Seminar on Modeling Concepts in Distributed Systems

Concepts and Models in Distributed Systems

Prof. Walter Kriha
HDM-Stuttgart

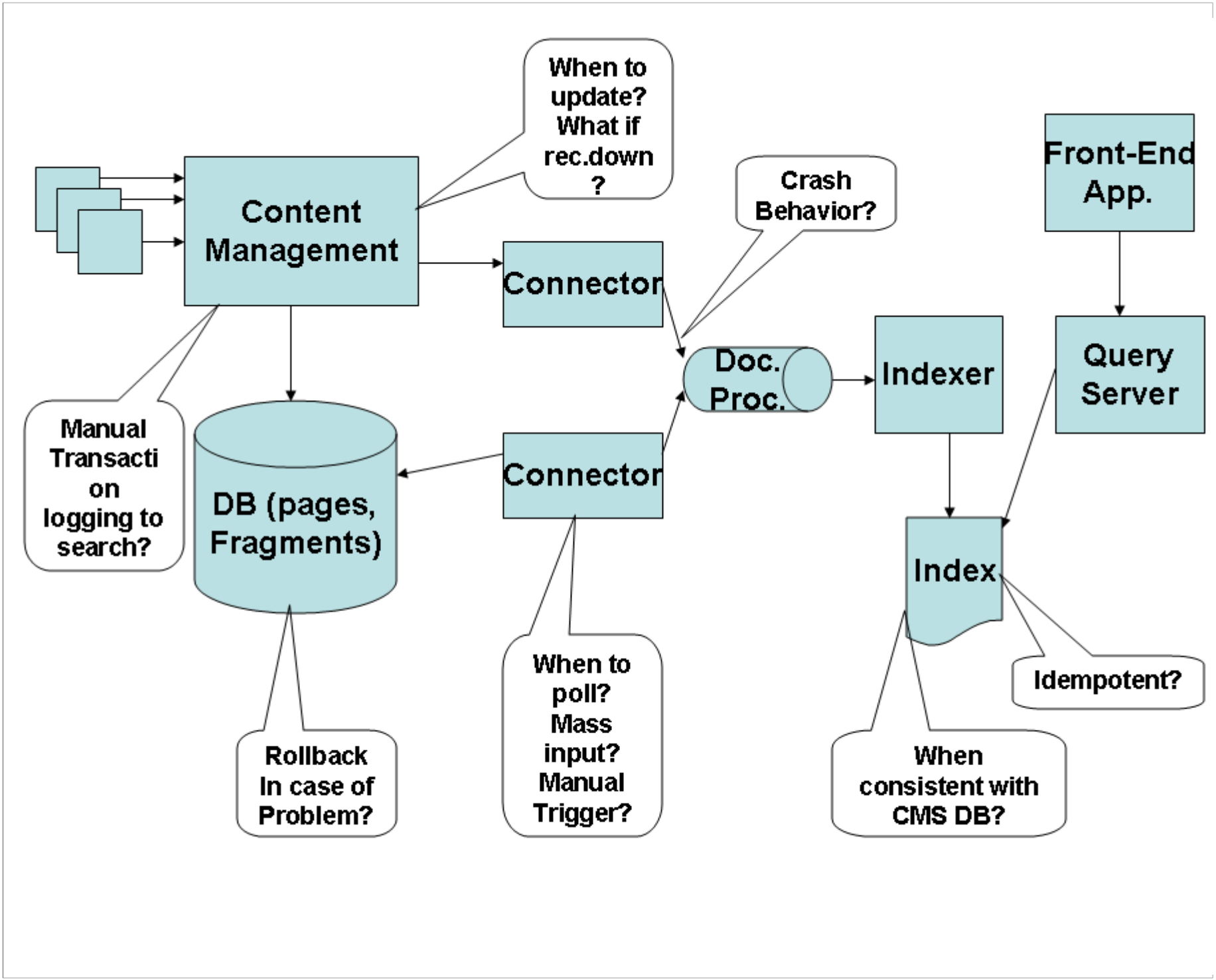
A practical Example: Connecting two Systems

The combination of a content management system and a search engine quickly shows a surprising number of unclear situations:

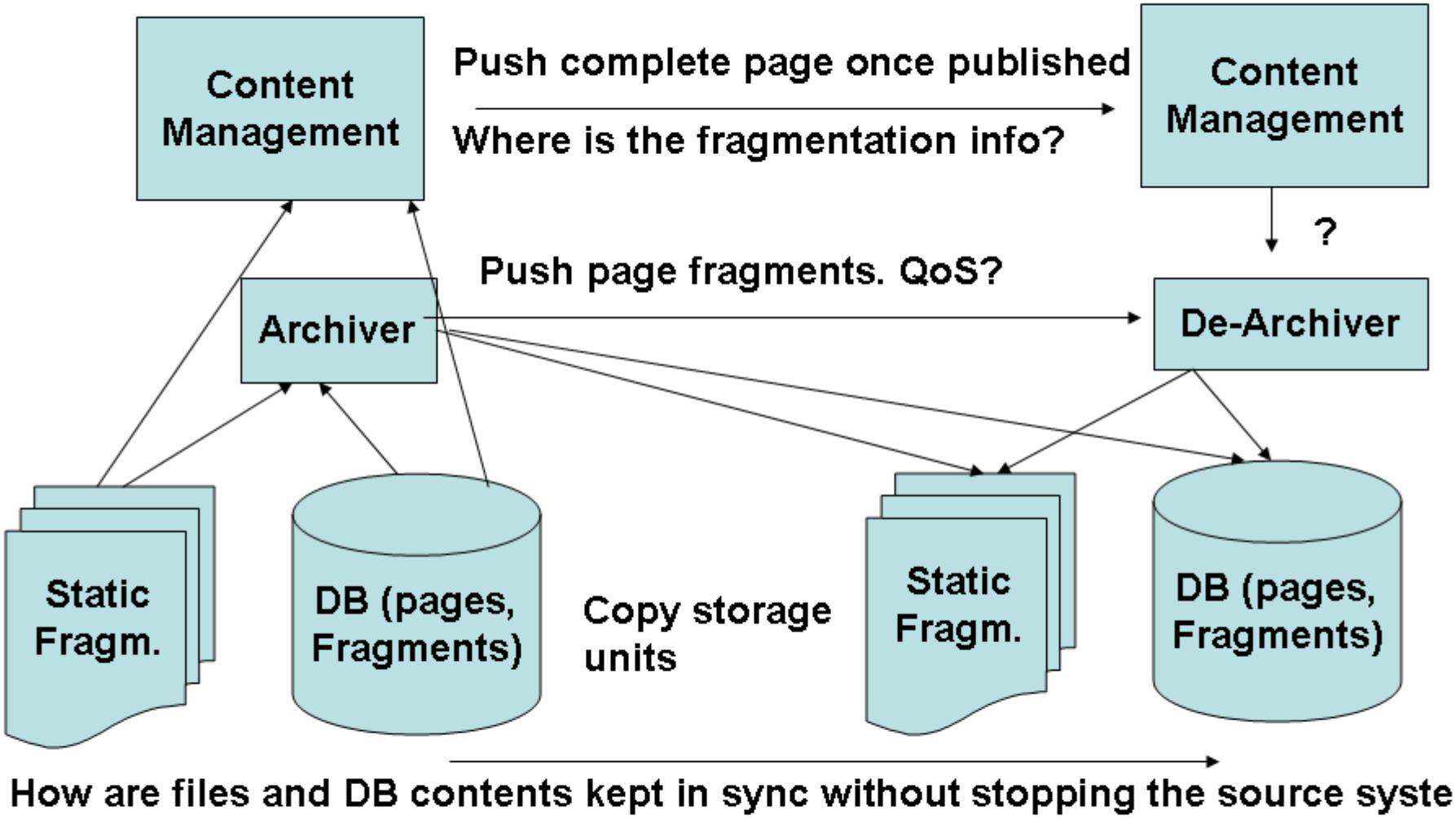
-The interfaces behave unclear in case of problems like overruns, partner-system failure etc.

- The whole system shows different phases (boot, communication/work, system failure(s) with recovery not defined.

The danger lies a) in lots of application level code needed to deal with these problems in an ad-hoc way and b) in the inability to determine the cause of errors quickly. A model should express basic delivery guarantees as well as timing based constraints. The use of specific middleware to deal with delivery problems makes the implementation much more reliable


When to update? What if rec.down ?
Content Management
Connector
Crash Behavior?
Front-End App.
Doc. Proc.
Indexer
Query Server
Manual Transacti on logging to search?
DB (pages, Fragments)
Connector
Index
Idempotent?
Rollback In case of Problem?
When to poll? Mass input? Manual Trigger?
When consistent with CMS DB?

Replication Levels



Replication of the complete page is useless because the page can no longer be disassembled and stored at the target CMS.

Only the archiver knows page fragments and where they should go. There is a transaction problem with file fragments because only the target DB supports transactions. An alternative would be to use a „de-archiver“ at the target but the TX problem persists.

Low level file and DB copy is critical because of changes during the copy process. This is not a transaction with respect to both sources. The advantage lies in the fact that the replication code does not need to know any CMS specifics.

Modeling Interaction

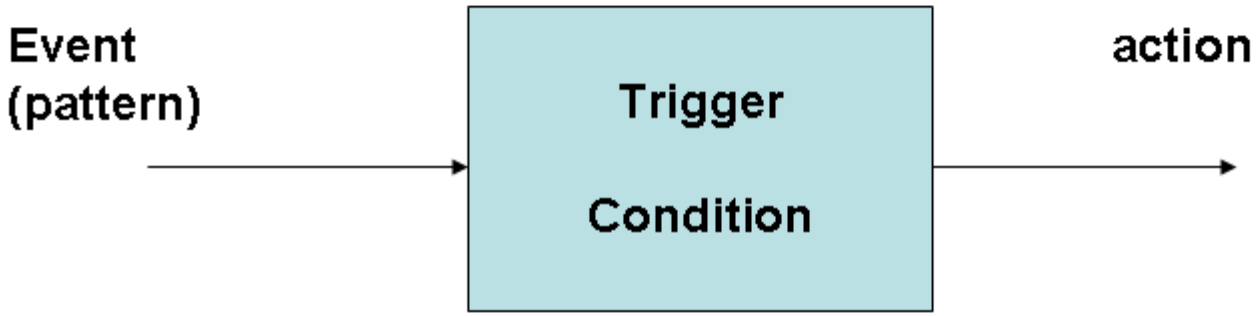
Interaction Models according to Mühl et.al.

		Consumer initiated	Producer initiated
Adressee	Direct	Request/Reply	callback
	Indirect	Anonymous Request/Reply	Event-based

Expecting an immediate „reply“ makes interaction logically synchronous – NOT the fact that the implementation might be done through a synchronous mechanism. This makes an architecture synchronous by implementation (like with naive implementations of the observer pattern).

Modeling Events

Trigger Example: Event-Condition-Action (ECA)



Leads automatically to a state machine like modeling of the problem domain!

Modeling of event-based Systems with traces

S0 (A0) -> S1 (A1) -> S2 (A2) -> S3 (A3) -> S4 (A4) …

Or with collapsed states and actions:

S0, S1, S2, S3, S4, …..

A trace is a sequence of states (see Mühl et.al. 25 ff). Properties of traces can be described with temporal logic. Even concurrent distributed processes can be described that way.

Composite Event Language

A language which allows the detection of:

- Individual events or their negation
- Event order (Concatenation, sequence, iteration, alternation
-Timed events
- Parallelization

See Mühl, pg. 238ff. Or Luckham

Modeling Time

Logical Time in Distributed Systems

Definition:

Logical time in distributed systems is defined by the transitive closure of all partially ordered internal event relations of all processes with the partially ordered external event relation.

If a is logical smaller than b then a comes before b or a is potentially causal for b.

(Birman 256ff)

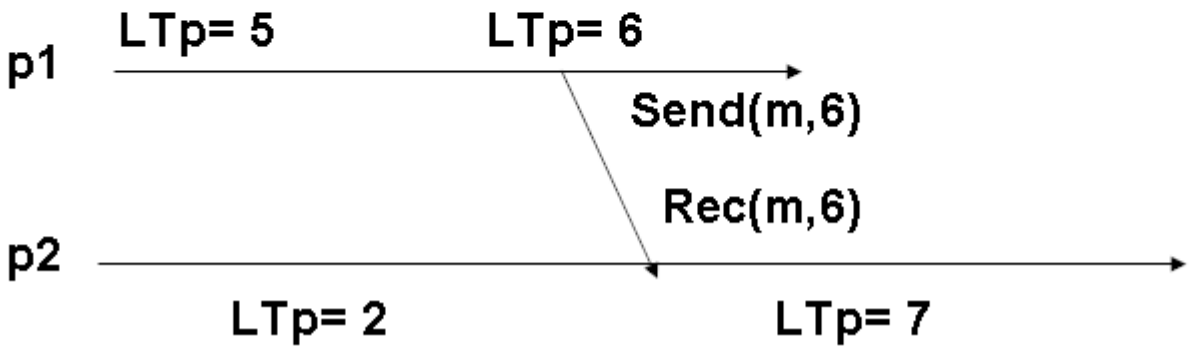
Logical (Lamport) Clocks

Let LTp be the value of the logical time for process p. Then

1. If (LTp < LTm) LTp = LTm +1 // take over „time“ from message plus one
2. If (LTp ≥ LTm) LTp = LTp +1 // increase internal „time“
3. Internal events cause: LTp = LTp + 1 // increase internal time per int.event

Processes count events using an internal counter. This counter is incremented and copied into every message sent. On receiving a message with such a counter value the internal counter is adjusted (increased) to reflect „later“. Applications are responsible for the granularity of events that „count“ for time.

Lamport Clock Example



Receiver p2 adjusts its logical clock for the value transmitted in the event plus 1. Simple logical clocks can show causality that is not really the case.

Vector Clocks

Vector clocks need to transmitt a vector of processes and their latest event time with each message. They can only be used for static membership protocols. (Birman 260ff)

Time and Order: Interval Timestamps

T = |Tlow; Thigh|

Total order: T1 << T2 if T1high < T2low

Partial order: T1 < T2 if (T1high < T2high) V (T1high = T2high AND T1low < T2low)

Formal Specification of event-based Systems with temporal logic

Operators:

-Always: a predicate or expression holds for all sub-traces or places within a trace

-Eventually: a predicate of expression will hold for at least one place in a subtrace or one subtrace

-Next: a predicate or expression holds for the second place in a subtrace or for the second subtrace

-Logical operators and quantifiers

-Atomic Predicates P

A specification is a set of traces. A system in compliance with a specification will only show behavior (traces) which are part of this set.

(see Mühl et.al. 25 ff).

Examples of specifications of event-based Systems with temporal logic

Operators:

-Always: a predicate or expression holds for all sub-traces or places within a trace

-Eventually: a predicate of expression will hold for at least one place in a subtrace or one subtrace

-Next: a predicate or expression holds for the second place in a subtrace or for the second subtrace

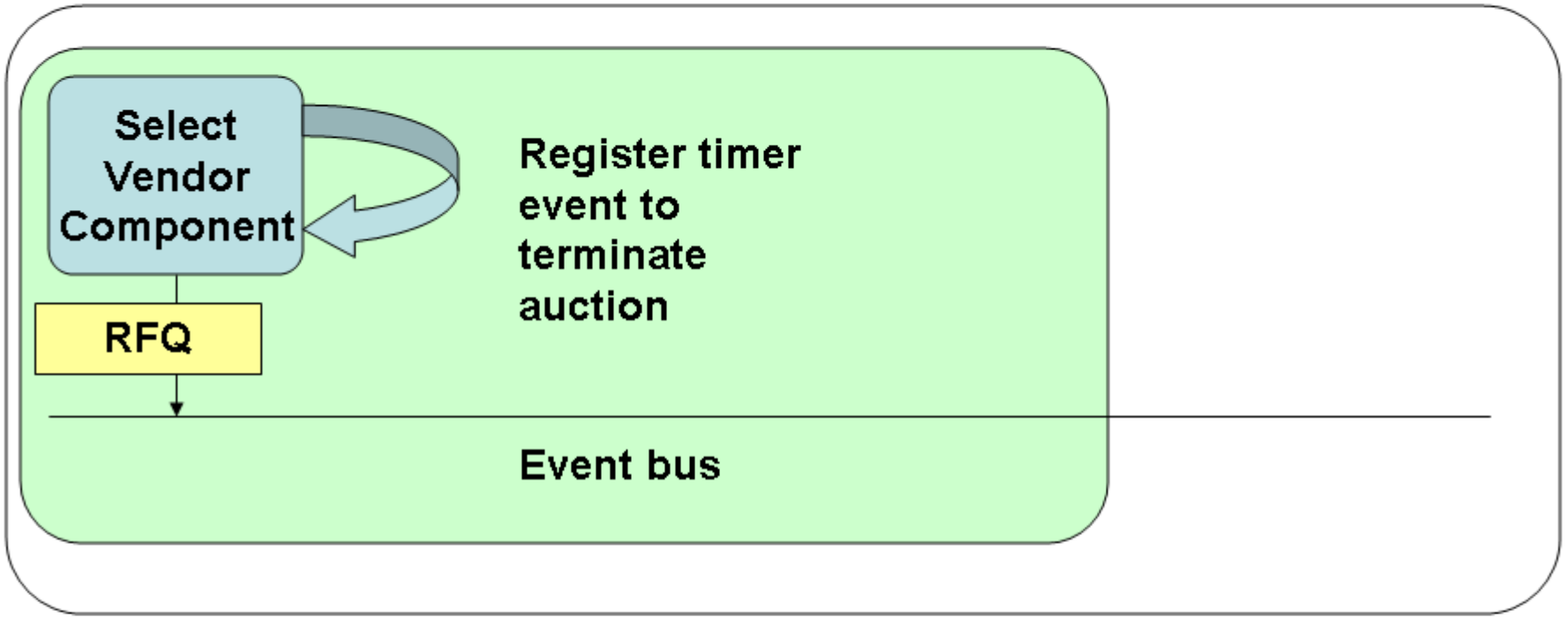
-Logical operators and quantifiers

-Atomic Predicates P

A specification is a set of traces. A system in compliance with a specification will only show behavior (traces) which are part of this set.

(see Mühl et.al. 25 ff).

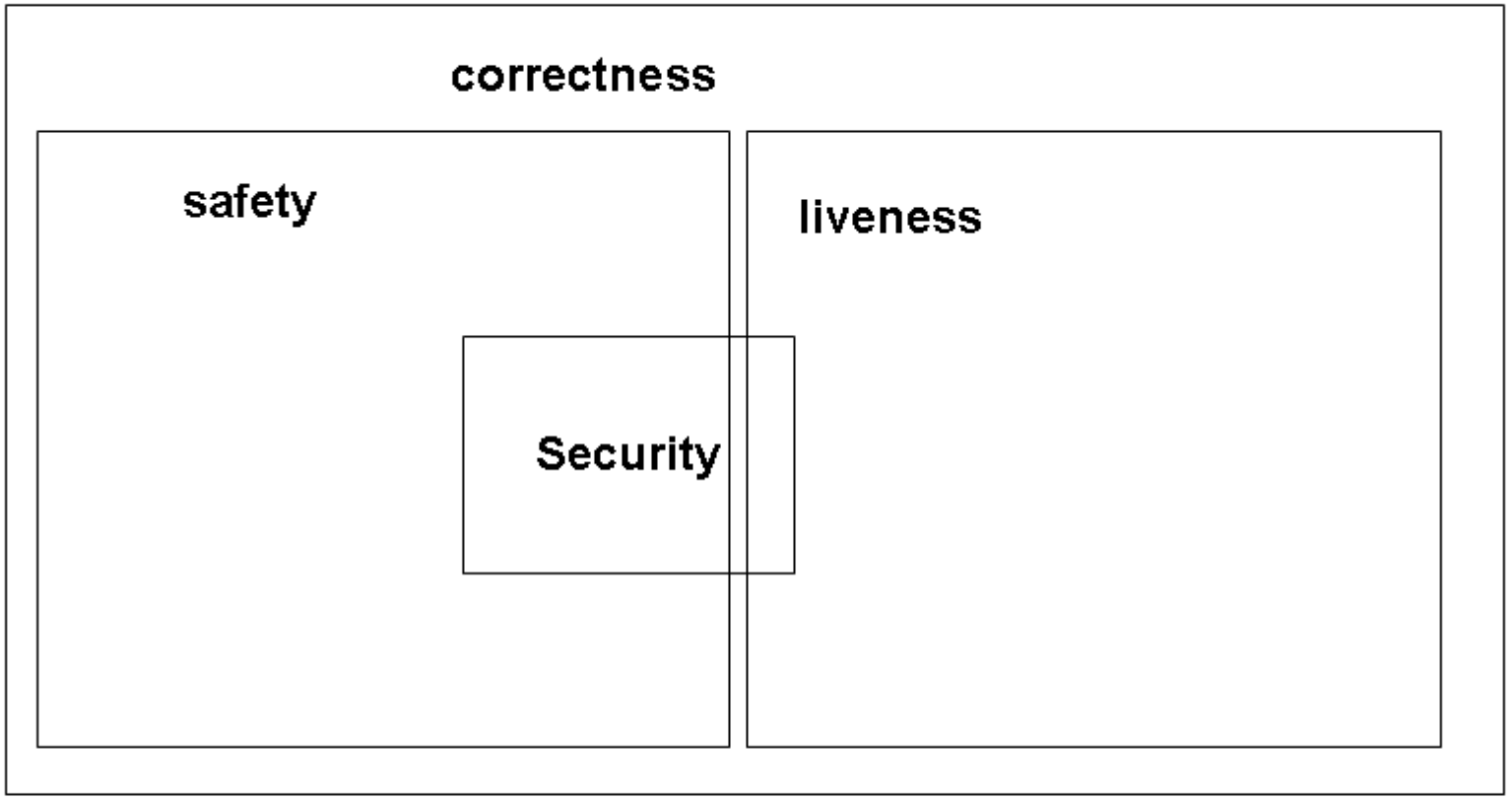
Modeling of internal state changes with timer events



Time limits for process steps can easily be created by registering timer events. Many systems model time based effects this way (see VRML timers for 3D effects etc.)

Modeling Basic System Qualities

Liveness and Safety in Distributed Event-based Systems



The correctness of a system consists of safety (expressed by things that should NOT happen) and liveness (expressed as things that SHOULD happen). Security issues can be expressed as both.

Modeling Causality

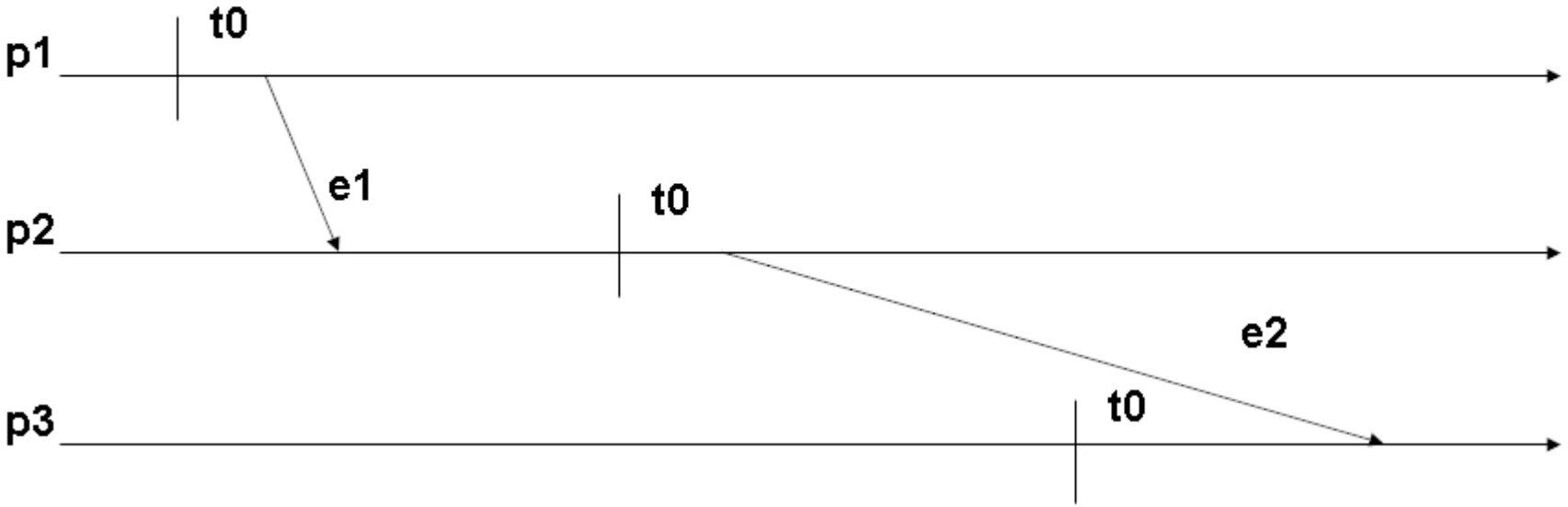
Time and Causality

If A causes B (A →B) then B cannot happen before A.

No clock in a system should B give an earlier timestamp than A.

Of course, this is NOT true for distributed systems because of clock drift between clocks.

No Global Time in Distributed Systems



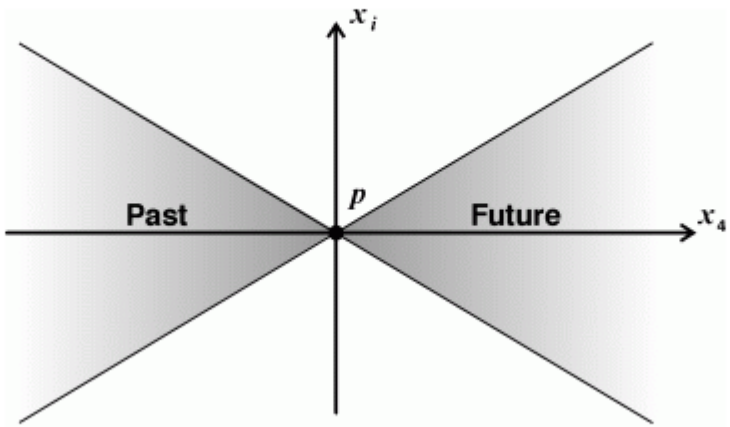
The processes p1-p3 run on different clocks. The clock skew is visible in the distances of t0 on each time line. T0 represents a moment in absolute (theoretical) time in this distributed system. For p2 it looks like e1 is coming from the future (the sender timestamp is bigger than p2‘s current time). E2 looks ok for p3. Causal meta-data in the system can order the events properly. Alternatively logical clocks and vector clocks (see Birman) can be used to order events on each process and between processes. This does NOT require a central authority (like an event system for CEP can provide)

Causality

- Process and Causality (Sowa)
- Computational Causality

Computational causality is very different to other concepts of causality. In computer science visibility and reachability from constraints not unlike the speed of light in physical causality. But both concepts of causality require a theoretical base in any case. In other words: to say that A is a cause of B we always need some theory that relates A and B in some way.

Physical Causality



The light cone was introduced by Albert Einstein in the theory of relativity, which predicts that no information can propagate faster than the speed of light. In Figure 2, the values of functions at points that lie in the shaded area to the left of the point *p* can have a causal influence on the values at *p*. Similarly, the values at *p* can have a causal influence on the values in the shaded area to the right of *p*. The term *light cone* may be replaced by the more general term *cone of causal influence*, since anything outside the cone cannot influence or be influenced by anything that happens at *p*.

Definition of past and future: Let *P*=(*F*,*M*) be a continuous process, and *p* any point in *M*.

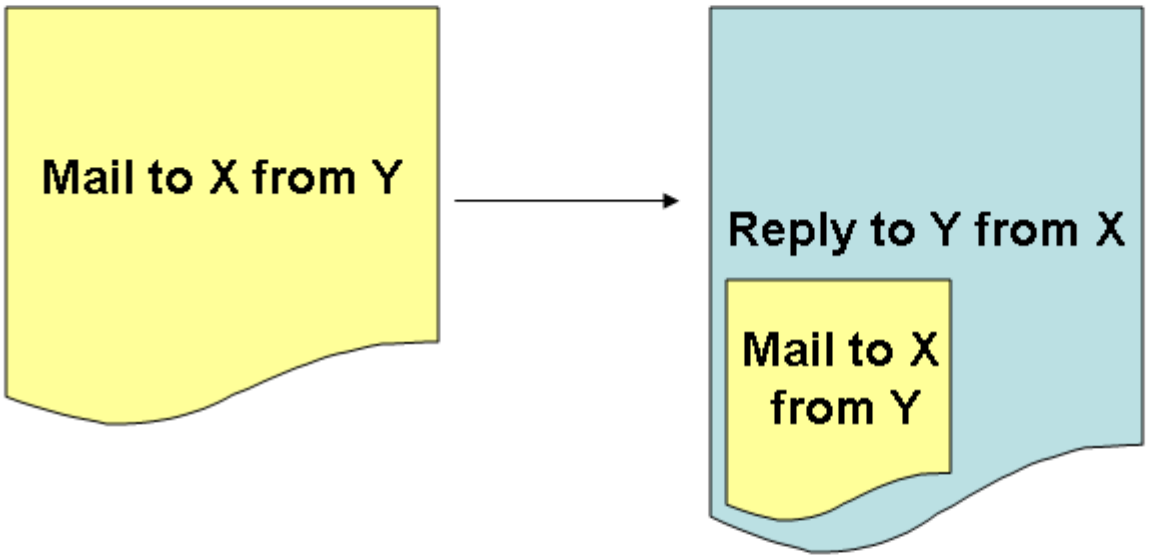
For any neighborhood *U* around *p*, with coordinates *xi*, the *cone of causal influence* with apex *p* is the set of all points *q* in *U*, with coordinates *yi*, which satisfy the following inequality: (*y*1-*x*1)2 + (*y*2-*x*2)2 + (*y*3-*x*3)2 £ *c*2(*y*4-*x*4)2

The constant *c* is called the *speed of information propagation*. Its upper limit is the speed of light in a vacuum.

The region of the cone of causal influence in which (*y*4-*x*4)<0 is called the *past* in *U* with respect to *p*, and the region in which (*y*4-*x*4)>0 is called the *future* in *U* with respect to *p*.

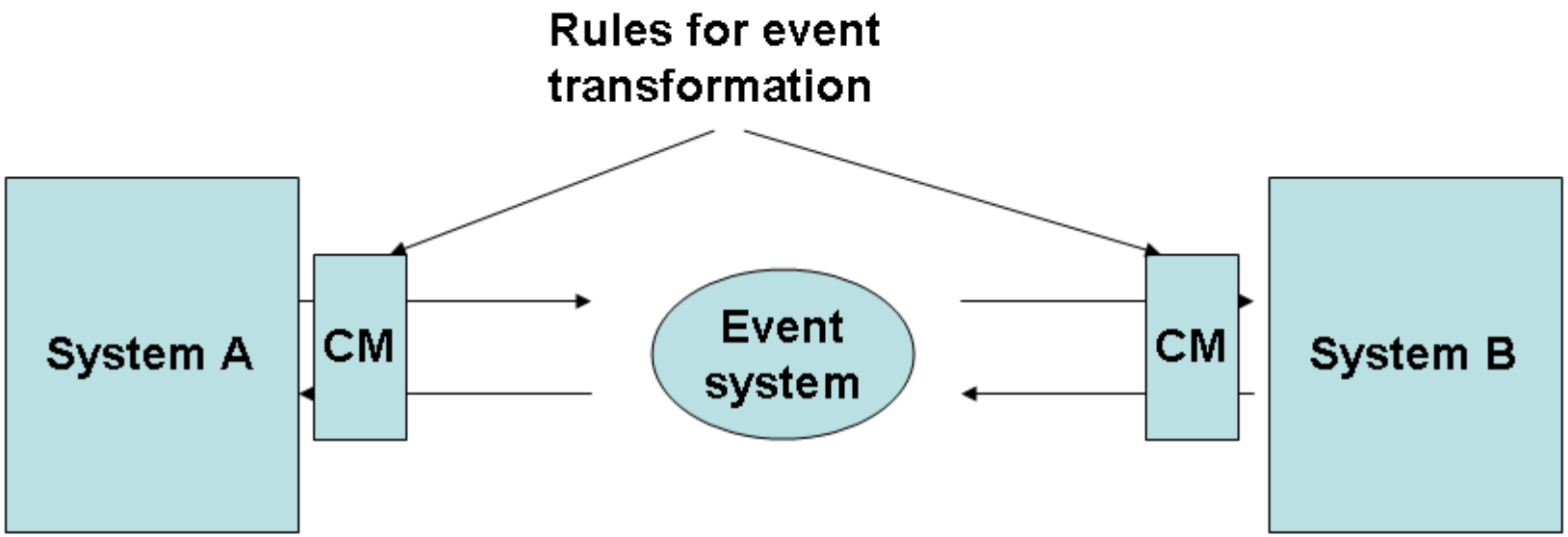
Although the light cone was first recognized as a fundamental concept in relativity, the general principle applies to any kind of reasoning about causality. If the speed of information propagation is *c*, the only events that can have a causal influence at a given point are those inside the past half of the cone determined by *c*. In fluid mechanics, the speed of sound is the normal maximum for information propagation. Therefore, a jet plane that is traveling faster than sound is outside the cone of causal influence for a molecule of air in its path. That molecule would not be influenced by the oncoming jet until it was hit by the leading edge of the plane or by its shockwave. (from J.Sowa, Process and Causality, http://www.jfsowa.com/ontology/causal.htm).

Computational Causality 1



Does a reply to a mail imply a causal relationship between both events? What if the reply comes long after the original mail and the author only used the reply-feature for convenience reasons (just re-use the senders mail address as a new target instead of typing the address). The original mail content would not have a connection to the reply content. But would the reply mail have happened without the original mail? Perhaps. But is there a computing relation between both? Perhaps.

Causal Models



Causal models can be retrofitted to monitor incoming and outgoing requests between collaborating enterprises. Models can use correlation ideas if they exist or create their own causal identifiers.

Modeling Failures

Failure Models

-Failstop: A machine fails completely AND the failure is reported to other machines reliably.

-Byzantine Errors: machines or part of machines, networks, applications fail in unpredictable ways and recover partially.

Many protocols to achieve consistency and availability make certain assumptions about failure models. This is rather obvious with transaction protocols which may assume failstop behavior by its participants if the protocol should terminate.