US 20210073689A1

(19) **United States**

(12) **Patent Application Publication**

**FINZI et al.**

(10) **Pub. No.: US 2021/0073689 A1**

(43) **Pub. Date: Mar. 11, 2021**

(54) **METHOD FOR GENERATING A SCHEDULE FOR MIXED CRITICAL COMPUTER NETWORKS**

(71) Applicant: **TTTech Computertechnik Aktiengesellschaft**, Wien (AT)

(72) Inventors: **Anaïs FINZI**, Wien (AT); **Silviu S. CRACIUNAS**, Wien (AT); **Ramon Serna OLIVER**, Wien (AT)

(21) Appl. No.: **17/011,250**

(22) Filed: **Sep. 3, 2020**

(30) **Foreign Application Priority Data**

Sep. 9, 2019 (EP) ..... 19196173.9

**Publication Classification**

(51) **Int. Cl.**

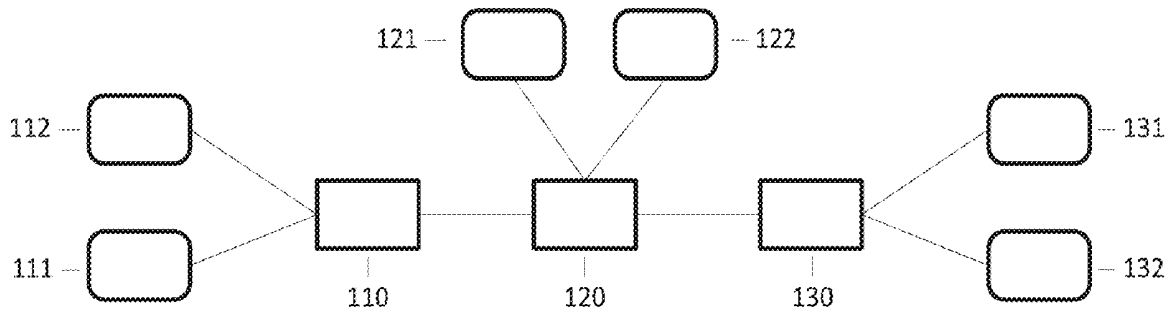
| | |
|---|---|
| ***G06Q 10/06*** | (2006.01) |
| ***H04L 29/08*** | (2006.01) |
| ***H04L 12/43*** | (2006.01) |
| ***H04L 12/875*** | (2006.01) |

(52) **U.S. Cl.**

CPC ..... ***G06Q 10/06*** (2013.01); ***H04L 47/564*** (2013.01); ***H04L 12/43*** (2013.01); ***H04L 67/325*** (2013.01)

(57) **ABSTRACT**

A method for generating a schedule for the transmission of time-triggered, TT, messages in a network, wherein said network communicates TT messages according to said schedule and based on a global, network-wide time, wherein said network communicates rate-constrained, RC messages, wherein for each of said RC messages real-time requirements are provided, wherein the method comprises: Step 1: setting the transmission time of all TT messages which are communicated in the network, and Step 2: executing a search function to find a set of TT transmission times so that the real-time requirements of all RC messages are fulfilled, and when all real-time requirements or at least real-time requirements for defined RC messages are fulfilled, generating in Step 3: the schedule based on the transmission times retrieved in Step 2, or executing Step 2 again when not all real-time requirements or not all real-time requirements for the defined RC messages are fulfilled.



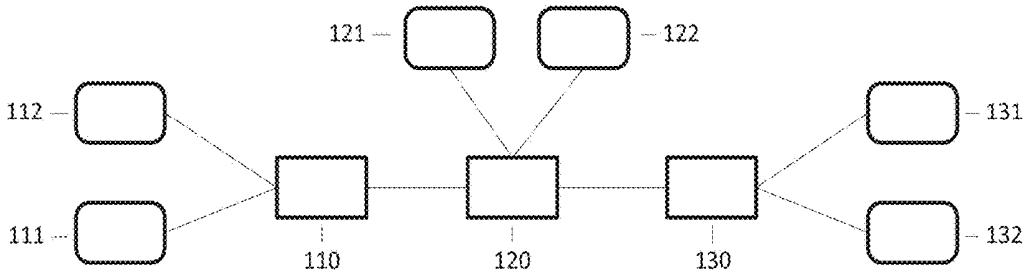



FIG. 1

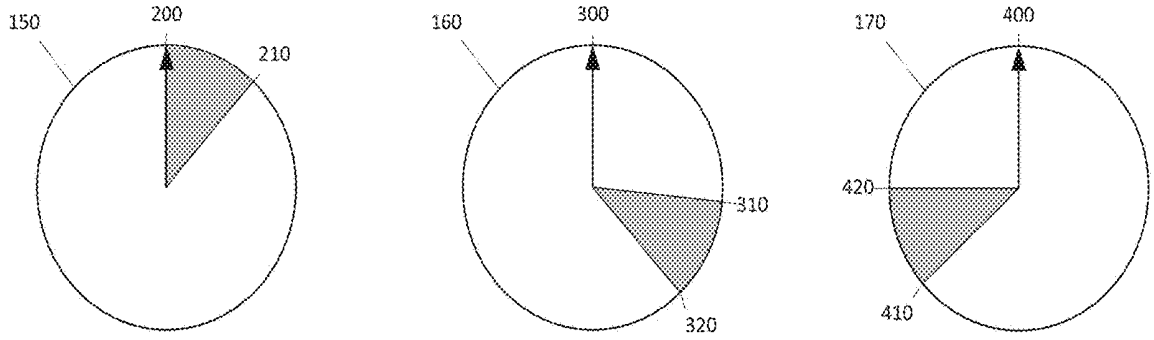



FIG. 2

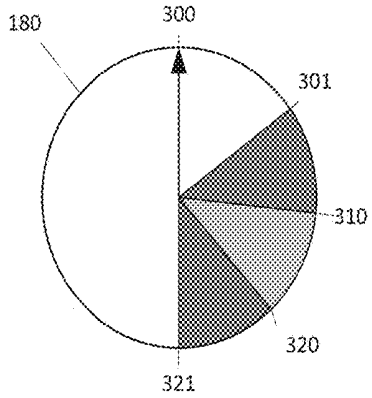



FIG. 3

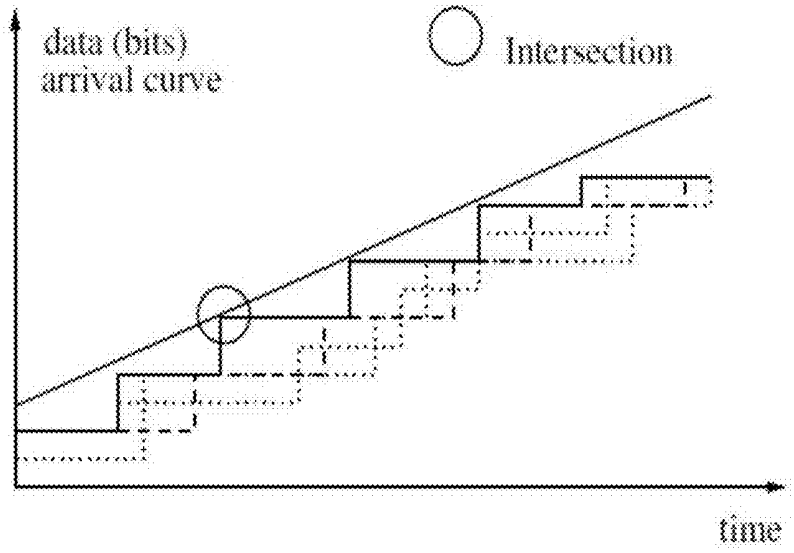

data (bits)
arrival curve
Intersection
time

FIG. 4

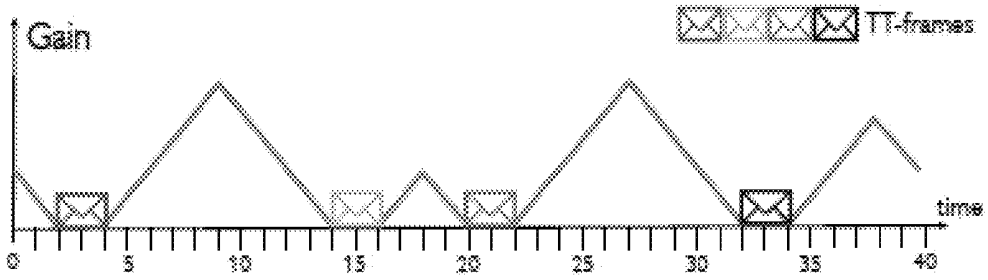

Gain
TT-frames
time
0
5
10
15
20
25
30
35
40

FIG. 5

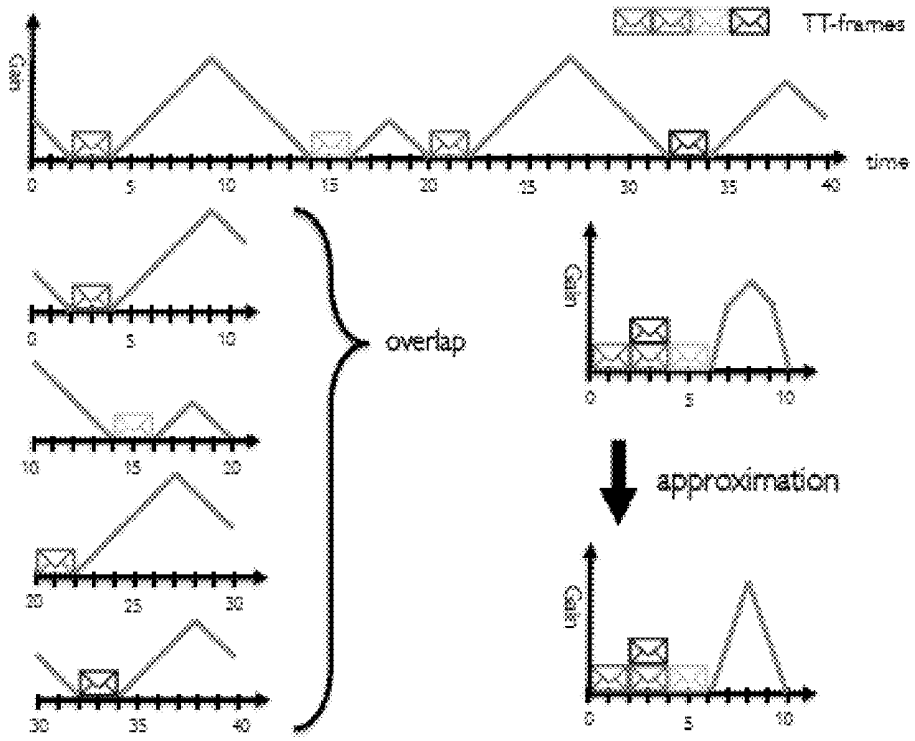

TT-frames
time
overlap
approximation

FIG. 6

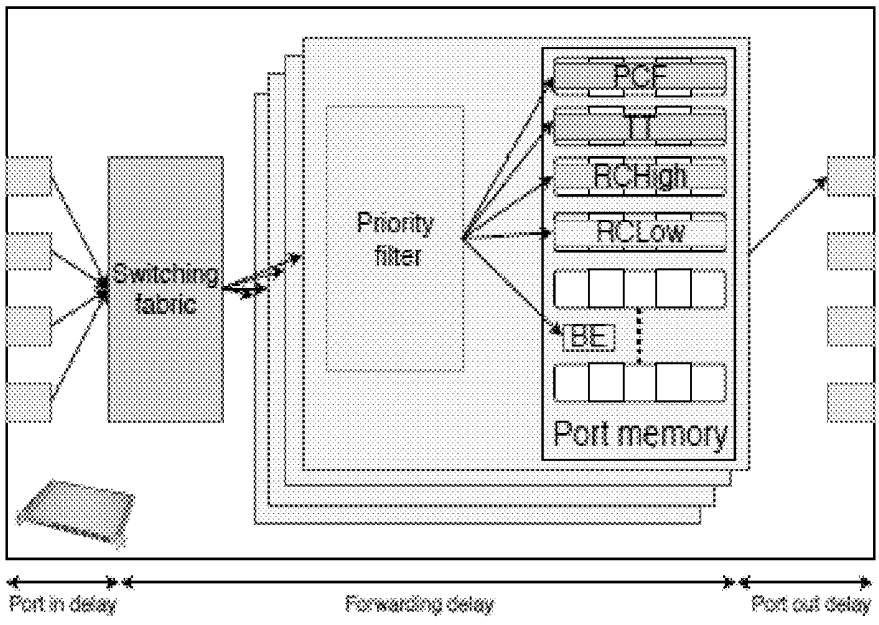

Switching fabric
Priority filter
PCF
TT
RCHigh
RCLow
BE
Port memory
Port in delay
Forwarding delay
Port out delay

FIG. 7

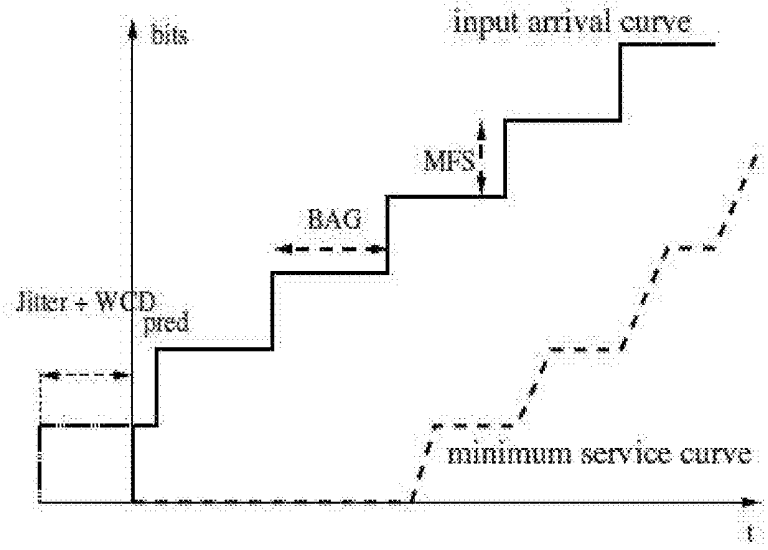

bits
input arrival curve
MFS
BAG
Jitter + WCD pred
minimum service curve
t

FIG. 8

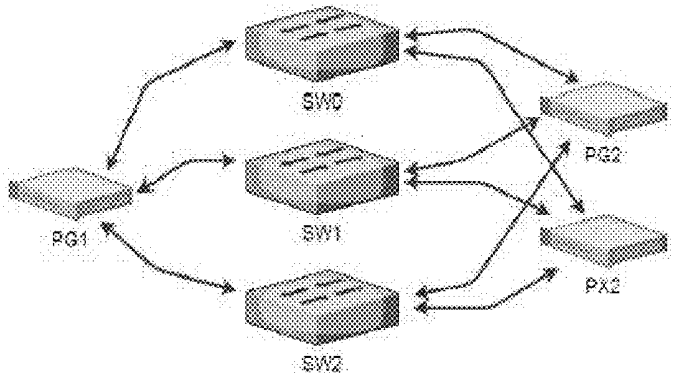

SW0
PG2
PG1
SW1
PX2
SW2

FIG. 9

# METHOD FOR GENERATING A SCHEDULE FOR MIXED CRITICAL COMPUTER NETWORKS

## CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This disclosure claims priority to and the benefit of EP Patent Application No. 19196173.9, filed Sep. 9, 2019, which is hereby incorporated by reference herein in its entirety.

## FIELD

[0002] The invention relates to a method for generating a schedule for the transmission of time-triggered, TT, messages in a network, for example in a TTEthernet or TSN network, wherein the network comprises components, for example nodes and starcouplers, wherein components of said network communicate time-triggered messages according to said schedule and based on a global, network-wide time, and wherein said components communicate rate-constrained, RC messages, wherein for each of said RC messages real-time requirements are provided.

[0003] Furthermore the invention relates to a computer network, for example a TTEthernet or TSN network, wherein the network comprises components, for example nodes and starcouplers, wherein components of said network communicate time-triggered messages according to said schedule and based on a global, network-wide time, and wherein said components communicate rate-constrained, RC messages, wherein for each of said RC messages real-time requirements are provided.

[0004] For example said components, in particular said nodes and star couplers, are arranged in a multi-hop fashion.

## BACKGROUND

[0005] A computer network, for instance an IEEE 802.3 Ethernet (Institute of Electrical and Electronics Engineers 2018) or TSN (Institute of Electrical and Electronics Engineers 2017), can carry both scheduled and unscheduled communication messages, whereby scheduled communication (time-triggered or TT) messages are transmitted from a sending entity (component) to one or more receiving components (entities) at predefined points in time in a network-wide well-defined time, while unscheduled communication messages are transmitted according to other criteria. Appropriate transmission protocols and mechanisms for message handling and message prioritization can be applied, whereby it is ensured that no interference of any scheduled or unscheduled communication message with any given scheduled communication message can occur.

[0006] Non-scheduled messages can be of two types: Rate-Constrained (RC) and Best-Effort (BE). RC traffic is sent with a minimum interarrival-time called BAG (Institute of Electrical and Electronics Engineers 2018) and typically has real-time requirements on the maximum allowed end-to-end latency and jitter. BE traffic does not have any real-time requirements.

[0007] In mixed-criticality networks (i.e., networks that have TT, RC and BE traffic), the communication schedule for TT messages influences the timely behaviour of RC and BE messages due to the fact that TT messages have higher priority over RC and BE messages and are sent first, hence delaying the transmission of RC and BE messages. Hence, the T communication schedule may adversely affect the transmission of RC messages such that they do not adhere to their real-time requirements.

## SUMMARY

[0008] It is an objective of the invention to generate a schedule for networks, in which at least time-triggered messages and RC messages are communicated.

[0009] This object is achieved a method as mentioned above, wherein according to the invention said method is characterized by the following steps:

[0010] Step 1: setting the transmission time of all TT messages which are communicated in the network, and

[0011] Step 2 executing a search function to find a set of TT transmission times so that the real-time requirements of all RC messages are fulfilled, and in the case, that all real-time requirements or at least real-time requirements for defined RC messages are fulfilled, generating in a

[0012] Step 3: the schedule based on the transmission times retrieved in Step 2,

[0013] or executing Step 2 again in case that not all real-time requirements or not all real-time requirements for the defined RC messages are fulfilled.

[0014] For example, it may be provided that the real-time requirements demand or at least demand that the end-to-end travel time for defined RC flows, in particular for each RC flow fulfills its deadline.

[0015] According to the invention it may be provided to evenly space the T transmission times to reduce the impact of TT flows on RC flows so that for example the RC worst-case end-to-end travel times (time needed for a message to go from its source to its destination) may be smaller than the deadlines.

[0016] Further advantages of the invention, which alone or in any arbitrary combination may be realised, are described in the following:

[0017] The transmission times of all the TT messages may be set, in particular for one TT flow after the other, using an optimization function, preferably with an SMT-solver.

[0018] The real-time requirements for the RC messages may comprise an end-to-end delay bound, and wherein the search function in Step 2 is loop based with the following loop-steps:

[0019] loop-step 1: comparing the worst-case end-to-end delay bound of each RC flow with its corresponding deadline, wherein in case that the delay bound is larger than the corresponding deadline according to the RC requirements for said RC flow, the next two loop steps are executed:

[0020] loop step 2 identifying TT transmission times to be modified, and

[0021] loop step 3: computing of the transmission times of the selected TT flows in loop step 2, preferably for each output port of the flow path of a selected TT flow, for example using an optimization function within the SMT-solver.

[0022] Here, loop step 2 will help to guide the search to the most promising parts of a solution space.

[0023] If in loop step 1 the delay bound is smaller, a solution is found and the search stops.

[0024] A record of the previously explored option can be kept, which may be used to guide the search into

unexplored parts of the solution space and/or to be used as an additional stopping condition for the search.

**[0025]** The search function (search algorithm) comprises an identification of TT transmission times to be modified based on a Network Calculus framework and an arrival curve in each output port, which represents the maximum amount of data that can arrive in an output port in any time interval of TT traffic detailed. This method is detailed in L. X. Zhao, H. G. Xiong, Z. Zheng, and Q. Li. *Improving worst-case latency analysis for rate-constrained traffic in the time-triggered Ethernet network.*

**[0026]** A staircase curve for each output port of the network crossed by TT flows may be computed, wherein the staircase curves are approximated by linear curves, and wherein TT flows having an impact, in particular a large impact on RC flows, in particular on the real-time requirements of the RC flows, are identified by intersecting a staircase curve and its linear approximation, the flow most likely to have a large impact on RC flows is identified (see also detailed description under "findBestFlow( )"). This computing is detailed for example in L. X. Zhao.

**[0027]** The computation of the TT transmission times may be executed with the use of an optimization function, which preferably is added within the SMT-solver, and wherein a set of TT flows with a hyper-period, HP, which is the least common multiple of all flow periods of all TT flows, is considered, and wherein for each output port, where TT flows occur, a gain function is defined. Defining a gain function may guide the search of a good transmission time, i.e. a time which will have a low impact on RC end-to-end delays.

**[0028]** For the computation of transmission times, gain functions between the already scheduled TT frames may be determined, wherein with $t^{end}_k$ being the end of a frame transmission and $t^{start}_{k+1\ being}$ the start of the next transmission, for each TT frame already scheduled, the gain functions are determined by:

$$\forall\, t_k^{end} \le t < \frac{t_k^{end} + t_{k+1}^{start}}{2},\ \mathrm{gain}(t) = t - t_k^{end}$$

$$\forall\, t_{k+1}^{start} \ge t \ge \frac{t_k^{end} + t_{k+1}^{start}}{2},\ \mathrm{gain}(t) = t_{k+1}^{start} - t$$

**[0029]** wherein for each period of a flow of interest, preferably in each output port, the gain functions are summed and the abscise of the maximum value of the summed gains is selected, preferably by the SMT-solver, as the transmission time in the output port.

**[0030]** It may be provided that only gain functions in the acceptable times are summed, and wherein the resulting summed gain functions are approximated, in each output port, in that the abscise of the maximum value of the approximated gains is selected, preferably by the SMT-solver, as the transmission time in the output port. "Acceptable" time means time intervals where no frame are already scheduled and wherein the time interval is large enough to schedule a current frame. Accordingly, only a maximum of two linear functions per interval can be kept.

**[0031]** In particular, the preservation of real-time properties or the fulfilling of real-time requirements refers to the guaranteed end-to-end latency for the periodic transmission of messages between one sender node and one or several receiver nodes, via well-defined communication channels, known as virtual links (VLs). The end-to-end latency is bounded if the transmission methods ensure that the messages are transmitted at their scheduled point in time (within a small deviation derived from the clock synchronization imprecision) without occurring in contention with other scheduled or unscheduled transmissions.

**[0032]** The second step of the proposed method (in particular the search algorithm) may relate to changing an existing schedule for TT message transmissions, whereby the changes preserve the transmissions of already scheduled TT messages and guarantee the real-time requirements of RC messages.

**[0033]** In the context of this text the term "transmission times" of a TT message refers to transmission points in time/transmission instants, in particular the earliest transmission instant/point in time at which said TT message is transmitted. In particular, these are the earliest starts of transmissions allowed inside an output port, meaning the transmission of a message will start as soon as possible after this point in time (the transmission can sometimes be delayed by other messages).

**[0034]** In particular, it is provided that (a) the transmission of TT messages does not interfere with the transmission of other already established TT message transmissions; (b) the modification will not invalidate the real-time requirements of such TT messages, and (c) the modified transmission times of such TT messages allows RC messages to meet their real-time requirements.

**[0035]** In the network, preferably each node is connected to at least one star coupler via a physical link. The connection is realized via physical ports on each of the devices. Nodes and star couplers have a limited number of ports, and therefore a maximum number of connecting physical links.

**[0036]** All nodes and star couplers share a common notion of time by means of e.g. a time synchronization protocol like, for example, SAE AS6802 (SAE International 2011) or IEEE 802.1AS (Institute of Electrical and Electronics Engineers kein Datum).

**[0037]** Nodes communicate to each other by exchanging periodic time-triggered (TT) messages and rate-constrained (RC) messages.

**[0038]** A time-triggered message, characterized by a virtual link, has the following attributes:

**[0039]** A sender node

**[0040]** One or several receiver nodes

**[0041]** A period

**[0042]** A maximum message size

**[0043]** A maximum end-to-end latency

**[0044]** A rate-constrained message, characterized by a virtual link, has the following attributes:

**[0045]** A sender node

**[0046]** One or several receiver nodes

**[0047]** A minimum inter-arrival time

**[0048]** A maximum message size

**[0049]** A maximum end-to-end latency

**[0050]** A maximum jitter

**[0051]** The transmission of time-triggered messages may be characterized by their VLs and follows a schedule. Each VL is routed through the network. The routing process

consists of finding a multi-hop network path connecting the sender node with each of the receivers (using, for example Steiner Trees [Steiner, W. 2010. "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks." RTSS].

**[0052]** For each physical link on the network path, a frame may be scheduled for periodic transmission according to the respective message attributes.

**[0053]** Frames are scheduled sequentially such that the transmission point in time of any intermediate hop is not scheduled before the previous frame is received at the respective hop.

**[0054]** The schedule point in time of each frame may be an offset relative to the period, in which the frame will be transmitted. The transmissions are repeated periodically at the given offset.

**[0055]** The end-to-end latency is guaranteed if the last frames are received at the receiver nodes within the maximum end-to-end latency.

**[0056]** During operation, nodes and star couplers transmit TT frames on each link at their scheduled points in time (within a given time precision) following the schedule for the respective link.

**[0057]** The schedule for a link comprises the scheduled points in time for all TT frames scheduled on that link.

**[0058]** The schedule repeats cyclically according to the network cycle, typically the least common multiple of the periods of all VLs.

**[0059]** RC frames are sent whenever there is no higher-priority TT message ready to be sent. Hence the timely behaviour of RC messages is influenced by the TT transmission schedule.

**[0060]** The generation of a network schedule comprising the periodic transmission of frames for all virtual links satisfying their end-to-end latency and without contention is a complex operation. Therefore, it is typically calculated and distributed offline (prior to operation). This implies that the information regarding the communication needs between nodes is also available prior to operation.

**[0061]** The transmission of time-triggered messages per se as characterized above during operation is prior art.

**[0062]** Given a time-triggered computer network as described above in operation the invention for example relates to a method to identify the TT messages that adversely impact the RC message transmission and to modify the transmission times of said TT messages in order to guarantee RC real-time requirements (these requirements may comprise the end-to-end latency, and/or jitter), wherein the transmission points in time of the respective TT frames are modified without altering the real-time properties (end-to-end latency) of the already scheduled transmissions and the real-time properties of said new VLs are guaranteed.

**[0063]** Additionally, the invention refers to a computer program comprising program code means for performing a method according to the invention when said program is run on a computer.

**[0064]** Finally, the invention relates to a computer program product comprising program code means stored on a computer readable medium for performing a method according to the invention when said program product is run on a computer.

**[0065]** In another embodiment, the invention relates to a method for generating a schedule for the transmission of flows in a network, said flows including time-triggered, TT flows, and rate-constrained, RC flows, each such flow comprising messages, respectively TT messages and RC messages, wherein the network comprises components, like nodes and starcouplers or other components that communicate messages between different components in the network, wherein the network is a time-triggered, TT, network or a time-sensitive, TSN network, and wherein the components of said network communicate time-triggered messages according to said schedule and based on a global, network-wide time, and wherein said components communicate rate-constrained, RC messages, wherein for each of said TT and RC messages real-time requirements are provided, and wherein for each of the RC messages, the real-time requirement provided comprises a first requirement which is a worst-case end-to-end delay bound for the transmission of the message.

**[0066]** The method may comprise the steps of

**[0067]** A1: a first search for a set of TT transmission times satisfying a first condition, the first condition being that with the set of TT transmission times applied to the TT messages, that the real-time requirements of all TT messages are fulfilled,

**[0068]** A2: a first determination, applied to the set of TT transmission times found during the first search, of whether the found set of T transmission times satisfies a second condition,

the second condition being that with the set of transmission times applied to the TT messages, that the real-time requirements of the messages of a first set of flows are fulfilled, wherein the first set of flows is a subset of the RC flows,

and wherein in the case where the found set of TT transmission times does not satisfy the second condition, the step A1 that follows include steps:

**[0069]** A11: a first selection of a second set of TT transmission times among the found set, the second set of TT transmission times being the TT transmission times that need to be modified in view of fulfilling the real-time requirements of the first set of flows,

**[0070]** A12: a computation of modified transmission times of the second set of T transmission times, the result of the computation being a new set of TT transmission times fulfilling the first condition for all TT messages.

**[0071]** A3: repetition, in a case where the found set of TT transmission times does not satisfy the second condition, of the steps A1 to A3 until a set of transmission times is found that satisfies the second condition, and

**[0072]** A4: generation of the schedule such that the schedule includes the found set of TT transmission times.

**[0073]** It may be provided that the first search, A1, includes a second selection for choosing, among a plurality of sets of TT transmission times satisfying the first condition, the one of the plurality of sets of TT transmission times satisfying the first condition that optimizes a first optimization function, wherein the first optimization function is checked with a solver, like a Satisfiability Modulo Theory, SMT, solver, or any other solver.

**[0074]** It may be provided that the step A11 comprises the steps

**[0075]** A111: a computation of a hyper-period, HP, which is the least common multiple of all flow periods of all TT flows,

**[0076]** A112 a third selection of the first set of flows on the basis of the hyper-period, and

**[0077]** A113: for each output port identified as being the output port of a node of the network such that at least one of the flows of the first set of flow passes through the considered output, a computation of a gain function.

**[0078]** It may be provided that, step A113 comprises the steps of

**[0079]** A1131: the computation of gain functions for already scheduled TT frames is according to a following formula, wherein with t-w being the end of a frame transmission and $t^{start}_{k+1\ being}$ the start of the next transmission:

$$\forall\, t_k^{end} \le t < \frac{t_k^{end} + t_{k+1}^{start}}{2},\ \mathrm{gain}(t) = t - t_k^{end}$$

$$\forall\, t_{k+1}^{start} \ge t \ge \frac{t_k^{end} + t_{k+1}^{start}}{2},\ \mathrm{gain}(t) = t_{k+1}^{start} - t$$

wherein the step A12 comprises the steps of:

**[0080]** A121: for at least one of the TT flows, a computation of at least one of the TT transmission times based on the gain functions computed for the already scheduled TT frames, which step A121 comprises the steps, for each period of the considered flow and for at least one output port through which the considered flow passes:

**[0081]** A1211: computing a total gain function by summing the gain function of all the flows passing through the considered output port, and

**[0082]** A1212: selecting as a TT transmission time in the output port a time t**0** for which the value of the total gain function is maximum over the period of the considered flow.

**[0083]** It may further be provided that step A1211 comprises

**[0084]** A12111: computing the gain function only for a period of the considered flow, and

**[0085]** A12112: approximating the total gain functions.

**[0086]** It may be provided that step A1 is guided by an algorithm based on a Network Calculus, NC, framework.

**[0087]** It may be provided that step A2 includes storing a record of the set of TT transmission times found.

**[0088]** It may be provided that the step A2 comprises steps:

**[0089]** A21: a computation, for the found set, of a staircase curve for each output port identified as being the output port of a node of the network such that the first set of flow passes through the considered output port,

**[0090]** A22: for each output port considered in step A21, an approximation of the staircase curve by a linear curve,

**[0091]** A23: for each output port considered in step A21, an intersection of the staircase curve with the linear curve,

**[0092]** A24: for an output port considered in step A21, determining an impact of a flow passing through the output port on the basis of the intersection of the staircase curve with the linear curve, and

wherein the first selection in step A11 includes identifying one of the flows of the first set of flows on the basis of the impact of said flow being quantified as large.

**[0093]** The invention refers also to a computer network, wherein the network is a time-triggered, TT, network or a time-sensitive, TSN network, wherein said computer network comprises components, like nodes and starcouplers or other components that communicate messages between different components in the system,
wherein said components communicate flows including time-triggered, TT flows, and rate-constrained, RC flows, each such flow comprising messages, respectively TT messages and RC messages,
and wherein components of said network communicate time-triggered messages according to a schedule and based on a global, network-wide time, and wherein said components communicate rate-constrained, RC messages, wherein for each of said TT and RC messages real-time requirements are provided, wherein the network is adapted to include means to generate a schedule generated with a method described above.

**[0094]** The invention further relates to a computer program comprising program code means for performing the steps of the described method when said computer program is run on a computer.

**[0095]** Finally the invention relates to a computer program product comprising program code means stored on a computer readable medium for performing the method when said computer program product is run on a computer.

## BRIEF DESCRIPTION OF THE FIGURES

**[0096]** In the following, in order to further demonstrate the present invention, illustrative and non-restrictive embodiments are discussed, as shown in the drawings, which show:

**[0097]** FIG. **1** an example of a time-triggered network,

**[0098]** FIG. **2** the periodic transmission of TT messages,

**[0099]** FIG. **3** a detailed representation of the transmission cycle depicted in FIG. **2**,

**[0100]** FIG. **4** an example of computing of a TT flows arrival curve,

**[0101]** FIG. **5** the computation of gain functions,

**[0102]** FIG. **6** the approximation of a gain function,

**[0103]** FIG. **7** an example of a switch architecture,

**[0104]** FIG. **8** an example of a traffic model, and

**[0105]** FIG. **9** an example of a use case.

## DETAILED DESCRIPTION

**[0106]** We discuss some of the many implementations of the invention next. If not stated otherwise, all details described in connection with a specific example are not only valid in connection with this example, but apply to the general scope of protection of the invention.

**[0107]** FIG. **1** shows an example of a time-triggered network which comprises or consists of three time-triggered star couplers (i.e. switches) **110**, **120**, and **130**, and six nodes (i.e. end systems) **111**, **112**, **121**, **122**, **131**, **132**. All systems are connected via lines (which preferably are bidirectional) as shown in the figure and have a common time-base, for example as defined in TTEthernet (Institute of Electrical and Electronics Engineers 2018). Time-triggered (TT) messages are transmitted following a pre-configured global cyclic schedule in coexistence with other traffic (rate-constrained and best effort traffic).

**[0108]** A global distributed schedule determines exact points in time for the transmission of TT messages between the network systems (also denoted as "devices" or "components"; such systems are, for example, the mentioned nodes and star couplers), in a way that the transmissions through the shared lines is realized without contention. The calculation of the schedule is computationally intense, and therefore it is typically performed offline (i.e. prior to the system start-up) and distributed fully or partially to each of the systems of the network. At run-time, the global time base, within a known precision, is available to all systems, and used to execute the schedule in a cyclic and coordinated manner.

**[0109]** Non-scheduled traffic is transmitted during the sparse time between scheduled transmissions, in a way that the interference to scheduled transmissions is either avoided or bounded to a known maximum delay.

**[0110]** The transmission of scheduled messages is logically organized according to the concept of virtual links (VL). A VL defines one sender node (i.e. end system) and one or multiple receivers, as well as a physical path between them. The transmission of messages in a VL originates at the sender and propagates through the physical path (communication lines) until the receiver end system nodes are reached. Each of these propagation steps implies a scheduled transmission after the reception of the previous message. Additional constraints may be provided for VLs, for example a maximum end-to-end transmission deadline, referring to the maximum allowed interval for the propagation of messages—from sender to receiver(s).

**[0111]** A time-triggered message, characterized by a virtual link, has the following attributes:

**[0112]** A sender node

**[0113]** One or several receiver nodes

**[0114]** A period

**[0115]** A maximum message size

**[0116]** A maximum end-to-end latency

**[0117]** A rate-constrained message, characterized by a virtual link, has the following attributes:

**[0118]** A sender node

**[0119]** One or several receiver nodes

**[0120]** A minimum inter-arrival time

**[0121]** A maximum message size

**[0122]** A maximum end-to-end latency

**[0123]** A maximum jitter

**[0124]** Let $vl_a$ be a TT virtual link with sender node **121** and receiver node **131** in the network depicted in FIG. **1**. The network path is hence composed by the sequence {121, 120, 130, 131}, as extracted from FIG. **1**. Let the period as well as the end-to-end delay of $vl_a$ be 100 ms.

**[0125]** FIG. **2** illustrates the periodic transmission of TT messages from $vl_a$ within the transmission cycles of the three nodes conforming the respective network path (subfigures **150**, **160**, and **170**). Note that in this example only the transmission events are scheduled, hence node **131**, the receiver, does not appear. Respectively, **150** represents the cyclic transmission of messages from $vl_a$ in node **121**, **160** that in node **120**, and **170** in node **130**. Each of the subfigures represent a time cycle of 100 ms starting from the top most point in time (**200**, **300**, and **400**) and progressing clockwise. The clock is synchronized globally at every cycle; hence the time progression is homologous for all systems, within a known synchronization precision.

**[0126]** In essence, at time **200** a transmission event for a TT message of $vl_a$ occurs at node **121** which initiates the transmission of a message taking place until time **210**. Node **120** receives the message and transmits the succeeding message at time **310**, being the transmission finished by, at most, **320**. Similarly, node **130** transmits a succeeding message at **410**, finishing by **420**. This transmission cycle repeats endlessly in a coordinated manner.

**[0127]** Note that event **310** can only occur after event **210**, as the message transmission in node **120** directly depends on the previous reception of the message transmitted by node **121**. Analogously, event **410** depends on the occurrence of event **320**. In essence, for the second and following messages propagated along the network path of a VL, the transmission can only be initiated after the previous message of the VL has arrived at the current node.

**[0128]** FIG. **3** provides a detailed representation of the transmission cycle depicted in FIG. **2**, including all other scheduled transmissions in the same communication line for node **120** (dark gray), in addition to those of $vl_a$ (light gray) already depicted in FIG. **2**. We observe that additional transmissions occur between events **301-310**, and **320-321**. Note that a transmission cycle defines the outgoing transmissions of a message towards one single communication line of the respective node (often referred as ports). Therefore, **180** depicts only the transmission cycle of node **120** for the communication line between nodes **120** and **130**. Also note that similarly to the sequential transmissions depicted in FIG. **2**, the transmissions of messages shown in FIG. **3** have dependencies to previous transmissions from nodes **121**, **122** or **110** as part of VL paths traversing node **120**.

**[0129]** The schedule of the TT frames on the timeline may have a major impact on the delays experienced by RC flows since TT traffic has a higher priority. Typically, the end-to-end latency of RC messages in such networks is analyzed through methods like network calculus (A. Van Bemten, W. Kellerer. 2016. *Network Calculus: A Comprehensive Guide*. TUM. https://mediatum.ub.tum.de/doc/1328613/1328613.pdf). An extension of this analysis which considers the TT message schedule is presented in. A preferred timing analysis, which may be used in this invention, is based on the Network Calculus framework [L. Zhao, P. Pop, Q. Li, J. Chen, and H. Xiong, "Timing analysis of rate constrained traffic in TTEthernet using network calculus," Real-Time Systems, 2017.] which may be used to compute upper delay and backlog bounds. These bounds depend on the traffic arrival described by a so-called arrival curve $\alpha$, which represents the maximum amount of data that can arrive in any time interval, and on the resource availability described by a curve, the so-called minimum service curve $\beta$, which represents the minimum amount of data that can be sent in any time interval.

**[0130]** Proposed Method

**[0131]** The schedule of the TT frames on the timeline may have a major impact on the delays experienced by RC flows since TT traffic has a higher priority. According to the present invention it is checked that the RC end-to-end latencies computed with the current T schedule are fulfilling the RC deadlines, using a feedback loop that preferably uses an RC network calculus analysis (A. Van Bemten, W. Kellerer. 2016. *Network Calculus: A Comprehensive Guide*. TUM. https://mediatum.ub.tum.de/doc/1328613/1328613.pdf) to check that the RC end-to-end latencies computed with the current TT schedule are fulfilling the RC deadlines.

If the RC latencies are larger than the RC deadlines, the problematic TT messages are rescheduled.

**[0132]** Evenly spacing out T message placement may lead to a lower impact on RC message end-to-end latencies. Hence, it may be provided that an optimization metric is built that tries to evenly space out TT frames on the timeline and use optimization objectives to drive the modification of the TT message schedule.

**[0133]** Firstly, all the TT flows are scheduled, preferably according to said optimization metric. Secondly, an RC analysis is provided to determine if the RC traffic fulfills the deadline requirements. If this is the case, the method/algorithm stops. If this is not the case, the algorithm identifies the TT messages most likely to be causing delays and attempt to find a better offset, i.e., a modification of transmission times of identified TT messages, preferably using an optimization metric. This second step is repeated until an appropriate schedule is found or a stopping condition is reached.

**[0134]** The search may stop when a set of offsets fulfilling the deadline conditions is found or when the search algorithm has tried to reschedule all the possible flows without finding an unexplored set of offsets.

**[0135]** Identification of the TT Message to Reschedule (FIG. **4**)

**[0136]** The impact of TT flows may be represented by an arrival curve of the TT traffic, which may be computed using formulas detailed in (L. Zhao, P. Pop, Q. Li, J. Chen, and H. Xiong, "Timing analysis of rate constrained traffic in TTEthernet using network calculus," Real-Time Systems, 2017). The main aspect is to compute the impact of TT flows in all possible situations and keep the maximum values, as illustrated in FIG. **4**. The dotted lines in FIG. **4** are the different situations and the plain line staircase represents the maximum of the dotted lines, representing the final TT flows maximum arrival curve. From this staircase arrival curve, a linear arrival curve may be approximated. The rate [the increase of the linear curve] of the linear curve is the same as the rate of the staircase curve over a period. Concerning the initial burst [the value of the linear curve at t=0] of the linear curve, it is computed such as the linear curve is always superior or equal to the staircase curve, with at least one intersection, as illustrated in FIG. **4**. As a result, by modifying the value of the offset associated to the staircase intersection point, the value of the burst of the linear curve may be reduced, and consequently, the impact of TT traffic on RC flows is reduced. A so-called diversification list may be used to keep track of TT flows that have already been explored and changed in order not to run into an endless loop by trying to modify the schedule of the same TT flow(s). To select the flow most likely to have the most impact, the number of times a flow is an intersection can be computed and the flow with the most occurrences that is not in the diversification list may be selected. Finally, if no flow has been found, the first flow in the list of TT flows that is not in the diversification list will be selected.

**[0137]** Generation of the TT Offsets (FIG. **5**)

**[0138]** To compute the offset leading to a minimum impact of TT flows on RC flows, it may be provided that the frames are spread over a hyperperiod (that is the least common multiple of all scheduled TT message periods), in order to reduce the initial burst of the aggregated TT flows. Here, the period is 10 and the hyperperiod HP is 40.

**[0139]** To achieve this goal, in particular in each output port with TT flows, gain functions between the already scheduled TT frames may be defined, as illustrated in an example in FIG. **5**. When looking at the time-line of scheduled transmissions between 0 and the hyperperiod, we denote $t_k^{end}$ to be the end of the transmission of a TT frame and $t_{k+1}^{start}$ to be the start of the next transmission of a TT frame on the time-line. For each TT frame k already scheduled (not including the TT frames of the TT flow identified as described above, which may be denoted by TT flow i, the gain functions are determined by:

$$\forall\, t_k^{end} \le t < \frac{t_k^{end} + t_{k+1}^{start}}{2},\ \mathrm{gain}(t) = t - t_k^{end}$$

$$\forall\, t_{k+1}^{start} \ge t \ge \frac{t_k^{end} + t_{k+1}^{start}}{2},\ \mathrm{gain}(t) = t_{k+1}^{start} - t$$

**[0140]** For a selected TT flow from the identification step its transmission times along the route may be modified as follows:

**[0141]** the at least one, preferably all of the following correctness conditions have to be kept:

**[0142]** Contention-Free Constraints

**[0143]** Path-Dependent Constraints

**[0144]** Simultaneous Relay Constraints

**[0145]** End-to-end Transmission constraints

**[0146]** Application Level constraints

**[0147]** Protocol-Control Flow Constraints

**[0148]** Domain-Specific constraints

**[0149]** An optimization condition for the frames of the identified TT flow that drive the placement of the frames to be as close as possible to the maximum gain for that frame is added.

**[0150]** The above conditions are described in detail in [Steiner, W. 2010. "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks." RTSS.].

**[0151]** Optionally, a condition for each port that the new offset is different from the old offset may be added.

**[0152]** Approximation (FIG. **6**)

**[0153]** The method presented above may require a large number of optimization functions. In between each two scheduled TT frames, a number of optimization functions is necessary to define the two linear parts of the gain, i.e., to define the upper and lower bounds of the new offset and the value associated with the gain. When the number of TT flows increases, so does the number of assertions needed, resulting in an increased runtime of the scheduler.

**[0154]** Accordingly, the following may be provided: only a period Ti of a flow i (the flow to be rescheduled) is considered and the values excluded by the currently scheduled frames are computed, i.e., if a frame is being transmitted, or if the inter-frame gap is too small to transmit the frame of flow i. Then, the gains, preferably only on the acceptable times, are summed up, as illustrated in FIG. **6**. Finally, to further reduce the number of linear parts of the gain function (which means reducing the number of assertions and hence the computation time), the remaining gain functions may be approximated to keep only a maximum of two linear functions per interval [between two already scheduled frames or between 0 and a frame or between a frame and the hyperperiod (in particular, between 0 and the

end of the hyperperiod)], as illustrated in FIG. **6**, here with Ti=10 and hyperperiod HP=40.

**[0155]** In the example described in FIG. **5** and FIG. **6**, the number of assertions has been reduced from 24 to 6, which should improve the timing performances.

**[0156]** Detailed Explanation of an Example Method

**[0157]** In the following a concrete example of the method according the invention will be explained. All details, even if they are in the context of this example described as mandatory, may be provided or implemented in the most general scope of the invention.

**[0158]** 1 Theoretical Background

**[0159]** 1.1 TTEthernet

**[0160]** TTEthernet is an extension to standard Ethernet currently used in mixed-criticality real-time applications in the aerospace domain but also in emerging industrial automation systems. TTEthernet features three types of traffic that represent different criticality levels: TT traffic (used for highly critical traffic with guaranteed end-to-end latency and minimal jitter), RC traffic with deterministic quality of service (QoS) guarantees and non-critical traffic (i.e. BE traffic). The TTEthernet [7] standard is based on the use of global time synchronization to send Time Triggered (TT) frames at precise, predefined times (encoded in a local schedule table that is part of a globally computed schedule) to ensure the lowest contention and delays. Hence, the TT flows are defined by their size, period, and offsets (the times their transmission should start) in each output port. The synchronization flows have the highest priority in TTEthernet networks, the next priority is used by the TT flows, the two next priorities are used by the AFDX $RC_{HIGH}$ and $RC_{LOW}$ traffic, respectively, while the remaining 4 lowest priorities are reserved for Best-Effort (BE) communication (cf. FIG. **7**).

**[0161]** 1.2 SMT-Based TTEthernet Schedule Synthesis

**[0162]** Here, we briefly describe, based on [11, 6, 10] the SMT-based approach for computing communication schedules for TTEthernet.

**[0163]** Satisfiability Modulo Theories (SMT) is a well-known method, similar to SAT, used to check the satisfiability of first-order logical formulas based on a background theory such as linear integer arithmetic ($\mathcal{LA}\,\mathbb{Z}$) or bit-vectors ($\mathcal{BV}$) [1], [9]. Current state-of-the-art TTEthernet scheduling approaches [6], use SMT solvers for generating the static schedule by generating assertions to the context of an SMT solver representing the necessary and sufficient correctness conditions. The result of the SMT solver is a solution for the given constraints, representing values for the offsets of individual frames on each device. Additionally, modern SMT solvers like Z3 [2] offer the possibility to include optimization criteria that find the optimal solution given minimization or maximization condition(s).

**[0164]** Our algorithm, which builds on top of the work in [11], uses SMT solvers to find out the schedule for a network based on the inherent TTEthernet technology constraints and optional user constraints. Steiner [11] proposes an incremental backtracking algorithm, which we also implemented, that splits the scheduling problem in small increments, scheduling subsets of flows at a time. If a partial solution is found for the subset, additional frames and constraints are added until either the complete schedule is found or a solution to a partial problem cannot be found. In the case of in-feasibility, the problem is backtracked and the size of the added subset is increased. In the worst case the algorithm backtracks to the root, scheduling the complete set of frames in one step. Our algorithm adds each flow from the input to the SMT context in sequence following the algorithm described in [11]. We refer the reader to [11] for a description and formalization of the correctness constraints for the SMT-based TTEthernet scheduler which we use in our method.

**[0165]** While the SMT-based approach retains the optimality property of SAT- or MiP-based solutions, the major downside is that the SMT solver acts as a black box and we are only able to influence it by defining constraints (assertions) and optimization criteria. Moreover, we are constrained to the expressiveness of first order logic. Hence, state-of-the-art SMT-based schedulers only take into account the TT flow constraints. As a result, the RC traffic can be strongly impacted by the placement of the scheduled TT frames and miss their required deadlines.

**[0166]** We propose a feedback-based approach that attempts to drive the SMT solver via optimization criteria towards placing TT frames on the timeline such that RC traffic will adhere to the latency and backlog requirements.

**[0167]** Please note that, while the proposed method is tailored to TTEthernet, the main feedback-based approach can also be used in Time-Sensitive Networks (TSN) which is the new standardized deterministic Ethernet solution for industrial communication systems. For TSN, existing SMT-based solutions (e.g. [10]) can be extended using our proposed method to consider AVB traffic shaped by CBS by including the network calculus-based AVB analysis results from [12].

**[0168]** 1.3 Network Calculus

**[0169]** The timing analyses detailed in this paper are based on the Network Calculus framework [8] which is used to compute upper delay and backlog bounds. These bounds depend on the traffic arrival described by the so-called arrival curve α, which represents the maximum amount of data that can arrive in any time interval, and on the resource availability described by the so-called minimum service curve δ, which represents the minimum amount of data that can be sent in any time interval.

**[0170]** Definition 1 (Arrival Curve) [8] A function α(t) is an arrival curve for a data flow with an input cumulative function A(t), i.e., the number of bits received until time t, iff:

$$\forall t, A(t) \leq (A \otimes^{1} \alpha)(t)$$

[1] $f \otimes g(t) = \inf_{0 \leq s \leq t} \{f(t-s)+g(s)\}$

**[0171]** Definition 2 (Strict minimum service curve) [8] The function β is the minimum strict service curve for a data flow with an output cumulative function A*, if for any backlogged period]s,t][2]: $A^*(t)-A^*(s) \geq \beta(t-s)$

[2] ]s,t] is called backlogged period if $A(\tau)-A^*(\tau)>0, \forall \tau \in ]s,t]$

**[0172]** To compute the main performance metrics, we need the following results:

**[0173]** Theorem 1 (Performance Bounds) [8] Consider a flow F constrained by an arrival curve α crossing a system S that offers a minimum service curve β. The performance bounds obtained at any time t are:

Backlog[3]: $\forall t: q(t) \leq v(\alpha,\beta)$

Delay[4]: $\forall t: d(t) \leq h(\alpha,\beta)$

Output arrival curve[5]: $\alpha^*(t) = (\alpha \oslash \beta)(t)$

[3] v: maximal vertical distance

[4] h: maximal horizontal distance

[5] $f \oslash g(t) = \sup_{s \geq 0} \{f(t+s)-g(s)\}$

**[0174]** Theorem 2 (Concatenation-Pay Bursts Only Once) [8] Assume a flow crossing two servers with respective service curves $\beta_1$ and $\beta_2$. The system composed of the concatenation of the two servers offers a service curve $\beta_1 \otimes \beta_2$.

**[0175]** Theorem 3 (Left-over service curve) [3] Consider a system with the strict service curve $\beta$ and m flows crossing it, $f_1, f_2, \ldots, f_m$. The maximum packet length of $f_i$ is $l_{i,max}$ and $f_i$ is $\alpha_i$-constrained. The flows are scheduled by the NP-SP policy, where priority of $f_i$>priority of $f_j \Leftrightarrow i<j$. For each $i \in \{1, \ldots, m\}$, the strict service curve of $f_i$ is given by[6]:

$$\left(\beta - \sum_{j<i} \alpha_j - \max_{k \geq i} l_{k,max}\right)^{+}$$

**[0176]** The traffic contracts are generally enforced using a leaky-bucket shaper, i.e., the traffic flow is (r,b)-constrained where r and b are the maximum rate and burst, and the arrival curve is $\alpha(t)=r \cdot t+b$. A common model of the minimum service curve is the rate-latency curve $\beta_{R,T}$, defined as $\beta_{R,T}(t)=R \cdot (t-T)^{+}$, where R is the output transmission capacity, T is the system latency, and $(x)^{+}$ denotes the maximum between x and 0. The resulting output arrival curve is then: $\alpha^*(t)=r \cdot (t+T)+b$

[6] $g^{+}(t)=\max\{0, \sup_{0 \leq s \leq t} g(s)\}$

**[0177]** An improvement of the leaky bucket shaper is to take into account the link capacity of the input ports in order to obtain a more accurate input arrival curve of the flows [5]. For example, a flow arriving with a maximum burst b and rate r, from a link with a capacity $C_{in}$, has an input arrival curve $\alpha(t)=\min(C_{in} \cdot t, r \cdot t+b)$.

**[0178]** Extensions for the RC traffic class analysis for TTEthernet that consider the impact by the TT traffic class schedule have been introduced in [14, 13, 4].

**[0179]** 2 Feedback Loop

**[0180]** An optimal solution would be to integrate the Network Calculus framework within the SMT solver to directly consider the constraints of the RC traffic. However, this is not possible due to the non-linearity of Network Calculus. In particular, the computation of the arrival curve of TT frames as described in [14] make use of several minimizations, maximizations and upper-bound values to compute the overall arrival curve of TT flows.

**[0181]** The current state-of-the-art SMT-based schedulers do not consider RC flows. However, the placement of the TT frames on the timeline may have a major impact on the delays experienced by RC flows since TT traffic has a higher priority. The main idea of our proposal (described in more detail below) is to use a feedback loop that uses the RC network calculus analysis to check the TT schedule and, if necessary, reschedule problematic flows.

**[0182]** First all the TT flows are scheduled using the optimization function of the SMT-solver to spread the TT frames. Our observation is that evenly spacing out TT frame placement leads to a lower impact on RC flows. Hence, we build an optimization metric that tries to evenly space out TT frames on the timeline and use optimization objectives to drive the scheduling and rescheduling of TT flows via the SMT solver.

**[0183]** Secondly, the RC analysis is called to determine if RC traffic fulfills the deadline requirements. If this is the case, the algorithm stops. If this is not the case, the algorithm identify the flow most likely to be causing delays and attempt to find a better offset, i.e., reschedules, using the optimization function of SMT.

**[0184]** This second step is repeated until an appropriate schedule is found or a stopping condition is reached.

**[0185]** The additional advantage of such a solution is that it is complementary to the existing implementation and can be readily integrated into existing tools. It does not require modification of the existing constraints, just the addition of optimization objectives and the implementation of the feedback loop.

**[0186]** 2.1 General Algorithm

**[0187]** The general algorithm is detailed in Alg. 1. To modify the values of the offsets, we propose to add an optimization constraint to the SMT scheduler, implemented in the function smtComputeOffset(flow) in Section 1.2.3. In particular, we propose two methods for the computation of the optimization function. They are detailed in Sections 1.2.4 and 1.2.5.

**[0188]** First, in Ag. 1, we set an initial offset for each flow (ranked from smallest to largest period) using smtComputeOffset(flow), from Line 1 to Line 3. Then, we check whether the RC deadlines are fulfilled in Line 4. If it is not the case, we attempt to modify the offset of a flow selected by the function findBestFlow( ) (describe in Section 1.2.2), in Lines 8 and 9.

**[0189]** If the new set of offsets has already been explored (Line 10), we use a while loop to explore possible offsets until all the flows have been tested or a new set has been found (Line 11). While no acceptable solution is found in Line 11, we add the flow to the diversification list, look for a new flow that is not in the diversification list, and compute the new offsets as before (Lines 12 to 15). If the new set of offsets has not already been explored, then the diversification list is reset (Lines 16 to 18). Finally, we check if the RC deadlines are now fulfilled and update the list of explored offsets (Lines 19 and 20).

Algorithm 1: General algorithm

```
Require: flows_TT, flows_RC
 1: for flow in flows_TT do
 2:     smtComputeOffset(flow)
 3: end for
 4: rcDeadlinesFulfilled=computeRCDelays(flows_RC)
 5: listExploredOffset=[]
 6: diversification=[]
 7: while rcDeadlinesFulfilled==False and diversification.size <
    flows_TT.size do
 8:     flow = findBestFlow( )
 9:     smtComputeOffset(flow)
10:     if getCurrentOffsets( ) in listExploredOffset then
11:         while getCurrentOffsets( ) in listExploredOffset and
    diversification.size < flows_TT.size do
12:             diversification.add(flow)
13:             flow = findBestFlow(diversification)
14.             smtComputeOffset(flow)
15:         end while
16:     else
17:         diversification==[]
18:     end if
19:     rcDeadlinesFulfilled=computeRCDelays(flows_RC)
20:     listExploredOffset.add(getCurrentOffsets())
21: end while
```

**[0190]** 2.2 findBestFlow( )

**[0191]** The impact of TT flows is represented by the arrival curve of the TT traffic, which is computed using the formulas detailed in [14]. The main idea is to compute the impact of TT flows in all the possible situations and keep the maximum values, as illustrated in FIG. **4**. The dotted lines are the different situations and the plain line staircase is the maximum of the dotted lines, representing the final TT flows maximum arrival curve, denoted $\alpha_{TT}^{k}(t)$.

**[0192]** From this staircase arrival curve, we are able to approximate a linear arrival curve. The rate of the linear curve is the same as the rate of the staircase curve over a period. Concerning the initial burst of the linear curve, it is computed such as the linear curve is always superior or equal to the staircase curve, with at least one intersection, as illustrated in FIG. **4**. As a result, by modifying the value of the offset associated to the staircase intersection point, we may be able to reduce the value of the burst of the linear curve, and consequently, reduce the impact of TT traffic on RC flows. The intersection points are computed within the findIntersections( ) function.

Algorithm 2: findBestFlow( )

```
Require: flows_TT,flows_RC, Data_paths,diversification
Ensure: bestFlow
 1: listCard={ }
 2: for flowTT in flows_TT do
 3:    listCard[flowTT]=0
 4: end for
 5: for flowRC in flows_RC do
 6:    for path in flowRC paths do
 7:       for port in path do
 8:          flowsTTInter=findIntersections( )
 9:          for flowTT in flowsTTInter do
10:             listCard[flowTT]+=1
11:          end for
12:       end for
13:    end for
14: end for
15: bestFlow=NULL
16: if not isEmpty(listCard) then
17:    bestFlow=findMaxCardinal(listCard,diversification)
18: end if
19: while bestFlow==NULL or bestFlow in diversification do
20:    bestFlow = flows_TT.getNext( )
21: end while
```

**[0193]** To select the flow most likely to have the best impact (the algorithm is detailed in Algo. 2), we compute the number of times a flow is an intersection (from Line 5 to 14) and we select the flow with the most occurrences that is not in the diversification list, in Line 17. Finally, if no flow has been found, we select the first flow in the list of TT flows that is not in the diversification list, in Lines 19 to 21.

**[0194]** 2.3 smtComputeOffset( )

**[0195]** We consider a set of TT flows with a hyper-period HP (computed as the least common multiple of all flow periods). Our goal is to define the offset x of flow i with maximum frame sizes $MFS_i$ and period $T_i$. All the preexisting constraints from [11] are enforced. In this section, we only present the constraints added for the optimization of the offsets. Additionally, if the flow has already been scheduled, we also add the constraint that x must differ from the current offset.

**[0196]** We denote $C_{out}$ the capacity of the output port. For each TT frame k already scheduled, we define $t_k^{start}$ and $t_k^{end}$ the start and the end of the transmission. We have

$$m_k = \frac{t_k^{end} + t_{k+1}^{start}}{2}.$$

The function newSymbolicInt( ) creates a symbolic variable that will be used as an unknown by the SMT solver.

**[0197]** In Alg. 3, for each port in each path of the TT flow i, we compute the constraints defining the gains, then they are added to the existing SMT constraints in Line 6. Finally offsets are computed by the SMT solver in Line 7. The computation of the new constraints in a port. (i.e. Line 4) is described in Algo. 4.

Algorithm 3: Constraint computation for a path of TT flow i

```
Require: TT flow i, path
Ensure: Offsets_i
1: sumGain=newSymbolicInt( )
2: constraints=preexisting constraints from [11]
3: for port in path do
4:    (constraints, sumGain)=computeConstraintInPort (TT flow i,
   constraints, sumGain, Data_port)
5: end for
6: constraints.add(maximize(sumGain))
7: Offsets_i=smtSolve(constraints)
```

**[0198]** 2.4 computeConstraintsInPort( ): Variant 1

**[0199]** To compute the offset leading to the minimum impact of TT flows on RC flows, we must spread the frames over the hyper-period in order to reduce the initial burst of the aggregated TT flows.

**[0200]** To achieve this goal, in each output port with TT flows, we define gain functions between the already scheduled TT frames, as illustrated in FIG. **5**, with $T_i$=10 and an Hyper-period HP=40. We denote $t_k^{end}$ the end of a frame transmission and $t_{k+1}^{start}$ the start of the next transmission. For each TT frame k≠i already scheduled, the gain functions are determined by:

$$\forall\, t_k^{end} \le t < \frac{t_k^{end} + t_{k+1}^{start}}{2},\ \text{gain}(t) = t - t_k^{end}$$

$$\forall\, t_{k+1}^{start} \ge t \ge \frac{t_k^{end} + t_{k+1}^{start}}{2},\ \text{gain}(t) = t_{k+1}^{start} - t$$

**[0201]** Alg. 4 describes the definition of the gain function illustrated in FIG. **5**. In particular, Line 7 defines the outer bounds of a triangle, Line 8 (resp. Line 9) defines the right (resp. left) side of the triangle.

**[0202]** However, this method requires a large number of assertions since SMT optimization constraints can only be defined as linear curves in $\mathcal{LA}\ \mathbb{Z}$. Hence, in between two frames,

Algorithm 4:
Constraint computation in a port p: computeConstraintsInPort( )

```
Require: TT flow i, constraints, sumGain, Data_p
Ensure: constraints, sumGain
1:  x= newSymbolicInt( )
2:  J = HP / T_i
```

-continued

Algorithm 4:
Constraint computation in a port p: computeConstraintsInPort( )

```
3:  for j in range(0,J) do
4:    gain_j = newSymbolicInt( )
5:    for frame k in TT flows do
6:      if t_k^end >= j · T_i and t_{k+1}^start <= (j + 1) · T_i then
7:        A = And(x + j · T_i + MFS_i/C_out <= t_k^end , t_{k+1}^start <= x + j · T_i)
8:        B = And(x + j · T_i >= m_k , gain_j = t_{k+1}^start − x − j · T_i)
9:        C = And(x + j · T_i <= m_k , gain_j = x + j · T_i − t_k^end)
10:       constraints.add(And(A, Or(B, C)))
11:       sumGain+ = gain_j
12:     end if
13:   end for
14: end for
```

**[0203]** 6 assertions are necessary to define the two linear parts of the gain, i.e., to define the upper and lower bounds of x and the value associated to the gain. When the number of TT flows increases, so does the number of assertions needed, resulting in an increased runtime of the scheduler [6].

**[0204]** Consequently, we propose a second variant of the algorithm to reduce the number of assertions through preprocessing and hence improve the runtime of our approach.

**[0205]** 2.5 computeConstraintsInPort( ): Variant 2

**[0206]** The main idea of the improvement is to consider only the period $T_i$ of the flow i and compute the values excluded by the currently scheduled frames, i.e., if a frame is being transmitted, or if the inter-frame gap is too small to transmit the frame of flow i. Then, the gains (only on the acceptable times) are summed up, as illustrated in FIG. **6**. Finally, to further reduce the number of linear parts of the gain function (which means reducing the number of assertions and hence the computation time), we approximate the remaining gain functions to keep only a maximum of two linear functions per interval, as illustrated in FIG. **6**, with $T_i$=10 and HP=40.

**[0207]** In the example described in FIG. **5** and FIG. **6**, the number of assertions has been reduced from 24 to 6, which should improve the timing performances. In the next section, we will present experimental results to validate these concepts and assess the gain in performance between the two variants.

**[0208]** 3 Performance Analysis

**[0209]** In this section, we do a performance analysis based on a real-world use-case in order to compare the two selected variants of our solution. First, we present preliminary assumptions, e.g., the model of the flow, the TTEthernet switch and end-systems, and delay computations. Then, after presenting our case study, we do a comparison of the two methods for a TTEthernet network.

**[0210]** 3.1 Preliminaries and Assumptions

**[0211]** Switch and End-System model: we consider the TTEthernet switch architecture described in FIG. **7**. Th input port delay is the amount of time necessary for a frame i to arrive at a rate $C_{in}$:

$$\frac{MFS_i}{C_{in}}.$$

We consider the delay in the switch starts after the frame has been fully received. The forwarding process is defined by a minimum (best-case) and maximum (worst-case) delay, denoted $WCD_{fwd}^{n}$, and $BCD_{fwd}^{n}$, in a switch or end-system $n \in \{sw, es\}$. All the flows are using the shuffling integration policy in the switches and end-systems, i.e, the TT frames can be delayed by lower priority frames.

**[0212]** TTEthernet output ports: the impact of the TT traffic on RC traffic is computed using the input arrival curves proposed in [14]. Then, the service curves are computed using Th. 3.

**[0213]** Traffic model: to compute the delay bounds within each node (output port, switch sw or end-system es), we use Th. 1 under the following assumptions: (i) staircase arrival curves for the traffic flows at the input of node n. For a flow i, we define the Maximum Frame Size $MFS_i$ and the Bandwidth Allocation Gap $BAG_i$ (the period and generally also the deadline). The initial arrival curve sent by the traffic source

$$\alpha_I(t) = \sum_{i \in I} MFS_i \cdot \left\lceil \frac{t + J_i}{BAG_i} \right\rceil.$$

For each class I, the aggregate traffic has an input arrival curve in the node $n \in \{es, sw\}$: $\alpha_I^n(t)=\min\{C_{I,in}^n \cdot t, \Sigma_{i \in I, i \in n} \alpha_i^n(t)\}$, where the maximum input rate $C_{I,in}^n$ is the sum of the capacities $C_{in}$ of the input links of node n crossed by traffic of class I, and $\alpha_i^n(t)$ is the input arrival curve of flow i in node n. $\alpha_i^n(t)$ is a staircase curve as illustrated in FIG. **8**, with $WCD_{prec}$ the delays in the previous nodes. We considered that in the generating node n, $C_{I,in}^n=+\infty$: (ii) the offered service curve by node n to the traffic class I is a staircase curve as illustrated in FIG. **8**. This results from using Th. 3 with staircase arrival curves.

**[0214]** Delay bound computation: with this method, we can use $BCD_{fwd}^{n}$ to obtain a tighter input arrival curve in the output port. The input arrival curve of an aggregate flow I in the output port port in a node $n \in \{es, sw\}$ is:

$$\alpha_i^{port}(t)=\min\{C_{I,in}^n \cdot (t+\delta_{fwd}^n), \alpha_I^n(t+\delta_{fwd}^n)\}$$

with $\delta_{fwd}^n=WCD_{fwd}^n-BCD_{fwd}^n$. Then, according to Th. 1, we can compute the worst-case delay bound as the maximum horizontal distance between $\alpha_I^{port}(t)$ and $\beta_I^{port}(t)=R_I^{port} \cdot (t-T_I^{port})^+$. The delay in the switch (or end-system) n is then: $WCD_I^n=WCD_I^{port}+WCD_{fwd}^n$ and the output arrival curve is $\alpha_I^{n,*}(t)=\alpha_I^n(t+\delta_{fwd}^n+WCD_I^{port})$.

**[0215]** End to end delay bounds: finally, the end-to-end delay of a flow is obtained by summing the delays along the path in the end-systems, input ports, switches and links along the path of the flow.

**[0216]** 3.2 Case Study

**[0217]** We consider a real-world project AERO1[7] depicted in FIG. **9**, with 3 end-systems, i.e., PG**1**, PG**2** and PX**2**, and 3 switches, i.e., SW**0**, SW**1**, SW**2**. The forwarding delays are described in Table 1.1, and the link capacity is 100 Mbps.

[7]Please note that the realistic test cases have been anonymized due to contractual obligations

**[0218]** We consider 12 flows for TT (resp. RC) traffic, with a redundancy level 3, i.e. 36 VLs, with identical period (resp. BAG) of 4 ins and MFS=1518 bytes. The RC traffic has a default maximum initial jitter of 100 μs. Six VLs are generated in PG**1** with PG**2**

TABLE 1.1

Forwarding delays

| Node n | $WCD_{fwd}^{n}$ (s) | $BCD_{fwd}^{n}$ (s) |
|---|---|---|
| End System | $2.43 \cdot 10^{-6}$ | $2.18 \cdot 10^{-6}$ |
| Switch | $2.50 \cdot 10^{-6}$ | $2.41 \cdot 10^{-6}$ |

and PX**2** as destinations. The other six are generated in PG**2** with PG**1** and PX**2** as destinations. The default deadline of a flow is equal to the period.

**[0219]** In order to assess both methods, we explore different scenarios by varying two parameters, i.e., the TT deadline and RC jitter, as illustrated in Table 1.2. As highlighted in the traffic model, increasing the initial jitter increases the input arrival curve of the generated traffic, resulting in the increase of the RC end-to-end delays for identical TT offsets. Consequently, increasing the RC jitter also increases the constraints on the RC deadline and delay. We choose to vary the jitter because it is a minor parameter compared to MFS and BAG, allowing us to remain close to the default scenario.

**[0220]** The results of various scenarios will highlight the advantages and drawbacks of each method.

**[0221]** We focus on the maximum number of iterations $n_{max}$, on the maximum time of execution $t_{max}$, on the iteration where the best result is found $n_{best}$, on the time necessary to find the best result $t_{best}$, and on the schedulability sched. We define the schedulability as the percentage of schedulable flows. Additionally, to ensure the algorithm ends within an acceptable time frame, we have added a constraint in Line 7 of Algo. 1 to limit the number of iterations to 2 000.

TABLE 1.2

Scenarios description

| scenario | TT deadline (s) | RC Jitter (s) |
|---|---|---|
| default | 0.004 | $100 \cdot 10^{-6}$ |
| 1.1 | 0.004 | $800 \cdot 10^{-6}$ |
| 1.2 | 0.004 | $900 \cdot 10^{-6}$ |
| 1.3 | 0.004 | $1000 \cdot 10^{-6}$ |
| 1.4 | 0.004 | $1100 \cdot 10^{-6}$ |
| 1.5 | 0.004 | $1200 \cdot 10^{-6}$ |
| 2.1 | 0.002 | $100 \cdot 10^{-6}$ |
| 2.2 | 0.002 | $300 \cdot 10^{-6}$ |
| 2.3 | 0.002 | $400 \cdot 10^{-6}$ |

**[0222]** 3.3 Results

**[0223]** First, we compute the results of all 9 scenarios without the optimization methods, i.e., with the current implementation. Results show that the computation is done in about $t_{max}$=1.80 s and that 18 out of 36 RC VLs are not schedulable, i.e., their delay is greater than their deadline. Hence, the schedulability of RC flows is 50% without the optimization methods for all 9 scenarios.

**[0224]** In Table 1.3, we present the results of the scenarios finding an acceptable solution, i.e., all the RC flows are schedulable.

**[0225]** In Table 1.4, we present the results of scenarios not finding an acceptable solution, i.e., not all the RC flows are schedulable.

TABLE 1.3

Results with a schedulability of 100% for both methods

| scenario | method | $n_{max}$ | $t_{max}$ (s) |
|---|---|---|---|
| default | 1 | 1 | 101 |
| default | 2 | 1 | 2.5 |
| 1.1 | 1 | 1 | 95 |
| 1.1 | 2 | 3 | 3 |
| 1.2 | 1 | 4 | 174 |
| 1.2 | 2 | 691 | 256 |
| 1.3 | 1 | 4 | 170 |
| 1.3 | 2 | 1110 | 508 |
| 2.1 | 1 | 8 | 46 |
| 2.1 | 2 | 1 | 2.3 |
| 2.2 | 1 | 1841 | 3934 |
| 2.2 | 2 | 123 | 76 |

TABLE 1.4

Results without a schedulability of 100% for both methods

| scenario | method | $n_{max}$ | $t_{max}$ (s) | $n_{best}$ | $t_{best}$ (s) | sched |
|---|---|---|---|---|---|---|
| 1.4 | 1 | 55 | 604 | 55 | 604 | 100% |
| 1.4 | 2 | 2000 | 1190 | 1053 | 616 | 83% |
| 1.5 | 1 | 2000 | 8621 | 1 | 115 | 50% |
| 1.5 | 2 | 2000 | 1180 | 1535 | 882 | 66% |
| 2.3 | 1 | 2000 | 3968 | 255 | 891 | 83% |
| 2.3 | 2 | 2000 | 1023 | 660 | 350 | 83% |

**[0226]** First, concerning the default scenario, we can see from Table 1.3 that both methods find solutions after the first iteration ($n_{max}$=1). Hence, the schedulability is increased from 50% to 100%. Additionally, $t_{max}$ shows that, as expected, method 2 is faster than method 1 (up to 51 times faster in scenario 2.2), and slightly slower than the current method without the optimization.

**[0227]** When studying all 9 scenarios, we notice that method 1 tends to find the best solutions in less iterations (i.e. $n_{max}$ and $n_{best}$ in scenarios 1.1, 1.2, 1.3, 1.4, and 2.3), but not always (i.e. scenarios 2.1 and 2.2). This is most likely due to the approximations done on the gain functions in method 2.

**[0228]** However, less iterations does not necessary result in better timing performances (i.e. $t_{max}$ and $t_{best}$ in scenarios 1.1 and 2.3) because, as expected, an iteration with method 2 is faster than with method 1, e.g. about 4 times faster in scenario 2.3 and up to 40 times faster in the default scenario.

**[0229]** When the schedulability is not 100% with both methods (see Table 1.4), we can see that the best method, i.e. with the best schedulability, depends on the scenario: method 1 for scenario 1.4 and method 2 for scenario 1.5. This again highlights that either method can find the best offsets.

**[0230]** Also, it is interesting to compare the maximum time to explore all 2 000 options: in scenario 1.5, it takes $t_{max}$=8621 s for the first method to finish, and only 1180 s for method 2 to assess that no solution gives 100% schedulability. Additionally, the schedulability with method 1. i.e., the longest to finish, is lower than with method 2, i.e., the fastest method.

**[0231]** Finally, we can conclude that while method 1 is less complex to implement and generally finds results in less iterations, method 2 is the best method to find solutions in an acceptable time frame. Additionally, we see that our feedback-based method is able to generate schedules for TT flows such that the schedulability of RC traffic is increased for all 9 scenarios (even up to 100%).

## References

[1] Clark Barrett, Roberto Sebastiani, Sanjit Seshia, and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, volume 185. IOS Press, 2009.

[2] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. vz - an optimizing SMT solver. In *Proc. TACAS*. Springer, 2015.

[3] Anne Bouillard, Laurent Jouhet, and Eric Thierry. Service curves in Network Calculus: dos and don'ts. Research report, INRIA, 2009.

[4] M. Boyer, H. Daigmorte, N. Navet, and J. Migge. Performance impact of the interactions between time-triggered and rate-constrained transmissions in TTEthernet. In *Proc. ERTS*, 2016.

[5] Marc Boyer, Jörn Migge, and Nicolas Navet. An efficient and simple class of functions to model arrival curves of packetized flows. In *Proc. WTTT*, 2011.

[6] Silviu S. Craciunas and Ramon Serna Oliver. Combined task- and network-level scheduling for distributed time-triggered systems. *Real-Time Systems*, 52(2), 2016.

[7] Hermann Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer. The Time-Triggered Ethernet (TTE) Design. *Proc. ISORC*, 2005.

[8] J.Y. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*. Springer-Verlag, 2001.

[9] Roberto Sebastiani. Lazy satisfiability modulo theories. *JSAT*, 3(3-4):141–224, 2007.

[10] Ramon Serna Oliver, Silviu S. Craciunas, and Wilfried Steiner. IEEE 802.1Qbv Gate Control List Synthesis using Array Theory Encoding. In *Proc. RTAS*. IEEE, 2018.

[11] Wilfried Steiner. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In *Proc. RTSS*. IEEE, 2010.

[12] L. Zhao, P. Pop, Z. Zheng, and Q. Li. Timing analysis of AVB traffic in TSN networks using network calculus. In *Proc. RTAS*, 2018.

[13] L. X. Zhao, H. G. Xiong, Z. Zheng, and Q. Li. Improving worst-case latency analysis for rate-constrained traffic in the time-triggered ethernet network. *IEEE Communications Letters*, 18(11), 2014.

[14] Luxi Zhao, Paul Pop, Qiao Li, Junyan Chen, and Huagang Xiong. Timing analysis of rate-constrained traffic in TTEthernet using network calculus. *Real-Time Systems*, 53(2):254–287, 2017.

We claim:

**1**. A method for generating a schedule for the transmission of time-triggered, TT, messages in a network comprising a TTEthernet or TSN network, wherein the network comprises components comprising nodes and starcouplers, wherein the components of said network communicate time-triggered messages according to said schedule and based on a global, network-wide time, and wherein said components communicate rate-constrained, RC messages, wherein for each of said RC messages real-time requirements are provided, wherein the method comprises the steps of:

Step 1: setting the transmission time of all TT messages which are communicated in the network, and

Step 2 executing a search function to find a set of TT transmission times so that real-time requirements of all RC messages are fulfilled, and when all real-time requirements or at least real-time requirements for defined RC messages are fulfilled, generating in

Step 3: the schedule based on the transmission times retrieved in Step 2, or executing Step 2 again in case that not all real-time requirements or not all real-time requirements for the defined RC messages are fulfilled.

**2**. The method according to claim **1**, wherein the transmission times of all the TT messages are set for one TT flow after the other, using an optimization function with an SMT-solver.

**3**. The method according to claim **1**, wherein the real-time requirements for the RC messages comprise an end-to-end delay bound, and wherein the search function in Step 2 is loop based with the following loop-steps:

loop-step 1: comparing the worst-case end-to-end delay bound of each RC flow with its corresponding deadline, wherein in case that the delay bound is larger than the corresponding deadline according to the RC requirements for said RC flow, the next two loop steps are executed:

loop step 2: identifying TT transmission times to be modified, and

loop step 3: computing of the transmission times of the selected TT flows in loop step 2 for each output port of the flow path of a selected TT flow using an optimization function within the SMT-solver.

**4**. The method according to claim **1**, wherein a record of the previously explored option can be kept, which may be used to guide the search into unexplored parts of the solution space and/or to be used as an additional stopping condition for the search.

**5**. The method according to claim **1**, wherein the search function comprises the identification of TT transmission times to be modified based on a Network Calculus framework and an arrival curve in each output port, which represents the maximum amount of data that can arrive in an output port in any time interval of TT traffic detailed.

**6**. The method according to claim **5**, wherein a staircase curve for each output port of the network crossed by TT flows is computed, and wherein the staircase curves are approximated by linear curves, and wherein TT flows having a large impact on the real-time requirements of the RC flows are identified by intersecting a staircase curve and its linear approximation, the flow most likely to have a large impact on RC flows is identified.

**7**. The method according to claim **1**, wherein the computation of the TT transmission times is executed with the use of an optimization function, which is added within the SMT-solver, and wherein a set of TT flows with a hyper-period, HP, which is the least common multiple of all flow periods of all TT flows, is considered, and wherein for each output port, where TT flows occur, a gain function is defined.

**8**. The method according to claim **7**, wherein for the computation of transmission times gain functions between the already scheduled TT frames are determined, wherein with $t_{end}^{k}$ being the end of a frame transmission and $t^{start}_{k+1}$ *being* the start of the next transmission, for each TT frame already scheduled, the gain functions are determined by:

$$\forall\, t_k^{end} \le t < \frac{t_k^{end} + t_{k+1}^{start}}{2},\ \mathrm{gain}(t) = t - t_k^{end}$$

$$\forall\, t_{k+1}^{start} \ge t \ge \frac{t_k^{end} + t_{k+1}^{start}}{2},\ \mathrm{gain}(t) = t_{k+1}^{start} - t$$

wherein for each period of a flow of interest in each output port, the gain functions are summed and the abscise of the maximum value of the summed gains is selected by the SMT-solver as the transmission time in the output port.

**9**. The method according to claim **7**, wherein only gain functions in the acceptable times are summed, and wherein the resulting summed gain functions are approximated, in each output port, in that the abscise of the maximum value of the approximated gains is selected by the SMT-solver, as the transmission time in the output port.

**10**. A computer network comprising a TTEthernet or TSN network, wherein the network comprises components comprising nodes and starcouplers, wherein components of said network communicate time-triggered messages according to a schedule and based on a global, network-wide time, and wherein said components communicate rate-constrained, RC messages, wherein for each of said RC messages real-time requirements are provided, wherein the schedule is generated with a method according to claim **1**.

**11**. A computer program comprising program code for performing all steps of claim **1** when said program is run on a computer.

**12**. A computer program product comprising program code stored on a computer readable medium for performing the method of claim **1** when said program product is run on a computer.

* * * * *