Chapter 6

Accurate Event Composition

'What, then, is time? If no one asks me, I know what it is.
If I wish to explain it to him who asks me, I do not know'
Augustine, Confessions

For the integration of events, i.e., the composition of events from different sources, an ENS requires information about occurrence times and order of events. In this chapter, we analyze event composition under the aspect of temporal order. Several aspects influence the information an ENS has about event time and order: the observation method, the timestamping method, the time system, and the observer strategy. In the preceding chapters, we assumed access to continuous observation and correct time sources. However, a realization of our event algebra in a system's filter engine demands the consideration of observation and timestamping strategies to ensure accurate identification of composite events.

The ordering of events within a single process is easily understood and influences our way of perceiving the concept of time. This concept of order changes fundamentally when considering events in a distributed system. In a distributed environment, a set of distinct processes communicates by exchanging messages; the message delay is not negligible compared to the time between events in a single process. Additionally, messages can outrun each other (message overtaking). It is sometimes difficult or even impossible to determine which one of two events occurred first. For the detection of composite events in a distributed system of event sources and event observers, the following information is needed: the local (partial) order of events on the same network site, the total (global) order of events, and the real occurrence times of events (i.e., the real time of the event occurrences). If this information is not available or inaccurate, the service's users may be notified about false events or events may be missed. We additionally evaluate the completeness of event information to estimate the time span between the event occurrence until the event information reaches the ENS.

In this chapter, we analyze how the observation and timestamping methods affect the accuracy of the event information at an ENS. Additionally, the influence of time systems and of a distributed ENS architecture is evaluated. We propose a strategy to minimize the temporal delays and to enhance the accuracy of event integration from different sources. In Section 6.1, we introduce event observation and timestamping methods and briefly review time systems for distributed system. We discuss each of the event detection methods in detail. Additionally, we analyze how a distributed network of ENS servers influences the composition accuracy. In Section 6.2, we summarize the advantages and disadvantages of the analyzed event detection methods regarding the correct event composition. In Section 6.3, we introduce a method that determines how an ENS should handle temporal event information and propose an extension for existing timestamping methods. These extensions lead to higher accuracy in the detection of primitive and composite events.

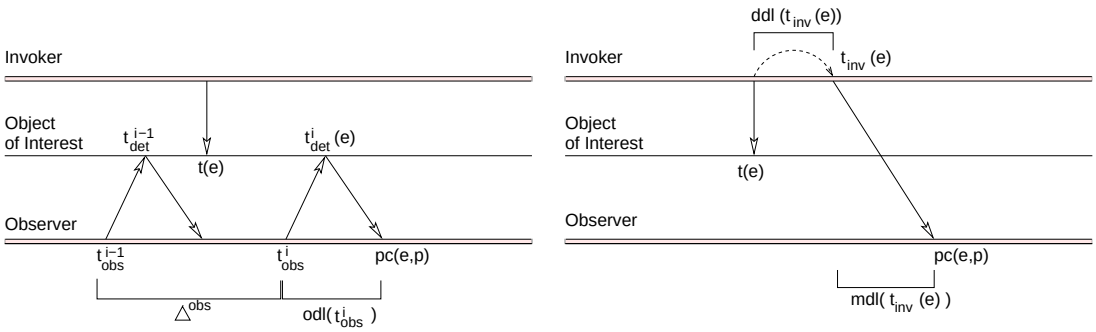
6.1 Methods for Event Detection and Ordering

In this section, we introduce methods for event detection, i.e., event observation and timestamping, and briefly review time systems for distributed systems. The methods are then analyzed for their accuracy of event timestamps and event ordering. Event detection methods are combinations of the observation strategies (as discussed in Chapter 3) and timestamping methods. We consider an invoker-process ip at a provider's site that causes an event e in an object repository at a certain event time $t(e)$. An observer process op at the ENS may learn about the occurrence of the event at perception time $pc(e, op)$. We consider active and passive event observation strategies: Events are either observed by active observers or they are reported to a passive observer by the event invoker. Figure 6.1 shows the time-lines for active and passive observation. These time-lines are sections of event notification sequences as introduced in Chapter 3.

Using active observation[1], an observer-process op retrieves information about events on an object of interest by sending requests to the object (repository) at time t_i^{obs}. A scheduled observer retrieves events on a regular basis, the observer period Δ^{obs} is fixed. Here, $i \in \mathbb{N}$ is a reference number for each of the active observations. The time period between the start of a request (e.g., send a query to the database) and the time of perception $pc(e, op)$ of the result by the observer process is called observation delay $odl(t_i^{obs})$. The event is detected at $t_i^{det}(e)$ and reported to the observer at $pc(e, op) = t_i^{obs} + odl(t_i^{obs})$. The maximal observation delay is referred to as $maxOdl(op) := \underset{i,e}{max}(odl(t_i^{obs}))$. Observers can also react on irregular triggers. Then, the observation period Δ^{obs} depends on i. If the observer delay is longer than the observation period $maxOdl(op) > \Delta^{obs}$, two cases have to be distinguished: The system prevents or allows for message overtaking. Message overtaking may also be prevented by the observer by adjusting the observation period.

Using passive observation, an invoker causes the event e at a certain time $t(e)$ and sends at time $t^{inv}(e)$ a message reporting the event to the observer. The time between event invocation and message

[1]The delays caused by active observation cannot be neglected. If the observation frequency (polling frequency) is too high, the observation may be inefficient. However, if the observation frequency is too low, events my not be detected in a timely manner and event information is falsified.



(a) Active Observer

(b) Passive Observer

Figure 6.1: Event observation time-lines

sending is the detection delay $ddl(t^{inv}(e))$. The event is reported to the observer at $pc(e,p) = t^{inv}(e) + mdl(t^{inv}(e))$, $mdl(t)$ denotes the message delay. The maximal detection delay and message delay for an invoker ip and observer op is defined as $maxDdl(ip) := \max_e(ddl(t^{inv}(e)))$ and $maxMdl(ip, op) := \max_e(mdl(t^{inv}(e)))$, respectively.

We assume that events are observed only once within our system, i.e., one event message reports the occurrence of an event. Events are assigned to a certain detection interval I_{det}, they are not detected twice by the same observer. Similarly to active observation, we have to distinguish systems that allow for or prevent message overtaking.

The timestamp can be attached to an event (or the message reporting the event) in various ways. Before we introduce and analyze the timestamping methods in Section 6.1.2, we argue in the next section that for the analysis, information is required about time systems that are typically used in ENS.

6.1.1 Time Systems

A timestamp is attached to an event message; the moment of timestamping depends on the event detection method. This timestamp underlies an error caused by the time system. We briefly review time systems for distributed systems to highlight their influence on the timestamp accuracy. Two main directions can be distinguished for assigning time and order to events in a distributed environment: *logical time* and *real-time mechanisms*. Logical time [Lam78, Mat88] focuses on the logical (causal) order of events. Using logical time, it is not possible to obtain real-time information about the occurrence times of events. Therefore, logical time is not sufficient for ENS.

In real-time mechanisms, a common solution is the creation of a virtual clock at each site using local hardware clocks (see [LCB99]). Virtual clocks count real-time units, e.g., seconds. A hardware clock typically consists of an oscillator and a counting register that is incremented by the ticks of the oscillator. The instant of time at which an event occurs will be called physical time. The reference time RT is a granular representation of dense physical time. A local clock within a time service is defined as follows:

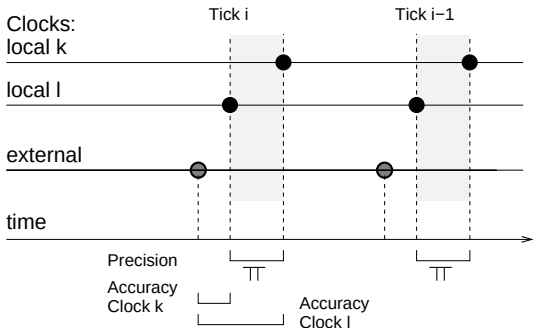


Figure 6.2: Synchronization of external and local clocks

Definition 6.1 (Virtual Clock $C(t)$***)***
A virtual clock is represented by a function $C(t) : RT \rightarrow CT, CT \subset RT$ *that maps reference time to clock-time* CT*. The clock has a granularity* g *by which the counter can be incremented.*

To be correct*, a local hardware clock requires a limited drift rate: Let* s, t *be two reference times* $s, t \in RT$ *with* $s \leq t$*. Then holds* $(1 - \rho)(t - s) - g \leq C(t) - C(s) \leq (1 + \rho)(t - s) + g$*, where the constant* ρ *describes the drift and* g *the granularity of the reference time.*

The *external clock synchronization* aims at maintaining a given maximal time deviation between local time and reference time; the *internal clock synchronization* aims at reaching a consistency between virtual clocks, including compensation for frequency differences (skew). Their measures are called accuracy and precision, respectively.

Accuracy α**:** the maximum deviation of local clock reading from real-time (cf. Figure 6.2).

Precision Π**:** the maximum difference of simultaneous local clock readings (cf. Figure 6.2).

Accuracy α always implies precision $\pi \leq 2\alpha$. For virtual clocks generally holds $g << \alpha$. Several real-time mechanisms have been proposed, we briefly introduce the four main approaches:

2g-Precedence model: The 2g-precedence model [KFG93] defines a global time approximation using a combination of temporal global timestamps and logical local timestamps. Schwiderski [Sch96] enhanced the model for distributed event ordering and composite event detection by introducing 2g-precedence-based sequence and concurrency operators. The detection of event order within a distributed system bases on a comparison of the states of local and global systems clocks. Events from different sites can only be ordered if they are at least two clock ticks apart (2g). The model is widely used in ENS, e.g., in EVE [TGD97], COBEA [MB98], the Cambridge Event Architecture (CEA) [BBHM96], and Object Monitor [HS97].

The 2g-precedence model is applicable for closed networks with interconnected servers (for internal clock synchronization). However, event handling is non-deterministic in the case of concurrent or unrelated events. Additionally, the violation of the granularity condition ($2g$) may lead to false detection of events.

Weakly Synchronized Clocks: In this category fall time systems that use unsynchronized system clocks or radio controlled clocks that are occasionally synchronized, e.g., based on a predefined schedule.

These time systems are typically found in many real-world applications; an example are facility management system that use a daily synchronization resulting in an accuracy of about one second. The clocks are not internally synchronized.

Network Time Protocol: The network time protocol (NTP) is an Internet standard for real-time mechanisms [Mil91]. It consists of a time-service architecture and a distribution protocol. Primary time servers are directly connected to a primary reference source, such as time-code receiver or a calibrated atom clock. The network time protocol provides clients with an average accuracy of $\alpha \leq 10msec$ [SS97]. For mobile devices, the accuracy may deteriorate.

The NTP allows for assignment of real-time timestamps with given maximal errors. When an event e occurs at physical time $t(e)$ the systems calls a time system function to assign a time to the event. The time system returns the time $C(t)$ as reading of the local clock and the global timestamp is attached to the event. The NTP service does not automatically provide a global order of the events. Global and partial order of the events can be obtained with a certain accuracy based on the timestamps. However, in open distributed environments, where not all servers are interconnected, event ordering based on the NTP may lead to false event detection [LCB99].

Interval-based time system: Several methods additionally consider the possibility of partial ordering in a distributed environment. These approaches better scale to open systems. Spurious events and ambiguous event consumption are avoided. These methods define event order based on intervals [Mar85, SS97, LCB99].

Liebig et al. [LCB99] propose an event timestamping mechanism for large-scale, loosely coupled distributed systems. They use accuracy intervals with reliable error bounds for timestamping events that reflect the inherent inaccuracy in time measurements. Similar to 2g-precedence, the accuracy intervals depend on both local and global timestamps but are created using the NTP. The interval-based time system are only applicable to ENS with active providers (passive observation). We show in the next sections that for active observation, an extended ordering method is required.

As seen in this brief overview, existing time systems provide different precision and accuracy characteristics. For an integrating event notification system, sources using various time systems have to be supported. Therefore, information about the time system and its characteristics is crucial for accurate event detection and composition. Consequently, our analysis also covers the influence of the time system on the composition accuracy.

6.1.2 Distributed Event Detection

We identify four event detection methods (I to IV), i.e., methods for event observation and timestamping, as shown in Table 6.1. In method II, we further distinguish the available temporal event information. We now describe each of the event detection methods, their applicability and influence on the accuracy of event information (timestamp, partial and global order, and completeness of information). We also analyze the influence of message delays on the composition accuracy. We use the characteristic parameters shown in Table 6.2 for the analysis of event detection methods. We introduce the notion of *timestamp accuracy* $ta(e)$: Whereas the timestamp delay tdl describes the time span between event

No	Observation		Timestamping			
	Active Observer	Passive Observer	set by			set at
			Object	Invoker	Observer	
I	X		X			Occurrence time
II a	X				X	Detection time
II b						Request / Answer time
II c						Answer time
III		X		X		Occurrence time
IV		X			X	Perception time

Table 6.1: Event detection: observation and timestamping methods

time and timestamping, the timestamp accuracy describes the difference of timestamp value and event time. Note that an *accuracy interval* (as introduced by Liebig et al. [LCB99]) describes the real-time equivalence of the time stamp; it does not necessarily equal the real-time representation of the detection interval, which we use for event ordering. For each method, we estimate the parameter values in order to analyze the method's influence on the accuracy of the event information. The first method is described in detail, the subsequent methods only briefly.

Parameter	Description
Event-time $t(e)$	occurrence time of the event e
Timestamp $ts(e,p)$	point in time associated with event (message) e to indicate its occurrence time, assigned by a process p
Detection Interval I_{det}	time interval between the upper and lower limit for the event-time as inferred from the event detection
Timestamp Delay tdl	time between event and timestamp $tdl(e,p) = ts(e,p) - t(e)$, estimates the error of the timestamp
Perception Time $pc(e,op)$	point in time an observer process op learns about the occurrence of event e
Perception Delay pdl	time between event e and perception of the event by an observer process op: $pdl(e,op) = pc(e,p) - t(e)$
Timestamp accuracy $ta(e)$	difference of timestamp-value (in real time) and event occurrence time

Table 6.2: Characteristic parameters of event detection methods

I Active Observation, Timestamp by Object. When the invoker causes the event at a certain time $t(e)$, this invocation time is attached to the object (e.g., the last-modified value of a file). This object-timestamp is read by the observer. New events can possibly overwrite or change the effects of older events, e.g., when a file is changed, the last-modified information is overwritten. Additionally, this method is not applicable for delete operations, because the timestamp is deleted together with the object.

The detection interval is a single point in time $I_{det} = [t(e), t(e)]$, the timestamp delay is zero. The perception delay is less or equal $\Delta^{obs} + odl(t_i^{obs})$ (see Figure 6.1(a)). The object carries the time of event invocation as timestamp according to local invoker time C_{inv}. The timestamp accuracy $ta(e) = t(e) - ts(e)$ depends on the invoker clock accuracy. For an externally synchronized clock as introduced before, it exists an α_{inv} so that the real time of the event time can be estimated as accuracy interval

$$t(e) \in [C_{inv} - \alpha_{inv}; C_{inv} + \alpha_{inv}]_{rt} \triangleq I_{det}, \quad \text{and} \quad ta(e) = 2\alpha$$

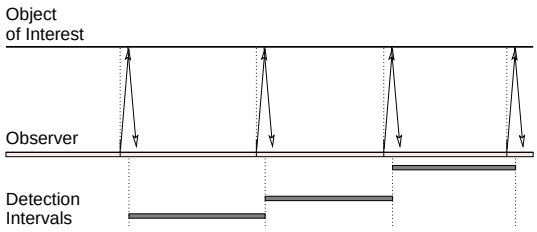


Figure 6.3: IIa: Timestamp by active observer, detection time accessible

Here, the detection interval equals the accuracy interval. The $\triangleq$ operator indicates the reference to a real-time equivalent of a parameter. To compare event times with local observer times, the event time expressed in observer time can be estimated as $C_{obs}(e) \in [C_{inv} - \alpha_{inv} - \alpha_{obs}; C_{inv} + \alpha_{inv} + \alpha_{obs}]$. If observation delays are not limited, the completeness of the event information cannot be ensured. Due to the different numbers of requests and answers, the observer is aware of the incompleteness.

The real occurrence time of the events (i.e., the real-time of the event occurrence) can be estimated based on the real-time interval of the timestamp by invoker. The local order of events can be determined based on the local invoker timestamps $C_{inv}(e)$. Events in disjunct accuracy intervals can also be ordered globally. At perception time $C_i^{obs} + odl(C_i^{obs})$ the observer has complete information about events that happened before $C_{inv} - \alpha_{inv} - \alpha_{obs}$.

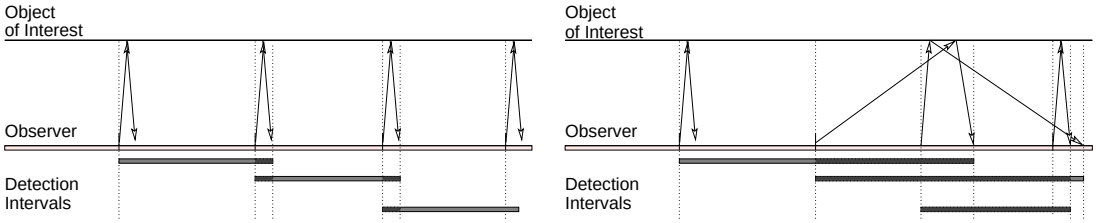
If the observer has no access to invoker information (timestamping method and time system), timestamps can only be used as event counters without relation to real time. Then, events cannot be globally ordered, real occurrence times are not available and the completeness of information at a certain point is not known. Events cannot be matched against time-dependent profiles and composite event profiles.[2]

In this case, the observation times should additionally be stored (as in IIb) to enable an estimation of the event time based on the observer time: Events in the same observation period can only be globally ordered with non-intersecting real-time observation periods. The real occurrence time can be estimated by the (real) observation intervals. Completeness can be estimated independent of the invoker time of events. Therefore, for each event, the observation period must also be considered in the ENS.

II Active Observation, Timestamp by Observer. The observer may have access to one or more of the following temporal information: event detection time, request time, and answer time. Consequently, we distinguish three cases: (a) the detection time is accessible, (b) only request and answer time are accessible, or (c) request and answer time are accessible but independent of each other.

IIa Detection Time t_i^{det} accessible: The observer learns at the perception time $pc(e, op) = t_i^{obs} + odl(t_i^{obs})$ about the events that happened in $I_{det} = (t_{i-1}^{det}, t_i^{det}]$. Equal timestamp values are assigned to all events observed within a detection interval: $ts(e, op) = t_i^{det}$. The timestamp is defined by the observer, but given in invoker time: $ts(e, p) = C_{inv,i}^{det}, i \in \mathbb{N}$. The accuracy interval

[2] We assume that user profiles are defined according either to real time or to observer time.



(a) without message overtaking, $\Delta^{obs} \geq maxOdl$ (b) with message overtaking, $\Delta^{obs} \leq maxOdl$

Figure 6.4: IIb: Timestamp by active observer, request and answer time accessible

is a subset of the real-time representation of the detection interval:

$$\begin{aligned} ts(e) &\in [C^{det}_{inv,i} - \alpha_{inv}, C^{det}_{inv,i} + \alpha_{inv}]_{rt} \\ t(e) &\in [C^{det}_{inv,i-1} - \alpha_{inv}, C^{det}_{inv,i} + \alpha_{inv}]_{rt} \triangleq I_{det}, \\ ta(e) &\leq \frac{\Delta^{obs} + odl(C^{obs}_i) + g}{1-\rho} \end{aligned}$$

The perception time and perception delay are similar to method I. The real occurrence time of an event is estimated by its real-time detection interval. The partial order of events within the detection interval is not known. Events from different detection intervals can be ordered by their timestamps (see Figure 6.3). For global ordering, the real-time representations of detection intervals have to be used. The influence of the time system on the completeness of event information is similar to method I.

Information about the invoker time system is crucial to obtain occurrence time and order of events. If the invoker time system parameters are not available to the system, timestamping method IIb should be used.

IIb Only Request and Answer Time Accessible: The observer learns at time $pc(e, op) = t^{obs}_i + odl(t^{obs}_i)$ about the events that happened in $(t^{det}_{i-1}, t^{det}_i]$. Because the event detection time t^{det}_i is not known to the service, the detection interval is $I_{det} = (t^{obs}_{i-1}, pc(e, op))$. The timestamp for each event is set to its perception time $ts(e, op) = pc(e, op)$. The timestamp is independent of the invoker time. The accuracy interval (detection interval) is

$$\begin{aligned} ts(e) &\in [C^{obs}_i + odl(C^{obs}_i) - \alpha_{obs}, C^{obs}_i odl(C^{obs}_i) + \alpha_{obs}]_{rt} \\ t(e) &\in [C^{obs}_{i-1} - \alpha_{inv}, C^{obs}_i odl(C^{obs}_i) + \alpha_{obs}]_{rt} \triangleq I_{det}, \\ ta(e) &\leq \frac{\Delta^{obs} + odl(C^{obs}_i) + g}{1-\rho} \end{aligned}$$

The real-time estimation of the perception interval may be used as an approximation of the event occurrence time.

Detection intervals overlap with adjacent intervals, the overlaps are displayed in dark gray in Figure 6.4(a). For complete information about events in a detection interval, the observer has to await the results of the subsequent observation: Events occurring in $[t^{obs}_i, t^{obs}_i + odl(t^{obs}_i))$ can be detected either in $(t^{obs}_{i-1}, t^{obs}_i + odl(t^{obs}_i))$ or in $(t^{obs}_i, t^{obs}_{i+1} + odl(t^{obs}_{i+1}))$. Without message overtaking,

events can be partially ordered by their timestamps. If the observer period is shorter than the maximal observation delay ($\Delta^{obs} \leq maxOdl$) message overtaking may occur (see Figure 6.4(b)). In this case, events with overlapping detection intervals can neither be ordered partially nor globally, because the observation order cannot be determined.

The global order of events is obtained similar to method IIa. The influence of the time system on the completeness of event information is similar to method I. Note that no information about the invoker time system is required with this method.

IIc Only Independent Request and Answer Time Accessible: Similar to method IIb, the observer learns at time $pc(e,p) = t_i^{obs} + odl(t_i^{obs})$ about the events that happened in $(t_{i-1}^{det}, t_i^{det}]$. The timestamp is $ts(e,p) = pc(e,p)$. The sets of starting points t_i^{obs} and perception points $pc(e,p)$ are independent. If the observation delay is limited by the observer period ($odl^{obs} \leq \Delta^{obs}$), starting points and perception points can be related. Then the events are in the same order as the messages reporting them and the characteristics of this method are similar to those of method IIb.

If the maximal observation delay is longer than the observer period or no upper limit is known for the delay, no local order can be assigned to the observed events. The detection interval is then $I_{det} = (-\infty, pc(e,p))$ [3], timestamp delay and perception delay may be infinite. This case is not further considered here.

For all active scheduled observations, the following applies: If the observer period is longer then the observation delay $\Delta^{obs} \geq maxOdl$ then the perception delay is bound by the observation period length $pdl(e,p) \leq 2\Delta^{obs}$.

III Passive Observation, Timestamp by Invoker: The invoker reports event and time of event invocation $t(e)$ to the observer; the timestamp is $ts(e,p) = t(e)$ and the detection interval $I_{det} = [t(e), t(e)]$. The timestamp delay is zero, the perception delay is $pdl(e, op) = ddl(t^{inv}(e)) + mdl(t^{inv}(e))$ (see Figure 6.1(b)). Similar to method I, the timestamp is expressed in invoker time $ts(e) = C_{inv}(e)$. The real-time interpretation of the detection interval equals the accuracy interval:

$$t(e) \in [C_{inv}(e) - \alpha_{inv}; C_{inv}(e) + \alpha_{inv}] \quad \text{and} \quad ta(e) \leq \alpha_{inv}$$

Very similar to method I, the ENS has to use invoker timestamp for local order and the real-time equivalents of the invoker timestamp (accuracy interval) for global order and the estimation of real occurrence time. For the completeness interval at perception time the real-time mappings of observer and invoker times have to be considered. The completeness of the event information depends on the delays ddl and mdl, as discussed in Section 6.1. To estimate the detection delay ddl, further information about the invoker time system is required. No information about missed events is available. If the detection and message delays are not limited it is not possible to ensure the completeness of the event information for the observer. The observer is not aware of the incompleteness.

Observation accuracies depend on information about invoker time and observer time. If the observer has no information about invoker time system, the timestamps can only be used as event counters

[3] In a system implementation, $-\infty$ would be mapped to the starting time of the system t_{start}

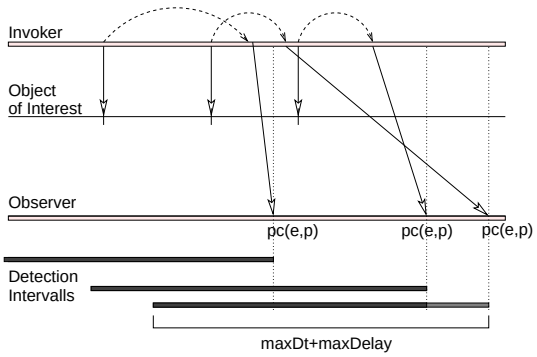


Figure 6.5: IV: Passive observer, timestamp by observer, unlimited delays

without relation to real time. Additionally, events cannot be globally ordered and the completeness of the information at a certain point is not known. Therefore, events cannot be matched properly against time-dependent profiles and composite event profiles.

IV Passive Observation, Timestamp by Observer: The invoker reports the event at a certain time pc(e,p) to the observer without stating the time of event invocation. The timestamp is $ts(e,p) = pc(e,p)$, the detection interval can only be estimated as $I_{det} = [pc(e,p) - maxDdl - maxMdl(ip,p), pc(e,p))$. The timestamp delay is estimated $tdl(e,p) \leq maxDdl + maxMdl(ip,p)$. If detection delay ddl and message delay mdl are limited and no message overtaking occurs, the event can be partially orderer by their timestamps. With message overtaking, only events with non-intersecting detection intervals can be partially ordered. For global order, only events with non-intersecting (real time) detection intervals can be ordered. As before, the real-time equivalences have to be considered. The event time and order of events is determined similar to method IIa.

If either the delays are not limited or if message overtaking occurs, the local order and the occurrence time of the events cannot be determined (see Figure 6.5). The detection interval is then $(-\infty, pc(e,p))$. The timestamp delay is estimated as $tdl(e,p) \leq pc(e,p) - t_{start}$.

Similar to method III, the completeness of the event information depends on the delays ddl and mdl. Information about invoker time may influence the estimation of ddl and mdl. If the delays are not known or not limited, the events cannot be ordered, neither their real occurrence time nor the completeness are known. Information about missed events is not available.

6.1.3 Influence of ENS Architecture

We study how the ENS architecture additionally the influences timestamp accuracy and ordering of events. For events from different providers, partial order based on local timestamps can only be obtained for events at the same provider site. As shown in the previous section, timestamps given by the observer depend the on observation strategy and may not necessarily reflect the order of the event occurrences. For the observation of several providers, a circular scheduled observation by a single observer is conceivable. This may lead to a lower timestamp of an event that happened after an earlier event on a different site that is observed later. Thus, events that are observed within the same observer turn cannot be ordered.

By use of multiple observers with different time systems, the events from different sites can only be globally ordered according to their real-time detection intervals. Local order can only be obtained for events at the same provider's site.[4] Event notification systems can form a hierarchy or a network of systems. Here, the ENS act as providers and clients for each other. Each of the systems may employ different observation and timestamping methods and time systems. The time systems of provider and system site do not directly influence global order, because that order is based on real-time equivalents of timestamps and on detection intervals. But the time systems influence the accuracy of local time and, therefore, the set of events that can be ordered. Each time-system adds its share to the inaccuracy of the considered times. Only directly synchronized clocks can avoid the effect due to a precision $\pi \leq 2\alpha$.

The hierarchical architecture influences profile matching and perception time depending on the service's event operators. For example, the hierarchy influences the result for selecting the first event instance. Additionally, profiles can only be matched with a certain error bound, which is increased by the number of involved network nodes. Thus, the ordering problem is not always decidable. Heterogeneous networks additionally aggravate duplicate detection. Timestamps and their accuracies depend on the timestamping components, such as invoker or first observer. The accuracy is not influenced by the routing of the event messages within the network to the client. To ensure a correct filtering, the filter of each ENS has to have information about the time system the event's timestamp is based on. The event message also has to carry information about the detection interval to determine a global order. To incorporate all events observed in the network of services, the filter component requires information about the maximal perception delay, which depends on the network architecture.

If the event servers in an ENS hierarchy do not propagate the event timestamps (as is customary, e.g., in a digital library context) each component has to define a new timestamp and the inaccuracy grows with the sum of the detection delays of all concerned components. Time reference and ordering become almost impossible.

6.2 Analysis of Event Detection and Ordering

In the previous section, we introduced several methods for event observation and timestamping. Each of these methods has advantages and disadvantages when used for event composition. In this section, we provide an overview of our findings regarding the accuracy of event filtering under different event detection methods and time systems. As result of our analysis, we propose both a strategy for the handling of event timestamps and an extensions to the timestamping used by event notification systems. These extension are introduced in the next section.

We consider the following aspects in our analysis: (a) detection of the occurrence of events, (b) estimation of occurrence time of events (timestamp accuracy), (c) detection of event order, (d) estimation of the completeness of the event information at the observer, and (e) the influence on event composition.

a) Occurrence of events: Different observation strategies and timestamping methods can lead to missing events or falsified events. The underlying time systems have indirect influence on the observed events, e.g., by influencing scheduling times.

[4] We do not consider the problem of duplicate event observation introduced by the use of several observers for the same object.

No	Timestamp	Detection Interval	Timestamp Delay
I	$t(e)$	$[t(e), t(e)]$	$tdl(e,p) = 0$
II a	t_i^{det}	$(t_{i-1}^{det}, t_i^{det}]$	$tdl(e,p) \leq \Delta^{obs} + maxOdl$
II b c	$t_i^{obs} + odl(t_i^{obs})$	$(t_{i-1}^{obs}, t_i^{obs} + odl(t_i^{obs}))$	
III	$t(e)$	$[t(e), t(e)]$	$tdl(e,p) = 0$
IV	$t(e,p)$	$[t(e), pc(e,p))$	$tdl(e,p) = ddl(t^{inv}(e)) + mdl(t^{inv}(e))$

Table 6.3: Timestamps, detection intervals, and timestamp delays

b) Accuracy of timestamps: The real time of an event is estimated using the real-time interpretation of the detection interval, which is influenced by the event detection method and the time system. As expected, the detection method has a higher impact on the real-time estimation of the event time than the accuracy of the underlying time system. In Table 6.3, we give an overview of the timestamps, detection intervals and timestamp delays resulting from the various event detection methods. Best results are achieved for methods I and III. The timestamp delays in II and IV depend on network delays, e.g., observation and message delays. For Internet-scale services, no upper limit on network delays can be ensured.

The timestamp accuracies have been estimated with in the descriptions of the methods in Section 6.1, they are not repeated in the figure. For the methods I and III, the accuracy is directly influenced by the time system; for methods IIa-c, the accuracy depends mainly on observation; for method IV, the accuracy depends on notification and observation. If the time-system gains more influence than the observation (e.g., the time-system error is greater than the observation period) the combined error increases.

c) Order of events: Without message overtaking, events can be partially ordered according to their local timestamps. Events with identical local timestamps cannot be ordered. With message overtaking, only events with disjunct (local) detection intervals can be partially ordered. For partial order, the methods I and III provide the most accurate ordering: *All* events can be partially ordered. Methods IIa-c and IV depend on message delays; of these, IIa achieves best results because the detection intervals do not overlap.

The properties of the local time-system determine local accuracy. This local accuracy is more precise than the global accuracy. The local accuracy depends heavily on the underlying time system. Global order depends on the time detection interval. Events with disjunct (real-time) detection intervals can be ordered globally. Global ordering and occurrence time, therefore, depend on information about the local invoker time and the local observer time. If invoker time information is available, the methods I and III have best accuracy results: Most information about occurred events is available. Using method I, events can be falsified or missed due to the observation interval — the method I may be improved by combination with method IIb. Falsification may also happen using method III if the detection delay is larger than the time between the events $ddl > \frac{1}{freq_{events}}$. If $\alpha_{inv} > \frac{1}{freq_{events}}$, the events with non-disjunct accuracy intervals cannot be ordered.

Method IIa should be used for applications with preference on event time, e.g., with focus on time events. Method IIb should be used if event ordering is more important than accurate event time, because

No		Partial order			Global order			Occurrence time			Comments
		delays		no	delays		no	delays		no	
		lim	ulim	Inv	lim	ulim	Inv	lim	ulim	Inv	
I		×	×	-	×	-	-	×	-	-	Improved by combin. with IIb
II	a	×	×	-	×	-	-	×	-	-	IIb to be preferred
	b	×	×	-	×	×	×	×	×	-	
	c	(×)	-	-	(×)	-	-	(×)	-	-	Small influence of time system
III		×	×	-	×	-	-	×	-	-	
IV		(×)	-	-	(×)	-	-	(×)	-	-	Invoker information required

Table 6.4: Available information under limited (lim) delays, unlimited ($ulim$) delays, or unknown invoker time system ($noInv$)

of the independence of invoker time. For IIc, the time system has only small influence. For method IV, further invoker information is necessary.

If no information about the local invoker clock is available, only method IIb supports global ordering. Then, we recommend a combination of the methods I and IIb.

d) Completeness of event information: We consider the completeness of event information available to the ENS at a given point in time. Complete information means that all events that occurred before a given point in time are known to the system. The completeness of the event information at a given point in time depends on the observation method and the transmission delays. It is not directly affected by the time system.

Table 6.4 shows the ordering and time information available to the service using the different detection methods. We distinguish for each method, whether the observation and network delays are limited or unlimited and whether information about the invoker time system is available.

First, we consider the case of limited network delays (without message overtaking). The column 'completeness' in Table 6.5 indicates if the information known to the system at perception time covers all events that happened before $pc(e,p) - pdl(e,p)$. Active observation implies larger perception delays based on the sum of observation periods (and the observation delay) for all observing instances between provider and client. *Delete events* may be missed and events on the same object within an observation period are observed with falsified results as a single event. Most information is available using method I. For passive observation, all event occurrences may be known to the service if $ddl < \frac{1}{freq_{events}}$. The timestamp delays are expected to be smaller than in active observation. Events are not falsified but event information could be lost without the ENS being aware of this missed event information. Completeness of information depends on the provider's support.

For unlimited network delays and possible message overtaking, ENS with active observers have to provide a strategy to cope with requests that have a very long observation delay, e.g., by reposting the request after a certain time. In ENS with passive observers, event information can be lost without the observer being aware of these outstanding or lost messages. Additional use of logical clocks could locally indicate missed events; we introduce our extended timestamping concept in the next section.

The time system influences the accuracy of the timestamp and, indirectly, the perception delay. Scheduled observers might change their predefined order: Observation times, timestamps, and event

No	Perception Time	Perception Delay	Compl.
I	$t_i^{obs} + odl(t_i^{obs})$	$pdl(e,p) \leq \Delta^{obs} + maxOdl$	yes
II a			no
II b			no
II c			no
III	$t^{inv}(e) + mdl(t^{inv}(e))$	$pdl(e,p) \leq maxDdl + maxMdl(ip,p)$	yes
IV	$t^{inv}(e) + ddl(t^{inv}(e)) + mdl(t^{inv}(e))$	$pdl(e,p) \leq maxDdl + maxOdl(p)$	yes

Table 6.5: Completeness and perception delays

order are influenced. This effect is more significant in a hierachy/network architecture of a group of interacting ENS. Each ENS adds its share to the perception delay. Additionally, network parameters such as message delays have a great influence on the perception delay and, thus, indirectly on the number of observed events.

e) Detection of composite events: Primitive events can be missed or not be clearly identified due to falsified event observations and long message delays. The observation interval length and the accuracy of the time system give a measure for the possible differentiation between events. Time events underlie a certain maximal error accuracy α.

The errors in the primitive event detection influence the event composition. For example, a disjunction profile is matched by the event first received by the filter; thus, the result is influenced by detection methods and time systems, and network architecture. For a conjunction profile, the effect of the time systems on the time constraint may lead to matching errors. For a sequence profile, the order of events is crucial. It depends on observation strategy, timestamping method, time systems, and on the ENS network architecture. The order cannot always be determined. The ordering is therefore only valid with a certain maximal error, which depends on observation interval and time system accuracy. The service has to implement a strategy to cope with events of unknown order. The matching of passive profiles heavily depends on the correct determination of the time interval. This is only possible with a certain accuracy and, therefore, matching of negation constraints underlies a certain error. Therefore, negation may lead to the detection of spurious events.

The parameters for composite events introduced in Chapter 5 additionally influence the accuracy of event detection: For determining the n^{th} event within a duplicate list, the order of events has to be determined. Similar argumentation as for event sequences applies. The observation and network delay additionally influence the notification delay for a service, because the filter has to await complete event information about a considered interval. As we have shown, this may take several observation cycles.

6.3 Event Handling and Extended Timestamping Method

We have shown that the real-time representations of the detection intervals have to be used for event ordering. Our approach is an extension of Liebigs interval-based filtering [LCB99] that can only be used for active observation (using method III). In this case, the detection interval equals the accuracy interval. The detection interval has to be determined based on the time system information.

Timestamp Handling. We propose the following strategy for the handling of timestamps: Local timestamps have to be stored in order to obtain a partial order of events from the same site, which is of higher accuracy than the global order. Global order requires information about time system and event detection method. Consequently, the following information should be exchanged together with the event messages: local timestamp, detection interval, and time system information.

If no time system information is available, timestamps based on this time system can only be used for partial ordering but not for real-time estimation and global ordering. A new timestamp has to be obtained by the next component within the event server hierarchy. This new timestamp and its meta information (e.g., detection interval and time system) have to be forwarded in the hierarchy.

Extended Timestamping Method. In order to support the detection of missing and falsified events and to support ordering of observed events, we propose the additional use of simple logical clocks that extend the employed time systems by an event counter. This extended time system leads to higher accuracy in the detection of (composite) events. We propose three approaches that increase the quality of event information delivered to the client.

1. A simple approach is the implementation of a local event counter that is called by the invoker at each event occurrence. That *logical counter time* is then associated to the event and attached to the affected object, if available. The advantages are evident: The fact of missed or falsified events is made transparent to the observer. When observing logical counter times with gaps in the enumeration, the skipped numbers refer to events that have not been observed. It is not possible to distinguish between missed and falsified events.

2. An advanced but still simple approach are logical event counters distinguished according to the event form. For example, separate counters for delete events and content events are introduced. Skipped numbers in the content event enumeration now refer to falsifying events. Still, the affected objects cannot be identified.

3. The most advanced approach is the consideration of different event counters for different objects. That approach would only support object-related events, but it clearly indicates falsified events. The information gained could be used for more sophisticated profiles, considering also the number of missed or falsified events.

The observed uncertainty of event information should be made transparent to the user. Thus, the introduction of an extended time system enables more precise event filtering and a greater transparency.

6.4 Related Work

In this section, we briefly review related approaches for including awareness of accurate composite event detection in event notification services. Selected issues of this chapter have already been addressed in the respective sections, e.g., time systems and event order.

Event ordering and composition in active databases takes advantage of the centralized architecture inherent to most systems: Sentinel [CM94] uses a global history log, Ode [GJ91] and SAMOS [GD94]

use timestamps for event identification and require a total ordering. These techniques can only be used for partial ordering in our context. Logical ordering of events based on logical clocks has been studied, among others, by Lamport [Lam78], Fidge [Fid88], Matern [Mat88], and Hrischuk and Woodside [HW96]. Logical clocks may provide event ordering but do not preserve real-time information.

A first approach to address the problems of imprecise timestamps and composition of events in a distributed environment was proposed by Liebig et al. [LCB99, LBB99]. This approach uses accuracy intervals with reliable error bounds for timestamping events that reflect the inherent inaccuracy in time measurements. It is based on the Network Time Protocol (NTP). It is not sufficient for an integrating service, because of its restriction to passive observation (method III). Additionally, timestamping methods and active observation are not covered.

To the best of our knowledge, the interrelation of observation strategies, timestamping methods, and time systems for event detection has not been considered so far. Our findings provide a contribution to the effective implementation of integrating ENS in real-world applications.

6.5 Summary

In this chapter, we analyzed the influence of event detection methods and time systems on the accuracy of the event composition. We discussed six methods to observe events and set timestamps while using active and passive observers. The different time systems typically used by event notification services have been studied briefly. The interplay of event observation and timestamping methods with the time system of the ENS has been analyzed. Special emphasis has been given to the aspects of event detection and ordering, the occurrence times of events, the order of events, and the completeness of event information obtained by the service. Selected results of our analysis have been published in [Hin01].

We proposed a strategy for the handling of timestamps of events that helps to minimize the temporal delays and to increase the event detection accuracy. This strategy describes a new interval-based ordering mechanism for events in a open distributed network that covers active and passive event observation. Additionally, we proposed three extensions to the time systems used by event notification systems. The extensions lead to higher accuracy in the detection of primitive and composite events. In that way, an ENS with those extensions supports the accurate integration of event information from different sources. The analysis in this chapter and the proposed timestamp handling answer to the requirements R2 and R5 as defined in Chapter 2 at Page 19. The implementation of our event handling strategy and time system is discussed in Chapter 8.