Endlessly Circulating Messages in IEEE 1588-2008 Systems

David Broman, Patricia Derler, Ankush Desai, John C. Eidson and Sanjit A. Seshia
University of California, Berkeley, CA, USA

***Abstract*—This paper studies conditions where messages endlessly circulate in a system of IEEE 1588-2008 clocks. The study is based on two independent analysis techniques. One uses a discrete event simulation environment for describing the operation of the best master clock (BMC) algorithm in IEEE 1588-2008. The second uses a model checking tool. We discuss several cases illustrating conditions under which circulating messages occur and the effectiveness of measures to squelch these messages. This paper demonstrates that one or more of the squelching mechanisms must be implemented.**[1]

***Index Terms*—Discrete event simulations, IEEE 1588-2008, Modeling, Rogue messages**

I. Introduction

This paper studies conditions where messages endlessly circulate in a system of IEEE 1588-2008 clocks. IEEE 1588-2008 [1] is a protocol that synchronizes clocks in a distributed system. It is widely used in the telecommunications [2], industrial [3], power [4], and test [5] industries.

The possibility of messages endlessly circulating in a communications system containing loops, also referred to as *rogue* messages, is not new and techniques have been devised to detect and squelch these messages [6]. To date there are no detailed studies of conditions under which this problem might occur in an IEEE 1588 timing system.

The study is based on two independent analyses. One uses a discrete event simulation for modeling the operation of the best master clock (BMC) algorithm in IEEE 1588-2008, *the standard*. The second uses a model checking tool.

In section II we present the basics of the BMC algorithm and the measures in IEEE 1588 designed to prevent rogue messages from occurring. In section III we describe the two independent analysis methods used in this study. In section IV we present several cases illustrating conditions under which rogue messages occur and the effectiveness of measures to squelch these messages. Section V summarizes the conclusions of the study and makes recommendations for future work.

[1]Broman, Derler and Eidson's work was supported in part by the iCyPhy Research Center (Industrial Cyber-Physical Systems, supported by IBM and United Technologies), and the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley (supported by the National Science Foundation, NSF awards #0720882 (CSR-EHS: PRET) and #0931843 (ActionWebs), the Naval Research Laboratory (NRL #N0013-12-1-G015), and the following companies: Bosch, National Instruments, and Toyota). Eidson also received support from Calnex Solutions. Desai and Seshia's work was supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

II. IEEE 1588-2008 BMC algorithm basics

In IEEE 1588-2008 the communication topology is established by a distributed algorithm, the BMC algorithm, that creates a spanning tree with the best clock at the root of the tree. The relevant specifications of the BMC algorithm are found in section 9.3.2 of the standard [1]. The BMC algorithm has two functions: the election of the best clock in the system as the *grandmaster* which will be the root of the spanning tree and the source of time for the system, and the creation of the spanning tree. The algorithm is completely distributed and operates based on data contained in *Announce messages* which are exchanged between ports on the clocks in the system and on locally maintained data describing each clock. Announce messages contain data characterizing the clock that the sending port considers the best clock in the system and a measure, *stepsRemoved*, of the distance of the sending port from the presumed best clock. These data are, in order of precedence in determining the best clock, the *grandmasterPriority1*, *grandmasterClockQuality*, *grandmasterPriority2*, *grandmasterIdentity*, and *stepsRemoved*. All are specified in section 13.5 of the standard.

Once the timing topology is established by the BMC algorithm, each clock synchronizes to its master by exchanging *timing* messages between connected ports. This paper is not concerned with this aspect of the protocol.

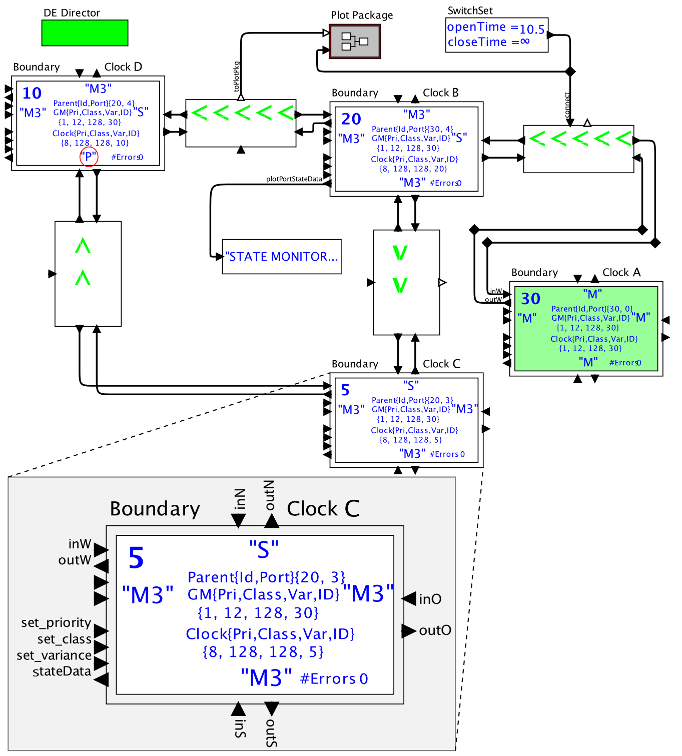
The principal operations of the BMC algorithm are:

- Announce messages are periodically exchanged between ports with the period set by the parameter, *Announce Interval*, AI.
- At each port the best of the Announce messages received during the last AI are compared using the BMC algorithm *dataset comparison algorithm (DCA)* to determine the best of these messages $e_r best$, see Figures 27 and 28 of the standard.
- Each clock then uses the DCA to determine the best Announce message received by the clock, i.e., $ebest$.
- On each port the DCA is used to compare both $e_r best$ and $ebest$ to locally maintained data characterizing the clock and then, based on Figure 26 of the standard, determining the state of the port.

Devices based on the standard have been deployed in the field since 2008, and since 2002 based on an earlier version with a similar BMC algorithm, with no known cases where the algorithm failed.

During the creation of the standard there was considerable discussion on the possibility of rogue messages occurring. As a result, three potential squelching mechanisms are specified in the standard: a special pre-master state with a duration based on the value of the stepsRemoved field in the received Announce message, the foreign master mechanism, and a stepsRemoved threshold value above which the message will be discarded. There are three types of clocks specified in the standard: *ordinary* clocks, *transparent* clocks, and *boundary* clocks. Ordinary clocks have a single stateful port and will be either a source or sink for time. Transparent clocks have multiple ports and provide corrections to timing messages reflecting the time these messages spent traversing the transparent clock. Transparent clocks have no effect on the operation of the BMC algorithm. Boundary clocks have multiple stateful ports. The stepsRemoved attribute of Announce messages is incremented by one at each boundary clock thus providing a measure of the distance from the presumed grandmaster.

Rogue messages are believed to exist only in cyclic systems so that an ever increasing stepsRemoved attribute should indicate their presence. In normal operation the BMC algorithm reduces a cyclic topology to a spanning tree. The issue is whether there are conditions where rogue messages can be generated for example due to reconfiguration of the topology due to a new communication path being established, and old path broken, or by a change in the attributes of one of the clocks.



DE Director
Plot Package
SwitchSet
openTime =10.5
closeTime =∞
toPlotPkg
connect
Boundary Clock D
10 "M3" "M3" Parent{Id,Port}{20, 4} GM{Pri,Class,Var,ID} "S" {1, 12, 128, 30} Clock{Pri,Class,Var,ID} {8, 128, 128, 10} "P" #Errors0
Boundary Clock B
20 "M3" "M3" Parent{Id,Port}{30, 4} GM{Pri,Class,Var,ID} "S" {1, 12, 128, 30} Clock{Pri,Class,Var,ID} {8, 128, 128, 20} "M3" #Errors0
plotPortStateData
"STATE MONITOR...
Boundary Clock A
30 "M" "M" Parent{Id,Port}{30, 0} GM{Pri,Class,Var,ID} "M" {1, 12, 128, 30} Clock{Pri,Class,Var,ID} {1, 12, 128, 30} "M" #Errors0
inW
outW
Boundary Clock C
5 "S" "M3" Parent{Id,Port}{20, 3} GM{Pri,Class,Var,ID} "M3" {1, 12, 128, 30} Clock{Pri,Class,Var,ID} {8, 128, 128, 5} "M3" #Errors 0
Boundary
inN
outN
Clock C
inW
outW
set_priority
set_class
set_variance
stateData
5
"S"
"M3"
Parent{Id,Port}{20, 3}
GM{Pri,Class,Var,ID}
"M3"
{1, 12, 128, 30}
Clock{Pri,Class,Var,ID}
{8, 128, 128, 5}
"M3" #Errors 0
inO
outO
inS
outS

Fig. 1. Model of a simple network of clocks.

III. Analysis of the BMC Algorithm

To study this problem two very different engines were used to analyze the BMC algorithm and to explore both the normal operation as well as to discover conditions causing rogue messages. This section provides a brief description of each engine.

A. The Ptolemy II Simulation

Ptolemy II is a general actor model based simulation engine developed at the University of California at Berkeley [7]. Ptolemy allows simulations incorporating different models of computation such as continuous, discrete event, and synchronous reactive. In modeling the BMC algorithm, the *discrete event (DE)* model of computation is used. The model is quite general and allows us to explore options, such as eliminating the pre-master state, that are being considered by the P1588 standards group for inclusion in the upcoming edition.

The DE model implements all aspects of the BMC algorithm: the exchange of Announce messages, the determination of port states, and the maintenance of BMC algorithm related data in the datasets described in the standard. The only abstraction is that of the BMC algorithm attributes contained in Announce messages and compared by the DCA, only the grandmasterPriority1, grandmasterClockClass, grandmasterVariance, and grandmasterIdentity attributes are modeled. The grandmasterVariance is used as a proxy for the other attributes inferior to grandmasterClockClass. The grandmasterClockClass and grandmasterVariance are members of the grandmasterClockQuality attribute.

The model itself is too complex to describe in detail in this paper. In all the simulations we have conducted, the results are consistent with the expected behavior. Figure 1 shows a screenshot of the top level of the simulation model for the initial example discussed in section IV.

The principal features shown in the figure from top to bottom are (names in *italics* correspond to the actor names in the figure):

- A *Plot package* actor for displaying messages and events,
- A *SwitchSet* actor that permits opening or closing of a connection at specified times,
- Connector actors (no names but connect the Boundary Clocks) that allow changing the topology and provide an interface for observing messages,
- An actor *DE Token Monitor3* that displays clock attributes as a function of time,
- Four *Boundary Clocks* each showing the state of its ports and the values of attributes priority1, clockClass, clockVariance, clockIdentity both for the presumed grandmaster, the upper set, and for itself, the lower set in the clock icons. The green clock icon indicates the current grandmaster of the system. Each clock is identified by a letter, i.e., A, B, C, and D. The number in the upper left corner of each icon is the clockIdentity.
- The *DE Director*: The Ptolemy II component that determines the model of computation by controlling the firing and computation rules for each actor.

- The port state of each clock. The states relevant for this study are M and M3 for master, S for slave, and P for passive. Only ports in the master state issue Announce messages.

B. Verification Using Model Checking

Simulation-based tools can, for a specific test scenario, explore a single execution trace. The approach described in previous section simulated a few scenarios that can potentially lead to rogue messages. However, this approach cannot consider all possible environmental non-determinism and ambiguities that may exist in the standard.

Model checking is an automated, algorithmic method to systematically explore all possible executions of a model to check if it satisfies the required specification. We used model checking to exhaustively explore the state space of the abstract BMC model in the presence of scheduling non-determinism and clock dynamics. We used P [8], a domain-specific language for writing protocols. Models in P are collection of state machines interacting by exchanging messages. We created an abstract model of the BMC algorithm using P. Each clock and port is modeled as a separate state machine in P. Timeouts in the protocol were modeled as non-deterministic choice operations. The asynchronous execution of clock and port state machines is captured by exploring all possible interleavings of these state machines.

To verify the presence of rogue messages in situations where network topology changes (due to creation or deletion of links), we created a separate *failure* state machine that breaks a link between two nodes. The *failure* machine is non-deterministically interleaved to inject link failures at different points in the evolution of the BMC algorithm.

The property we checked was that for the 4-clock configuration of Figure 5, there exists an execution trace where the stepsRemoved value in a message exceeds 15. For this clock configuration the presence of such a trace indicates the presence of a rogue message; see the discussion of Figure 2 in Section IV-A. We checked this property on the composition of the abstract BMC model and the failure machine. The model checker automatically generated a counterexample that confirms the presence of messages with increasing stepsRemoved value. We also noticed that the messages were circulating in clockwise direction in the ring.

IV. Analysis results

Using our analysis engines we have determined several conditions that lead to rogue messages. While most of the examples have been examined using the Ptolemy-based model, in all cases where we have used both engines the results agree. In particular the results agree for the example extensively discussed in the Appendix.

The processes that lead to rogue messages are quite complex. We describe one example in some detail and present the results of several more. In all of these examples, the announce interval, i.e. the time between announce messages, is 1 second.

A. Simple Disconnect Model

In this section we discuss the behavior of the system of Figure 1 when clock A, the grandmaster, is disconnected, such that beginning with the announce interval starting at time 11, messages from clock A no longer reach the East port of clock B. Clock A initially became grandmaster because its priority1 value is less than that of any other clock in the system. Clock C is the second best clock by virtue of having a lower clockIdentity than clocks B or D. In this simulation the announce receipt timeout, ART, is 3 announce intervals and neither the pre-master state nor the foreign master features are turned on. The pre-master feature requires that a port wait for stepsRemoved announce intervals prior to entering the master state. The *foreign master threshold (FMT)* requires that Announce messages received on a port be present for a parameterized number of announce intervals before being considered. The squelch limit on rogue message stepsRemoved is set at 12 rather than the 255 specified in the standard.

Figure 2 plots the gmIdentity and stepsRemoved values in Announce messages sent from clock D to clock B, i.e., in the clockwise direction and with time measured in seconds.

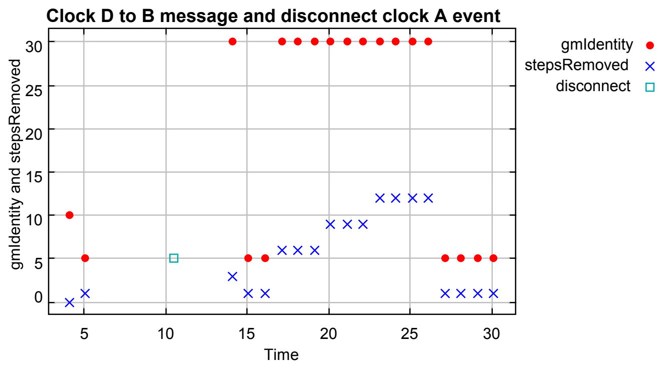


Fig. 2. Clockwise message in the simple disconnect model, ART=3.

The crosses in the figure show the Announce message stepsRemoved values increasing by 3, the number of boundary clocks in the cycle, starting a short time after the disconnect at time 11. The dots show values of the grandmasterClockIdentity field of the Announce messages. Note that this value of 30 references Clock A which is no longer part of the system after time 11. Also shown is the squelching of the rogue messages when the stepsRemoved value reaches 12 causing the system to reconfigure such that Clock C with identity = 5 is the grandmaster.

Figure 1 illustrates the configuration at time = 9 just prior to the disconnect of Clock A. At that time the network is in a stable condition with clock A the grandmaster and the cycle broken by the BMC algorithm by the South port on clock D being in the passive state and therefore not transmitting Announce messages. Note that in clocks B, C, and D the identity field shown in the middle row, labeled GM, of the Figure 1 clock icons is 30 corresponding to the identity of clock A. Clearly the rogue messages are generated during the transient when the network is reconfiguring after the disconnect of clock A. For those interested, the details of this process are discussed in the Appendix.

For this example, if the *foreign master threshold (FMT)* value is 2 rather than 0, then no rogue message is generated. Likewise, if FMT is 0, but the pre-master state mechanism of the standard is turned on, no rogue message is generated.

B. *More Complex Disconnect Models*

We have also investigated networks containing more than the three clocks in the cycle of the initial example shown in Figure 1. The additional clocks were added to the cycle between clocks C and D of Figure 1. The simulation results are shown in Table I.

Announce Receipt Timeout Values (in units of announce interval timeouts)						
2	3	4	5	6	7	12
3 Clocks in the cycle: ratio and direction						
2:1CW	3:0CW	3:0CW	3:0CW	3:0CW	3:0CW	3:0CW
4 Clocks in the cycle No rogue messages generated						
5 Clocks in the cycle						
2:3CW	3:2CW	4:1CW	5:0CW	5:0CW	5:0CW	5:0CW
6 Clocks in the cycle No rogue messages generated						
7 Clocks in the cycle: ratio and direction						
2:5CW	3:4CW	4:3CW	5:2CW	6:1CW	7:0CW	7:0CW

TABLE I
SUMMARY OF DISCONNECT MODEL BEHAVIORS

In Table I the values of the announce receipt timeout are shown for each of the columns. The table is divided into 5 sections each representing a system with different numbers of boundary clocks in the cyclic path. In each section the row entries of the form x:yCW indicate that there are x number of messages referencing the disconnected clock A circulating in the clockwise direction for every y messages referencing the best clock remaining in the cycle after the disconnect, i.e., clock C. These values were observed on the link between Clocks D and B. In all cases the simulation was run three times. Once with the rogue limit = 30 but with the FMT and pre-master features disabled. The second and third runs were respectively with FMT = 2, and the pre-master feature enabled. In all cases these mechanisms squelched rogue messages. The simulations for topologies with an even number of clocks failed to generate rogue messages for the values of announce receipt timeout tested. Note that this failure to generate rogue messages did not involve any of the squelch mechanisms but is a property of the configuration.

As noted in the detailed analysis in the Appendix, small changes in the clock properties can result in rogue messages circulating in the counter clockwise direction. As noted in the table it is also possible for circulating messages to consist a combination of rogue messages, i.e., those referencing the disconnected clock and messages referencing the best remaining clock in the system. For example, Figures 3 and 4 show the announce message values of grandmasterClockIdentity and stepsRemoved for the case where there are 5 clocks in the system and ART = 3 (second column in the 5 clock section of Table I). Here the ratio is 3:2 with the three consecutive rogue messages referencing Clock A with identity 30 followed by two consecutive messages referencing Clock B, identity 5, the best clock remaining in the system after the disconnect. In both figures, time is measured in seconds.

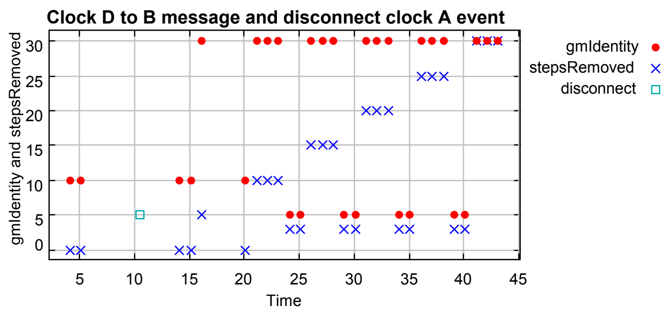


Fig. 3. Clockwise Messages with 5 clocks and ART = 3 in the simple disconnect model.

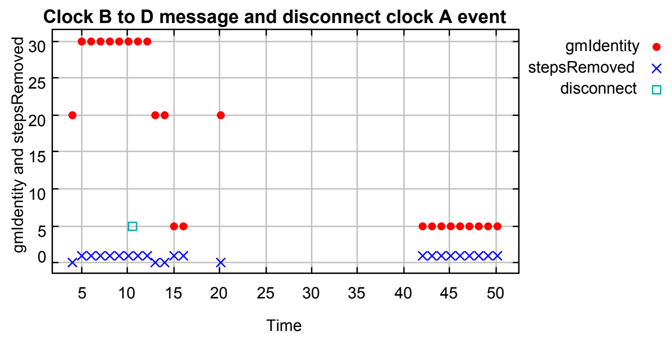


Fig. 4. Counterclockwise Messages with 5 clocks and ART = 3 in the simple disconnect model.

C. *Discussion of Results*

From the data presented in sections IV-A and IV-B the following observations can be made on the disconnect models shown:

- There are definitely conditions that spawn rogue messages. For the models studied it appears that these occur in cycles that include odd numbers of clocks even though these cycles may be broken initially by the action of the BMC algorithm setting one port in the cycle in the passive state.
- There are conditions where rogue messages apparently do not occur, e.g. for cycles that include even numbers of clocks, at least over the range studied.
- Depending on the details of the clock properties rogue messages can circulate in either direction.
- Depending on the relative values of the announce receipt timeout and the number of clocks, the rogue messages can consist of only messages referencing the disconnected clock or may consist of a mixture of these and messages referencing the remaining best clock in the cycle.
- In all cases studied so far where rogue messages can

be generated, the stepsRemoved-based rogue message limit mechanism successfully squelches these messages. In all these cases the presence of either the foreign master mechanism with threshold = 2 or the pre-master mechanism effectively prevents rogue messages from being established.
- Rogue messages are a definite hazard to the protocol. When rogue messages are circulating the referenced grandmaster is not present in the system. In addition with configurations such as discussed in reference to Figures 3 and 4, the network configuration is not even static but changes with time. Any timing messages issued based on such configurations would not produce a stable timescale.

It should be noted that these observations provide existence examples. We cannot prove that the results noted extend beyond the examples shown. Likewise we cannot conclude that the results for the BMC algorithm of the standard will hold for other algorithms. However the existence examples do indicate that squelching mechanisms must be provided in the standard.

As noted in section II, to our knowledge no cases of rogue messages have been reported. We pose several plausible reasons for this:

- The first and most obvious is that rogue messages do not occur in all topologies and conditions.
- Essentially all of the fielded applications are thought to implement both the foreign master and pre-master mechanisms, which at least for the examples presented here, squelch rogue messages.
- Finally the BMC algorithm of the standard operates on top of whatever topology exists. For example in simple installations such as a laboratory setup, the physical connections are likely to be tree structured and it seems reasonable to assume that in non-cyclic topologies rogue messages cannot occur. In other cases underlying spanning tree algorithms, e.g. at layer 2 in an Ethernet, may have reduced the standard's BMC algorithm operations to a non-cyclic environment.

V. Conclusions

We have provided existence examples of rogue messages endlessly circulating in plausible topologies and circumstances. We have also shown that the mechanisms currently in the standard are successful in squelching these rogue messages. There is serious discussion in the standards committee responsible for IEEE 1588 of eliminating the pre-master and possibly the foreign master mechanisms. The former since it exacts an N^2 penalty on reconfiguration time and the latter due to its complexity. This study demonstrates that these suggestions need careful evaluation. Certainly it should be a requirement to retain the stepsRemoved limit mechanism.

References

[1] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std. 1588-2008*.

[2] J.-L. Ferrant, M. Gilson, S. Jobert, M. Mayer, L. Montini, M. Ouellette, S. Rodrigues, and S. Ruffini, *Synchronous Ethernet and IEEE 1588 in Telecoms: Next Generation Synchronization Networks*. John Wiley & Sons, 2013.

[3] K. Harris, "An application of ieee 1588 to industrial automation," *Precision Clock Synchronization for Measurement, Control and Communication*, pp. 71–76, 2008.

[4] D. Broman, P. Derler, and J. Eidson, "Temporal issues in cyber-physical systems," *Journal of the Indian Institute of Science*, vol. 93, no. 3, pp. 389–402, 2013.

[5] N.-J. Jacobsen, "Lan-xi–the next generation of data acquisition systems," in *Structural Dynamics, Volume 3*. Springer, 2011, pp. 821–829.

[6] F. De Pellegrini, D. Starobinski, M. G. Karpovsky, and L. B. Levitin, "Scalable cycle-breaking algorithms for gigabit ethernet backbones," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4. IEEE, 2004, pp. 2175–2184.

[7] C. Ptolemaeus, "System design, modeling, and simulation using Ptolemy II," *Ptolemy. org*.

[8] A. Desai, V. Gupta, E. K. Jackson, S. Qadeer, S. K. Rajamani, and D. Zufferey, "P: Safe asynchronous event-driven programming," in *Proceedings of PLDI*, 2013.

VI. Appendix

The detailed state transitions and messages passed during the reconfiguration of the network of Figure 1 are shown in Figures 5 to 11. In these figures the port states and dataset values are those established based on the Announce message of the previous AI. The local clock datasets are D: x,y,z while the parent datasets (the dataset reflecting the presumed grandmaster) are P:x,y,z where x is the priority1, y is the identity, and z is the stepsRemoved value. The messages shown are those sent based on these states and dataset values and which will determine the states and datasets of the next AI. Note that the time = τ_i parent dataset value for stepsRemoved is one greater than the value in the previous incoming Announce message on the port, i.e., at the time = τ_{i-1}.

Figure 5 shows the first two AI after the disconnect (time 11 and 12) hence the dotted line connecting clocks A and B. Note that no messages pass between clocks A and B for the rest of this example.

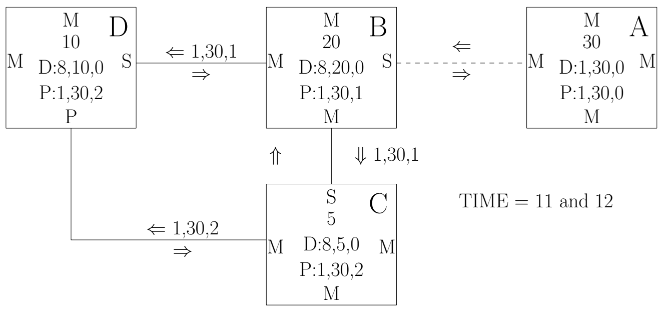


Fig. 5. Network state at AI = 11 and 12.

No visible changes are apparent until time = 13, Figure 6, which shows all ports of clock B in the master state as a result of an ART on the East port due to the disconnect of the previous grandmaster clock A. Note that clock C continues to transmit Announce messages referencing the now disconnected grandmaster clock C since nothing has caused an update to the datasets of clock C.

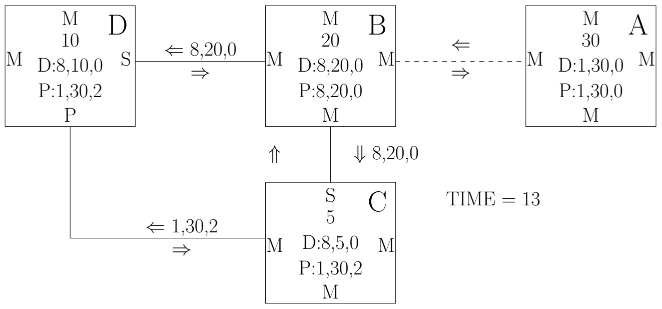


Fig. 6. Network state at AI = 13.

Here we see why the rogue message circulates clockwise since it is effectively initiated by clock C rather than by clock D whose South port is in the passive state. Indeed if the identity of clock C is changed to 15, then during the steady state prior to the disconnect the passive port will be the West port of the clock C, i.e., the asymmetry of the steady state topology is reversed. This results is the rogue message circulating in the counter clockwise direction.

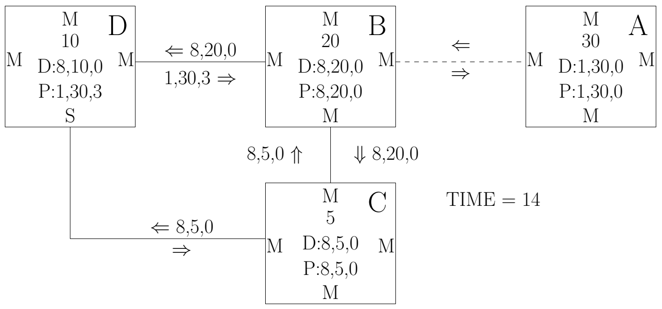


Fig. 7. Network state at AI = 14.

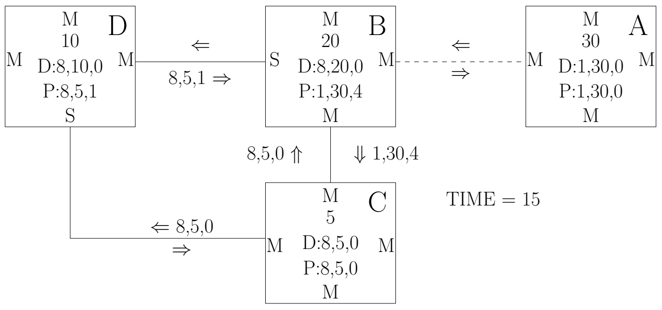


Fig. 8. Network state at AI = 15.

From time =13 to time = 18 a succession of changes in the parent datasets of the clocks occur with the end result that at time = 18, Figure 11, all parent datasets reflect the now absent grandmaster clock A and all the Announce messages reflect these values. Note that the stepsRemoved values in the datasets and the corresponding Announce messages increase by one when traversing each boundary clock in the clockwise direction. Therefore observing the Announce messages at any point in the system, for example between clocks D and B, will show the stepsRemoved values incremented by three every third message as illustrated in Figure 2.

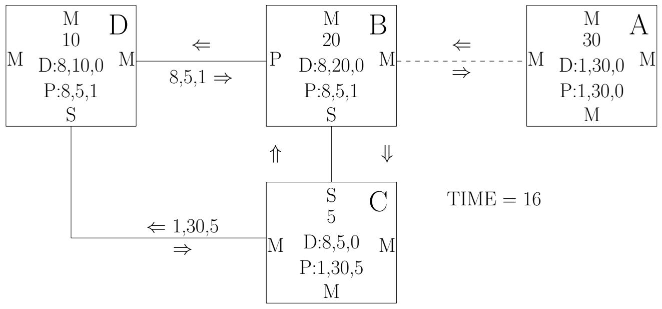


Fig. 9. Network state at AI = 16.

The progression of states for the stepsRemoved squelch mechanism, not illustrated, results from ARTs on each of the ports receiving rogue messages. The first Announce message with stepsRemoved = 12, the threshold, occurs on the West port of clock B at time = 22 and causes clock B ports to enter the master state at time = 25 due to the ART. This in turn causes clock B to send Announce messages on all ports resulting in clock C becoming the best clock in the system. By time =28 a new stable state is reached with the cycle broken by a passive state at the West port of clock B.

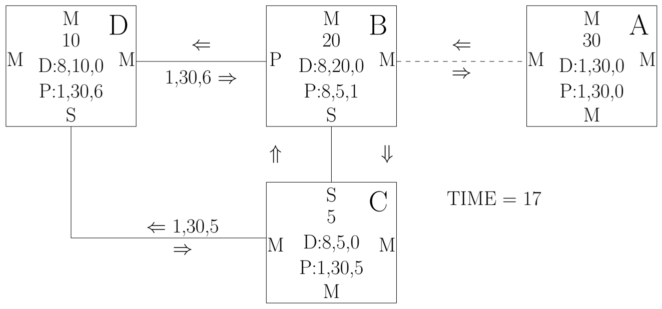


Fig. 10. Network state at AI = 17.

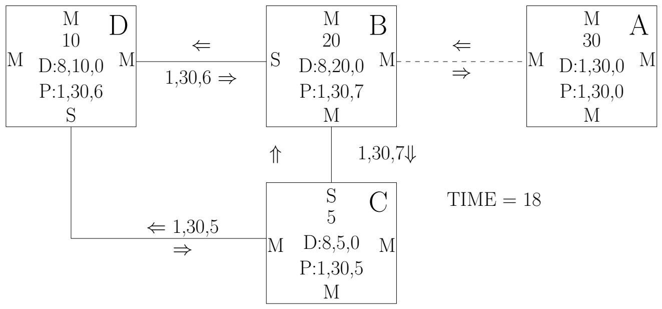


Fig. 11. Network state at AI = 18.