DISTRIBUTED EVENT-DRIVEN MONITORING

INTRO

Daniel Tovarňák

LAB OF SOFTWARE ARCHITECTURES
AND INFORMATION SYSTEMS

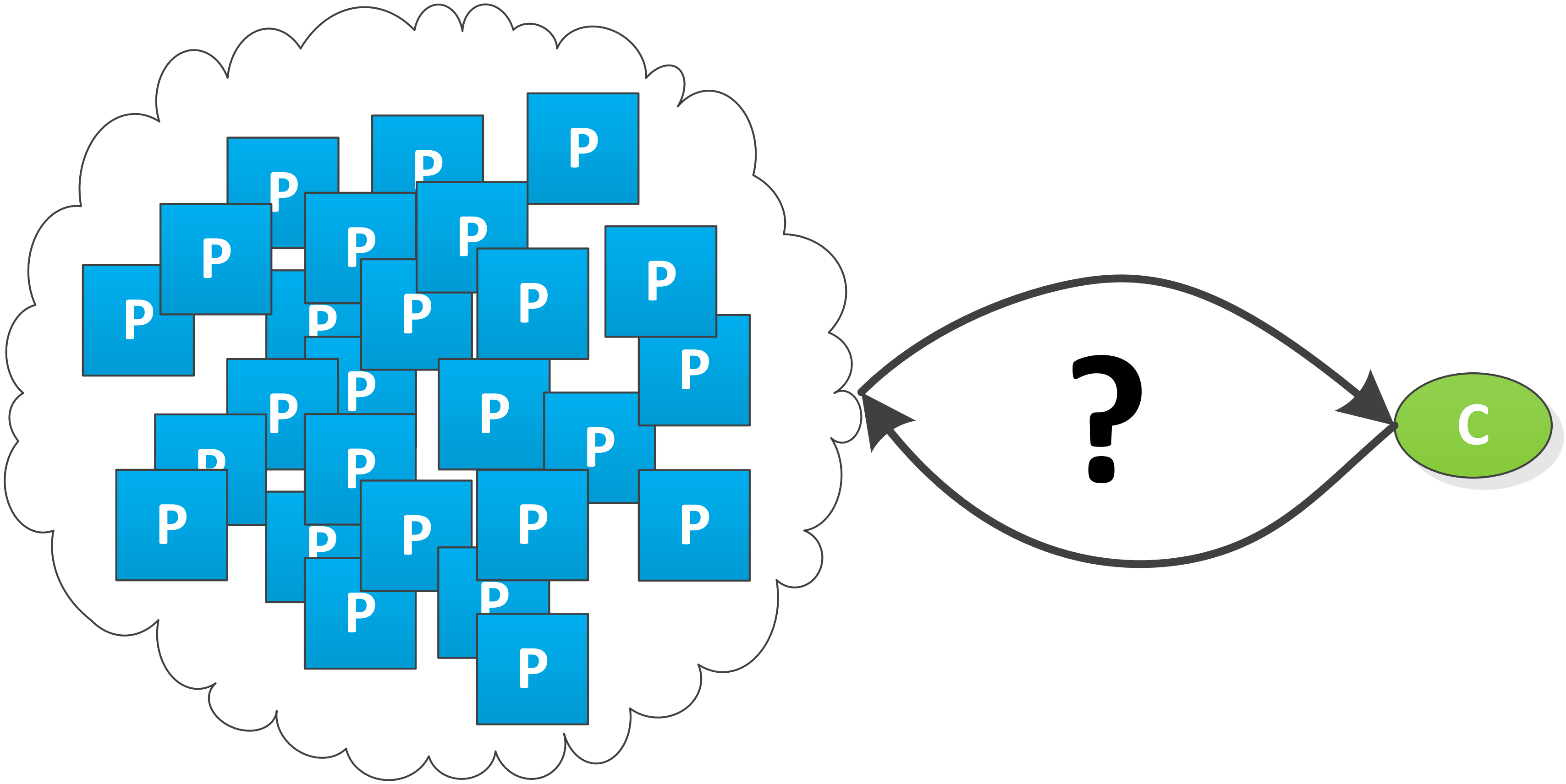
FACULTY OF INFORMATICS
MASARYK UNIVERSITY

Monitoring (of distributed infrastructure)

- *Continuous and systematic collection, analysis, and evaluation of data related to the state and behavior of respective constituents of said infrastructure.*

- Enterprise networks
- Internet of Things
- *Smart Grid (energy grid)*
- **Cloud infrastructure**

Monitoring in General



Goal – intelligent behavior monitoring

- Detection of (known) **behavior patterns** in the produced monitoring data in real-time
  - Dictionary attack, DDoS detection, Job state

- Monitoring information: *User Bob has logged in*
- Pattern: *User **X** failed to log in 1000 times within 1 minute*

- **Low overhead** imposed on monitored machines and network

- Several problems hinder achievement of such a goal

Monitoring of Cloud infrastructure

- **Huge volumes** of data produced by many distributed producers (virtual machines)

- **High variability** of monitoring data
  - Hardware, OS, Middle-ware, Web server, Application-level

- The entity of interest is usually **spread** across many computing nodes
  - Hadoop job, Custom distributed algorithm, Replicated DB

- Specific **trust model**

Problems

- Technical
  - mainly with respect to the monitoring data production
  - e.g. logging in natural language

- Conceptual
  - related to 3V of Big Data
  - e.g. scalability, and query expressiveness/complexity

Monitoring data collection

- *Huge volumes of data (up to 1MB/s per VM)*
  - *typically 100-1000 producers*
- Centralized
  - Limited scalability
- Selective (eg. Publish-subscribe)
  - Still centralized (data-wise)
- Distributed (eg. Hadoop Distributed File System)
  - Possible solution, in combination with pub-sub

Distributed processing

- Traditional DBMSs (distributed or not) are not very suitable for **continuous queries** (from the performance perspective)

- Solutions based on distributed collection and batch processing (MapReduce) have **high latency** (~mins)

- **Off-line** vs. **On-line** algorithms

Distributed Event-driven Monitoring Model

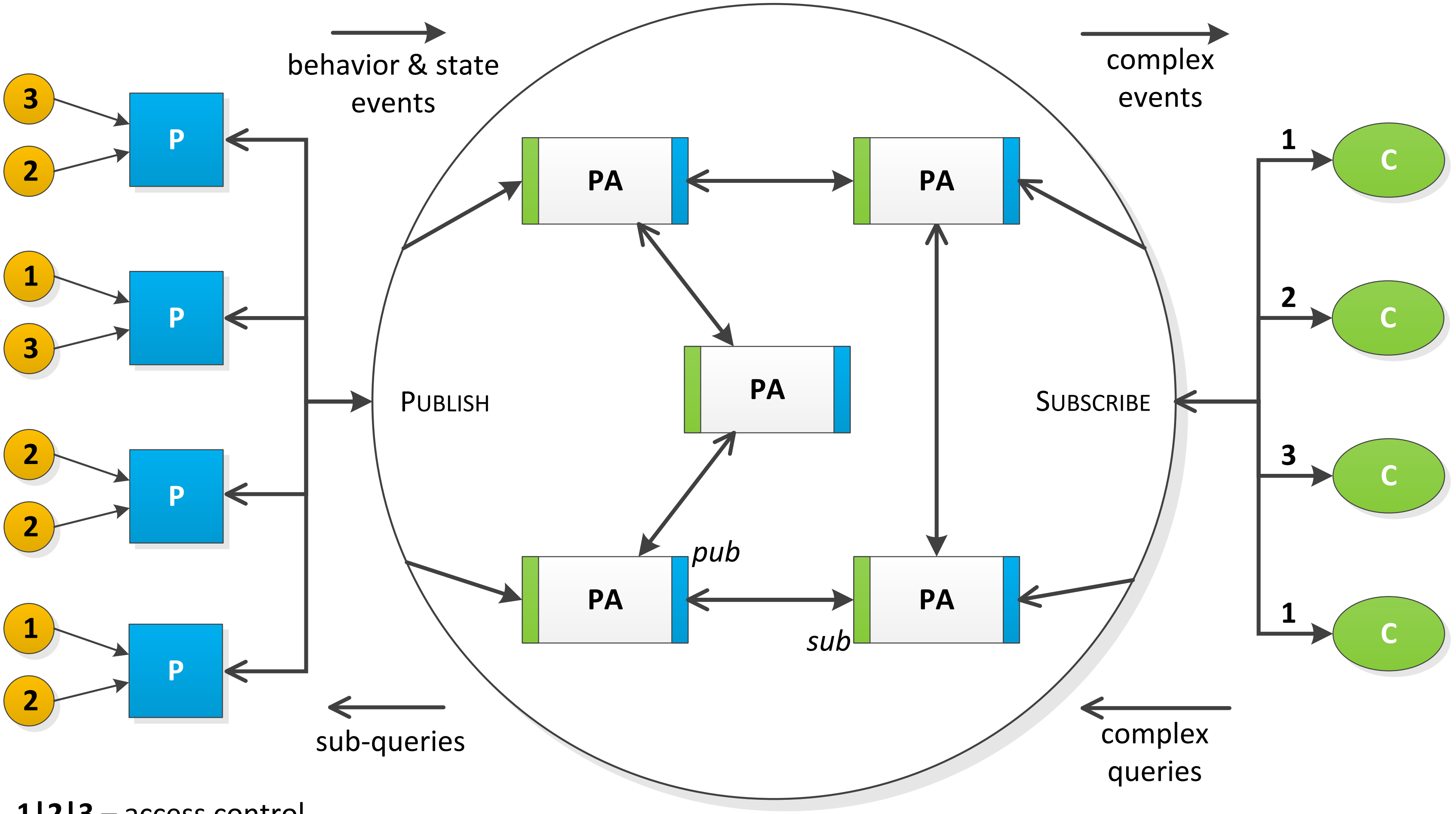
- Stream (online) processing of monitoring data in the form of events – everything is an event
- Techniques and algorithms for complex event processing
- **Fully distributed** processing using special variant of publish-subscribe (pattern-based)

Event-driven

- We consider everything to be an event
  - Measurement/metric (it is a *predefined* change)
  - State (its change)
  - Event (duh…)

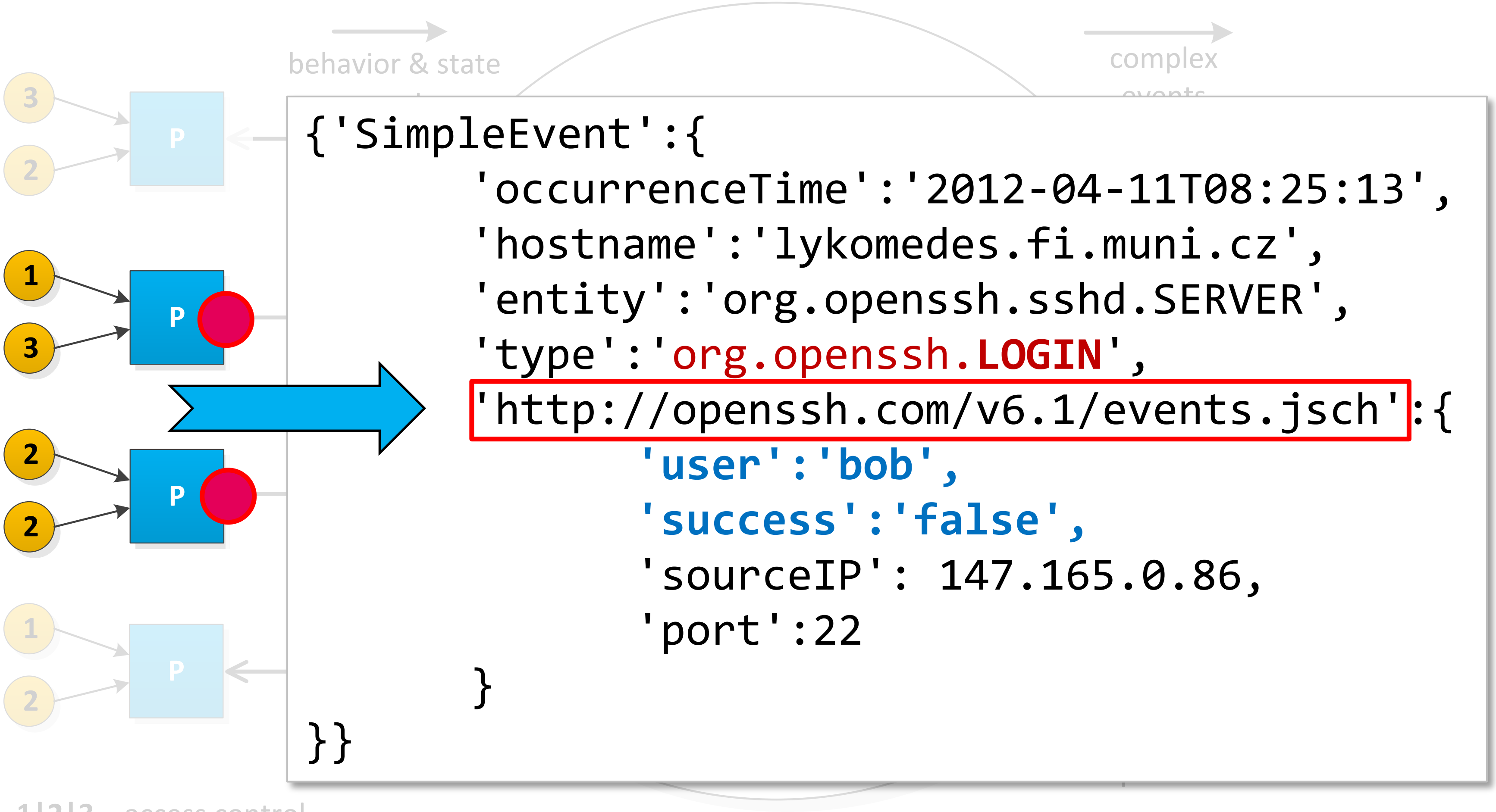
- Complex Event Processing
  - simple events are composed into more complex ones
  - **final complex event = detected pattern**

Distributed Event-driven Monitoring Model



1|2|3 – access control

Distributed Event-driven Monitoring Model

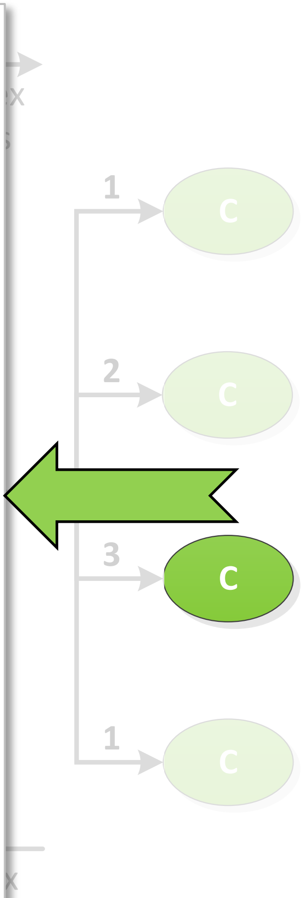


```
{'SimpleEvent':{
        'occurrenceTime':'2012-04-11T08:25:13',
        'hostname':'lykomedes.fi.muni.cz',
        'entity':'org.openssh.sshd.SERVER',
        'type':'org.openssh.LOGIN',
        'http://openssh.com/v6.1/events.jsch':{
                'user':'bob',
                'success':'false',
                'sourceIP': 147.165.0.86,
                'port':22
        }
}}
```

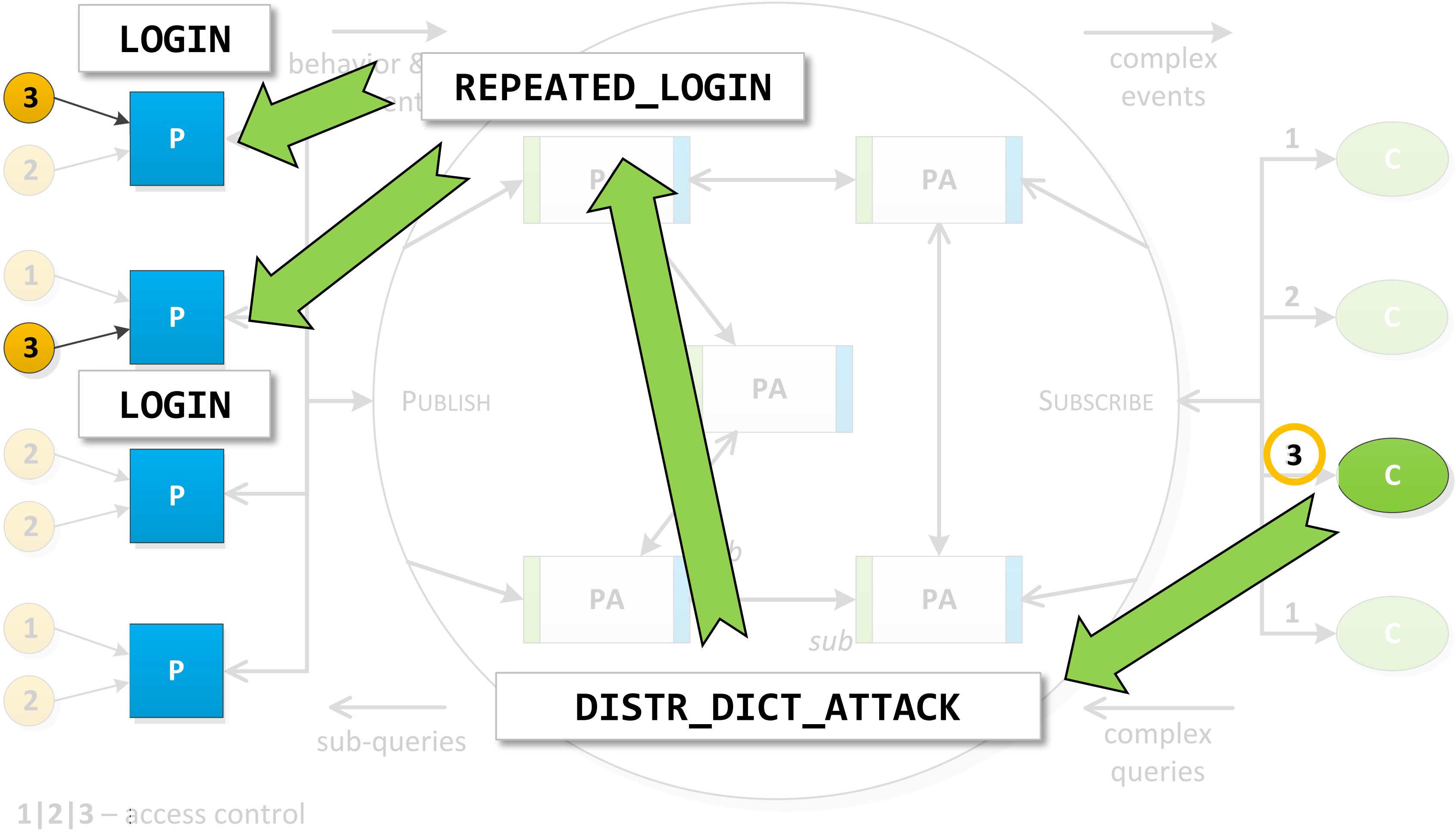
1|2|3 – access control

Distributed Event-driven Monitoring Model

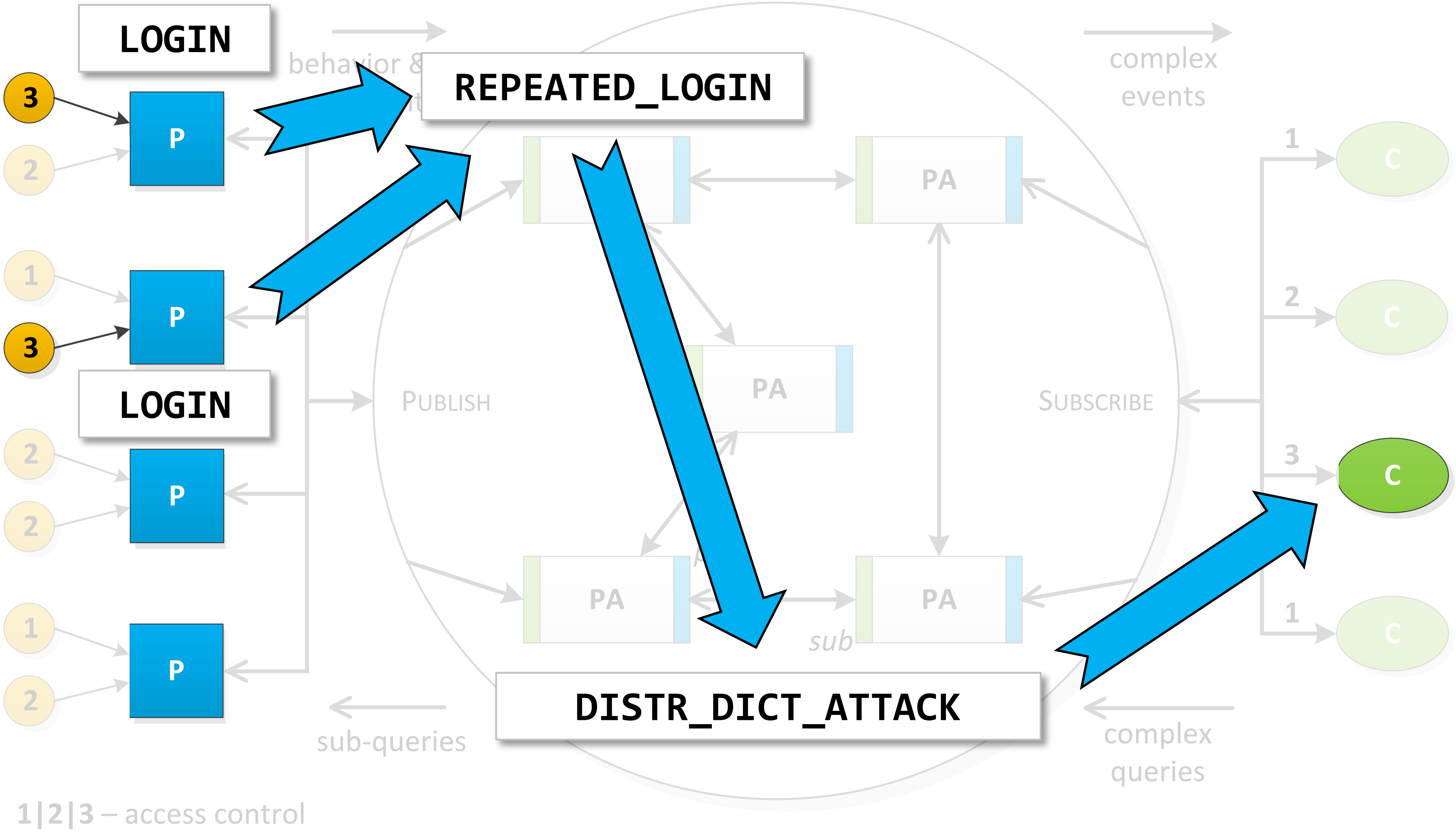
```
Subscribe for DISTR_DICT_ATTACK=

select count(*) as hostsNumber
from RepeatedLoginEvent.win:time(2 min)
where hostsNumber > 10
group by hostname

AND REPEATED_LOGIN=

select hostname, username,
      success, count(*) as attempts
from LoginEvent.win:time(60 sec)
where attempts > 1000, success=false
group by hostname, username
```



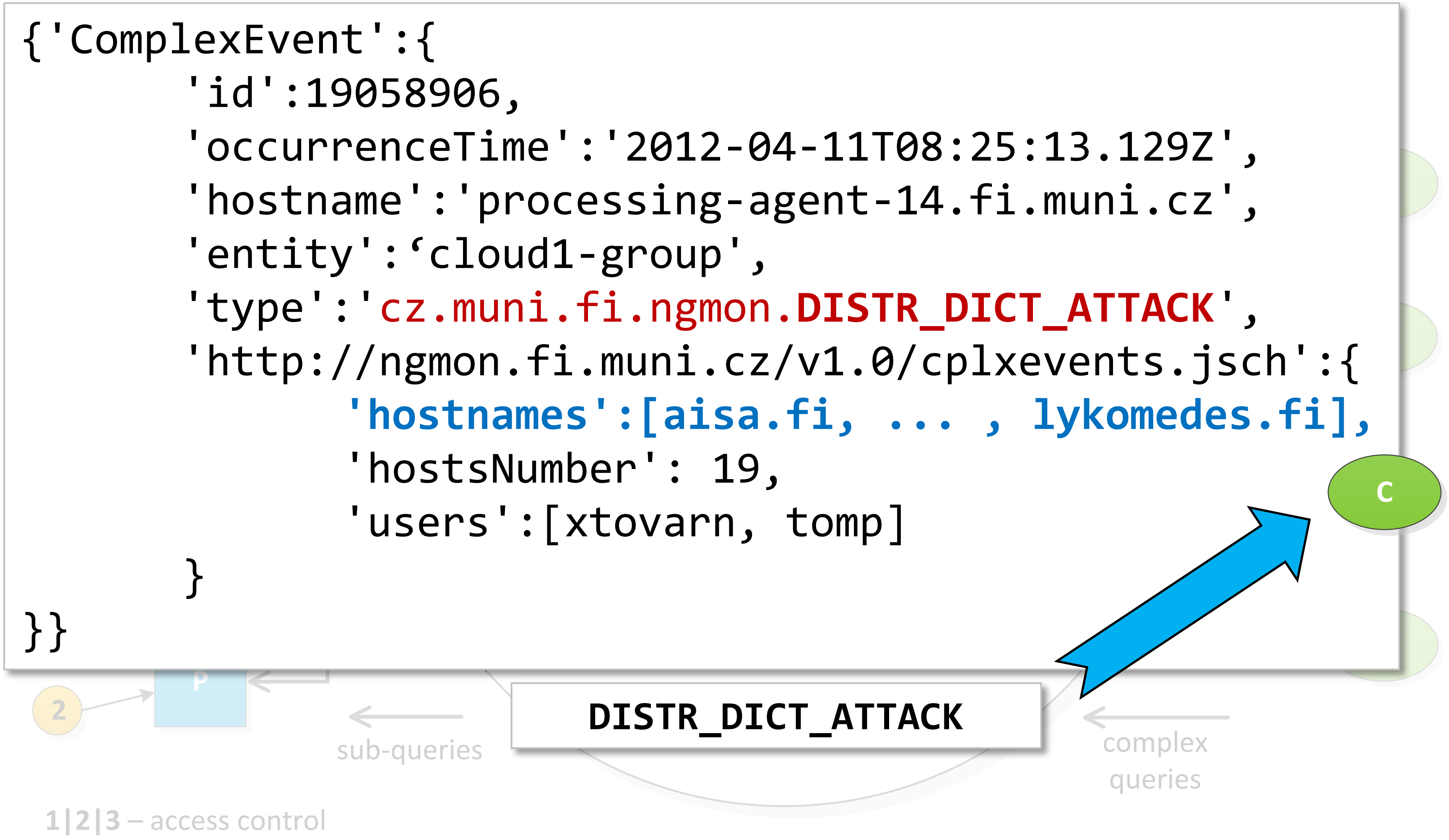
Distributed Event-driven Monitoring Model



Distributed Event-driven Monitoring Model



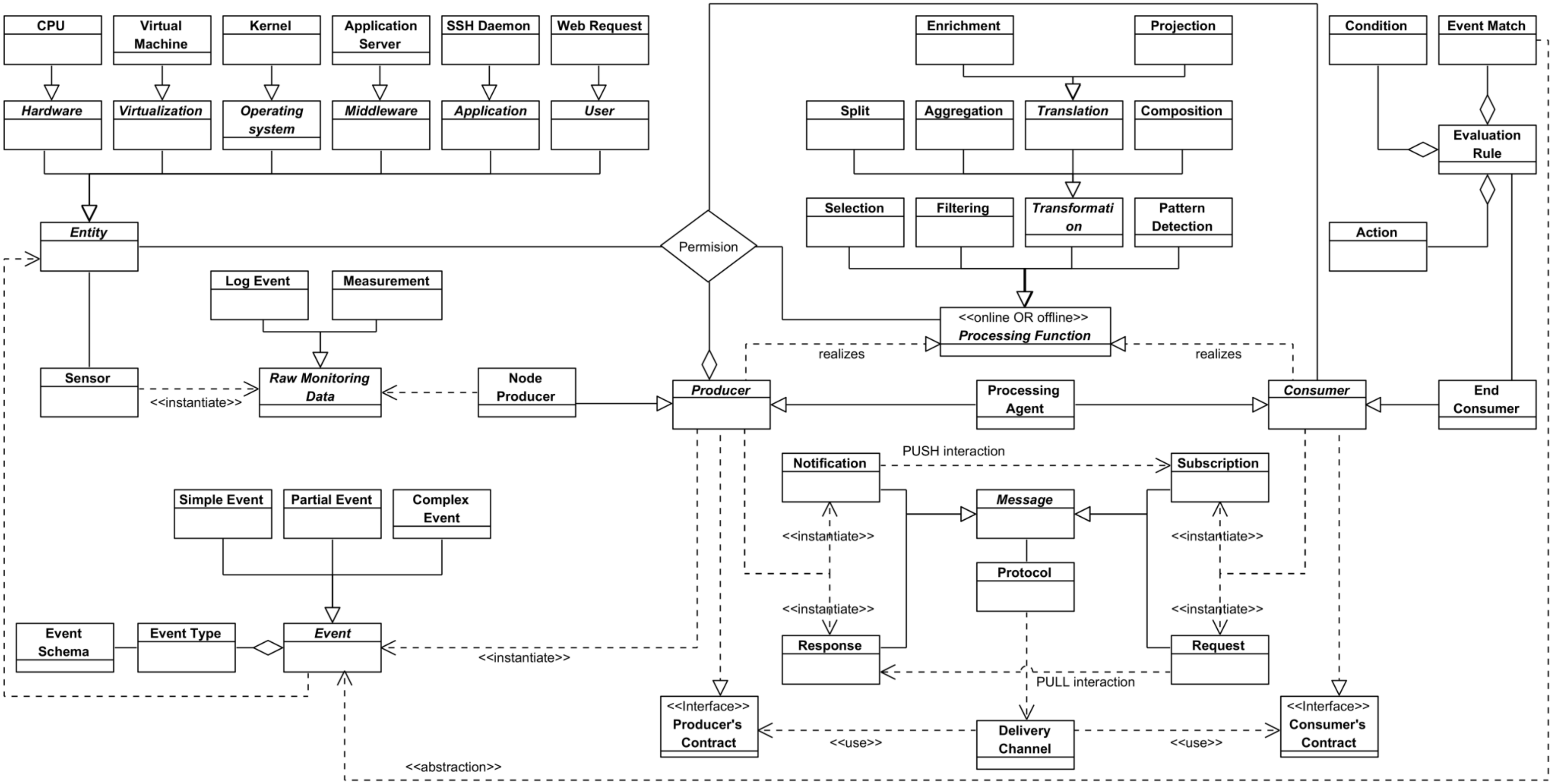
Distributed Event-driven Monitoring Model



```
{'ComplexEvent':{
        'id':19058906,
        'occurrenceTime':'2012-04-11T08:25:13.129Z',
        'hostname':'processing-agent-14.fi.muni.cz',
        'entity':‘cloud1-group',
        'type':'cz.muni.fi.ngmon.DISTR_DICT_ATTACK',
        'http://ngmon.fi.muni.cz/v1.0/cplxevents.jsch':{
                'hostnames':[aisa.fi, ... , lykomedes.fi],
                'hostsNumber': 19,
                'users':[xtovarn, tomp]
        }
}}
```

1|2|3 – access control

Different representation of the model

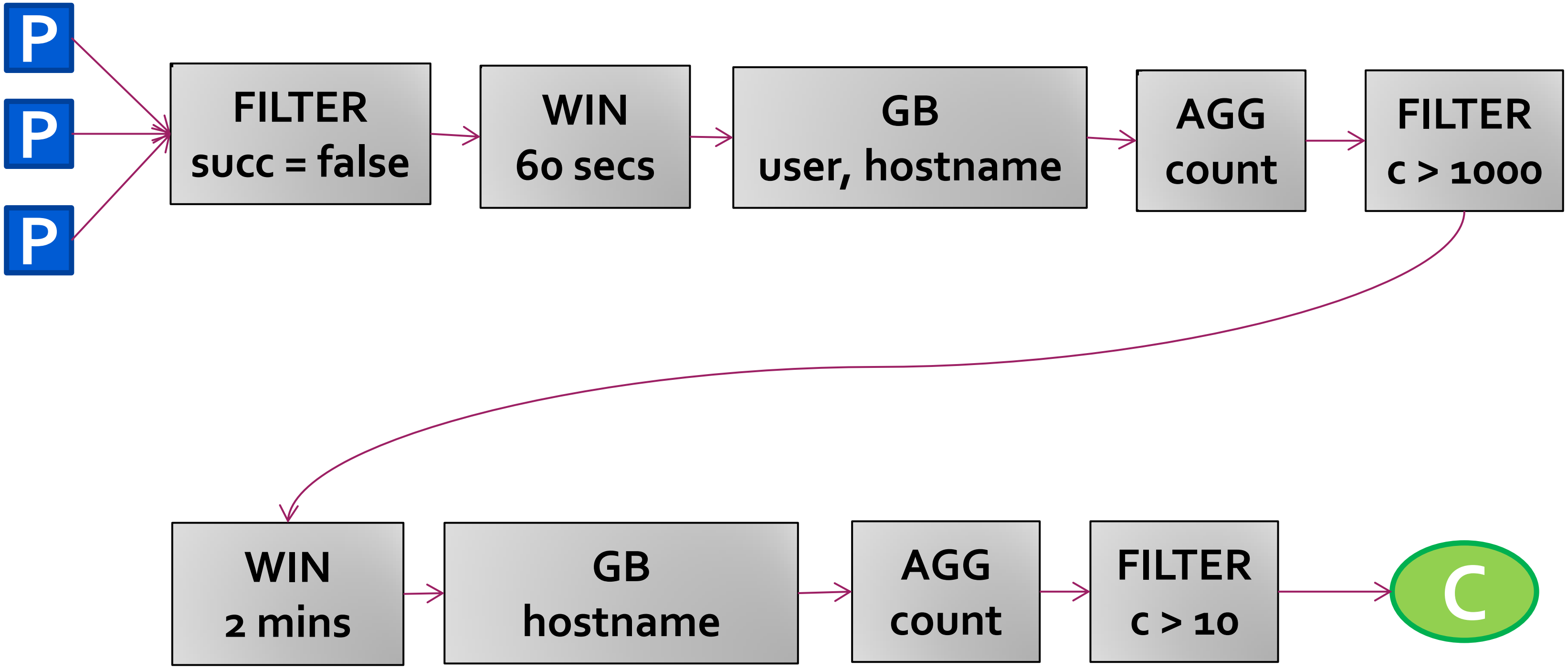


Event Processing Agents

- Processing agent performs one or more **processing functions -- operators**

- Filter
- Time window
  - sliding-tuple, sliding, tumble
- Aggregation (+ group by)
  - sum, count, stdev, min, max
- Sequence detection
- Multi-way JOIN

Box-And-Arrows Queries



P

P

P

FILTER
succ = false

WIN
60 secs

GB
user, hostname

AGG
count

FILTER
c > 1000

WIN
2 mins

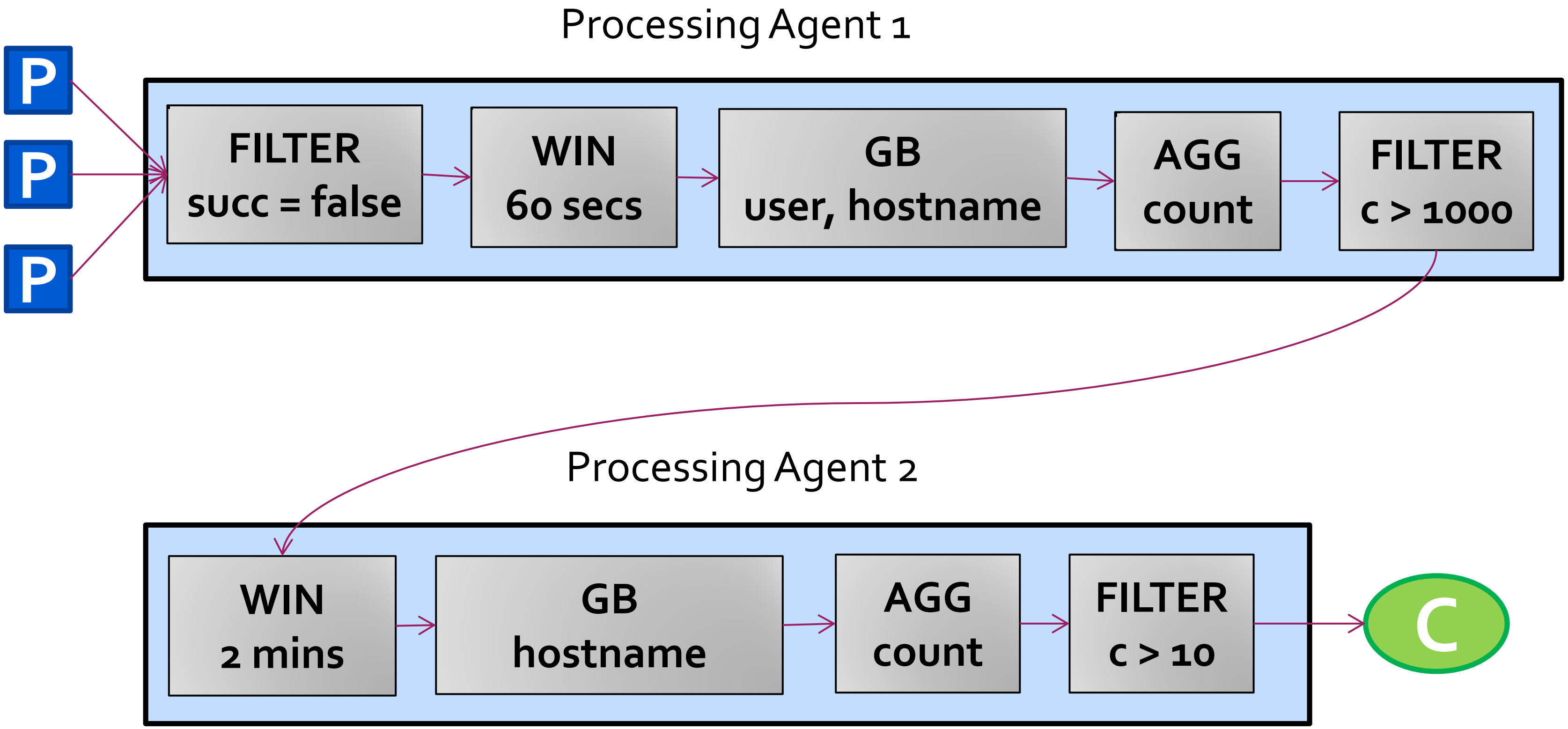
GB
hostname

AGG
count

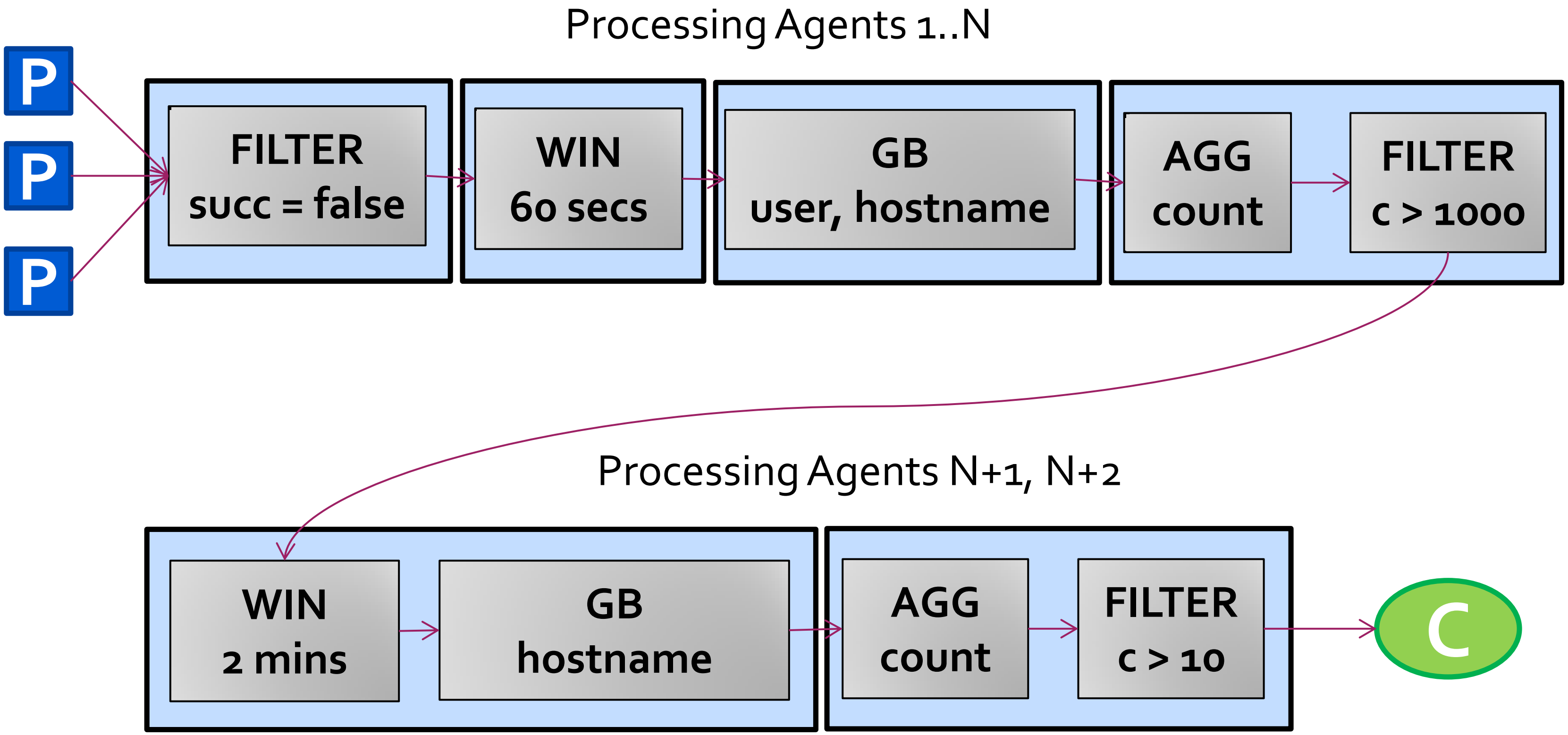
FILTER
c > 10

C

Box-And-Arrows Queries



Box-And-Arrows Queries



Processing Agents 1..N

P

P

P

FILTER
succ = false

WIN
60 secs

GB
user, hostname

AGG
count

FILTER
c > 1000

Processing Agents N+1, N+2

WIN
2 mins

GB
hostname

AGG
count

FILTER
c > 10

C

Models

- Event Processing Algebra
  - simple EP operator algebra
  - time and space complexity of each operator

- Distributed monitoring (meta?)model (static, dyn.)
  - best operators distribution
    - (w.r.t. available nodes, bandwidth, ever)
  - latency (minimize)
  - throughput (maximize)

- What data (from where) are needed to detect the pattern?
  - which producers, what events?

Prototype Implementation – Current state

- Prototype of distributed variant (simple static deployment with known patterns)
  - as the number of *monitored* nodes grows, new *monitoring* nodes can be added – almost linear scalability

- Typical CEP engine is able to process 50k-100k events per second

- Distributed engine/algorithm under development
  - Lightweigth engine (limited set of operators for monitoring)
  - Erlang is used – scalability, reliability, robustness

Summary - DEDMM

- Our goal is **behavior monitoring** of many distributed producers in real-time

- The model introduces paradigm shift towards **online** data processing utilizing complex event processing and detection

- We aim at **fully-distributed** event processing

Extension to Smart Grid

- **Considerable volumes** of data produced by relatively static set of producers

- **Moderate variability** of monitoring data
  - primarily measurements

- **Unreliable** and **slow** communication channels
  - GPRS (EDGE)

Simulation environment for Smart Grid

- Joint collaboration of Mycroft Mind, CERIT-SC MU, ČEZ, and Lasaris FI MU
- 3,500,000 smart meters simulated in CERIT Cloud (unique project in Europe)
- Several concepts presented today were used for the simulation environment monitoring