STUDIA INFORMATICA 2011
Volume 32 Number 2B (97)

Artur OPALIŃSKI
Gdańsk University of Technology, Faculty of Electrical and Control Engineering

DISTRIBUTED REPRESENTATION OF INFORMATION ON CYCLIC EVENTS

Summary. A representation of information on cyclic events has been proposed which is advantageous for computing environments where a distributed set of Receivers reacts to cyclic events generated by distributed sources. In such scenario no immanent central information repository exist on event timing or volume. Receivers are able to learn the event cycles without communicating with each other, merely on the basis of the fact that an event at a given instant of time has or has not been acted upon by other Receivers.

Keywords: distributed processing, cyclic events, multi-agent systems

ROZPROSZONA REPREZENTACJA INFORMACJI O ZDARZENIACH CYKLICZNYCH

Streszczenie. W artykule zaproponowano sposób reprezentacji informacji o zdarzeniach zachodzących cyklicznie, przydatny dla środowisk, w których rozproszony zbiór Odbiorników obsługuje cykliczne zdarzenia generowane przez rozproszone źródła. Mimo braku scentralizowanej informacji o ilości i czasie występowania zdarzeń, Odbiorniki wykrywają cykliczność zdarzeń bez potrzeby komunikacji między sobą, a jedynie na podstawie informacji, że zdarzenie zostało lub nie zostało obsłużone przez inne Odbiorniki.

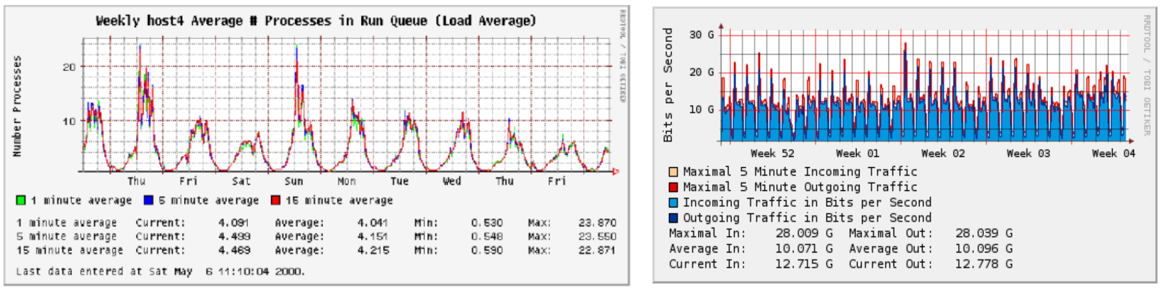
Słowa kluczowe: przetwarzanie rozproszone, zdarzenia cykliczne, systemy wieloagentowe

1. Introduction

In some computing environments distributed set of Receivers reacts to events originating from distributed sources. Such working schema is very typical and basic eg. in [1, 2, 3]. No immanent central repository usually exists with the information on volume, type and timing of events. Information on individual event patterns would intrinsically be averaged during aggregation and requires Fourier analysis or other forms of harmonics analysis [4] to restore it. Also collecting distributed information and disseminating results of processing may be challenging in distributed environments [5].

The natural cycles of human activity and technical factors may cause events to repeat in a cyclic manner in some scenarios, as depicted in Fig. 1, especially if the sources are numerous and diversified.

In this paper a mechanism for distributed repository of information on such events is researched. This mechanism allows for learning, representing and updating the information on timing and volume of events independently by individual Receivers. Receivers are able to achieve this goal without explicit communication, just based on timing analysis of events received by each participating Receiver.



```
1 minute average   Current:   4.091   Average:   4.041   Min:   0.530   Max:   23.870
5 minute average   Current:   4.499   Average:   4.151   Min:   0.548   Max:   23.550
15 minute average  Current:   4.469   Average:   4.215   Min:   0.590   Max:   22.871

Last data entered at Sat May  6 11:10:04 2000.
```

```
Maximal In:    28.009 G  Maximal Out:   28.039 G
Average In:    10.071 G  Average Out:   10.096 G
Current In:    12.715 G  Current Out:   12.778 G
```

Fig. 1. Examples of cyclic events addressed in this paper, manifesting some daily and weekly periodicity [6]

Rys. 1. Przykłady cyklicznych zdarzeń rozważanych w artykule, zachodzących w rytmach dobowym i tygodniowym [6]

2. Solution

The solution proposed in this paper considers cyclic events. It is based on the assumption that when similar events occur in higher quantities then they can be represented by almost periodic functions [7], specifically that events feature some time sequence during period T, and that such similar yet not identical sequences repeat with a period being a multiply of T. The period T is sometimes called “almost-period”.

The events are received separately even when they occur in higher quantities, so timing and other properties of each individual event can be easily identified by Receivers. For this to work Receivers should be software agents that are capable of learning the pattern of events they acted upon. It is assumed that Receivers are capable to react to at most one event at once. The following is considered crucial for the Receiver to learn the pattern of events:

- events should appear cyclically for the given Receiver,

- Receiver must posses an internal representation of cyclic events, encompassing many different cycles revealed during a common stretch of time,
- events which prove not to be cyclic in the time regarded should be forgotten as mere incidents. While these events will not be solidly learned, this does not preclude that they will be acted upon accordingly by Receiver.

The example Receiver internal representation of cyclic events is described below. Each Receiver has two attributes:

- the common almost-period T assumed for all events this Receiver encounters,
- the multiply n of the almost-period T. The value $T{\cdot}n$ is the period of all similar event sequences regarded.

Each Receiver attribute values may derive from its capabilities and are assumed constant. The Receiver will detect and separately represent internally all event cycles with periods $k{\cdot}T$, $1 \le k \le n$. All events not belonging to any of these cycles will be treated as random incidents by this Receiver, meaning they will be acted upon accordingly by Receiver but not recognized as cyclic and not solidly learned by this Receiver.

Each Receiver tracks discrete absolute time t and calculates its internal time t_R modulo $T{\cdot}n$:

$$t_R \equiv t \pmod{T{\cdot}n} \tag{1}$$

Every time a Receiver encounters an event at internal time instant t_{RE}, it treats the event as cyclic. If this event does not belong yet to any known cycle, Receiver assumes that the cycle of this event started during previous $T{\cdot}n$ period, at time instant:

$$t_{B\,prev} = T \cdot n - \left(\left(\left\lfloor \frac{t_{RE}}{T} \right\rfloor + 1 \right) \cdot T - t_{RE} \right) = T \cdot \left(n - \left\lfloor \frac{t_{RE}}{T} + 1 \right\rfloor \right) + t_{RE} \tag{2}$$

of that period, i.e. in the last almost-period T of the previous $T{\cdot}n$ period. Therefore the Receiver calculates event base time t_{BE} and event almost-period multiply k_E using the floor function denoted as $\lfloor\ \rfloor$ and stores the resulting values as:

$$k_E = \left\lfloor \frac{t_{RE}}{T} \right\rfloor + 1 \quad , k_E < n \tag{3}$$

$$t_{BE} = t_{RE} \quad , t_{BE} < n \cdot T$$

The internal representation of cyclic events by the Receiver comprises only pairs of almost-period multiply k_E and base time t_{BE} for all cycles learned so far by this very Receiver. This allows for easy determination if an event received belongs to an already known cycle. In case of receiving an event at time instant belonging to a known cycle, only the corresponding base time t_{BE} is updated with the Receiver current internal time $t_R = t_{RE}$.

Example a) in fig. 2 presents events constituting a cycle with k_E=1 and initial t_{BE} in the range $0 \leq t_{BE} < T$. Events in all figures are represented by Kronecker delta which is described for discrete time instants i,j as:

$$\delta_{i,j} = \begin{cases} 1, & if\ i = j \\ 0, & if\ i \neq j \end{cases} \tag{4}$$

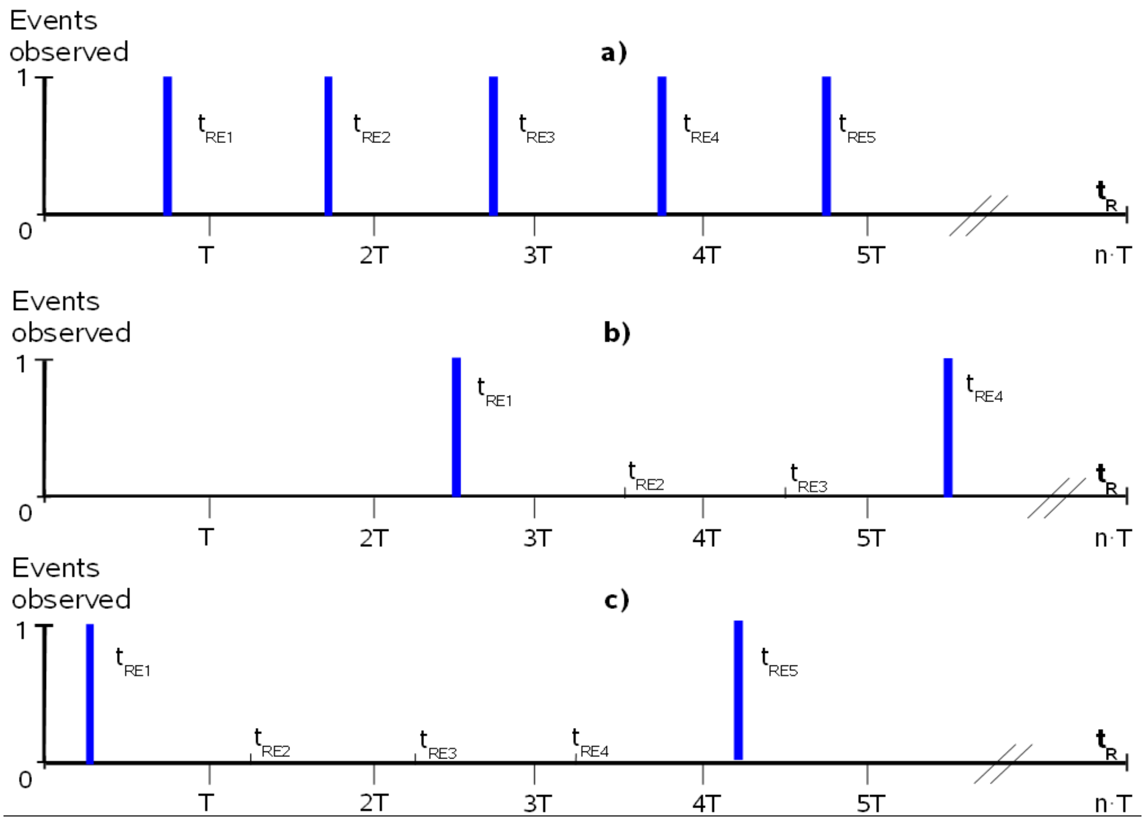


Fig. 2. Example event cycles observed by a Receiver: a) k_E=1 and initial t_{BE} in the range $0 \leq t_{BE} < T$, b) k_E=3 and initial t_{BE} in the range $2T \leq t_{BE} < 3T$, c) k_E=4 and initial t_{BE} in the range $0 \leq t_{BE} < T$

Rys. 2. Przykłady cykli zdarzeń z punktu widzenia Receivera: a) k_E=1 i początkowa wartość t_{BE} w zakresie $0 \leq t_{BE} < T$, b) k_E=3 i początkowa wartość t_{BE} w zakresie $2T \leq t_{BE} < 3T$, c) k_E=4 i początkowa wartość t_{BE} w zakresie $0 \leq t_{BE} < T$

At the instant t_{RE1} the first event is noted by the Receiver. According to (3) this event is assumed to belong to a new cycle with k_E=1 and $t_{BE} = t_{RE1}$. All later encounters with events fitting this known cycle at time instants t_{RE2} , t_{RE3} ,t_{RE4} , etc. results in updating t_{BE} only.

Example b) in Fig. 2 presents events constituting a cycle with k_E=3 and initial t_{BE} in the range $2T \leq t_{BE} < 3T$. At the instant t_{RE1} the first event is noted by the Receiver. As in the example a) according to (3) this event is assumed to belong to a new cycle with k_E=3 and $t_{BE} = t_{RE1}$ end the following events in this cycle result in updating t_{BE} only.

If an expected event of a known cycle has been missed, it is assumed that the formerly received events of this cycle have been mistaken as belonging to a common cycle. It is assumed instead that these events belonged to separate cycles with common longer period but with different base times t_{BE}. So new cycle representations are created for each of these past events instead of a common one. The common almost-period multiply k_E corresponding to these cycles gets incremented, while each new i-th cycle gets a different base time t_{BE} :

$$k_E := \left\lfloor \frac{t_{RE}}{T} \right\rfloor + 1 \quad , if\ k_E < n \tag{5}$$

$$t_{BEi} := t_{RE} - i \cdot T \quad , i = 1\ ..\ \left\lfloor \frac{t_{BE}}{T \cdot k_E} \right\rfloor .$$

If the resulting almost-period multiply k_E would exceed $T{\cdot}n$ then it is assumed that the events were non-periodic incidents or that they belong to cycles with period exceeding $T{\cdot}n$. Such cycles will not be stored any longer, but this does not preclude the Receiver usual reaction to the event received.

Example c) in Fig. 2 presents events constituting a cycle with k_E=4 and initial t_{BE} in the range $0 \leq t_{BE} < T$. At the instant t_{RE1} the first event is noted by the Receiver and is erroneously assumed based on (3) to belong to an identical cycle as in example a) in Fig.2, i.e. a cycle with k_E=1 and $t_{BE}= t_{RE1}$. When no expected event of this cycle appears at t_{RE2} Receiver recalculates the values according to (5), achieving k_E=2 and $t_{BE}= t_{RE1} -T$ (possibly in previous $n{\cdot}T$ period). At time instants t_{RE3} , t_{RE4} , Receiver recalculates the values according to (5) again arriving finally at the correct values k_E=4 and $t_{BE}= t_{RE1}$.

Priority queue mechanism can be used to ensure that events belonging to a known cycle will appear cyclically for any given Receiver. To achieve that, any Receiver expecting event of a known cycle enqueues with the highest priority exactly for the expected time instant of the event. Highest priority Receiver will be dequeued and notified on event. Therefore, the Receiver will not miss the event as long as the event happens.

A Receiver not expecting any event enqueues with the lowest priority to only learn new cycles not tapped by other Receivers. Enqueueing at the given priority level must be an atomic operation.

3. Verification

The approach has been verified with a C++ program published on the authors web page [8]. The verification program implements the part of Receiver code pertaining to event representation, event Generator, queue mechanism and a perfect discrete time. The time is perfect in the sense that time tics always differ by 1 and that every time tic is observable by all the program components.

Verification has been conducted with Receivers with identical T and n attribute values.

Whenever a Receiver

- missed an expected event,
- or encountered an expected event,
- or encountered an unexpected event

it presented its number and status, current global time t, Receiver time t_R, and event cycle attributes (t_{BE} and k_E).

The verification program has been validated first against simulated single and double event cycles similar to those presented in fig.2 using the values presented in table 1.

Table 1

Validation data

Receiver			Event cycles	
Number of Receivers and event cycles (R)	T	n	k_E	initial t_{BE}
1	5	2	1..10	0..(k_E-1) · T
2	6	3	k_{E1}=2 , k_{E2}=3	0..(k_{Ei}-1) · T

After successful validation the program has been run with the values presented in table 2 in order to verify the approach. It has been assumed that operating without the need to learn new cycles with periods between T and $n{\cdot}T$ after time three times longer than the multiply of all cycles involved is an adequate positive verification result.

Table 2

Verification data

Receiver			Event cycles	
Number of Receivers and event cycles (R)	T	n	k_E	initial t_{BE}
1	5	2	1..12	0..(k_E-1) · T
2	5	3	k_{E1}=2 , k_{E2}=3	0..(k_{Ei}-1) · T + random value from (0,T)
10	5	2	k_{Ei}=i , i=1..10	random value from (0,T)
10	5	7	k_{Ei}=i , i=1..10	0..(k_{Ei}-1) · T + random value from (0,T)

Verification results were positive for every combination listed in Table 2. The Receivers were able to detect and learn all cycles needed to represent the events observed. Event cycles with periods shorter than T and event cycles with periods longer than $n{\cdot}T$ have not been solidly learned as expected.

4. Conclusion

It has been verified that Receivers are able to independently represent event cycles without communicating with each other and only based on the information on events they observed themselves. Through attributes the representation method allows to control how Receivers share information on event cycles.

The approach requires that events are not directed to Receivers stochastically, so a queue mechanism is used instead. The influence of different queuing methods and the influence of other cycle representations needs additional research.

Representation of detected event cycles could be fed directly to event prediction, e.g. to provide resources timely or to resolve conflicts if events from different cycles would line up for one Receiver at the same instant of time. Further research needs to address these challenges.

It is worth noting that event cycles need not be correlated with time. It is conceivable to correlate the events reception with any other discrete value which is known to all Receivers. This has not been explored yet.

Future work could also be devoted to analyzing the appropriateness of the suggested event cycles representation to filtering events based on their periods.

BIBLIOGRAPHY

1. Berson A.: Client/server architecture. McGraw-Hill, Inc., New York 1996.
2. Huhns M. N., Singh M. P., Burstein M., Decker K., Durfee K. E., Finin T., Gasser, T. L., Goradia H., Jennings P. N., Kiran L., Nakashima H., Van Dyke P. H., Rosenschein J. S., Ruvinsky A., Sukthankar G., Swarup S., Sycara K., Tambe M., Wagner T., Zavafa L.: Research directions for service-oriented multiagent systems. IEEE Transactions on Internet Computing, Vol. 9, Issue 6, 2005.
3. Bailey D., Wright E.: Practical SCADA for Industry. Newnes, 2003.
4. Smith S. W.: The Scientist & Engineer's Guide to Digital Signal Processing. Technical Pub., California 1997.
5. Jordan H. F., Alaghband G.: Fundamentals of Parallel Processing, Prentice Hall, 2002.
6. Random google.com image search with “weekly server load”.
7. Stoiński S.: Funkcje prawie okresowe. Wydawnictwo Naukowe UAM, Poznań 2008.
8. Verification program URL, https://sites.google.com/site/flecabinet/downloads/BDAS11-.zip.

Recenzenci: Dr inż. Dariusz R. Augustyn
Dr inż. Katarzyna Harężlak

Wpłynęło do Redakcji 31 stycznia 2011 r.

Omówienie

Wiele środowisk przetwarzania architektury do rozproszonego przetwarzania równoległego, na przykład szeroko stosowana architektura klient-server, czy układy sterowania i monitoringu wykorzystują pojęcie zdarzenia, które jest obsługiwane przez przeznaczone do tego elementy, określane w niniejszym artykule jako Odbiorniki (*Receivers*). Ponieważ zarówno źródła zdarzeń, jak i obsługujące je Odbiorniki są rozproszone, często brak jest scentralizowanej informacji na temat ilości, rozłożenia w czasie i typach zdarzeń. Utrudnia to planowanie obsługi tych zdarzeń. Jednocześnie tworzenie scentralizowanych zbiorów informacji na temat licznych zdarzeń wymaga dość intensywnej komunikacji oraz może prowadzić do agregowania danych dotyczących zdarzeń, które to dane następnie trzeba analizować, np. metodami DSP [4].

W niniejszym artykule proponuje się rozproszoną reprezentację przechowywania informacji o pewnych cyklicznych (patrz rys. 1) zdarzeniach. Sposób analizy informacji o zdarzeniach i jej reprezentacji nie wymaga od Odbiornika wiedzy o innych zdarzeniach niż zdarzenia obsłużone przez niego samego. Umożliwia to unikniecie dodatkowej komunikacji, a jednocześnie sam fakt obsłużenia lub nieobsłużenia zdarzenia stanowi informację wspólną dla zbioru Odbiorników. Proponowana metoda pozwala na gromadzenie rozproszonej wiedzy. Pomyślną weryfikację metody przeprowadzono, tworząc program weryfikujący [8] w języku C++.

Address

Artur OPALIŃSKI: Politechnika Gdańska, Wydział Elektrotechniki i Automatyki, ul. Narutowicza 11/12, 80-233 Gdańsk, Polska, Artur.Opalinski@pg.gda.pl.