10-2017

Memetic-Based Schedule Synthesis for Communication on Time-Triggered Embedded Systems

Heyuan Shi
Tsinghua University

Kun Tang
Central South University

Chengbao Lio
Chinese Academy of Sciences

Xiaoyu Song
Portland State University, songx@pdx.edu

Chao Hu
Central South University

See next page for additional authors

Citation Details

Shi, H., Tang, K., Liu, C., Song, X., Hu, C., & Sun, J. (2017). Memetic-based schedule synthesis for communication on time-triggered embedded systems. International Journal of Distributed Sensor Networks, 13(10), 1550147717738167.

Research Article

International Journal of
Distributed
Sensor Networks

International Journal of Distributed Sensor Networks
2017, Vol. 13(10)
© The Author(s) 2017
DOI: 10.1177/1550147717738167
journals.sagepub.com/home/ijdsn
SAGE

Memetic-based schedule synthesis for communication on time-triggered embedded systems

Heyuan Shi[1], Kun Tang[2], Chengbao Liu[3,4], Xiaoyu Song[5], Chao Hu[2,6] and Jiaguang Sun[1]

Abstract
Time-triggered systems play an important role in industrial embedded systems. The time-triggered network is deployed on the time-triggered network-on-chip implementation. It ensures the safety-critical industrial communication for real-time embedded multiprocessor systems. To guarantee the safety-critical requirements for communication, each message is transmitted by a predefined static schedule. However, synthesizing a feasible schedule is a challenge because both spatial and temporal constraints should be considered. This article presents a novel memetic-based schedule synthesis algorithm to derive a feasible schedule by determining the offset of messages on the time-triggered network-on-chip. Memetic-based schedule synthesis algorithm is based on memetic algorithm, which incorporates local search in the iterations of general genetic algorithm. We compare memetic-based schedule synthesis algorithm with genetic algorithm in different scale of time-triggered network-on-chip and number of messages. The experimental results show that the memetic-based schedule synthesis algorithm is effective to synthesize a feasible schedule, and the failure schedule synthesized by memetic-based schedule synthesis algorithm is only 34.2% in average compared to the conventional genetic algorithm.

Keywords
Real-time systems, time-triggered networks, network-on-chip, scheduling, memetic algorithm

Date received: 29 May 2017; accepted: 21 September 2017

Handling Editor: Xiuzhen Cheng

Introduction

Time-triggered systems is one of the real-time systems for industrial multiprocessor systems.[1] At present, time-triggered systems are widely deployed in various fields, such as TTEthernet in avionic networks,[2,3] Time-triggered protocol (TTP) in aerospace industry[4] and GENESYS in embedded systems.[5] TTNoC integrates the time-triggered concepts into the network-on-chip (NoC) for safety-critical real-time embedded systems. Unlike the requirements for general networks, such as quality of service (QoS)[6] as well as quality of experience (QoE)[7] in wireless networks,[8] user privacy in social networks,[9] or energy consumptions and prolonged

[1]School of Software, Tsinghua University, Beijing, China
[2]School of Information Science and Engineering, Central South University, Changsha, China
[3]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[4]University of Chinese Academy of Sciences, Beijing, China
[5]Department of Electrical & Computer Engineering, Portland State University, Portland, OR, USA
[6]Information and Network Center, Central South University, Changsha, China

Corresponding author:
Kun Tang, School of Information Science and Engineering, Central South University, Changsha 410083, China.
Email: tk-0206@163.com

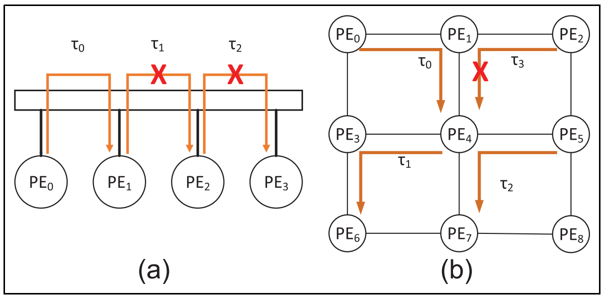

Figure 1. Two main architectures for a time-triggered embedded system: (a) bus-based architecture and (b) network-based architecture.

network lifetime in wireless sensor networks,[10–12] TTNoC provides the temporal predictability for safety-critical tasks, which ensures the tight and predictable communication latency.

There are two main architectures for a time-triggered embedded system, that is, bus-based and network-based architecture.[13] For the bus-based architecture, only one message is transmitted on the system at the same time. Therefore, each message owns its unique time slot for communication. For the network-based architecture, it allows a set of messages to be transmitted at the same time, provided that the physical links used by the messages are contention-free. Figure 1 shows the differences between bus-based and network-based time-triggered architectures. For the bus-based architecture, the message τ_0 is conflicted with τ_1 and τ_2 when they transmit at the same time. In other words, all the messages (τ_0, τ_1, and τ_2 in Figure 1(a)) must achieve temporal separation. In a network-based architecture, the messages (τ_0, τ_1, and τ_2 in Figure 1(b)) transmit at the same time because there is no spatial contention among the transmission of messages. The contention occurs at the link between PE_1 and PE_4 when τ_0 and τ_3 transmit at the same time.

TTNoC employs the network-based architecture. It consists of processing elements (PEs), switches, and physical links. PEs are interconnected by physical links and the switches on the TTNoC. A message for communication consists of header and data. Header includes the information for message communication on the TTNoC, for example, the addresses of source and destination. When the communication occurs on the TTNoC between two nodes, the switches forward the message based on a given route. To ensure the contention-free communication on the TTNoC, a schedule is required to determine the offset for each message. However, synthesizing such a schedule is rather complicated which is known as a nondeterministic polynomial (NP)-complete problem.[13,14] In this article, we give a population-based hybrid genetic algorithm (GA) coupled with an individual learning, called memetic-based schedule synthesis algorithm (MSSA), to synthesize a feasible schedule for the communication on TTNoC.

The main contributions of this article are as follows. We formally give a system model consists of architecture, message, and routing for the schedule synthesis. The Memetic-based Schedule Synthesis Algorithm (MSSA) is proposed to synthesize the schedule. The simulation based on different architecture scale and number of messages proves that the MSSA reduces the failure rate so that synthesizing a feasible schedule effectively.

The rest of this article is organized as follows. We first review the related work consists various algorithm to synthesize schedule based on different architectures in section "Related work." And the problem formulation consists of system model, and constraints are given in section "Problem formulation." Then, the detail of MSSA is proposed in section "Memetic-based schedule synthesis algorithm." The experimental results are shown in section "Evaluations." Finally, we conclude this article in section "Conclusion and future work."

Related work

There are many time-triggered applications in both bus-based and network-based architecture. Despite of different architectures, the communication on the time-triggered systems follows an predefined and statically configured schedule to ensure contention-free.[1] There are sufficient studies on schedule synthesis in time-triggered bus-based architecture.[4,15,16] However, in bus-based architecture, synthesizing a feasible schedule only need to consider the temporal constraints rather than spatial constraints.

For the network-based architecture, most of the studies focus on synthesizing a feasible schedule based on satisfiability modulo theories (SMT). The scheduling of the general time-triggered multi-hop networks was first considered based on SMT solver.[14] However, it is unrealistic for a pure SMT approach to synthesize a schedule for a large-scale time-triggered network. To tackle the limitation of scalability based on SMT, a decomposition approach[17] and an incremental method[14] were presented. Craciunas and Oliver[1,18] gave the solution combining task and network scheduling by SMT solver for distributed time-triggered systems. The schedule employs in time-triggered traffic in a general switched Ethernet network, which can also extend to specific Ethernet protocols such as Profinet and TTEthernet.

The schedule synthesis on network-based architecture in specific domains also appears in the various literature. Craciunas et al.[19] addressed the deterministic schedule synthesis for 802.1Qbv-compliant multi-hop switched networks. Zhang et al.[20] synthesized a schedule of a mixed-integer programming (MIP) model in Ethernet-based time-triggered systems. Ro et al.[21] synthesized the schedule on wireless networks by SMT solver. In time-triggered in-Vehicle networks, the holistic scheduling in system design and integration was studied.[22] The Unfixed Start Time (UST) algorithm, which schedules tasks and messages in a flexible way, was proposed.

Huang et al.[13] first studied the schedule synthesis specific to TTNoC. The problem was transformed into a special case of two-dimensional (2D) bin packing and formulated as an SMT instance. The schedule was synthesized by SMT solver with a First Fit Decreasing Height Decreasing Width (FFDHDW) algorithm to improve the algorithm performance of execution time. Scholer et al.[23] proposed an optimal scheduler based on a Boolean SAT solver for a TTNoC. Murshed et al.[24] gave a scheduling model based on mixed-integer linear programming (MILP), with both time-triggered and event-triggered messages and constraints on NoC. Freier et al.[25] proposed a heuristic algorithm for scheduling on the scalable communication NoC-like structure. The algorithm ensures that the NoC and the core are highly utilized with pseudo-polynomial time complexity.

Unlike the above studies based on SMT-like approach, we propose a meta-heuristic approach of memetic-based algorithm to synthesize a feasible schedule.

Problem formulation

In this section, we first introduce the basic notions and system model of TTNoC. And the constraints on TTNoC are given. Then, we formally state the schedule synthesis problem.

System model

Architecture model. The architecture of a TTNoC is modeled by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$. The set of vertices $\mathcal{V} = \{v_0, \dots, v_{m-1}\}$ represents m identical communication nodes (the PEs with switches). The edges $\mathcal{L} \subseteq \mathcal{V} \times \mathcal{V}$ comprise the communication physical links connecting the nodes. The physical links are full-duplex, allowing communication in both directions. Therefore, given $v_i, v_j \in \mathcal{V}$, $\langle v_i, v_j \rangle \in \mathcal{L}$ implies $\langle v_j, v_i \rangle \in \mathcal{L}$, where $\langle v_i, v_j \rangle$ is a pair denoting a directed physical link connecting two adjacent nodes, from a source node v_i to a sink node v_j.

Message model. We denote the set of n time-triggered messages on the TTNoC by $\Gamma = \{\tau_0, \dots, \tau_{n-1}\}$. Message τ_i is modeled by the tuple $\langle \tau_i.H, \tau_i.S, \tau_i.T, \tau_i.D \rangle$, where $\tau_i.H$ denotes the header of the message, $\tau_i.S$ denotes the size of the message, $\tau_i.T$ is the period, and $\tau_i.D$ is the relative deadline of the message. Only periodic messages are discussed in this article because periodic messages are typical in communication on real-time embedded systems.[13]

Routing model. Given two nodes $v, v' \in \mathcal{V}$, a *route* from v to v', denoted by $r^{\langle v, v' \rangle}$. The length n of the route is written as $|r^{\langle v, v' \rangle}|$. A route $r^{\langle v_i, v_j \rangle}$ represents a forwarding path for message communication from v_i to v_j on the TTNoC. Hence, its length $|r^{\langle v_i, v_j \rangle}|$ is the number of hops along this link.

Schedule synthesis model

The schedule s_i for message τ_i is defined by a tuple $\langle s_i.R, s_i.\phi, s_i.T, s_i.D, s_i.L \rangle$, where $s_i.R$ is the route along which the message τ_i is forwarded, $s_i.\phi$ is the offset of the message, $s_i.T$ denotes the period, $s_i.D$ denotes the relative deadline, and $s_i.L$ is the communication delay, respectively. The unit of offset, period, deadline, and delay is macrotick, which is the minimal time slot of communication and scheduling.

Let $MT_{\mathcal{G}}$ denote the macrotick, then we have the following formulas

$$s_i.T = \left\lceil \frac{\tau_i.T}{MT_{\mathcal{G}}} \right\rceil \tag{1}$$

$$s_i.D = \left\lceil \frac{\tau_i.D}{MT_{\mathcal{G}}} \right\rceil \tag{2}$$

Let $W_{\mathcal{G}}$ denote the width of physical links in the TTNoC. $SD_{\mathcal{G}}$ and $PD_{\mathcal{G}}$ represent the switching delay in the switches and the propagation delay between adjacent nodes, and then, we have that

$$s_i.L = \left\lceil \frac{|s_i.R| \times (SD_{\mathcal{G}} + PD_{\mathcal{G}})}{MT_{\mathcal{G}}} \times \left\lceil \frac{\tau_i.S + \tau_i.H}{\tau_i.H} \right\rceil \right\rceil \tag{3}$$

where $MT_{\mathcal{G}}$ denotes the value of macrotick, $|s_i.R|$ equals the number of hops, and $SD_{\mathcal{G}} + PD_{\mathcal{G}}$ is the forwarding delay of each hop. This formula is based on Duato et al.[26] with adopting macrotick as the unit. By above formulas, given s_i and its route $s_i.R$, $s_i.T$, $s_i.D$, and $s_i.L$ can be derived by formulas 1–3, respectively. In other words, we want to determine the offset $s_i.\phi$ for each message communication.

We define the whole schedule set as $\mathcal{S} = \{\mathcal{S}.\mathcal{R}, \mathcal{S}.\Phi, \mathcal{S}.\mathcal{T}, \mathcal{S}.\mathcal{D}, \mathcal{S}.\mathcal{L}\}$, where $\mathcal{S}.\mathcal{R}$ is the routes of all messages, $\mathcal{S}.\Phi$ is the offset of all messages, $\mathcal{S}.\mathcal{T}$ is the period of all messages, $\mathcal{S}.\mathcal{D}$ is the relative deadline of all messages, and $\mathcal{S}.\mathcal{L}$ is the delay of all messages. Therefore, the key problem is to derive $\mathcal{S}.\Phi = \{s_i.\phi | s_i \in \mathcal{S}\}$ based on the given system model with *contention-free constraints*.

Contention-free constraints

The constraints for general time-triggered multi-hop network are presented in Steiner.[14] The constraints on the TTNoC are similar to those on general time-triggered network that a schedule $\mathcal{S}$ should ensure the contention free in both temporal and spatial domains.

Offset constraints. For the schedule of a message $s_i \in \mathcal{S}$, the offset $s_i.\phi$ must have non-negative values and we should guarantee that the message transmission is completed before the deadline of the communication. Therefore, we have

Constraint 1. $(s_i.\phi \geq 0) \wedge (s_i.\phi + s_i.L \leq s_i.D)$.

Link constraints. For each communication, a message occupies links in terms of a given route until the whole message is received by the sink node. The contention will happen if messages which share the overlapped route forward simultaneously. To show the link constraints, we first give the definition of overlap. Given two schedules of messages $s_i, s_j \in \mathcal{S}$, we define *overlap*(s_i, s_j) to indicate whether there are any common physical links between the routes of s_i and s_j. *overlap*(s_i, s_j) is true if the routes of s_i and s_j share at least one hop of physical links or false otherwise

$$overlap(s_i, s_j) = \begin{cases} true & s_i.R \cap s_j.R \neq \varnothing \\ false & s_i.R \cap s_j.R = \varnothing \end{cases} \quad (4)$$

where $s_i.R$ and $s_j.R$ are routes of s_i and s_j, respectively. Then, we define the communication time for message τ_i in its kth period as an interval $s_i(k)$

$$\begin{aligned} &s_i(k) = [k \times s_i.T + s_i.\phi\ , k \times s_i.T + s_i.\phi + s_i.L) \\ &s_i(k) \in \mathcal{Z} \end{aligned} \quad (5)$$

The schedule $\mathcal{S}$ should ensure that two messages are never transmitted on the identical link at the same time. In other words, the synthesized schedule must guarantee that the links on TTNoC are contention-free. There comes our constraint for links:

Constraint 2. Given any $p, q \in \mathbb{N}, s_i, s_j \in \mathcal{S}$ such that $i \neq j$, $overlap(s_i, s_j) s_i(p) \cap s_j(q) = \varnothing$.

Problem statement

The problem of schedule synthesis can be formulated based on the schedule synthesis description and constraints, which are given as follows:

- The architecture of a TTNoC $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ consisting of the set of m communicating nodes $\mathcal{V}$ and interconnection links $\mathcal{L} \subseteq \mathcal{V} \times \mathcal{V}$.
- The set of n time-triggered messages Γ with $\langle \tau_i.H, \tau_i.S, \tau_i.T, \tau_i.D \rangle$ for each $\tau_i \in \Gamma$.
- The routes $\mathcal{S}.\mathcal{R}$ of the schedule $\mathcal{S}$ corresponding to Γ.

We try to determine the offsets $\mathcal{S}.\Phi$ for messages Γ on TTNoC $\mathcal{G}$, which satisfies Constraints 1 and 2. To solve the problem, we present the memetic-based schedule synthesis algorithm to synthesize offset $\mathcal{S}.\Phi$ which meet the contention-free constraints.

Memetic-based schedule synthesis algorithm

In this section, we first give the motivation of schedule synthesis on TTNoC, and components' representation of MSSA is also given. Then, the algorithm description is proposed, followed by discussion about the details, that is, fitness function and global and local search strategies.

Motivation

To show the schedule synthesis problem, we first give an example of messages communication on a 3×3 mesh TTNoC, as shown in Figure 2.

And Table 1 shows the schedule information of messages on the TTNoC. The nodes on the TTNoC is denoted as $\mathcal{V} = \{v_0, \ldots, v_8\}$. The set of messages for communication is $\Gamma = \{\tau_0, \tau_1, \tau_2, \tau_3, \tau_4\}$. The period is a positive power of two in terms of macrotick according to the timing specification of TTNoC.[13] We assume that the relative deadline equals its period for each message, and we adopt XY routing[26] as the routing strategy for simplicity. The schedule synthesis problem is to determine the offset to avoid contention among messages. In other words, for each message in this example, we try to determine the offset in its possible offset.

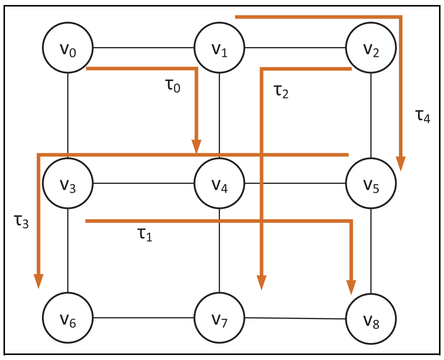

Figure 2. Motivation of messages communication on the TTNoC.

Table 1. The schedule information of messages in Figure 2.

Schedule	Route	Period	Deadline	Possible offset
s_0	$r_0^{\langle 0,4\rangle}$	2	1	$\{0,1\}$
s_1	$r_1^{\langle 3,8\rangle}$	4	1	$\{0,1,2,3\}$
s_2	$r_2^{\langle 2,7\rangle}$	4	1	$\{0,1,2,3\}$
s_3	$r_3^{\langle 5,6\rangle}$	8	2	$\{0,1,2,3,4,5,6\}$
s_4	$r_4^{\langle 1,5\rangle}$	8	1	$\{0,1,2,3,4,5,6,7\}$

Table 2. The component representation of the population consists of two individuals.

In.	0	1	2	3	4	5	6	7
in_0	$s_1.\phi$ $s_0.\phi$		$s_2.\phi$		$s_3.\phi$			$s_4.\phi$
in_1	$s_0.\phi$	$s_1.\phi$		$s_2.\phi$	$s_3.\phi$		$s_4.\phi$	

Components representation

The MSSA requires genetic representation which is similar with general GA. MSSA adopts the hyperperiod $Hp(\mathcal{S})$ as the definition of chromosome. The value of hyperperiod is the least common multiple of the period $\mathcal{S}.T$ among the schedule $\mathcal{S}$ for messages Γ, for example, the hyperperiod is 8 for the schedule on Table 1. We define the macrotick as a gene, and the location of a gene on a chromosome denotes the offset of a message. A chromosome represents a possible allocation for the offset of messages $\mathcal{S}.\Phi$. It should be noted that a single gene may consist of multiple offsets of messages since several messages without contention can be transmitted at the same time on the network-based TTNoC. Note that each individual is equipped with a score, which is evaluated by some fitness function. The fitness function is discussed later.

The population with two individuals in our motivation from Table 1 is proposed in Table 2. For individual in_0, the offset of message from s_0, s_1, s_2, s_3, s_4 is 0, 0, 2, 4, and 7, respectively.

Algorithm 1. MSSA description.

```
Input:
    the size of population $scale$
    the maximum number for iterations $N$
Output:
    best scheduling offsets $\mathcal{S}.\Phi$
 1: $pop \leftarrow initPop(scale)$
 2: $best \leftarrow selectBest(pop)$
 3: if $best.score = 0$ then
 4:     return $\mathcal{S}.\Phi \leftarrow best.chromosome$
 5: end if
 6: for $i = 0$ to $N - 1$ do
 7:   $pop \leftarrow crossover(pop)$
 8:   $pop \leftarrow mutation(pop)$
 9:   $pop \leftarrow localSearch(pop)$
10:   $best' \leftarrow selectBest(pop)$
11:   if $best'\ score < best\ score$ if
12:       $best \leftarrow best'$
13:   end if
14:   if $best\ score = 0$ if
15:       break
16:   end if
17:   $pop \leftarrow select(pop, scale)$
18: end for
19: return $\mathcal{S}.\Phi \leftarrow best\ chromosome$
```

Algorithm description

Memetic algorithm (MA) is widely used as a synergy of evolutionary or any population-based approach with separate individual learning or local improvement procedures for solution search.[27,28] Based on MA, MSSA consists global search and local search. We deploy the GA as the global search strategy. And the subset of infeasible schedule is chosen as the element to join the local search.

We formally give the algorithm as in Algorithm 1. When the MSSA starts, the offset of messages is generated randomly within the possible offsets by *initPop*(*scale*) (Line 1). The best individual *best* among the population is selected by *selectBest*(*pop*). If no feasible schedule exists in the initial *pop*, we start the iteration (Lines 6–18) until a feasible scheduling is derived (Lines 14–16) or an upperbound of the iteration number N is reached. For each iteration, the GA operations of *crossover*(*pop*) and *mutation*(*pop*) are employed as global search strategies. And the local search *localSearch*(*pop*) is employed after that. After the search, the best individual will be chosen (Lines 11–13). The iteration ends when the chromosome of *best* individual is a feasible schedule (Lines 14–16). The new population for the next iteration is generated by selection *select*(*pop*, *scale*) (Line 17). The individual with

Table 3. The route and overlapped set for schedule in Table 2.

Schedule $s_i \in \mathcal{S}$	Routing per hop $s_i.R$	Overlapped set $O(s_i)$
s_0	$\langle 0,1\rangle\langle 1,4\rangle$	s_2, s_4
s_1	$\langle 3,4\rangle\langle 4,5\rangle\langle 5,8\rangle$	s_4
s_2	$\langle 2,1\rangle\langle 1,4\rangle\langle 4,7\rangle$	s_0, s_4
s_3	$\langle 5,4\rangle\langle 4,3\rangle\langle 3,6\rangle$	$\varnothing$
s_4	$\langle 1,4\rangle\langle 4,5\rangle$	s_0, s_1, s_2

lower score owns more possibility to reserve for the next iteration.

Fitness function

The fitness function is to evaluate individuals in the population. To adopt the fitness function, we first determine the scenario of overlap among messages. And we transform chromosome to represent the communication time for messages in the hyperperiod. Then, the score of each individual can be acquired by the fitness function to evaluate the chromosome in Constraint 2.

By checking the forward routing for each message by Formula 4, we denote the set of overlapped schedule $O(s_i)$ for s_i based on Formula 4, and we have

$$O(s_i) = \{s_j | overlap(s_i, s_j), s_i, s_j \in \mathcal{S}\}$$

Table 3 shows the route based on XY routing and overlapped set for the messages depicted in Table 1. Since $overlap(s_0, s_1)$ and $overlap(s_0, s_3)$ are false while $overlap(s_0, s_2)$ and $overlap(s_0, s_4)$ are true, the overlapped set $O(s_0)$ is $\{s_2, s_4\}$.

The communication time of message in the kth period $s_i(k)$ is defined by Formula 5, so we have

$$D^{Hp}(s_i) = \{s_i(k) | k \in [0, Hp(\mathcal{S})/s_i.T - 1]\}$$

where $D^{Hp}(s_i)$ denotes the communication time in a hyperperiod. We define $D^{Hp}(\mathcal{S}) = \{D^{Hp}(s_i) | s_i \in \mathcal{S}\}$ that denotes the set of communication time for messages in a hyperperiod. Table 4 shows communication time for messages based on given individuals in Table 2.

For $s_i, s_j \in \mathcal{S}, s_i \neq s_j$, if s_i and s_j are overlapped, the times of conflicts between them should be derived to evaluate the chromosome. For $s_i \in \mathcal{S}$, we define $C(s_i)$ to derive the set of schedule conflicted with s_i. So, we have

$$C(s_i) = \{s_j | D^{Hp}(s_i) \cap D^{Hp}(s_j) \neq \varnothing, s_j \in O(s_i)\}$$

If there is no conflict with s_i on the its duration, then $C(s_i) = \varnothing$. We define the total times of conflicts for s_i as $times(s_i)$. And we have

Table 4. The communication time in a hyperperiod for the schedule in Table 2.

In.	0	1	2	3	4	5	6	7
in_0	τ_0		τ_0		τ_0		τ_0	τ_4
	τ_1		τ_2		τ_1	τ_3	τ_2	
					τ_3			
in_1	τ_0	τ_1	τ_0	τ_2	τ_0	τ_1	τ_0	τ_2
					τ_3	τ_3	τ_4	

Table 5. Evaluating individuals in Table 1 by the fitness function.

Individual *in*	Conflict $C(s_i)$	Conflict times $times(s_i)$	Score $fit(in)$
in_0	s_0, s_2	$times(s_0) = 2$	4
		$times(s_2) = 2$	
in_1	s_0, s_4	$times(s_0) = 1$	2
		$times(s_4) = 1$	

$$\begin{aligned} times(s_i) &= \sum_{s_j \in C(s_i)} conflict(s_i, s_j) \\ &= \sum_{s_j \in C(s_i)} |D^{Hp}(s_i) \cap D^{Hp}(s_j)| \end{aligned}$$

where the function $conflict(s_i, s_j)$ is to count the times of conflicts between s_i and s_j. We define $C(\mathcal{S}) = \{s_i | C(s_i) \neq \varnothing\}$ to denote the set of conflicting schedule. The fitness function is the sum of times of conflicts in $C(\mathcal{S})$. Thus, for a individual in, we have

$$fit(in) = \sum_{s_i \in C(\mathcal{S})} times(s_i)$$

Since the score represents the times of conflicts, the lower the score, the better the individual is, and the individual with score zero represents a feasible schedule. The score of two individuals in Table 2 is shown in Table 5.

Global search strategy

The general operation in a GA, such as crossover and mutation,[29] is used in the step of global search. For operation of crossover, we employ two of the individuals in_i and in_j among the population as parents to join the crossover in terms of their score. The lower score of an individual, the higher possibility for this individual to be selected as parent. The crossover generates a new individual called child, and its chromosome depends on the parents. The individual of parents with lower score owns higher possibility to determine the offset location $s_i.\phi$ for $s_i \in \mathcal{S}$ on the chromosome of child. For parents in_i and in_j, we define the function

Table 6. An example of the crossover between individuals in Table 2.

Pa.	0	1	2	3	4	5	6	7
in_0	$s_0.\phi$ $s_1.\phi$		$s_2.\phi$		$s_3.\phi$			$s_4.\phi$
in_1	$s_0.\phi$	$s_1.\phi$		$s_2.\phi$	$s_3.\phi$		$s_4.\phi$	
Ch.	0	1	2	3	4	5	6	7
in_2	$s_0.\phi$ $s_1.\phi$			$s_2.\phi$	$s_3.\phi$			$s_4.\phi$

Table 7. An example of the mutation for individuals in Table 2.

Before	0	1	2	3	4	5	6	7
in_0	$s_0.\phi$ $s_1.\phi$		$s_2.\phi$		$s_3.\phi$			$s_4.\phi$
After	0	1	2	3	4	5	6	7
in_0	$s_0.\phi$ $s_1.\phi$	$s_4.\phi$	$s_2.\phi$		$s_3.\phi$			

Table 8. Local search for individual in_0 in Table 2.

Before	0	1	2	3	4	5	6	7
in_0	$s_0.\phi$ $s_1.\phi$		$s_2.\phi$		$s_3.\phi$			$s_4.\phi$
Score	0	1	2	3	4	5	6	7
4	$s_0.\phi$ $s_1.\phi$		$s_2.\phi$		$s_3.\phi$			$s_4.\phi$
2	$s_1.\phi$	$s_0.\phi$	$s_2.\phi$		$s_3.\phi$			$s_4.\phi$
After	0	1	2	3	4	5	6	7
in_0	$s_0.\phi$ $s_1.\phi$	$s_4.\phi$	$s_2.\phi$		$s_3.\phi$			

$$Possibility(in_i) = 1 - \frac{in_i}{in_i + in_j}$$

to represent the possibility for in_i as a parent to determine the offset in the chromosome of child. Table 6 shows process of crossover. in_2 is generated by crossover based on the two individuals in Table 2 as parents. Because the score of in_0 is 4 while in_1 is 2, in_0 owns 67% while in_0 has 33% of possibility to determine the offset. A possible result of crossover is given: chromosome of in_2 with the offset of s_0, s_2, s_3 from in_1 and s_1, s_4 from in_0.

The operation of mutation randomly selects an offset and reallocate a new offset randomly in its possible offset. Table 7 is an example of mutation. It is noticed that the probability of mutation for each individual in our implement is 10%.

Local search strategy

Before the local search for each individual, the schedule $s_i \in \mathcal{S}$ with the maximum value of $times(s_i)$ is selected as local search element. If there are two or more individuals with the maximum times of conflicts, the element to be searched is selected stochastically among them. Then, the offset of message to be searched is reallocated in its possible offset. Each reallocated individual with minimum score replaces the previous individual.

Table 8 shows an example of local search for in_0 in Table 2. Since $times(s_0) = times(s_2) = 2$ in Table 5, we randomly select s_0 as the element. We locate the s_0 on its possible offsets 0 and 1. After evaluating individuals within possible offset $s_0.\phi$, we select the individual whose score is 2 as the updated in_0 because this individual owns the minimum score.

Evaluations

The network architecture we employed for simulation is a 2D mesh network whose scale is 3×3, 5×5, 7×7, 9×9, 11×11, and 13×13, respectively. The number of message in our evaluation is from 5 to 100. Each message owns its period and delay which is randomly generated. The period is randomly synthesized in $\{2^n | n \in [1, 10], n \in \mathcal{Z}\}$, and the delay is less than its period. We compare the MSSA with the GA in various architecture and number of messages, and for each

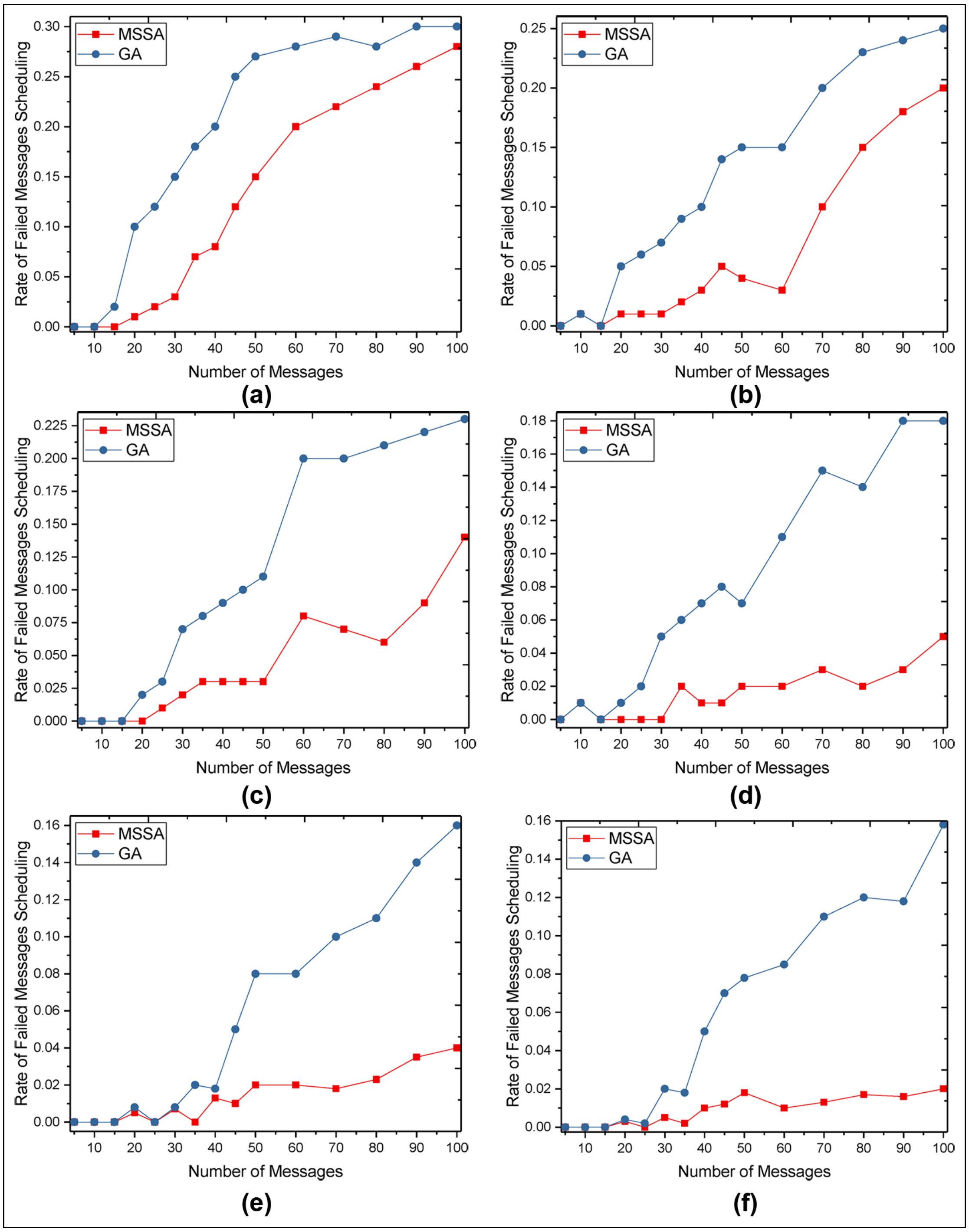

Figure 3. The failure rate of different number of messages on varying architectures: (a) 3 × 3 TTNoC, (b) 5 × 5 TTNoC, (c) 7 × 7 TTNoC, (d) 9 × 9 TTNoC, (e) 11 × 11 TTNoC, and (f) 13 × 13 TTNoC.

configuration of simulation, we repeat 15 times. The size of population in MSSA and GA is both 100.

Failure rate of schedule

The schedule of a message fails if the message communication faces contention with other messages based on this given schedule. To compare the performance of schedule synthesis of MSSA with GA, we define the failure schedule of messages as $\mathcal{S}.\Phi_{fail}$. And the failure rate of schedule R_{fail} as $|\mathcal{S}.\Phi_{fail}|/|\mathcal{S}.\Phi|$. Figure 3 shows R_{fail} for variable types of test cases by MSSA and GA, denoted as $R_{fail}(MSSA)$ and $R_{fail}(GA)$, respectively. The failure rate by the GA is higher than the MSSA in each

Table 9. The average failure rate of schedule in different scale of TTNoC.

Scale	$R_{fail}(GA)$	$R_{fail}(MSSA)$	$\frac{R_{fail}(GA)}{R_{fail}(MSSA)}$
3×3	0.1857	0.1261	0.6792
5×5	0.1226	0.0546	0.4452
7×7	0.0994	0.0345	0.3470
9×9	0.0721	0.0174	0.2411
11×11	0.0618	0.0120	0.1944
13×13	0.0557	0.0088	0.1582
Average	0.0995	0.0422	0.3442

configuration, especially when the scale of architecture and the number of messages are both relatively large, for example, for the case of 90 messages in 13×13 TTNoC, the rate is 11.9% for the GA while only 1.8% for the MSSA. It is proposed that the local search in the MSSA is the key step to significantly improve the performance.

We define $R_{fail}(MSSA)/R_{fail}(GA)$ in Table 9 to compare MSSA with GA in different scales of architecture. From the results, we can see that the $R_{fail}(MSSA)$ is about 34.4% of the $R_{fail}(GA)$. R_{fail} decreases when the scale of TTNoC architecture increases. The reason is that the link resource increases with the growth of architecture scale, resulting in decreasing the possibility of contention among the messages' transmission on the TTNoC. The failure rate of schedule increases when the number of messages on the TTNoC increases. It is because the limited link resources have to transmit more messages when there is more messages on the TTNoC.

Feasible cases

We consider the number of feasible cases in each experiment type. The schedule for all messages on TTNoC is feasible when there is no conflict between communication of messages in both spatial and temporal domains. In other words, a case of communication is feasible if each synthesized schedule of message $\mathcal{S}.\Phi$ is contention-free, otherwise it is an infeasible case.

Figure 4 presents the result of numbers of feasible cases in different scale of architecture after the schedule synthesis. For simple test cases with less than 15 messages, most of the cases are feasible. But for complex test cases of more than 60 messages, it is hard for MSSA to find a feasible schedule.

Discussion

Synthesizing a feasible schedule depends on the number of messages, the route of messages, and the scale of TTNoC. It is hard for MSSA to synthesize a feasible

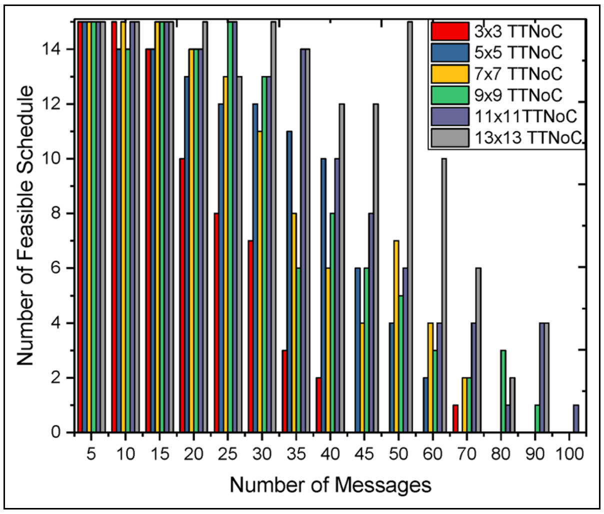

Figure 4. The number of feasible cases in different scale of TTNoC and number of messages.

scheduling when the set of messages is large and/or the scale of TTNoC architecture is small. However, except two reasons above, the mapping between messages and nodes as well as routing strategy also affect generating a feasible solution. The designer can manually change the mapping allocation or the routing strategy for the failure schedule or even design a larger scale of network architecture if needed. After these changes, the feasible scheduling can be generated by the MSSA iteratively.

Conclusion and future work

This article proposes an MSSA to synthesize the schedule for message communication on the TTNoC. MSSA deploys the GA as the global search strategy, and the subset of infeasible schedule is chosen as the element to minimize its contention times. We evaluate our MSSA and general GA with various randomly generated messages on varying scales of mesh-based TTNoC. The results show that our MSSA is effective to synthesize a schedule with less infeasible message communication, which is 34.2% compared with the general GA. In our future work, the mapping between messages and nodes and routing strategy will be integrated to our consideration before scheduling of messages on the TTNoC. And the different TTNoC architecture, for example, torus and hypercube, and different local search strategies, for example, simulated annealing, will also be implemented in our evaluations for the improvement of schedule synthesis performance.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research is sponsored in part by NSFC Program (No. 61527812), National Science and Technology Major Project (No. 2016ZX01038101), MIIT IT funds (Research and Application of TCN Key Technologies) of China, and The National Key Technology R&D Program (No. 2015BAG14B01-02).

References

1. Craciunas SS and Oliver RS. Combined task- and network-level scheduling for distributed time-triggered systems. *Real-Time Syst* 2016; 52(2): 161–200.
2. Steiner W, Bauer G, Hall B, et al. TTEthernet dataflow concept. In: *Proceedings of the eighth IEEE international symposium on networking computing and applications (NCA 2009)*, Cambridge, MA, 9–11 July 2009, pp.319–322. New York: IEEE.
3. Beji S, Hamadou S, Gherbi A, et al. SMT-based cost optimization approach for the integration of avionic functions in IMA and TTEthernet architectures. In: *18th IEEE/ACM international symposium on distributed simulation and real time applications (DS-RT 2014)*, Toulouse, 1–3 October 2014, pp.165–174. New York: IEEE.
4. Hu M, Luo J, Wang Y, et al. Scheduling periodic task graphs for safety-critical time-triggered avionic systems. *IEEE T Aero Elec Sys* 2015; 51(3): 2294–2304.
5. Kopetz H. GENESYS—a cross-domain architecture for dependable embedded systems. In: *5th Latin-American symposium on dependable computing (LADC 2011)*, São Paulo, 25–29 April 2011, pp.1–6. IEEE.
6. Zheng X, Cai Z, Li J, et al. Scheduling flows with multiple service frequency constraints. *IEEE Internet Things* 2017; 4(2): 496–504.
7. Zheng X, Cai Z, Li J, et al. An application-aware scheduling policy for real-time traffic. In: *35th IEEE international conference on distributed computing systems (ICDCS 2015)*, Columbus, OH, 29 June–2 July 2015, pp.421–430. New York: IEEE.
8. Zheng X, Cai Z, Li J, et al. A study on application-aware scheduling in wireless networks. *IEEE T Mobile Comput* 2017; 16: 1787–1801.
9. Cai Z, He Z, Guan X, et al. Collective data-sanitization for preventing sensitive information inference attacks in social networks. *IEEE T Depend Secure*. Epub ahead of print 26 September 2016. DOI: 10.1109/TDSC.2016.2613521.
10. Duan Z, Li W and Cai Z. Distributed auctions for task assignment and scheduling in mobile crowdsensing systems. In: *37th IEEE international conference on distributed computing systems (ICDCS 2017)*, Atlanta, GA, 5–8 June 2017, pp.635–644. New York: IEEE.
11. Shi T, Cheng S, Cai Z, et al. Exploring connected dominating sets in energy harvest networks. *IEEE ACM T Network* 2017; 25: 1803–1817.
12. He Z, Cai Z, Yu J, et al. Cost-efficient strategies for restraining rumor spreading in mobile social networks. *IEEE T Veh Technol* 2017; 66(3): 2789–2800.
13. Huang J, Blech JO, Raabe A, et al. Static scheduling of a time-triggered network-on-chip based on SMT solving. In: *2012 design, automation & test in Europe conference & exhibition (DATE 2012)*, Dresden, 12–16 March 2012, pp.509–514. New York: IEEE.
14. Steiner W. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In: *Proceedings of the 31st IEEE real-time systems symposium (RTSS 2010)*, San Diego, CA, 30 November–3 December 2010, pp.375–384. New York: IEEE.
15. Ding S. Scheduling approach for static segment using hybrid genetic algorithm in FlexRay systems. In: *10th IEEE international conference on computer and information technology (CIT 2010)*, Bradford, 29 June–1 July 2010, pp.2355–2360. New York: IEEE.
16. Lukasiewycz M, Schneider R, Goswami D, et al. Modular scheduling of distributed heterogeneous time-triggered automotive systems. In: *Proceedings of the 17th Asia and South Pacific design automation conference (ASP-DAC 2012)*, Sydney, NSW, Australia, 30 January–2 February 2012, pp.665–670. IEEE.
17. Pozo F, Steiner W, Rodriguez-Navas G, et al. A decomposition approach for SMT-based schedule synthesis for time-triggered networks. In: *20th IEEE conference on emerging technologies & factory automation (ETFA 2015)*, Luxembourg City, 8–11 September 2015, pp.1–8. New York: IEEE.
18. Craciunas SS and Oliver RS. SMT-based task- and network-level static schedule generation for time-triggered networked systems. In: *22nd international conference on real-time networks and systems (RTNS '14)*, Versaille, 8–10 October 2014, p.45. New York: ACM.
19. Craciunas SS, Oliver RS, Chmelik M, et al. Scheduling real-time communication in IEEE 802.1qbv time sensitive networks. In: *Proceedings of the 24th international conference on real-time networks and systems (RTNS 2016)*, Brest, 19–21 October 2016, pp.183–192. New York: ACM.
20. Zhang L, Goswami D, Schneider R, et al. Task- and network-level schedule co-synthesis of Ethernet-based time-triggered systems. In: *19th Asia and South Pacific design automation conference (ASP-DAC 2014)*, Singapore, 20–23 January 2014, pp.119–124. IEEE.
21. Ro JW, Roop PS and Malik A. Schedule synthesis for time-triggered multi-hop wireless networks with retransmissions. In: *IEEE 18th international symposium on real-time distributed computing (ISORC 2015)*, Auckland, New Zealand, 13–17 April 2015, pp.94–101. New York: IEEE.
22. Hu M, Luo J, Wang Y, et al. Holistic scheduling of real-time applications in time-triggered in-vehicle networks. *IEEE T Ind Inform* 2014; 10(3): 1817–1828.
23. Scholer C, Krenz-Baath R, Murshed A, et al. Optimal sat-based scheduler for time-triggered networks-on-a-chip. In: *10th IEEE international symposium on industrial embedded systems (SIES 2015)*, Siegen, 8–10 June 2015, pp.156–161. New York: IEEE.
24. Murshed A, Obermaisser R, Ahmadian H, et al. Scheduling and allocation of time-triggered and event-triggered services for multi-core processors with networks-on-a-chip. In: *13th IEEE international conference on industrial*

informatics (INDIN 2015), Cambridge, 22–24 July, pp.1424–1431. New York: IEEE.

25. Freier M and Chen J. Time-triggered communication scheduling analysis for real-time multicore systems. In: *10th IEEE international symposium on industrial embedded systems (SIES 2015)*, Siegen, 8–10 June 2015, pp.108–116. New York: IEEE.
26. Duato J, Yalamanchili S and Ni LM. *Interconnection networks—an engineering approach*. New York: IEEE, 1997.
27. Chen X, Ong Y, Lim M, et al. A multi-facet survey on memetic computation. *IEEE T Evolut Comput* 2011; 15(5): 591–607.
28. Ong Y, Lim M and Chen X. Memetic computation–past, present & future (research frontier). *IEEE Comp Int Mag* 2010; 5(2): 24–31.
29. Shi H, Song X, Gu M, et al. Task and participant scheduling of trading platforms in vehicular participatory sensing networks. *Sensors* 2016; 16(12): 2013.