e_eID2, _deviceRole, _devID, _lServerRole, _lServerID, REQ, signal(_devID), _lServerRole, _lServerID), t1, R(t1, t1+m) \wedge _eID1 \neq _eID2$

Examples of monitoring rules:

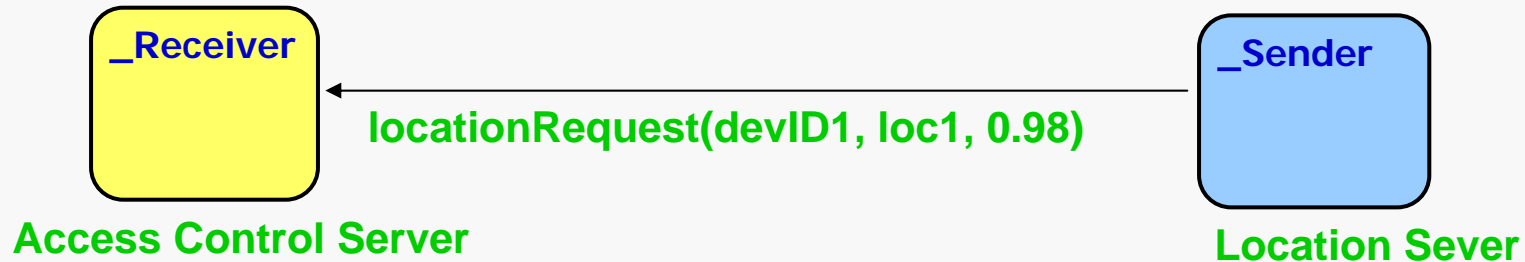
Rule for accuracy of location information



Condition: The accuracy of the device location information that is provided by the location server must always exceed a certain accuracy threshold

Examples of monitoring rules:

Rule for accuracy of location information



Condition: The accuracy of the device location information that is provided by the location server must always exceed a certain accuracy threshold

Rule 4

```
Happens(e(_eID1, _locationServerRole, _locationServerID, _controlServerRole,  
_controlServerID, RES, locationRequest(_dev, _loc, _prob), _controlServerRole,  
_controlServerID), t1, R(t1, t1))
```

\Rightarrow $_prob \geq AT$

Monitoring Process: Basics

- It is based on a generic **event calculus reasoning engine** (see [1,6,7,8])
- Consistency checking using
 - **Runtime events**
 - **Derived events** generated from assumptions by **deductive reasoning**
- Checks can relate to both **past** and **future** EC formulas
- Ability to analyse
 - events captured from **distributed sources** with **different clocks**
 - events arriving at the monitor **not in** the same **order** as the order of their capture

Monitoring process: diagnostic capabilities

- Given a **violation** of an S&D monitoring rule

$$R: E_1, E_2, E_3, \dots, E_n \Rightarrow E_{n+1}$$

Calculate beliefs in the genuineness of the events $E_1, E_2, \dots, \neg E_{n+1}$ which are involved in the violation since events might be the result of an **attack** or **fault**

- Approach
 - The genuineness of an event depends on the ability to find a **valid explanation** for it
 - An event explanation is a logical combination of other events and states of the system which would have the event as a consequence
 - An event explanation is considered to be valid if it has as consequences other events which have also been observed and are genuine
 - Possible event explanations are generated by **abductive reasoning** using the monitoring specifications of the active patterns of the system that is being monitored
 - Event genuineness is assessed by **beliefs** computed according the **Dempster-Shafer theory** of evidence

Monitoring process: threat detection capabilities

Detection of **potential violations** of S&D monitoring rules

$$R: E_1, E_2, E_3, \dots, E_n \Rightarrow E_{n+1}$$

Calculate belief that R will be violated given the observation of a subset of E_1, E_2, \dots, E_{n+1}

- Events might
 - Not be observed in the order they are expected by R
 - Be the result of an **attack** or **fault** (and therefore a belief in their genuineness needs to be estimated; see diagnosis)
- Approach
 - Use **DS beliefs** to measure the likelihood of events genuineness and the likelihood of conditional event occurrence
 - Negate the rule to get the exact pattern of events that violates it
 - Construct a **belief network** indicating how beliefs in the violation of the rule can be updated as partial evidence about events in the pattern emerges

Reaction to monitoring results (1)

- Focus: Automatic reactions taken at runtime by the SRF following monitoring results

Rule specification = EC formula + [(action₁, cnd₁), ..., (action_N, cnd_N)]

- Semantics:
 - Each of the actions (*action_i*) is executed only if the condition associated with it is also satisfied (*cnd_i*)
 - The actions are executed in the exact order that they appear in the rule specification
- The SRF supports only predefined types of actions
- Complex conditions may be associated with actions

- Action types
 - `DeactivatePattern()`
 - `RestartPattern()`
 - `NotifySRF(String external_SRF_ID, String Message)`
 - `NotifyApplication(String message)`
 - `StopMonitoringRules(String ruleID1, String ruleID2,... String ruleIDn)`
 - `StartMonitoringRules(String ruleID1, String ruleID2,... String ruleIDn)`
 - `Log()`

Conclusions

- SERENITY provides a monitoring infrastructure for checking at runtime whether certain preconditions and invariants required for the correct operation of S&D patterns are satisfied
- The conditions to be monitored are specified as monitoring rules in Event Calculus
- Monitoring rules are specified as part of S&D patterns and need to be accompanied by the actions that should be taken when they are violated
- The monitoring infrastructure provides basic monitoring and diagnosis capabilities
- Forthcoming versions of the monitoring infrastructure will also provide threat detection capabilities (i.e., detection of potential violations of monitoring rules)

Further readings (1)

1. Kloukinas C., Spanoudakis G., Mahbub K.: Estimating Event Lifetimes for Distributed Runtime Verification, 20th International Conference on Software Engineering and Knowledge Engineering, July 2008
2. Tsigritis T., Spanoudakis G.: Diagnosing Runtime Violations of Security & Dependability Properties, 20th International Conference on Software Engineering and Knowledge Engineering, July 2008
3. Amalio N., Spanoudakis G.: From Monitoring Templates to Security Monitoring and Threat Detection, 2nd International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2008), August 2008
4. Kloukinas C., Spanoudakis G., : A Pattern-Driven Framework for Monitoring Security and Dependability , 4th International Conference on Trust, Privacy and Security in Digital Business (TrustBus`07), Lecture Notes in Computer Science 4657/2007, DOI: 10.1007/978-3-540-74409-2_23, September 2007
5. Spanoudakis G., Kloukinas C., Androutsopoulos K.: Towards Security Monitoring Patterns, 22nd Annual International ACM Symposium on Applied Computing – Track on Software Verification, Seoul, South Korea, March 2007
6. Spanoudakis G. Mahbub K.: Non Intrusive Monitoring of Service Based Systems, International Journal of Cooperative Information Systems, 15(3), pp. 325-358, 2006

Further readings (2)

7. Androutsopoulos K., Ballas K., Kloukinas C., Mahbub K., and Spanoudakis G, "V1 of Dynamic Validation Prototype", Deliverable A4.D3.1, SERENITY Project. Available from http://www.serenity-forum.org/IMG/pdf/A4.D3.1_dynamic_validation_prototype_v1.2_final.pdf, 2006
8. Mahbub K., Spanoudakis G., Kloukinas C. (2007): "V2 of dynamic validation prototype". Deliverable A4.D3.3, SERENITY Project, Available from: [http://www.serenity-forum.org/IMG/pdf/A4.D3.3 - V2 of Dynamic validation Prototype.pdf](http://www.serenity-forum.org/IMG/pdf/A4.D3.3_-_V2_of_Dynamic_validation_Prototype.pdf), 2007
9. Spanoudakis G., Tsigkritis T. : "1st Version of Diagnosis Prototype". Deliverable A4.D5.1, SERENITY Project, Available from: http://www.serenity-forum.org/IMG/pdf/A4.D5.1_first_version_of_diagnosis_prototype_v1.1_final.pdf, 2008
10. Amalio N., DiGiacomo V., Kloukinas C., Spanoudakis G.: "Mechanisms for detecting potential S&D threats". Deliverable A4.D4.1, SERENITY Project, Available from: <http://www.serenity-forum.org>, 2008
11. Li K, et. Al: "Scenario S&D solutions v1", Deliverable A7.D4.2, SERENITY Project, Available from: <http://www.serenity-forum.org>, 2008

Thank you