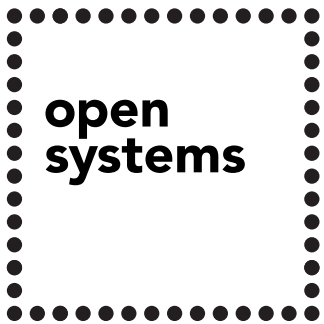
Event Correlation Engine

Master's Thesis – Final Presentation

Andreas Müller

Tutors:
Christoph Göldi, Bernhard Tellenbach

Supervisor:
Prof. B. Plattner



Event Correlation

Event Correlation

- Event
  - Any occurrence; anything, which happened
  - Computing: message, what happened when
- Correlation: analysis of co-relations
- Goal: gain higher-level knowledge
- Applications
  - Market data analysis
  - Algorithmic trading
  - Fraud detection
  - System log analysis
  - Network management

Event Correlation Engine (ECE)

- Application or toolkit to correlate events

Background

Background

- Master's thesis at Open Systems AG
- Open Systems provides managed security for customers around the world, operating a global network with more than 1500 hosts

Setup

- Hosts generate events from syslog messages and network observations, e.g.:
  - Network link down
  - CPU load high
- Events end up in tickets, which are handled manually

Motivation and Task

Problems

- Handling all events manually is time consuming and cumbersome
- Simple problems create too many events, important events may be overlooked
- Same problem has to be handled again and again

How to mitigate these problems?

$\Rightarrow$ Intelligently pre-process the events with an ECE

Task

- Choose or design, implement and evaluate an ECE suitable to extend the ticketing system of Open Systems

Event Pattern Analysis

Pattern Analysis: Goals

- Qualitative and quantitative overview of events
- Identification of frequent patterns

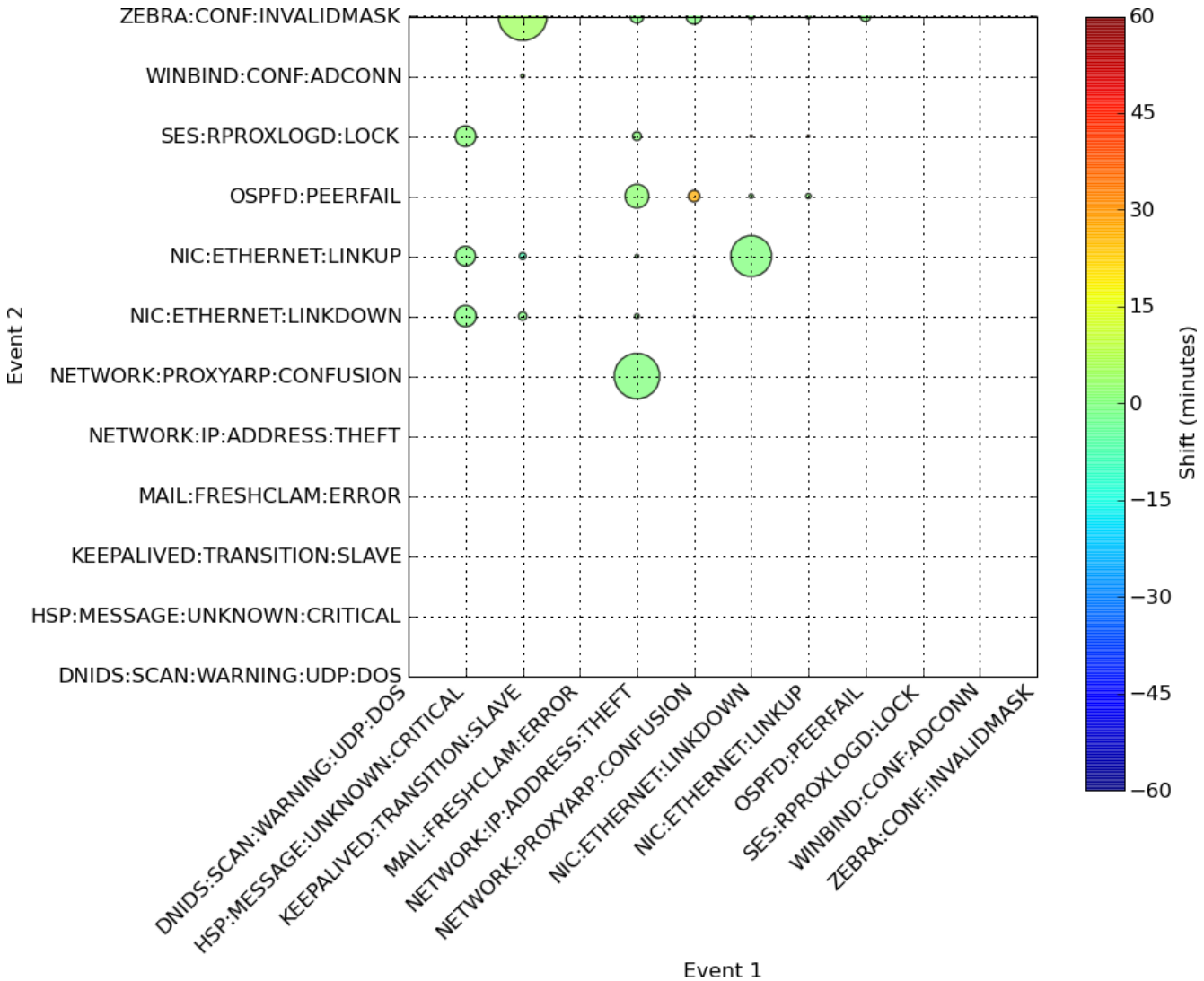
Statistical analysis

- E.g. event rate:
  - Average: 1-2 events per minute
  - Peaks: Up to 900 events per minute

Pattern identification

- E.g. temporal correlation (next slide)

Event Pattern Analysis: Temporal Correlation



Analysis of Existing Approaches

Correlation Approach

- Decisions of the correlation engine should be reproducible and understandable for humans
  - ⇒ Rule-based approach most suitable
- Finite-state machine would be suitable for some patterns
  - ⇒ Allow regular expressions on events

Existing Software

- No suitable software found
- Many concepts can be reused, e.g.:
  - Dynamic contexts
  - Combination of simple building blocks into powerful rules

Specification

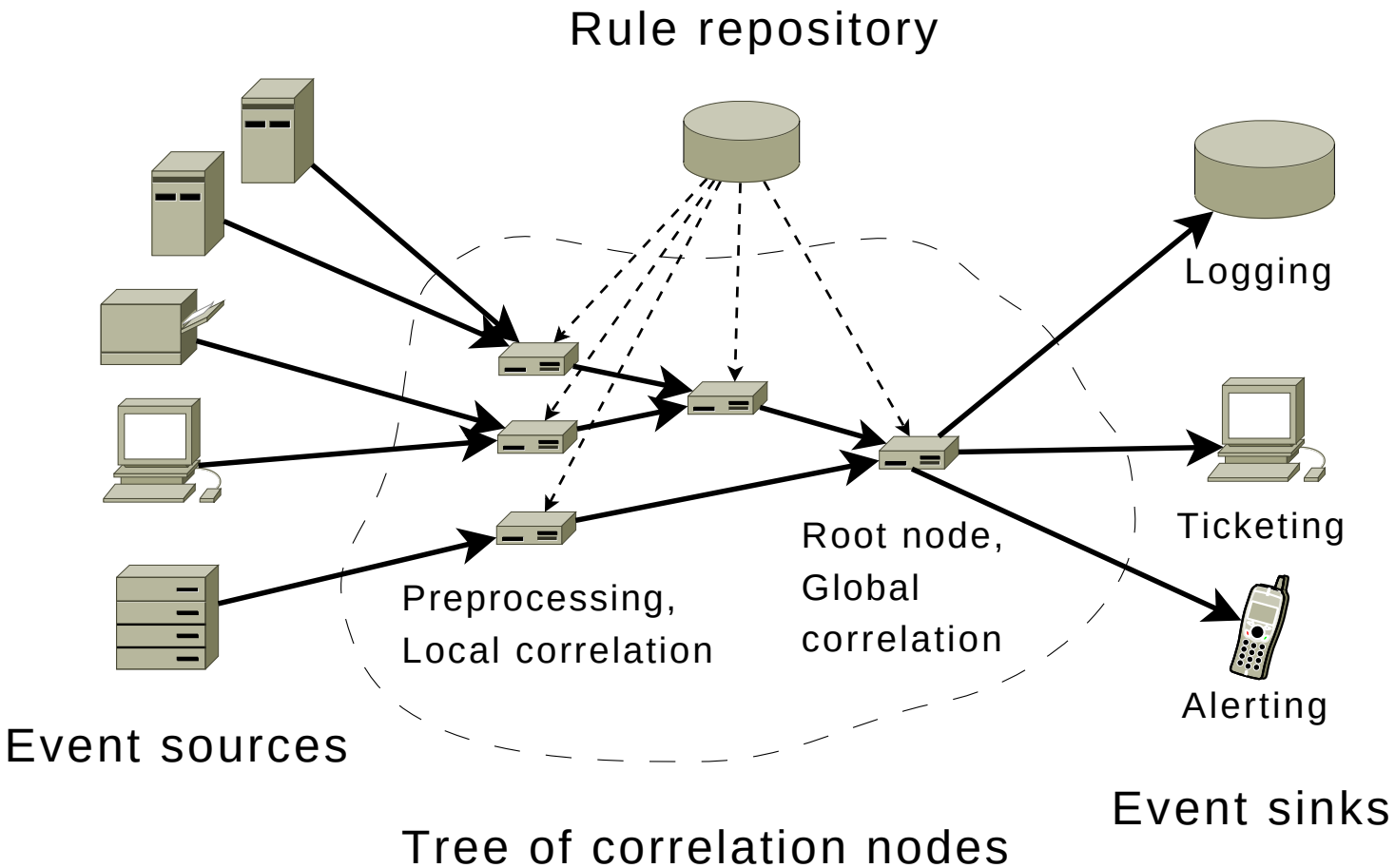
Requirements and design decisions

- Rule language should be easy to learn
  - ⇒ XML based rules
- Local and global correlation should be possible
  - ⇒ Tree of homogeneous correlation nodes

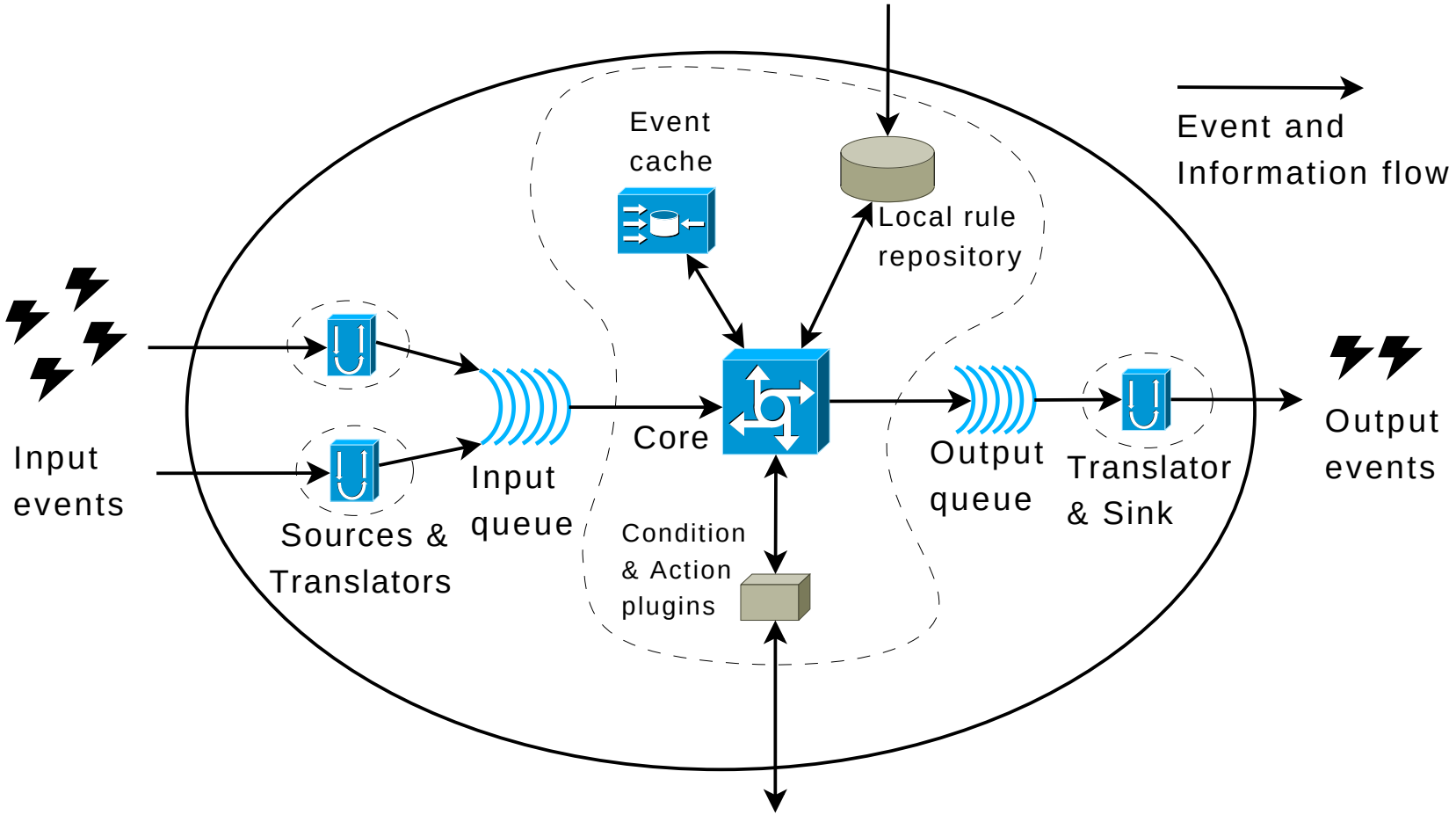
Implementation

- Functional programming to build rule functions from simple components
- Rapid prototyping was valued higher than execution speed
  - ⇒ Implemented in Python

Implementation: Node Tree



Implementation: Functional Overview



Testing and Evaluation

Testing and evaluation methods

- Profiling
- Unit tests
- Sanity checks
- Evaluation with random events
- Evaluation with replayed real events

Conclusions

Strengths

- Functional programming allows to build rule functions at startup
  - ⇒ No need for (slow and hard to debug) `eval()` during operation
- Regular expressions to match patterns suitable for FSMs
  - ⇒ As powerful as FSMs, but better known and easier to use
- Cache automatically decides how long to keep an event

Aptitude for real-world events

- Events of one month can be processed in $<$ 10 minutes
  - ⇒ Real-time operation no problem
- Simple compression can reduce the event volume by $>$ 50%
- Successful detection of complex patterns
  - ⇒ E.g. detection of successful failover transition with regexp

Outlook

Future development

- Support for rule creation (e.g. pattern mining, GUI tools)
- Central rule database with target scopes for each rule group
  - $\Rightarrow$ “Rule applies to all hosts in country X”
- Reducing complexity

Demo: Example Problem

Example: Irrelevant IP theft events

- Standby firewall becomes master, while primary firewall is still up
- Firewalls detect second host with same IP address
  - $\Rightarrow$ Events indicating IP address theft
- Duplicate master is also detected
  - Corresponding event often generated after IP theft event
- It is sufficient to solve the root problem (duplicate master)
  - $\Rightarrow$ IP address theft events are of no interest

Demo: Correlation Approach

Correlation behaviour

- Event indicating a duplicate master
  - ⇒ Represent this knowledge as context
  - ⇒ Suppress IP theft events from same host during last minute
- As long as context exists, suppress further IP theft events
- Event indicating duplicate master is gone
  - ⇒ Remove context

Demo

- Real events with anonymized host names
- One correlation node
  - First run: without correlation rules
  - Second run: rules with behaviour explained above

Questions?

Thank you for your attention.