Timestamped Event Stream System

Chronicle Recognition System

Christophe Dousson

orange™

& france telecom

Outline

Architecture

- interface producer vs. consumer

Feedback : control by consumers

- messages of the control protocol

Chronicles

- focusing when computing correlation rules

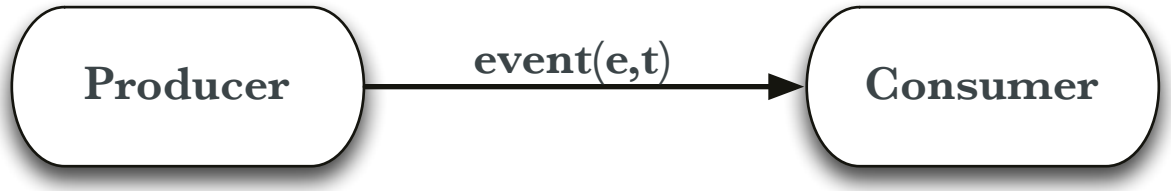
1

Architecture

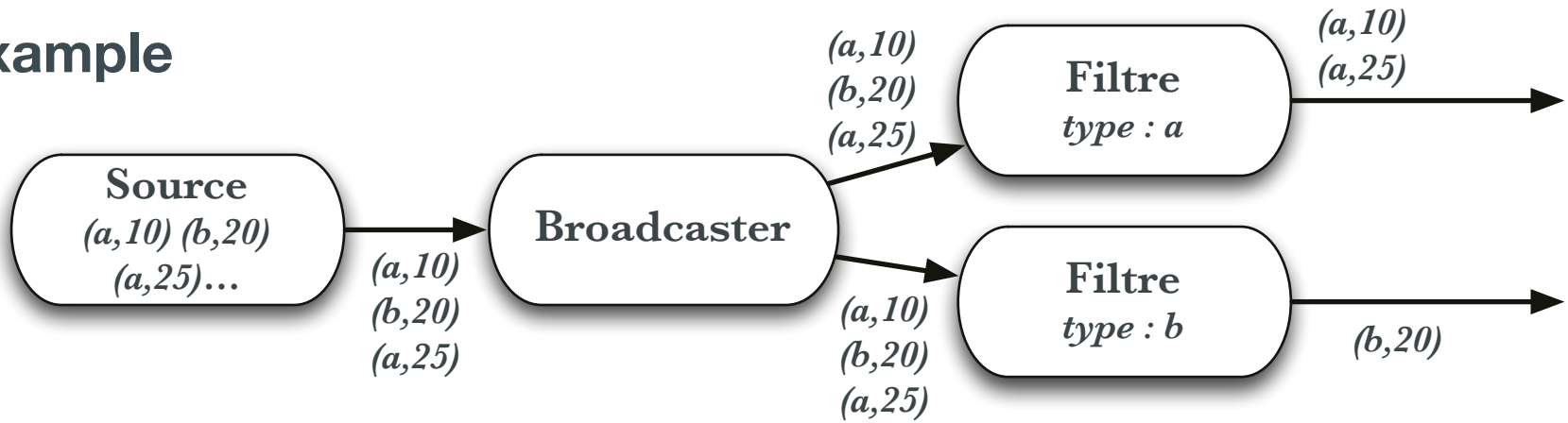
Producer/Consumer

Define the way of event transmission

- contract: an event **(e,t)** is sent only one time
  - *notice: distinct dates mean different events*



Example



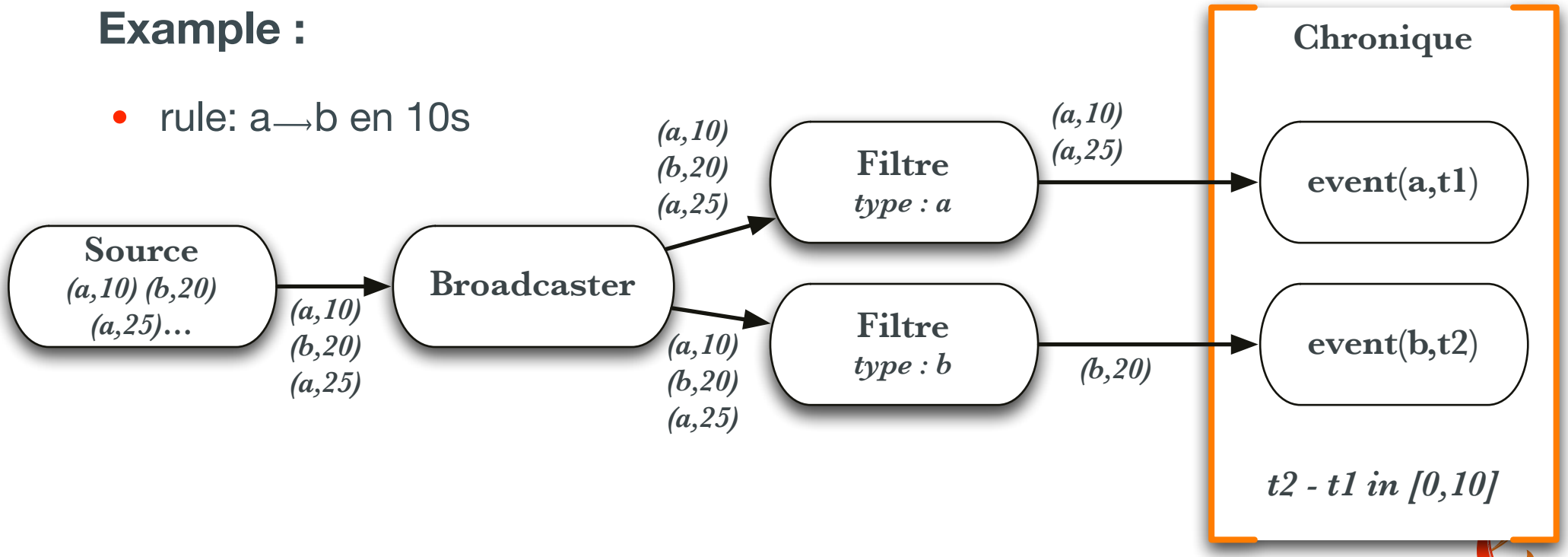
Application to the correlation of alarms

The correlation rule processing is based on

- a set of consumers (one per event of the rule)
- correlation information (e.g. the time constraint)

Example :

- rule: a⟶b en 10s

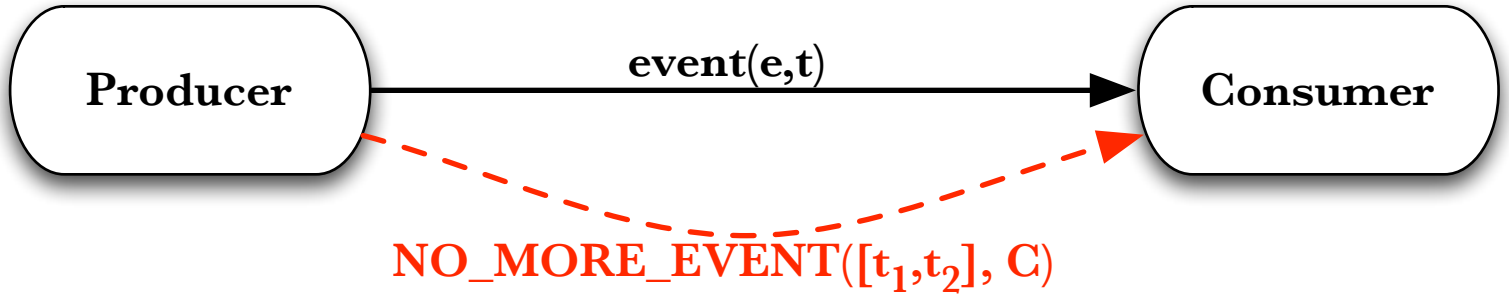


Control/sync. of observations

First control message : to sync. the streams

*contract: the producer will never send an event which satisfies C **and** with an occurrence date within TW*

- *This message could consider as a closure control message.*

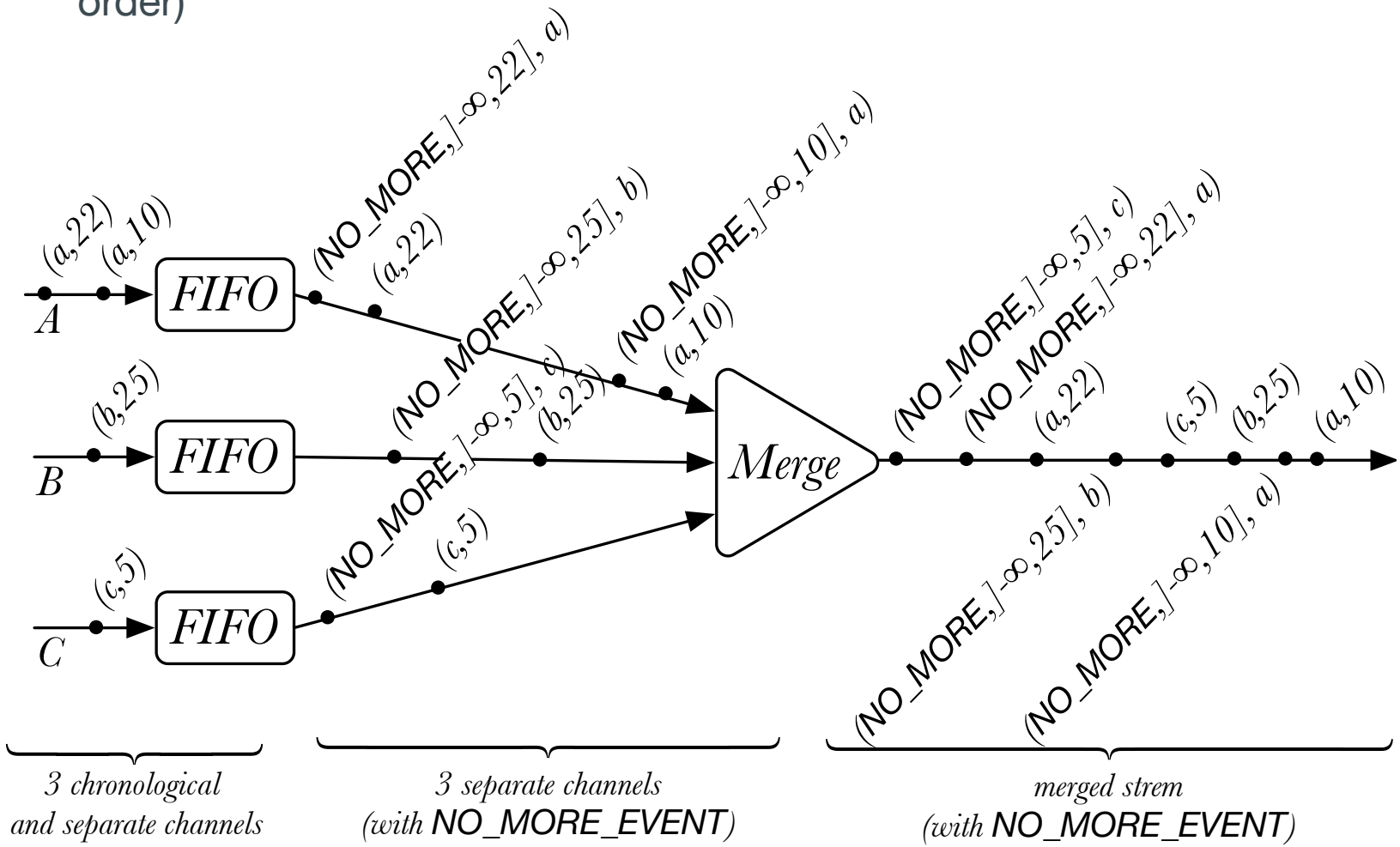


- **Advantage**: event transmission policy could be non chronological
  - anm([0,10]); … anm([100,200]);… anm([0,50])

NO_MORE_EVENT Usage (1/2): Multiple Sources

Example of user : streams joint

- hyp. each stream is FIFO (i.e. each source sends events in a chronological order)



NO_MORE_EVENT Usage (2/2): Deduced Events

Example:

- chronicle C
  - event(a,t) ∧ event(b,t') ∧ (t-t' in [90,120]) ➢ (c, t')

- Receiving (a,10)
  - a correlation context starts with t=10 and t'∈ [100,130]

- Receiving assertNoMore(]-∞, 50])
  - the previous context remains valid (and always waits for b)
  - **nevertheless we already be able to deduce assertNoMore(W, type:c)**

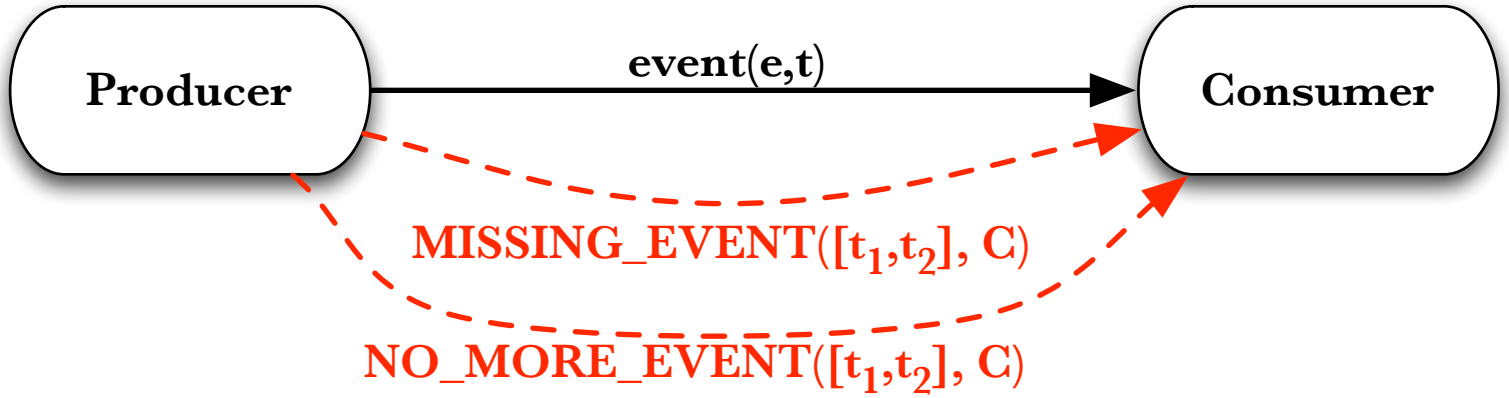
    with W = (]-∞,50]+90)\[100,130] =]-∞,100[∪]130,140]

Missed Observations

Control message: MISSING_EVENT

meaning: the producer indicates that some events may be missed (according to the specified condition and time window)

- It could be due to a sensor pb (uncertainty) or to a lost of events (full buffer) or to a action of the producer (for algorithmic reasons, …)



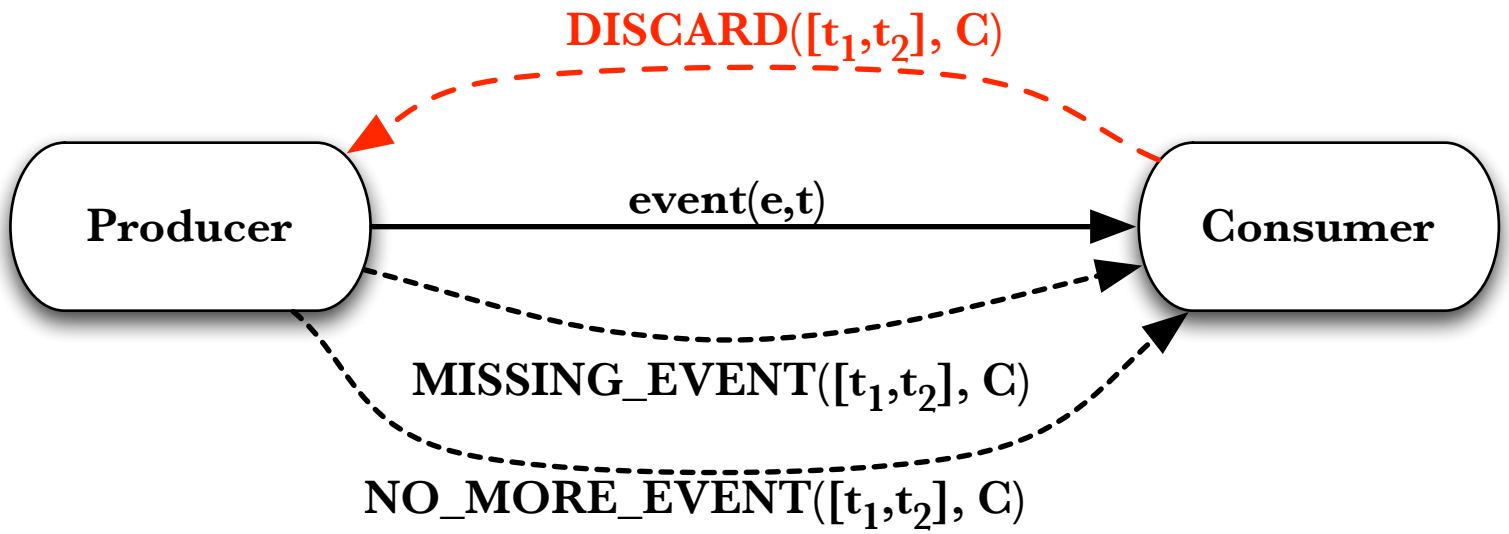
2

Control by the Consumer

(feedback)

Obsolete Events

The consumer indicates that it does not need some events

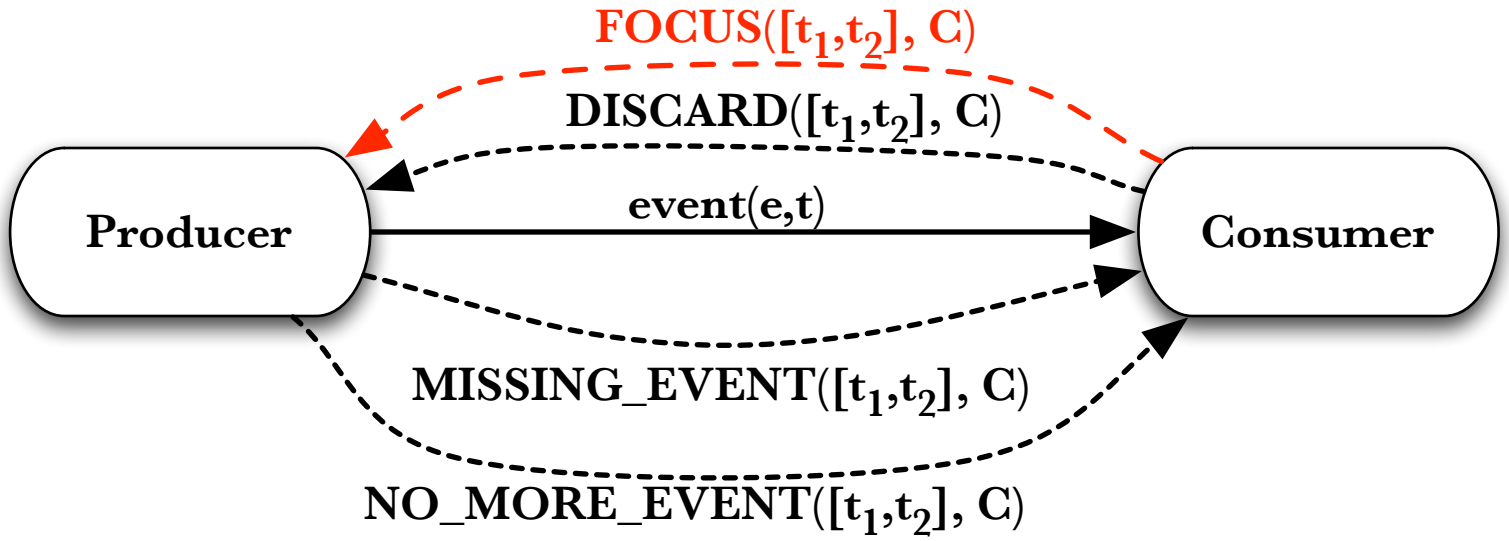


- *Note*: the producer could always send the events (i.e. ignore the DISCARD)
- *A contrario*, if the producer filters the events, it could inform the consumer with an appropriate NO_MORE_EVENT
- If the consumer retracts (REMOVE_DISCARD), a MISSING_EVENT message could be sent by the producer.

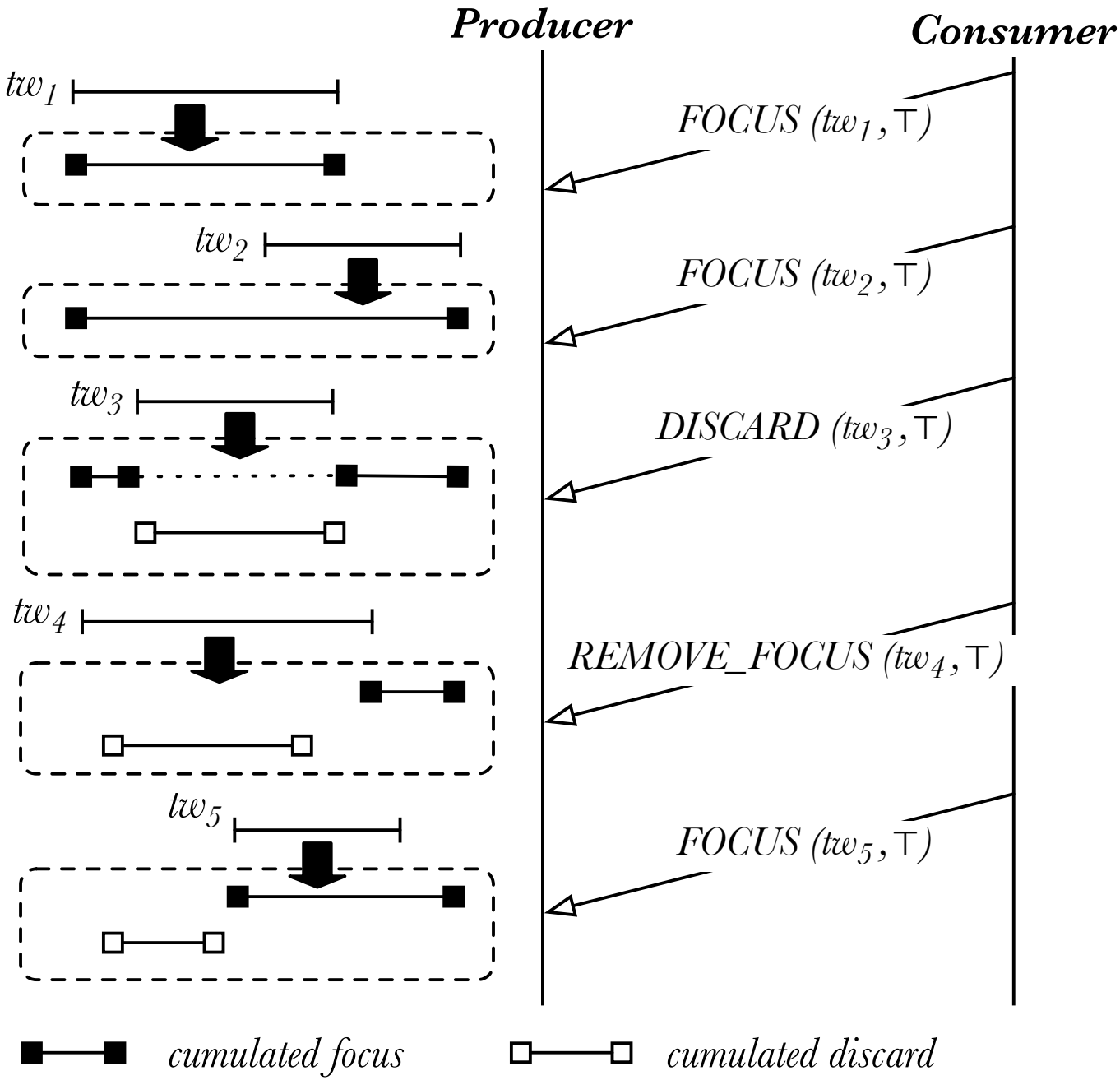
Focusing

The consumer indicates that some events are “urgent"

*contract: the producer should send the events **as soon as possible***

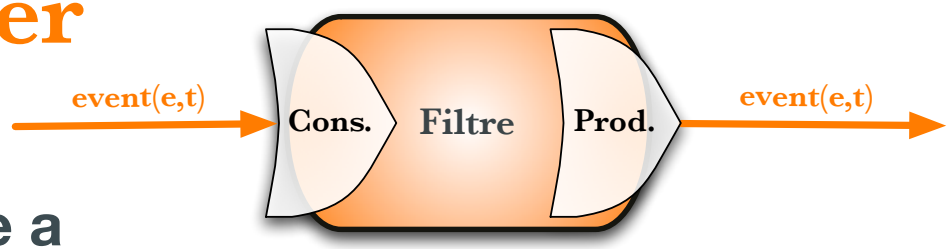


- A REMOVE_FOCUS message could cancel a focus
- Note: by default, a producer could delay its events as it desired if (and only if) it does not receive a FOCUS.



Behavior when Cumulating FOCUS & DISCARD

Example: impl. of a filter

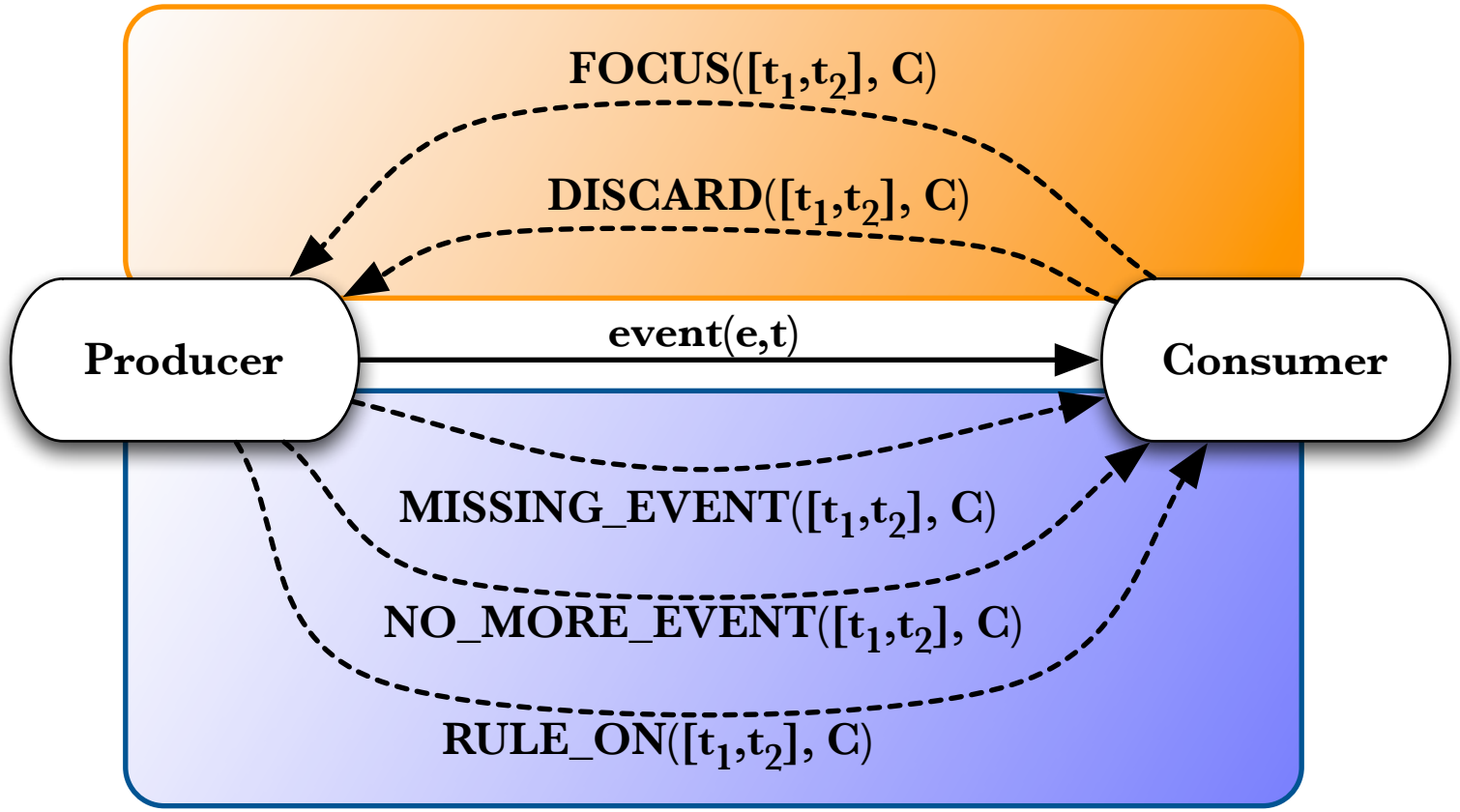


This is a component which compute a filtering window (union of received DISCARD) and removes all the events which match this time window.

- when a filter received a DISCARD(TW):
  - update: FILTER ← FILTER ∪ TW
  - send a NO_MORE_EVENT(TW) downstream and DISCARD(TW) upstream
- when a filter received a event (e,t):
  - if t ∉ FILTER, send (e,t) downstream (if not, do nothing)
- when a filter received NO_MORE_EVENT(TW):
  - if TW⊄FILTER, send NO_MORE_EVENT(TW) downstream
- when a filter received FOCUS(TW):
  - if TW⊄FILTER, send FOCUS(TW∩FILTRE) upstream

Summary

All control messages are <TYPE> (TimeWindow, Condition)



- *Note : the REMOVE_DISCARD and REMOVE FOCUS messages are not represented here (but they still exist)*

3

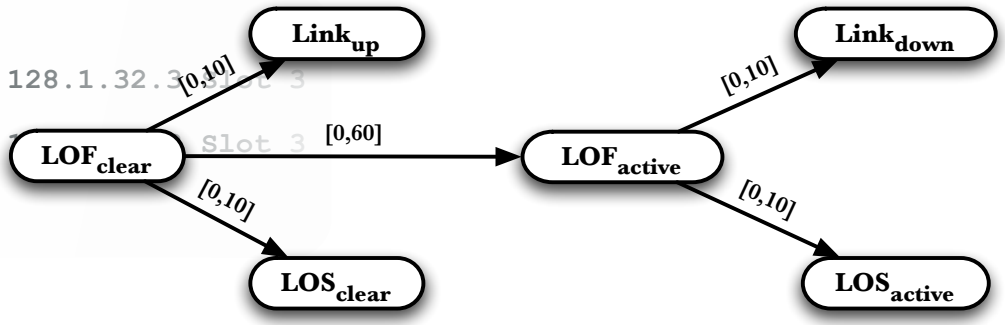
Chronicles

Temporal Correlation

```
21/09/2001 17:36:32 lyon-2 Slot 2 Link 1 has gone down
21/09/2001 17:36:33 clermont Slot 1 Link 0 Path Yellow Alarm Active
21/09/2001 17:36:33 clermont Slot 1 Link 0 Path Label Mismatch Alarm Active
21/09/2001 17:36:34 clermont Slot 1 Link 0 has gone down
21/09/2001 17:36:34 grenoble Slot 5 Link 0 Path Yellow Alarm Active
21/09/2001 17:36:34 svcQsaalUpDown trap received from 128.1.16.2 Slot 2
21/09/2001 17:36:35 grenoble Slot 5 Link 0 has gone down
21/09/2001 17:36:36 grenoble Slot 5 Link 0 Path AIS Alarm Active
21/09/2001 17:36:36 svcQsaalUpDown trap received from 128.1.29.5 Slot 5
21/09/2001 17:37:42 sta-4 Slot 11 Link 0 Loss of Frame Alarm Clear
21/09/2001 17:37:43 sta-4 Slot 11 Link 0 Loss of Signal Alarm Clear
21/09/2001 17:37:51 svcQsaalUpDown trap received from 128.1.26.1 Slot 1
21/09/2001 17:37:55 svcQsaalUpDown trap received from 128.1.4.5 Slot 5
21/09/2001 17:37:57 sta-4 Slot 11 Link 0 has come up
21/09/2001 17:38:05 sta-4 Slot 11 Link 0 Loss of Frame Alarm Active
21/09/2001 17:38:26 sta-4 Slot 11 Link 0 has gone down
21/09/2001 17:38:36 grenoble Slot 5 Link 0 Path AIS Alarm Clear
21/09/2001 17:38:52 sta-4 Slot 11 Link 0 Loss of Signal Alarm Active
21/09/2001 17:42:11 lyon-2 Slot 1 has gone down
21/09/2001 17:42:30 lyon-2 Slot 4 has gone down
21/09/2001 17:42:30 lyon-2 Slot 3 has gone down
21/09/2001 17:42:39 svcQsaalUpDown trap received from 128.1.4.5 Slot 5
21/09/2001 17:42:43 grenoble Slot 5 Link 0 Path Yellow Alarm Clear
21/09/2001 17:42:44 grenoble Slot 5 Link 0 has come up
21/09/2001 17:43:03 lyon-2 Slot 9 has gone down
21/09/2001 17:43:09 lyon-2 Slot 3 has come up
21/09/2001 17:43:09 svcQsaalUpDown trap received from 128.1.32.3 Slot 3
21/09/2001 17:43:20 lyon-2 Slot 0 has gone down
21/09/2001 17:43:21 svcQsaalUpDown trap received from [illegible] Slot 3
```

If receiving LOFclear and then LOSclear and LinkUp within 10 seconds, and…

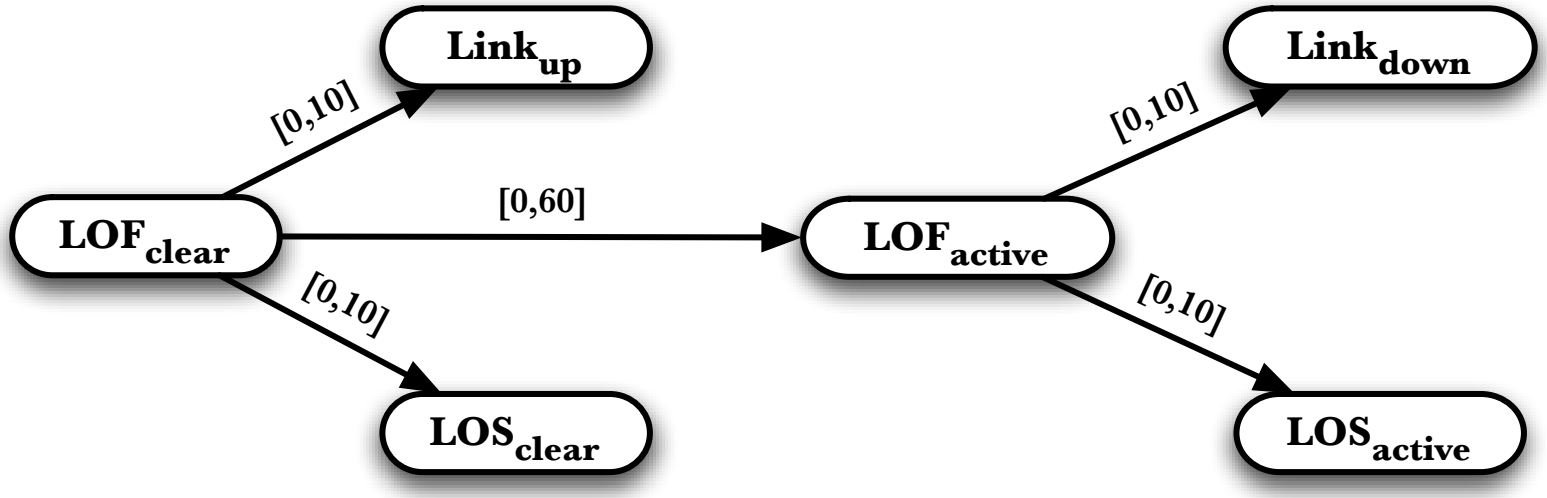
(we called a such temporal rule a ***chronicle***)



Temporal Correlation Rules

A such rule correlates a subset of events

- it is defined as a ***set of patterns*** of observable events…
- … with ***temporal constraints*** between their ***occurrence dates***
- example :



Temporal Correlation Formalism (basics)

Occurrence of event

- `event( M[196.3.2.4,?x,low], t)`

the “event” to wait for

occurrence date

Absence of event

- `noevent( M[102.0.0.1,?x,*], (t1, t2) )`

the “event” to reject

lower bound

upper bound

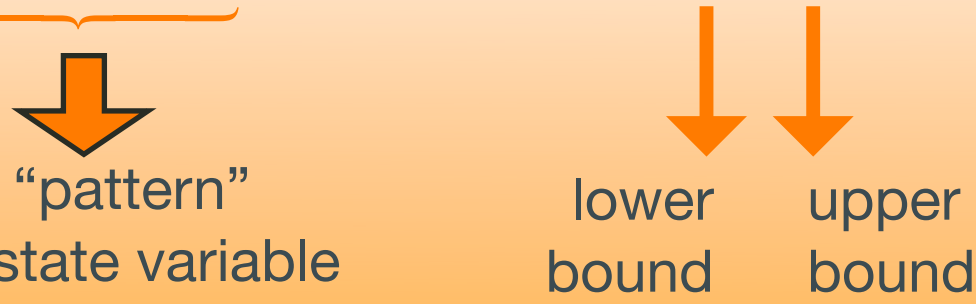
Temporal Correlation Formalism (more)

Change of state variable

- `event( Severity[123,?x]:(low,high), t)`

Persistency of state

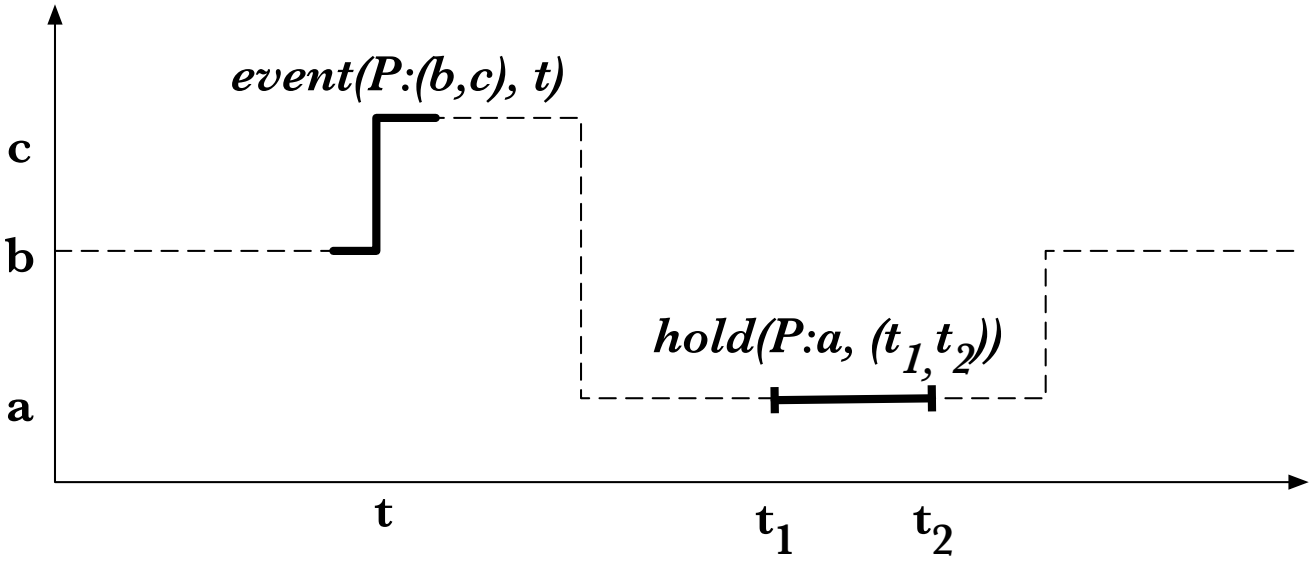
- `hold(Severity[123,?x]:low, (t1, t2) )`



Temporal Correlation Formalism (more)

Change of state variable

- `event( Severity[123,?x]:(low,high), t)`



the "pattern" of the state variable

lower bound

upper bound

An Example

Scenario : the battery becomes critical which is followed within 2 to 5 units of time by a loss of connectivity (good to bad) and a high level of IP packet loss less than 10 time units after the beginning. All of this scenario should be considered only for a WiFi connection.

```
chronicle DetectHandoverRequired_WIFI {
        event(SOC:(?,critical),t1)
```

```
}
```


An Example

Scenario : the battery becomes critical which is followed within 2 to 5 units of time by a loss of connectivity (good to bad) and a high level of IP packet loss less than 10 time units after the beginning. All of this scenario should be considered only for a WiFi connection.

```
chronicle DetectHandoverRequired_WIFI {
        event(SOC:(?,critical),t1)


        t2 - t1 in [2,5]


}
```

An Example

Scenario : the battery becomes critical which is followed within 2 to 5 units of time by a loss of connectivity (good to bad) and a high level of IP packet loss less than 10 time units after the beginning. All of this scenario should be considered only for a WiFi connection.

```
chronicle DetectHandoverRequired_WIFI {
        event(SOC:(?,critical),t1)
        event(link:(good,bad),t2)

        t2 - t1 in [2,5]

}
```

An Example

Scenario : the battery becomes critical which is followed within 2 to 5 units of time by a loss of connectivity (good to bad) and a high level of IP packet loss less than 10 time units after the beginning. All of this scenario should be considered only for a WiFi connection.

```
chronicle DetectHandoverRequired_WIFI {
        event(SOC:(?,critical),t1)
        event(link:(good,bad),t2)
        event(IPloss:(?,high),t3)

        t2 - t1 in [2,5]

    }
```

An Example

Scenario : the battery becomes critical which is followed within 2 to 5 units of time by a loss of connectivity (good to bad) and a high level of IP packet loss less than 10 time units after the beginning. All of this scenario should be considered only for a WiFi connection.

```
chronicle DetectHandoverRequired_WIFI {
        event(SOC:(?,critical),t1)
        event(link:(good,bad),t2)
        event(IPloss:(?,high),t3)

        t2 - t1 in [2,5]

        t3 - t1 in [0,10]

    }
```

An Example

Scenario : the battery becomes critical which is followed within 2 to 5 units of time by a loss of connectivity (good to bad) and a high level of IP packet loss less than 10 time units after the beginning. All of this scenario should be considered only for a WiFi connection.

```
chronicle DetectHandoverRequired_WIFI {
        event(SOC:(?,critical),t1)
        event(link:(good,bad),t2)
        event(IPloss:(?,high),t3)
        hold(connection:WiFi,(t1,t4))
        t2 - t1 in [2,5]
        t2 < t4
        t3 - t1 in [0,10]
        t3 < t4
    }
```

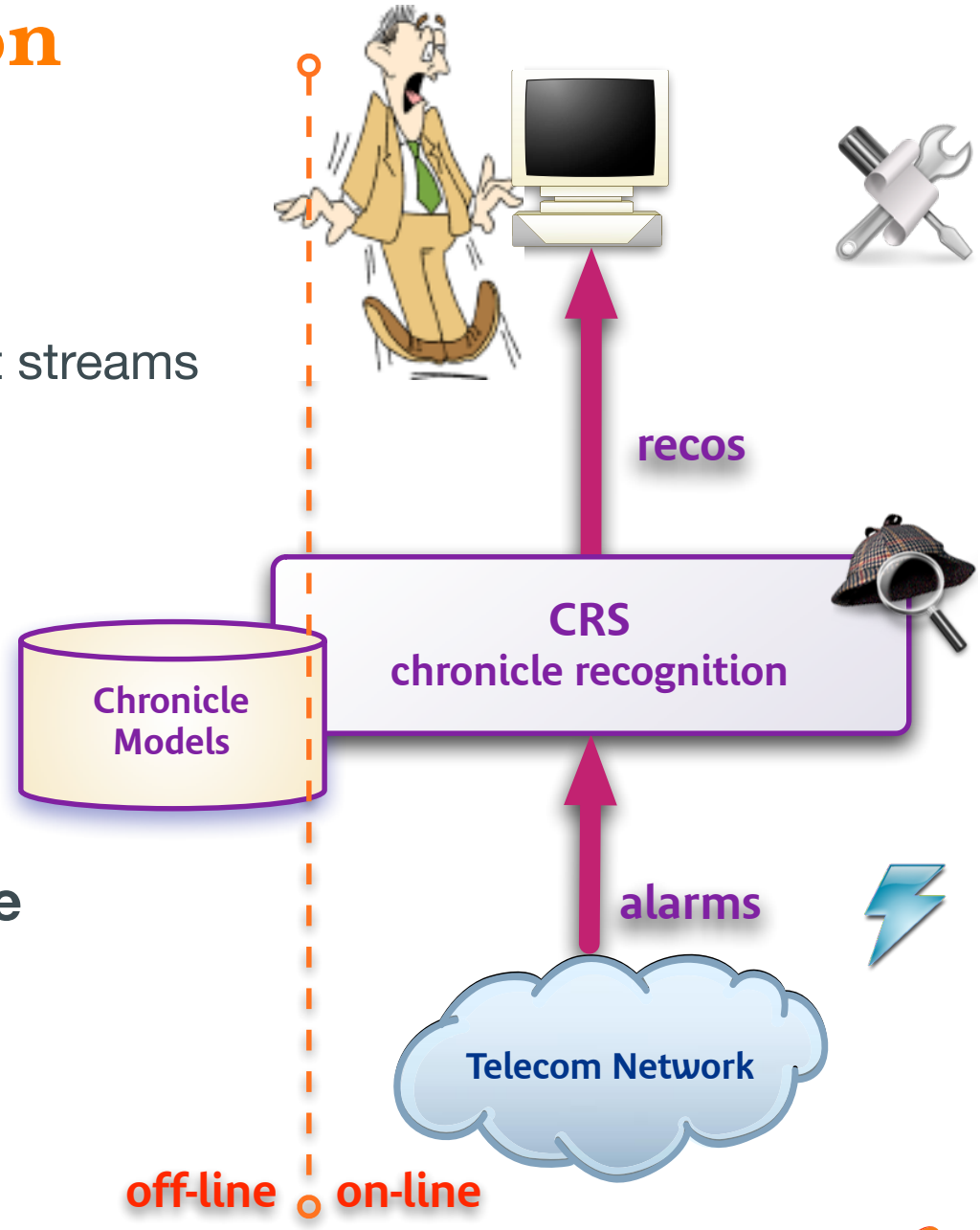
Chronicle Recognition

On-line Processing

- works on non-chronological event streams
- on the fly (one pass)
- 24h/24
- thousands evts/second

chronicle language is expressive

- event, noevent, time constraints
- non necessary complete model

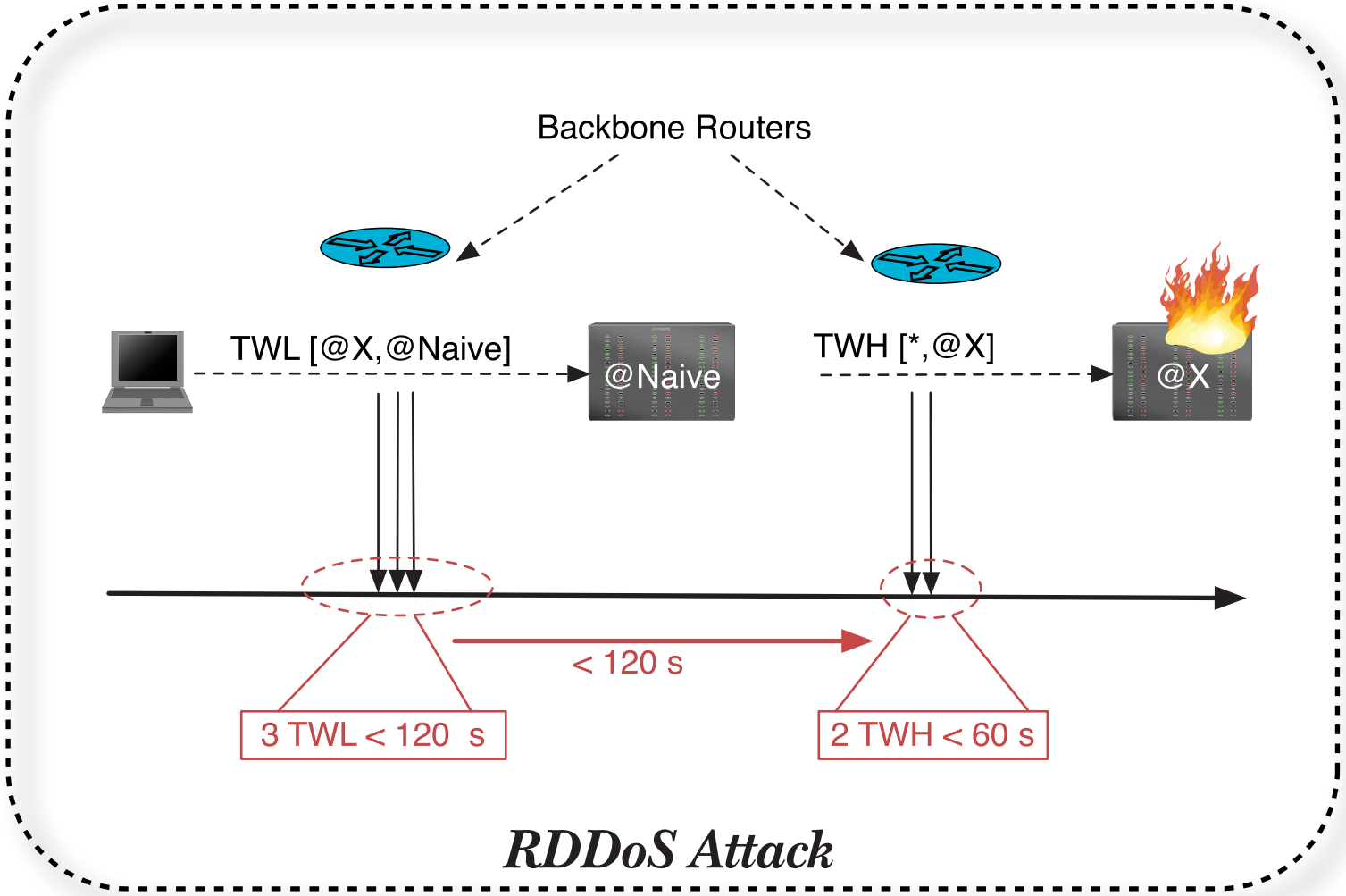


4

Usage for Correlation

(example)

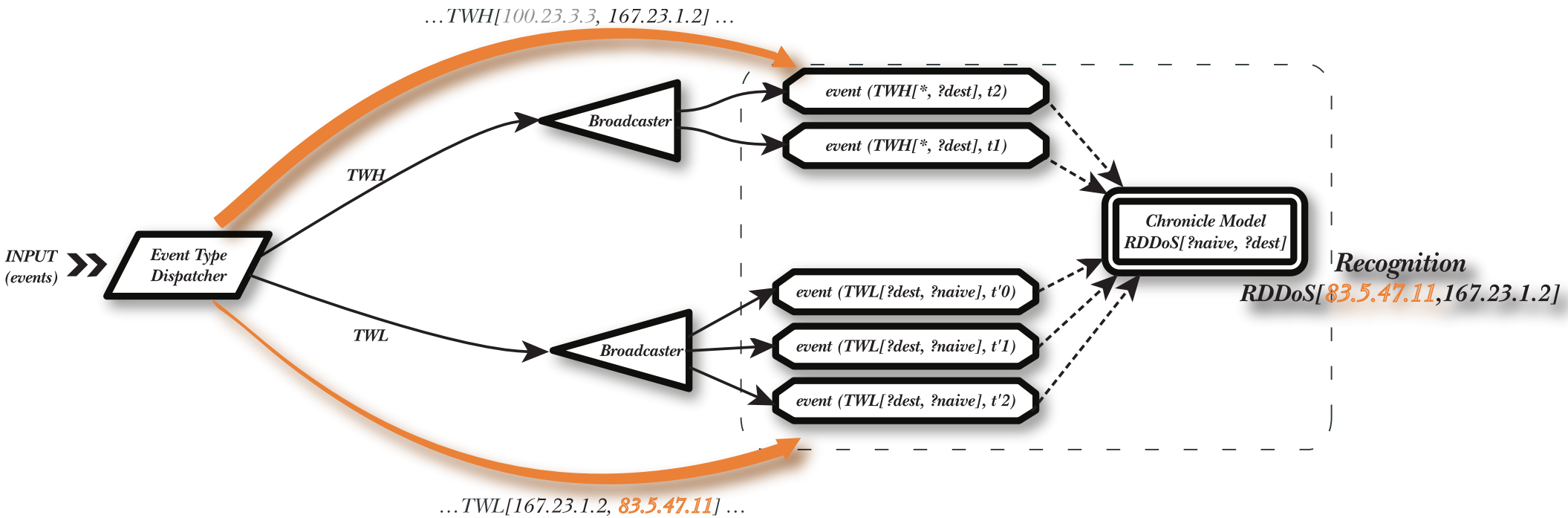
Attack detection (1/4)



Attack detection (2/4)

Classical behavior of correlation system

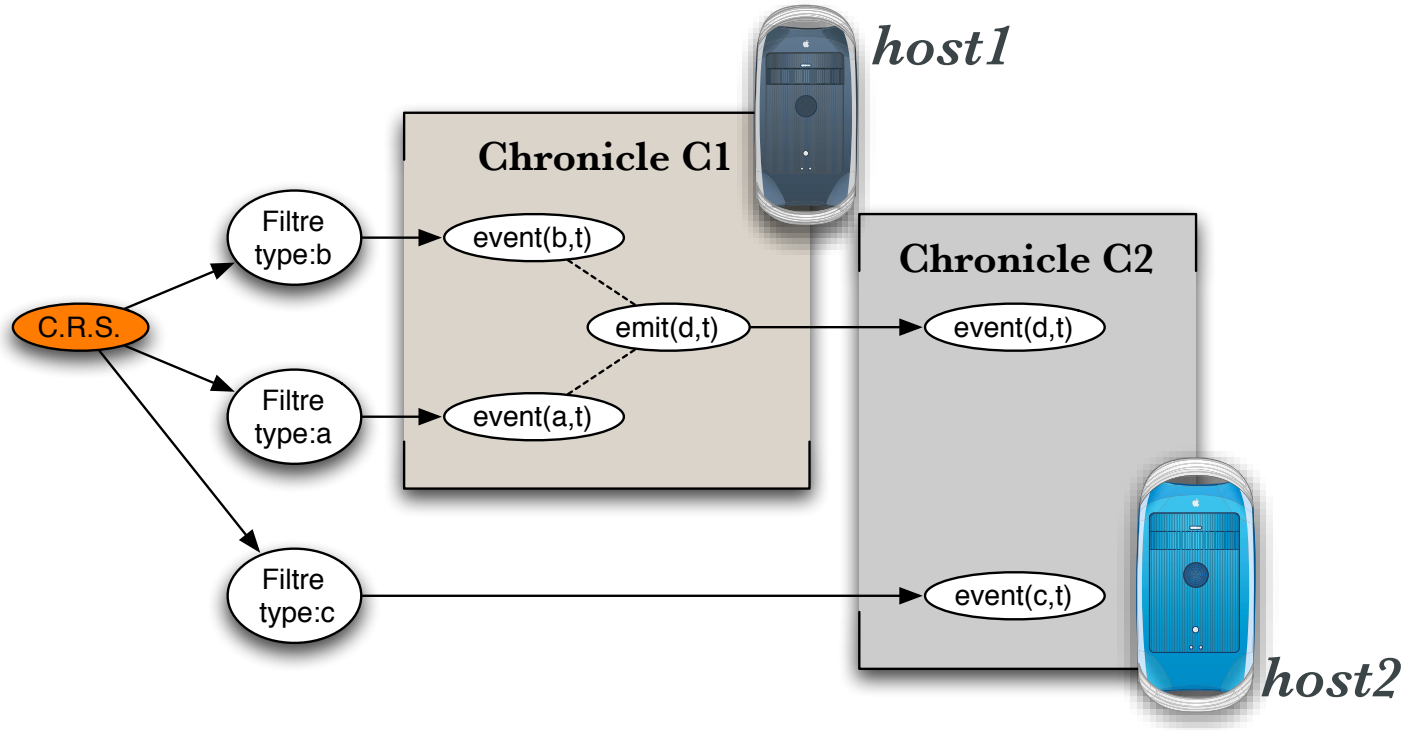
- events are integrated in their arrival order



Distributed processing

Principle

- Some temporal rules (chronicles) generates new event when recognized
- These chronicles are considered (and implemented) as producers



Conclusion

Modular architecture

- Addition of new components is easy: the processing is not only in the “code” but also in the structure (which could be automatically optimized and/or distributed)
- Impact on correlation tool (s.t. CRS): this is not a black box but a set of connected components → easy to distribute but also easy to modify the behavior (e.g. focusing)

Currently on progress…

- Hot-connection of components
- Multi-hosts components (at the moment, just RMI components available)
- On editing and debugging tools (graphical editing)
- On optimisation tool (of the structure)

Conclusion

Modular architecture

- Addition of new components is easy: the processing is not only in the “code” but also in the structure (which could be automatically optimized and/or distributed)
- Impact on correlation tool (s.t. CRS): this is not a black box but a set of connected components → easy to distribute but also easy to modify the behavior (e.g. focusing)

Currently on progress…

- Hot-connection of components ☺
- Multi-hosts components (at the moment, just RMI components available) ☹
- On editing and debugging tools (graphical editing) ☺
- On optimisation tool (of the structure) ☹


Thank
You !
Any question ?