J. Parallel Distrib. Comput. 83 (2015) 1–12

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems

Guoqi Xie [a,b,*], Renfa Li [a,b], Keqin Li [a,c]

[a] College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, 410082, China
[b] Key Laboratory for Embedded and Network Computing of Hunan Province, Hunan University, Changsha, Hunan, 410082, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

HIGHLIGHTS

- Introduce a completely heterogeneous network topology for accurate analysis on message and route.
- Make both the computation time and communication time relate to the specific processor.
- Consider the processor and route selection both from "upward" and "downward" comprehensively.
- Synchronize tasks and messages by the task earliest start time and the link finish time.
- End-to-end synchronized scheduling for tasks and messages for processor and route selection synchronously.

ARTICLE INFO

Article history:
Received 15 September 2014
Received in revised form
5 April 2015
Accepted 30 April 2015
Available online 7 May 2015

Keywords:
Heterogeneity
Networked embedded systems
DAG
Communication contention
Synchronized scheduling

ABSTRACT

Scheduling for a directed acyclic graph (DAG) on networked embedded systems is to maximize concurrency and minimize inter-processor communication for minimum end-to-end worst-case response time (WCRT). Time accuracy and synchronization are critical for scheduling on heterogeneous networked embedded systems, where computing and networking are both heterogeneous and deeply jointed. Most algorithms use the upward rank value for task prioritization, and the earliest finish time for processor selection. In order to obtain accurate and efficient schedules in heterogeneous networked systems, the above approaches can be improved. Moreover, synchronization with tasks and messages is critical for end-to-end WCRT. However, task scheduling and message scheduling are isolated in most approaches in communication contention environments. In this paper, a heterogeneity-driven task scheduling algorithm called Heterogeneous Selection Value (HSV) based on the classic model, and a heterogeneity-driven end-to-end synchronized scheduling algorithm called Heterogeneous Selection Value on Communication Contention (HSV_CC) based on the communication contention model are proposed to address the above problems. Both benchmark and extensive experimental evaluation demonstrate significant performance improvement of the proposed algorithms.

1. Introduction

1.1. Motivation

Over the last few years, the increasing number of processors, including sensors, actuators, and electronic control units (ECUs) in automobile, has increased the complexity of automotive electronic systems. Several vehicle heterogeneous buses, such as controller area network (CAN), FlexRay, Media Oriented System Transport (MOST), and Ethernet [23,30], have been deployed in automobile networks. Currently, there are at least about 70 ECUs with around 2500 signals in a luxury car [6,21]. Therefore, automobile electronic systems are heterogeneous systems, where computation and communication are both heterogeneous. Furthermore, parallel and distributed automobile functions with precedence constrained tasks have increased dramatically in automotive electronic systems [10,38]. Examples of active safety functions are x-by-wires and

* Corresponding author at: College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, 410082, China.
E-mail addresses: xgqman@126.com (G. Xie), lirenfa@vip.sina.com.cn (R. Li), lik@newpaltz.edu (K. Li).

http://dx.doi.org/10.1016/j.jpdc.2015.04.005
0743-7315/

adaptive cruise control (ACC). Therefore, automobile electronic systems can be described as heterogeneous networked embedded systems with heterogeneous computing, heterogeneous networking, and parallel processing.

More and more evidences show that the model for mapping functions is highly critical to the analysis of automobile electronic systems. Some models such as time chains [37], function chains [15], and task chains [13] have been employed for functions of automobiles. However, these are only suitable for simple applications. With increasing complexity and parallelization of automobile functions, a model for accurately reflecting the distributed characteristics of automobile applications is desirable.

An application with precedence constrained tasks at a high level of parallel and distributed computing is described as a directed acyclic graph (DAG), where nodes represent tasks, and edges represent communication messages between tasks. Undoubtedly, the DAG model can also be applied to heterogeneous networked embedded systems [18,19]. Scheduling tasks on processors with the objective of minimizing overall scheduling length (makespan) of a DAG is called the general multiprocessor scheduling problem. Scheduling tasks for fastest execution is a well-known NP-hard optimization problem, and many heuristic list scheduling algorithms have been proposed to generate near-optimal solutions of the problem in parallel and distributed systems [3,5,11,12,17,33].

List scheduling is the most popular method for precedence constrained task scheduling based on the DAG model. The basic idea of list scheduling is to put all tasks into a priority queue in a nonincreasing order, and a ready task with a higher priority is scheduled before a ready task with a lower priority. List scheduling is generally accepted as an attractive approach, since it combines the low complexity with good results [31,32].

Time accuracy is extremely strict in embedded real-time systems [1,29]. However, most DAG scheduling algorithms use the upward rank value for ordering tasks, where computation times for calculating the upward rank value are given by average values, and the earliest finish time (EFT) for selecting processors [31,32,35] from "downward", not from "upward". In order to obtain accurate and efficient schedules in heterogeneous networked systems, the above approaches can be improved.

Besides the accuracy of time, a more critical aspect is synchronized scheduling with tasks and messages, such that timing constraint with end-to-end worst-case response time (WCRT) is determined. End-to-end WCRT means a unified end-to-end execution and transmission path combined with tasks and messages to determine whether it can meet the real-time requirements [2]. There has been increasing interest and importance in end-to-end synchronized scheduling on networked embedded systems. However, although some approaches refer to synchronization, they do not take full advantage of DAG in parallel and distributed computing, and also do not create accurate and efficient schedules [18,19,22].

1.2. Our contributions

In this paper, we investigate end-to-end synchronized scheduling for precedence constrained tasks and messages on heterogeneous networked embedded systems. Our scheduling problems contain two topics, namely, scheduling precedence constrained tasks, and synchronized scheduling for tasks and messages. Accordingly, there are two types of scheduling models, namely, the classic model and the communication contention model.

The classic model for communication is described as follows [24,27].

(1) Every processor possesses a dedicated communication subsystem.
(2) The processors or the communication network are fully connected.
(3) All communications can be performed concurrently.

An important aspect missed by the classic model is the contention for communication resources. If a resource is occupied by one communication, any other communication requiring the same resource has to wait until it becomes available. In turn, the task depending on the delayed communication is also forced to wait. Thus, conflicts among communications generally result in a higher overall execution time [27]. The above scenarios are regarded as the communication contention problems. Hence, contention for communication resources is not considered in task scheduling.

In the communication contention model, however, computing systems and networking systems are deeply jointed. The contention for communication resources has a strong influence on the execution time of a parallel application, and the topology of a network is also required to be considered seriously. Therefore, we make the following agreement in this paper. Scheduling in the classical model is called DAG task scheduling, while scheduling in the communication contention model is treated as DAG task and message scheduling.

We will propose a novel DAG task scheduling algorithm called HSV (Heterogeneous Selection Value) with accurate analysis of time on computing heterogeneity by observing the upward rank value and the earliest finish time based on the classic model. We will also propose a novel DAG task and message scheduling algorithm called HSV_CC (Heterogeneous Selection Value on Communication Contention) with accurate analysis of time on networking heterogeneity, and synchronization of task and message by introducing a model of heterogeneous networks. Our scheduling problems and realistic data for experiments are also based on heterogeneous HS-CAN/LS-CAN (High-Speed-CAN/Low-Speed-CAN) networks connected by a gateway of automobiles.

The rest of the paper is organized as follows. In Section 2, we review related research. In Section 3, we propose and analyze a DAG task scheduling algorithm for the classic model. In Section 4, we propose and analyze a DAG task and message scheduling algorithm with synchronization for the communication contention model. In Section 5, we present simulation data and realistic data to verify the performance ratios of all our heuristic algorithms and approaches. We conclude the paper in Section 6.

2. Related work

The core idea of DAG list scheduling includes two phases, where the first phase is to order tasks according to a list in descending order of priorities, and the second phase is to assign each task to a proper processor. In [35], Topcuoglu et al. proposed the most famous and classical algorithms called HEFT (Heterogeneous Earliest Finish Time) and CPOP (Critical Path On a Processor) for scheduling tasks with precedence constraint on heterogeneous computing systems to reduce makespan to a minimum. This work has been extended in [3–5,12,31–33], and inspired substantial further investigation. The investigation in [16] is the latest work for above conditions, in which, an algorithm called CEFT (Constrained Earliest Finish Time) was proposed by defining and employing the constrained critical paths. Most algorithms use the upward rank value for ordering tasks, and the earliest finish time for selecting processors, where computation times and communication times are both based on average values for calculating. Therefore, these approaches do not obtain accurate and efficient schedules for networked embedded systems, where accuracy of time is extremely critical.

The main features of the above algorithms are only interested in task scheduling, and do not take message scheduling into

Table 1
Protocols for each communication bus standard with communication contention.

Bus standard	Protocol
LIN	Serial network
CAN	CSMA/CR (Carrier Sense Multiple Access/Collision Resolution)
FlexRay	TDMA (Time Division Multiple Access)
MOST	TDMA
Ethernet	CSMA/CD (Carrier Sense Multiple Access with Collision Detection)

consideration in their problems. That is, a simple classic model is used in the above algorithms to heavily idealize the target parallel systems. It is simplified that all communications can happen at the same time, and that all processors are fully connected. In other words, there is no contention for communication resources [24,25,28]. However, computing and networking are both heterogeneous, and have equal importance in networked embedded systems. The approach of merely considering task scheduling cannot reflect the realistic characteristics of such systems, and does not suffice for accurate and efficient scheduling. Experiments in [26] also showed that the consideration of contention for communication is essential for the generation of accurate and efficient schedules.

Scheduling with communication contention has also been studied extensively by many researchers. In early research, LogP is a communication contention model of parallel computation [8], and LogGPS is a communication contention model for synchronization analysis [14]. Recently, list scheduling of the communication contention model has also attracted many investigations. In [24,25], Sinnen et al. proposed a contention aware modified list scheduling scheme for realistic communication subsystem models of homogeneous computing systems. The algorithm resolves contention by appropriate communication routing. The approach in [28] was extended to reduce scheduling path based on task duplication [27]. In [34], Tang et al. presented the list scheduling algorithm for communication contention with arbitrary processor network (APN) on heterogeneous computing systems. In [7], Choudhury et al. even considered online scheduling of dynamic task graphs with communication contention for homogeneous multiprocessor embedded systems, where broadcast and point-to-point communication models were presented.

Even though the accuracy and efficiency of scheduling were significantly improved by considering communication contention, the above work cannot be applied to heterogeneous networked embedded systems for three reasons. First, these are mainly based on homogeneous computing environments. However, computing is fully heterogeneous in heterogeneous networked embedded systems. Second, the topology graph of a network mostly uses BTN (binary tree network) or LWT (LAN with switch) with only one communication route, or APN also called fully connected processors with $p(p-1)/2$ communication routes, where p represents the number of processors. However, the topology of embedded networks, e.g., automobile networks, are integrated with multiple heterogeneous networks and gateways for exchange messages; and the number of communication routes is different between different processors, namely, communication route is required to be computed by strict searching. Third and most important, end-to-end synchronized scheduling for tasks and messages is critical in networked embedded systems, such that timing constraints including end-to-end deadlines are satisfied. However, current and existing DAG scheduling algorithms with communication contention assume that tasks are not synchronized with the messages, where communication delays are possible and induce performance degradation on end-to-end WCRT [19]. Therefore, end-to-end WCRT should be rigorously considered and analyzed based on synchronized scheduling of tasks and messages for an application in networked embedded systems.

Table 2
Computation time matrix in Fig. 1.

Task	p_1	p_2	p_3
n_1	14	16	9
n_2	13	19	18
n_3	11	13	19
n_4	13	8	17
n_5	12	13	10
n_6	13	16	9
n_7	7	15	11
n_8	5	11	14
n_9	18	12	20
n_{10}	21	7	16

Both the classic model and communication contention model can be applied to some real applications. The classic model can be suitable for the heterogeneous network-on-chip (NoC) systems where there are particular communication links between two cores, namely, communication is parallel. The communication contention model is suitable for many heterogeneous automobile electronic systems where most communication bus standard (e.g., LIN, CAN, FlexRay, MOST, and Ethernet) are serial. Table 1 shows the protocols for each communication bus standard with communication contention.

3. Task scheduling

3.1. Application model

An application can be represented by a DAG $G = (N, E, W)$. N represents a set of n nodes, and each node $n_i \in N$ represents a task, which has different computation values on different processors. E is a set of communication edges, and each edge $e_{i,j} \in E$ represents the communication message from task n_i to n_j. Accordingly, $msg(e_{i,j})$ represents the size of message $e_{i,j}$. $pred(n_i)$ represents the set of n_i's immediate predecessor tasks. $ind(n_i)$ represents n_i's in-degree, which means the cardinality of $pred(n_i)$. A task is triggered to execute only if all its predecessor tasks have been executed. $succ(n_i)$ represents the set of n_i's immediate successor tasks. $outd(n_i)$ represents n_i's out-degree, which means the cardinality of $succ(n_i)$. The task which has no predecessor task is called n_{entry}, and the task which has no successor task is called n_{exit}. W is a $|N| \times |P|$ matrix, in which $w_{i,u}$ denotes the computation time to run task n_i on processor p_u.

Fig. 1 shows a benchmark of DAG with assigned task and edge weights. The benchmark was employed by many algorithms, e.g., CPOP [35], HEFT [35], HCNF (Heterogeneous Critical Node First) [5], and HCPFD (Heterogeneous Critical Parents with Fast Duplicator) [12]. Table 2 is a matrix of computation times in Fig. 1. The benchmark shows 10 tasks and 3 processors. The weight 18 of edge between n_1 and n_2 represents the message size denoted by $msg(e_{i,j}) = 12$. The weight 14 of n_1 and p_1 in Table 2 represents the computation time denoted by $w_{1,1} = 14$.

3.2. Task prioritizing

In DAG task scheduling, the computation time is the actual execution time of a task, while the communication time is the

Table 3
Values of HPRV in Fig. 1.

Task	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
$rank_u(n_i, p_1)$	113	81	79	88	77	65	45	37	52	21
$rank_u(n_i, p_2)$	103	67	75	64	58	60	39	29	32	7
$rank_u(n_i, p_3)$	110	83	86	89	72	65	44	41	49	16
$hrank_u(n_i)$	108.7	77	80	80.3	69	63.3	42.7	35.7	44.3	14.7
$outd(n_i)$	5	2	1	2	1	1	1	1	1	0
$HPRV(n_i)$	543.3	154	80	160.7	69	63.3	42.7	35.7	44.3	0

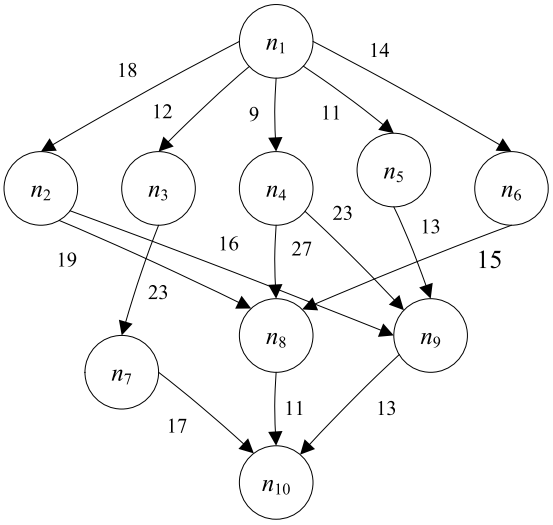

Fig. 1. A benchmark of DAG with 10 tasks.

average communication time of a message denoted by

$$\overline{c_{i,j}} = \frac{msg(e_{i,j})}{speed(e_{i,j})}, \tag{1}$$

where $msg(e_{i,j})$ means the size of message $e_{i,j}$ mentioned before, and $speed(e_{i,j})$ denotes the communication speed of $e_{i,j}$. All messages have the same communication speed in the classical model, which is denoted by 1, namely, the communication time has the same value as the size of a message.

As we know, the upward rank value of a task given by Eq. (2) is employed in many algorithms [12,33,35],

$$rank_u(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} \{\overline{c_{i,j}} + rank_u(n_j)\}, \tag{2}$$

and is considered as the common task prioritizing, where the tasks are ordered according to a nonincreasing order of $rank_u$.

We find that Eq. (2) uses the average computation time $\overline{w_i}$. However, in heterogeneous computing systems, each task has variable computation time on different processors. Hence, for accuracy, each task should have its own upward rank value on different processors. We define in Eq. (3) a new upward rank value for each task on different processors:

$$\begin{cases} rank_u(n_i, p_u) = \max\limits_{n_j \in succ(n_i)} \{rank_u(n_j, p_u) + w_{i,u} + \overline{c_{i,j}}\}; \\ rank_u(n_{exit}, p_u) = w_{exit,u}. \end{cases} \tag{3}$$

Then, $rank_u(n_i, p_u)$ for all processors is averaged as

$$hrank_u(n_i) = \frac{1}{|P|} \times \sum_{p_u \in P} rank_u(n_i, p_u). \tag{4}$$

Undoubtedly, $hrank_u(n_i)$ can get more accurate sorting results compared with $rank_u(n_i)$. However, it is not enough, and we give a new definition to explain.

Definition 1 (*Heterogeneous Priority Rank Value (HPRV)*)**.** The HPRV of task n_i represents the product of the out-degree and the average upward rank value for task n_i, namely,

$$HPRV(n_i) = outd(n_i) \times hrank_u(n_i). \tag{5}$$

The out-degree of a task also affects the task priority ordering. If a task with more immediate successor tasks is not executed preferentially, it may result in unreadiness of all its successors, and increase the scheduling length directly or indirectly. Therefore, the descending order of $HPRV(n_i)$ is treated as the task prioritizing in our approach.

Table 3 shows the values of each task's $rank_u(n_i, p_u)$, $hrank_u(n_i)$, and $HPRV(n_i)$ of the example. All tasks are ordered according to a nonincreasing order of $HPRV(n_i)$. Hence, the task ordering is $\{n_1, n_4, n_2, n_3, n_5, n_6, n_9, n_7, n_8, n_{10}\}$ in the example of Fig. 1.

3.3. Processor selection

As we know, the attributes $EST(n_j, p_u)$ and $EFT(n_j, p_u)$ represent the earliest start time (EST) and the earliest finish time of task n_j on processor p_u respectively. There are calculated by

$$\begin{cases} EST(n_{entry}, p_u) = 0; \\ EST(n_j, p_u) = \max\left(avail[p_u], \max\limits_{n_i \in pred(n_j)} \{AFT(n_i) + c_{i,j}\}\right); \end{cases} \tag{6}$$

and

$$EFT(n_j, p_u) = EST(n_j, p_u) + w_{j,u}. \tag{7}$$

$avail[p_u]$ is the earliest available time when processor p_u is ready for task execution. $c_{i,j}$ represents communication time. If n_i and n_j are allocated to the same processor, then $c_{i,j} = 0$, else $c_{i,j} = \overline{c_{i,j}}$. $AFT(n_i)$ is the actual finish time of task n_i. If n_i is the exit task, $AFT(n_{exit})$ is the schedule length of DAG. That is,

$$makespan = AFT(n_{exit}). \tag{8}$$

HEFT and other algorithms [31–33,35] only consider the earliest finish time as the processor selection criteria from "downward". However, it neglects one more important effective factor, that is, from "upward". A new concept for "upward" is defined as the longest distance exit time (LDET), and is given by

$$LDET(n_i, p_u) = rank_u(n_i, p_u) - w_{i,u}. \tag{9}$$

A task, which has smaller value of EFT but larger value of LDET, may not get expected result effectively. Since the topology of DAG also influences the overall scheduling length. Therefore, we are required to optimize the processor selection criteria not only from "downward", but also from "upward", and give a new definition for processor selection.

Definition 2 (*Heterogeneous Selection Value (HSV)*)**.** The HSV of task n_i means the product of EFT and LDET of task n_i executed on process p_u. That is,

$$HSV(n_i, p_u) = EFT(n_i, p_u) \times LDET(n_i, p_u). \tag{10}$$

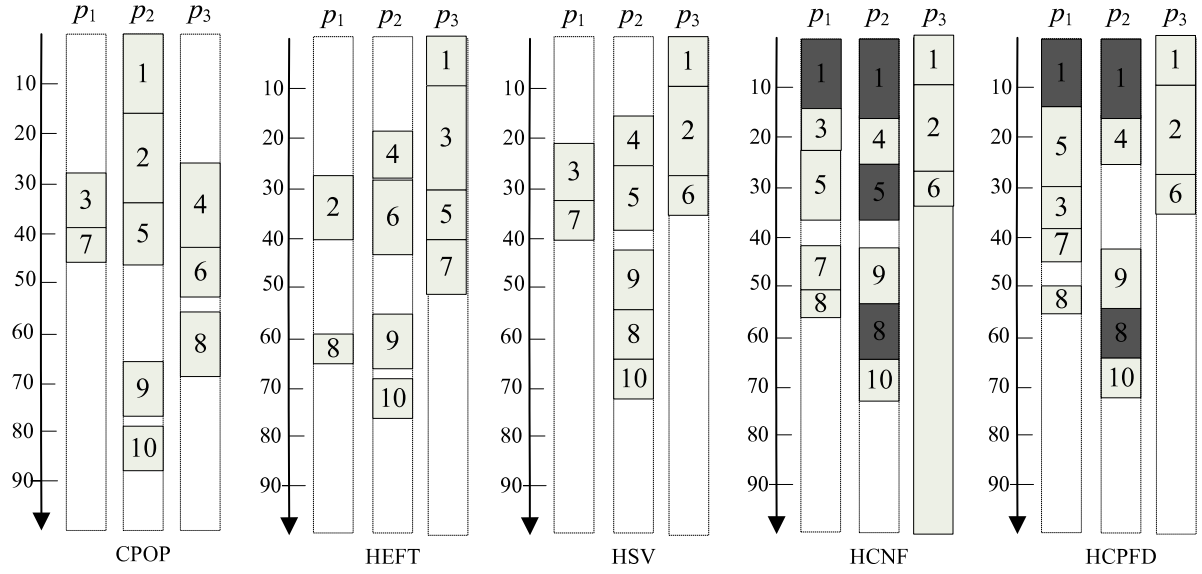

Fig. 2. Gantt charts of five task scheduling algorithms.

HSV considers the selection of processor from both "downward" and "upward" comprehensively, and this is the biggest improvement compared with most classical DAG algorithms [31,32,35], Therefore, we will propose the task scheduling algorithm called HSV in the next section.

3.4. The HSV algorithm

We describe the steps of the HSV algorithm in Algorithm 1.

Algorithm 1 HSV Algorithm

1: Compute each task's $rank_u(n_i, p_u)$, $hrank_u(n_i)$, and $HPRV(n_i)$, and put all tasks into a priority queue according to a nonincreasing order of $HPRV(n_i)$;
2: **while** there are tasks to be scheduled in the priority queue **do**
3: Select the task n_r with the maximum $HPRV(n_r)$;
4: Compute the $EFT(n_r, p_u)$, $LDET(n_r, p_u)$, and $HSV(n_r, p_u)$ of task n_r;
5: Assign task n_r to the processor p_u that has the minimum $HSV(n_r, p_u)$ using an insertion-based scheduling policy;
6: Mark n_r as a scheduled task;
7: **end while**

The time complexity of a scheduling algorithm for parallel applications with DAGs is usually expressed in terms of the number of tasks n and the number of processors p. The time complexity of the HSV algorithm is analyzed as follows. Computing $rank_u(n_i, p_u)$, $hrank_u(n_i)$, and $HPRV(n_i)$ must traverse all tasks and compare all processors, which can be done within $O(p \times n)$ time in an initialization phase. Scheduling all tasks must traverse all tasks, which can be done in $O(n)$ time. Computing the HSV values of all tasks can be done in $O(p \times n)$ time. Thus, the complexity of algorithm HSV is $O(p \times n^2)$, which is the same as the complexity of the HEFT and CPOP algorithms, and outperforms the CEFT algorithm which is $O(p \times n^3)$.

We list the Gantt graphs of some algorithms where the benchmark were employed, as shown in Fig. 2. The task ordering of the HSV algorithm is $\{n_1, n_4, n_2, n_3, n_5, n_6, n_9, n_7, n_8, n_{10}\}$, and the schedule length is 73. We can see that the makespan of the CPOP and HEFT algorithms are 86 and 80 respectively. The HSV algorithm has shorter makespan and less communication times. The makespan of the HCNF and HCPFD algorithms, which employ task duplication, are also 73 [5,12]. The HSV algorithm, without employing task duplication and increasing the time complexity, achieves the same schedule length as the HCNF and HCPFD algorithms.

4. Synchronized scheduling

Communication is concurrent, and it is isolated with computation in the classical model. In order to obtain a more realistic description, communication contention was introduced to schedule messages in most algorithms [24–26,28,34]. However, most of them ignore two critical issues, i.e., the heterogeneity of networking, and the synchronization of tasks and messages, which could also affect the scheduling length. Next, we will propose a novel synchronized scheduling algorithm, which includes the heterogeneity of networking and synchronization into consideration for more accurate and efficient schedules.

4.1. Heterogeneous networks

Automobile networks are no longer bus topologies completely, but a mixture of various types of network topologies, including bus, star, ring, tree, mesh types, etc. For example, CAN and FlexRay are usually configured with a bus topology, but they can be divided by means of gateways to form other topology over different domains. MOST in automotive networks is generally configured with a ring topology. Therefore, compared with many different types of single networks [24–26,28,34], the topology of an automotive network is a hybrid network topology, which consists of different network technologies with different bandwidth. Therefore, the same message transmitted in different links has different communication speeds.

We use an undirected graph $TG = \langle P, S, L\rangle$ to represent a heterogeneous network as follows. P is a set of heterogeneous processors mentioned before. S represents a set of gateways or switches. $L = \{l_1, l_2, \ldots, l_v\}$ represents a set of v links, and the value of a link represents its communication speed. Assume that we are given a message $e_{i,j}^{p_{src}, p_{dest}}$ to be transmitted from task n_i to task n_j, where p_{src} represents the source processor assigned to n_i, and p_{dest} represents the destination processor assigned to n_j. Therefore, there exist multiple communication routes between p_{src} and p_{dest} for message $e_{i,j}^{p_{src}, p_{dest}}$. We use $R^{p_{src}, p_{dest}} = \{r_1^{p_{src}, p_{dest}}, r_2^{p_{src}, p_{dest}}, \ldots, r_z^{p_{src}, p_{dest}}\}$ to denote the route set between p_{src} and p_{dest}. Each route, which may pass through multiple processors, gateways, and other

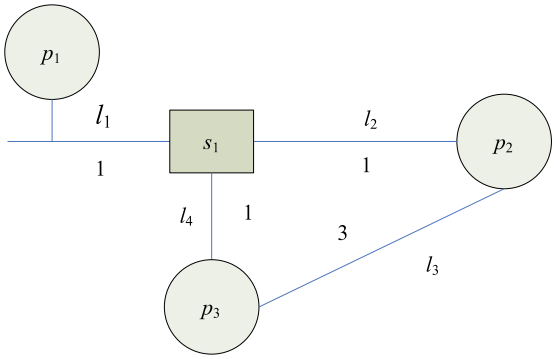

Fig. 3. A sample heterogeneous automobile network.

physical equipments, is composed of multiple links with different communication speeds. The routes of a message can be searched by the depth-first search (DFS) or the breadth-first search (BFS) algorithms [34].

Fig. 3 shows a simple example of automobile networks. There are 2 routes between p_1 and p_2, denoted by $R^{p_1,p_2} = \{r_1^{p_1,p_2}, r_2^{p_1,p_2}\}$, where $r_1^{p_1,p_2} = \{l_1^1, l_2^1\}$ and $r_2^{p_1,p_2} = \{l_1^2, l_4^2, l_3^2\}$ represent the links set of each route respectively, namely, route $r_1^{p_1,p_2}$ is composed of two links l_1 and l_2, and route $r_2^{p_1,p_2}$ is composed of three links l_1, l_4, and l_3. Furthermore, each link has different communication speed. Communication heterogeneity includes different communication protocols and speeds for different links. In this paper, we mainly focus on the heterogeneity of communication speeds. Undoubtedly, the heterogeneous network model can also be suitable for generic networks.

In the communication contention environments, the network topology of [28] is a star network topology with only 1 route between two processors; while APN presented in [34] considers that there are $p(p-1)/2$ routes between any processors. These are both too idealistic, and do not conform to the reality of heterogeneous automobile networks. Hence, the heterogeneous network topology proposed in this paper is according to the characteristics of automobile networks, where the number of routes between two processors belongs to the range of $[1, p(p-1)/2]$, and is based on strict searching approaches with BFS or DFS. This is the biggest difference with work in [28] and [34].

4.2. Task prioritizing in communication contention

We also find that Eq. (3) for calculating upward rank values use the average communication time. However, for the requirement of time accuracy, we recognize that each message has different communication times on different source processors. Thus, we have

$$\begin{cases} rank_u(n_i, p_{src}) = w_{i,src} + \max\limits_{n_j \in succ(n_i)} \{rank_u(n_j, p_{src}) + c_{i,j}^{p_{src}}\}; \\ rank_u(n_{exit}, p_{src}) = w_{exit,src}; \end{cases} \tag{11}$$

where $c_{i,j}^{p_{src}}$ represents the average communication time on source processor p_{src}. Therefore, the problem of obtaining more accurate value of $c_{i,j}^{p_{src}}$ is required to be solved.

First, assume that there exist z routes denoted by $R^{p_{src},p_{dest}} = \{r_1^{p_{src},p_{dest}}, r_2^{p_{src},p_{dest}}, \ldots, r_z^{p_{src},p_{dest}}\}$ for message $e_{i,j}^{p_{src},p_{dest}}$. In order to get $c_{i,j}^{p_{src}}$, we should obtain the communication speeds of $R^{p_{src},p_{dest}}$ denoted by $speed(R^{p_{src},p_{dest}})$ in advance. The value of $speed(R^{p_{src},p_{dest}})$ is the average value of speed for all routes, namely,

$$speed(R^{p_{src},p_{dest}}) = \frac{1}{z}\sum_{t=1}^{z} speed(r_t^{p_{src},p_{dest}}). \tag{12}$$

Table 4
Communication speeds of routes in Fig. 3.

Route	R^{p_1,p_2}	R^{p_1,p_3}	R^{p_2,p_3}	R^{p_2,p_1}	R^{p_3,p_1}	R^{p_3,p_2}
Speed	1	1	2	1	1	2

Table 5
Communication speeds of processors in Fig. 3.

Route	p_1	p_2	p_3
Speed	1.0	1.5	1.5

We assume that the gateway is only for message forwarding with cut-through communication scheme. Hence, the $speed(R^{p_{src},p_{dest}})$ is determined by a link with the minimum speed of $R^{p_{src}}$, namely,

$$\begin{aligned} speed(R^{p_{src},p_{dest}}) &= \frac{1}{z}\sum_{t=1}^{z} speed(r_t^{p_{src},p_{dest}}) \\ &= \frac{1}{z}\sum_{t=1}^{z} \min_{l_v \in L}\{speed(l_v^t)\}. \end{aligned} \tag{13}$$

For example, there are 2 routes with $\{l_1, l_4\}$ and $\{l_1, l_2, l_3\}$ between p_1 and p_3 of Fig. 3. In $\{l_1, l_2, l_3\}$, although the speed of l_3 is 3, it is restricted by the speed of l_1 and l_2, which is 1. Hence, the speed of route $\{l_1, l_2, l_3\}$ is 1, which is the same as l_1 and l_2.

According to Eq. (13), we can conclude the communication speeds for different routes of heterogeneous networks, as shown in Table 4.

Moreover, we let $speed(p_{src})$ represent the communication speed of source processor p_{src}, and the value of $speed(p_{src})$ is the average value of communication speed between p_{src} and other processors, namely,

$$speed(p_{src}) = \frac{1}{|P|-1}\sum_{p_{dest}\in P, p_{dest}\neq p_{src}} speed(R^{p_{src},p_{dest}}). \tag{14}$$

According to (14), we can get the communication speed of each source processor, as shown in Table 5.

To this step, we can get the value $c_{i,j}^{p_{src}}$ we require, namely,

$$c_{i,j}^{p_{src}} = \frac{msg(e_{i,j})}{speed(p_{src})}. \tag{15}$$

Compared with other algorithms based on communication contention, the biggest difference is that we make the communication time related to the specific processor, namely, each processor has its own average communication time. This approach is more accurate on time, and can obtain more reliable and efficient schedules than some approaches in [24,27], where the communication time is equal for all processors.

Similarly, in the environments of communication contention, we have

$$hrank_{ucc}(n_i) = \frac{1}{|P|} \times \sum_{p_u \in P} rank_{ucc}(n_i, p_u), \tag{16}$$

and

$$HPRV\_CC(n_i) = outd(n_i) \times hrank_{ucc}(n_i), \tag{17}$$

where $hrank_{ucc}$ represents the $hrank_u$ for communication contention. The HPRV_CC represents the heterogeneous priority rank value for communication contention, and it is employed for task prioritizing.

By means of Eq. (17), the priorities of tasks for the benchmark with communication contention is $\{n_1, n_4, n_2, n_3, n_5, n_6, n_9, n_7, n_8, n_{10}\}$.

4.3. Synchronization for processor selection

Similarly, the earliest start time is also critical for processor selection in communication contention environments. However, the value is not only determined by processor p_{dest}, but also by route $r_z^{p_{src},p_{dest}}$:

$$\begin{cases} EST(n_{entry}, p_{dest}, r_z^{p_{src},p_{dest}}) = 0; \\ EST(n_j, p_{dest}, r_z^{p_{src},p_{dest}}) = \max\{avail[p_{dest}], \\ \quad \max\limits_{n_i \in pred(n_j), proc(n_i)=p_{src}, p_{src} \in P} \{MFT(e_{i,j}^{p_{src},p_{dest}}, r_z^{p_{src},p_{dest}})\}\}; \end{cases} \tag{18}$$

where $MFT(e_{i,j}^{p_{src},p_{dest}}, r_z^{p_{src},p_{dest}})$ represents the finish time of message $e_{i,j}^{p_{src},p_{dest}}$ on route $r_z^{p_{src},p_{dest}}$. Even more, we use $MFT(e_{i,j}^{p_{src},p_{dest}}, r_z^{p_{src},p_{dest}})$ to represent the finish time of the last link l_{end}^z of route $r_z^{p_{src},p_{dest}}$. We have

$$\begin{cases} EST(n_{entry}, p_{dest}, r_z^{p_{src},p_{dest}}) = 0; \\ EST(n_j, p_{dest}, r_z^{p_{src},p_{dest}}) = \max\{avail[p_{dest}], \\ \quad \max\limits_{n_i \in pred(n_j), proc(n_i)=p_{src}, p_{src} \in P} \{LFT(e_{i,j}^{p_{src},p_{dest}}, l_{end}^z, r_z^{p_{src},p_{dest}})\}\}; \end{cases} \tag{19}$$

where $LFT(e_{i,j}^{p_{src},p_{dest}}, l_{end}^z, r_z^{p_{src},p_{dest}})$ represents the link finish time (LFT). A detailed definition and a calculation method of LFT will be presented later. Eq. (19) indicates that EST and LFT are closely related. When a task starts a message, it may not immediately transfer. Since the communication links to be transmitted may be occupied by others messages until there are idle.

Similarly, the earliest finish time of a task in communication contention environments is changed to

$$EFT(n_j, p_{dest}, r_z^{p_{src},p_{dest}}) = EST(n_j, p_{dest}, r_z^{p_{src},p_{dest}}) + w_{j,u}. \tag{20}$$

By means of the above analysis, we can synchronize tasks and messages. Since the earliest start time of a task depends on the message transmission time; while the message transmission time is restricted by the actual link finish time LFT. Therefore, for the value of LFT, we are required to obtain the link start time (LST) in advance.

The $LST(e_{i,j}^{p_{src},p_{dest}}, l_{x+1}^z, r_z^{p_{src},p_{dest}})$ represents the l_{x+1}^z's start time when message $e_{i,j}^{p_{src},p_{dest}}$ is transferred on route $r_z^{p_{src},p_{dest}}$. The recursive equation of LST is

$$\begin{cases} LST(e_{i,j}^{p_{src},p_{dest}}, l_1^z, r_z^{p_{src},p_{dest}}) = max\{AFT(n_i^{p_{src}}), avail(l_1^z)\}; \\ LST(e_{i,j}^{p_{src},p_{dest}}, l_{x+1}^z, r_z^{p_{src},p_{dest}}) \\ \quad = max\{LST(e_{i,j}^{p_{src},p_{dest}}, l_x^z, r_z^{p_{src},p_{dest}}), avail(l_{x+1}^z)\}. \end{cases} \tag{21}$$

Similarly, we use $LFT(e_{i,j}^{p_{src},p_{dest}}, l_{x+1}^z, r_z^{p_{src},p_{dest}})$ to represent link finish time, which is:

$$\begin{cases} LFT(e_{i,j}^{p_{src},p_{dest}}, l_1^z, r_z^{p_{src},p_{dest}}) \\ \quad = LST(e_{i,j}^{p_{src},p_{dest}}, l_1^z, r_z^{p_{src},p_{dest}}) + MCLT(e_{i,j}^{p_{src},p_{dest}}, l_1^z); \\ LFT(e_{i,j}^{p_{src},p_{dest}}, l_{x+1}^z, r_z^{p_{src},p_{dest}}) \\ \quad = max\{LFT(e_{i,j}^{p_{src},p_{dest}}, l_x^z, r_z^{p_{src},p_{dest}}), \\ LST(e_{i,j}^{p_{src},p_{dest}}, l_{x+1}^z, r_z^{p_{src},p_{dest}}) + MCLT(e_{i,j}^{p_{src},p_{dest}}, l_{x+1}^z)\}; \end{cases} \tag{22}$$

where $MLCT(e_{i,j}^{p_{src},p_{dest}}, l_x^z)$ represents the communication time of message $e_{i,j}^{p_{src},p_{dest}}$ on link l_x^z, namely,

$$MLCT(e_{i,j}^{p_{src},p_{dest}}, l_x^z) = \frac{w(e_{i,j}^{p_{src},p_{dest}})}{speed(l_x^z)}, \tag{23}$$

where $speed(l_x^z)$ represents the communication speed of link l_x^z.

Similarly, we are required to change the processor selection criteria in communication contention environments. That is,

$$LDET\_CC(n_i, p_u) = rank_{ucc}(n_i, p_u) - w_{i,u}, \tag{24}$$

and

$$HSV\_CC(n_j, p_{dest}, r_z^{p_{src},p_{dest}}) = EFT(n_j, p_{dest}, r_z^{p_{src},p_{dest}}) \times LDET\_CC(n_i, p_{dest}), \tag{25}$$

where LDET_CC represents the longest distance exit time for communication contention. HSV_CC represents the heterogeneous selection value for communication contention, and it is employed for processor selection in communication contention environments.

Based on the above analysis, we consider the synchronization of tasks and messages by the task earliest start time in communication contention environments, namely, we join the task and message from the view of end-to-end WCRT for an application, and the selections of processors and routes can be obtained simultaneously. Compared with the work in [24,27,28,34], where the task earliest time and the link finish time are calculated separately, our approach can obtain more accurate results. This is the maximum improvement compared with other approaches in communication contention environments.

4.4. The HSV_CC algorithm

We describe the steps of the HSV_CC algorithm in Algorithm 2.

Algorithm 2 HSV_CC Algorithm

1: Compute each task's $rank_{ucc}(n_i, p_u)$, $hrank_{ucc}(n_i)$, and $HPRV\_CC(n_i)$, and put all tasks into a priority queue according to a nonincreasing order of $HPRV\_CC(n_i)$;
2: **while** there are tasks to be scheduled in the priority queue **do**
3: Select the task n_r with the maximum $HPRV\_CC(n_r)$;
4: Compute the $HSV\_CC(n_r, p_u, r_z^{p_{src},p_u})$ of task n_r;
5: Assign task n_r to the processor p_u and route $r_z^{p_{src},p_u}$ that has the minimum $HSV\_CC(n_r, p_u, r_z^{p_{src},p_u})$ based on an insertion-based scheduling policy to synchronize selection for processor and route;
6: Mark n_r as a scheduled task;
7: **end while**

The time complexity of the HSV_CC algorithm is analyzed as follows. Traversing all tasks for scheduling all tasks can be done in $O(n)$ time. Computing the HSV_CC of each task can be done in $O(n \times p^3 \times l)$ time, where l represents the number of links, namely, traversing the immediate predecessor tasks for $O(p)$, traversing the routes for $O(p^2)$, and traversing the links for $O(l)$. Therefore, the complexity of algorithm HSV_CC is $O(n^2 \times p^3 \times l)$.

We also give the Gantt graph of the HSV_CC algorithm based on the benchmark, as shown in Fig. 4, where the left half is task to processor assignment and the right half is message to link assignment. The schedule length of the HSV_CC algorithm is 67, and the schedule length of the HSV algorithm is 73 (Fig. 2). Task priorities of the HSV and HSV_CC algorithms are not the same. HSV_CC reduces the scheduling length by 6 compared with the HSV algorithm, since the link l_3 has higher communication speed and transfers critical messages. This also indicates that the synchronized scheduling can obtain more accurate and efficient schedules than pure task scheduling to certain extent.

5. Experimental evaluation

5.1. Experimental metrics

The performance metrics chosen for comparison are schedule length ratio (SLR), speedup, and execution time. SLR is computed

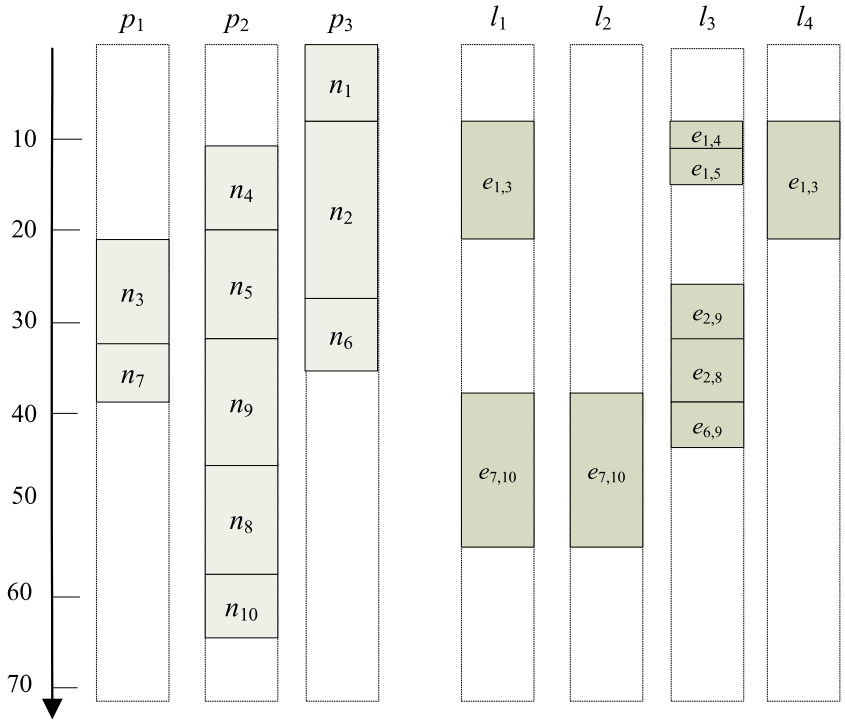

Fig. 4. Gantt chart of HSV_CC synchronized scheduling algorithm.

by dividing the real schedule length by the minimum execution time of all tasks in the critical path, as shown in Eq. (26):

$$SLR = \frac{makespan}{\sum_{n_i \in CP} \min_{p_u \in P}\{w_{i,u}\}}. \tag{26}$$

Speedup is computed by dividing the sequential execution time by the parallel execution time, as shown in Eq. (27):

$$Speedup = \frac{\min_{p_u \in P}\left\{\sum_{n_i \in N} w_{i,u}\right\}}{makespan}. \tag{27}$$

Meanwhile, we also consider the real computing environments, and choose WCRT as one of the metrics [36]. WCRT is the main performance evaluation of automotive electronic systems.

Random task graphs have been generated by TGFF (Task Graphs For Free) 3.5 [9], and programmed by Java to compare the results.

TGFF allows user to generate a variety of test DAGs with various characteristics depending on several input parameters, namely, number of tasks set $n = \{30, 40, 50, 60, 70, 80, 90, 100\}$, out degree set $\beta = \{1, 2, 3, 4, 5\}$, in-degree set $\gamma = \{1, 2, 3, 4, 5\}$, shape parameter set $\alpha = \{0.5, 1.0, 2.0\}$, range percentage of computation time set $\eta = \{0.1, 0.5, 1.0, 2.0, 4.0\}$, and communication to computation ratio (CCR) $CCR = \{0.1, 0.5, 1.0, 5.0, 10.0\}$.

η is the heterogeneity factor of actual processor computation power. A high percentage value of η causes significant differences among a task's computation times on processors, while a low percentage indicates that the expected execution time of a task is almost equal on any given processor in systems. Therefore, the larger the η, the larger differences in computation times on different processors, namely, the more obvious heterogeneity of computation. The computation time of task n_i on processor p_u is a random value in the following range:

$$w_i(1 - \eta/4) \le w_{i,u} \le w_i(1 + \eta/4). \tag{28}$$

CCR is basically the heterogeneity factor of communication for transmitting time. If $CCR < 1$, the computation is dominated in systems. However, if $CCR > 1$, and the larger the CCR, the larger proportion the communication time, namely, the more obvious heterogeneity of communication.

All algorithms are scheduled with fixed 3 processors, and all algorithms involved the HSV_CC algorithm use the network topology of Fig. 3.

Experimental results are shown with "box and whiskers graphs", which can show the concrete values with maximum point, minimum point, middle point (the median), and the middle points of the two halves (sub-medians) clearly.

5.2. Experiments on task scheduling

The algorithms for comparing with HSV are CPOP, HEFT, and CEFT, where the HEFT algorithm, the most quoted and famous algorithm, represents the characteristic of the "earliest finish time" type of algorithms. The CPOP algorithm, being from the same work with HEFT, represents the basic characteristic of the "critical path" approach. While CEFT, the latest proposed algorithm, is also suitable for comparison.

Experiment 1. We analyze the results of SLR and speedup for varying number of tasks respectively to verify the superiority of algorithms in scheduling performance. η is fixed with 1. CCR is also fixed with 1. The numbers of tasks are changed in the range of 30–100 with 10 increments. Values of 1000 experimental results are randomly selected from the sample space, as shown with box and whiskers graphs in Figs. 5 and 6.

It is observed that in all cases, the HSV algorithm has the best performance for both SLR and speedup as task number increases, and the best case reaches 25%. This is explained as follows. The processor selection criteria is HPRV, which considers the EFT from "downward" and LDET from "upward" comprehensively. Since all tasks on a critical path are assigned to one processor, CPOP has the worst performance. CEFT is also not good in performance, since

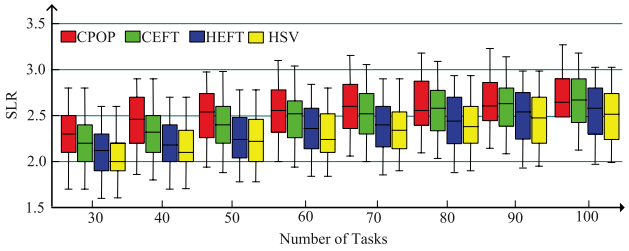

Fig. 5. SLR for varying number of tasks.

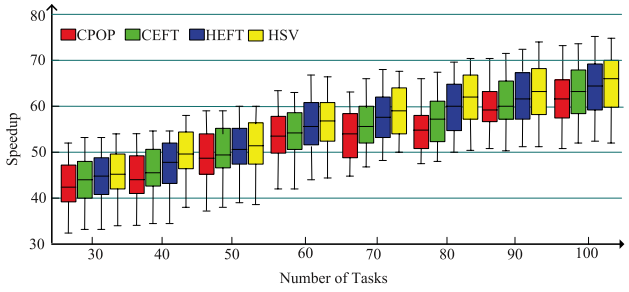

Fig. 6. Speedup for varying number of tasks.

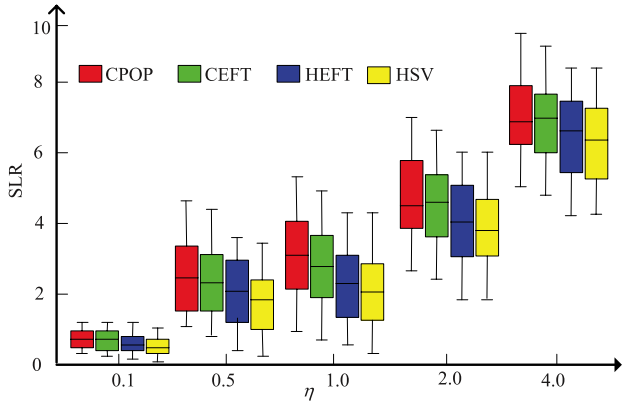

Fig. 7. SLR for varying η.

some tasks on constrained critical paths are also assigned to the same processors.

Experiment 2. We observe the results on SLR for varying η, which is a heterogeneity factor of computation, as shown in Fig. 7. The numbers of tasks is fixed with 50. *CCR* is fixed with 1. η is changed in the range of 0.1–4.0. Values of 1000 experimental results are also randomly selected from the sample space. The larger η is, the more heterogeneous of computation. As we expect, the HSV algorithm has better performance than other algorithms on SLR in all cases. Since the HSV algorithm optimizes the upward rank value for priority ordering phase, and considers the processor selection from both "upward" and "downward" comprehensively.

5.3. Experiments on synchronized scheduling

For the problem of different topologies with [28,34], we do not choose these algorithms for comparison. However, the HSV_CC algorithm is an improvement of the HSV algorithm by considering communication contention and synchronization. Therefore, the HSV algorithm is suitable for comparison with the HSV_CC algorithm. Since the HSV algorithm does not consider message scheduling, we assume that communication is concurrent, but with equal transmission speed.

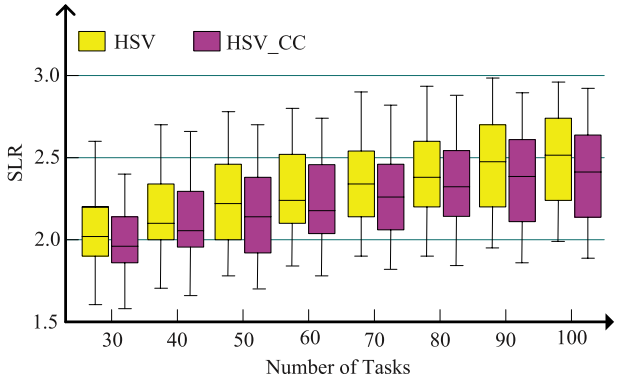

Fig. 8. SLR for different numbers of tasks.

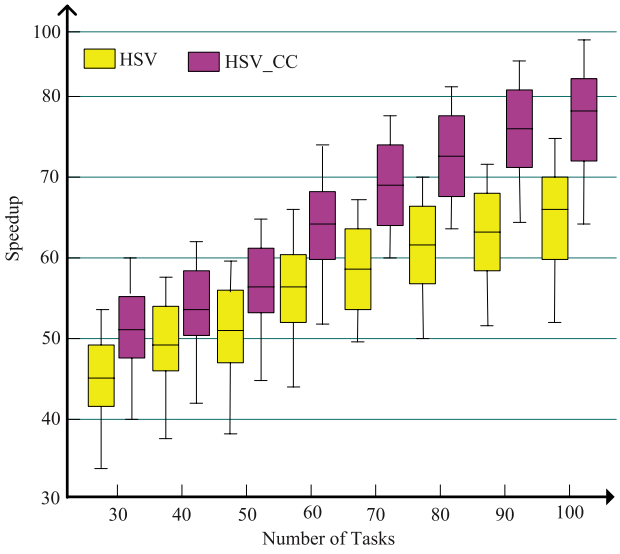

Fig. 9. Speedup for different numbers of tasks.

Experiment 3. We observe the results of SLR and speedup for varying number of tasks respectively to verify the superiority of the algorithms in scheduling performance and accuracy of scheduling results in communication contention environments. η is fixed with 1. *CCR* is also fixed with 1. The numbers of tasks are changed in the range of 30–100 with 10 increments. Values of 1000 experimental results are randomly selected from the sample space, as shown in Figs. 8 and 9.

It is observed that the HSV_CC algorithm has better performance than the HSV algorithm for both SLR and speedup as task number increases, and with the increasing number of tasks, the algorithm is more superior. This experiment verifies that the HSV_CC algorithm includes two important issues into consideration, namely, the heterogeneity of both computation and communication for task ordering, and synchronized scheduling with tasks and messages for communication contention.

Experiment 4. We observe the results the on SLR for varying CCR, which is a heterogeneity factor of communication. The numbers of tasks is fixed with 50. η is fixed with 1. *CCR* is changed in the range of 0.1–10.0. Values of 1000 experimental results are also randomly selected from the sample space, as shown in Fig. 10. The larger CCR is, the more heterogeneous of communication. As we expect, the HSV_CC algorithm has better performance than the HSV algorithm in all cases. The HSV_CC algorithm fully considers the heterogeneity of both computation and communication for task priority ordering phase, and the synchronization with tasks and messages for processor and route selection.

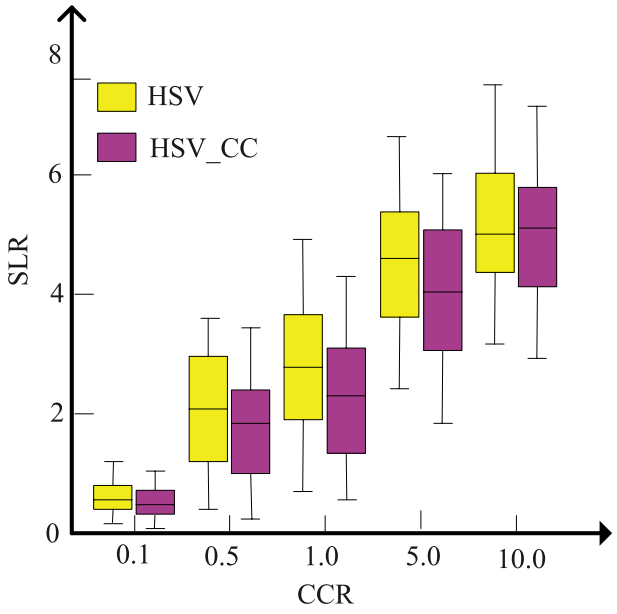

Fig. 10. SLR for varying CCR.

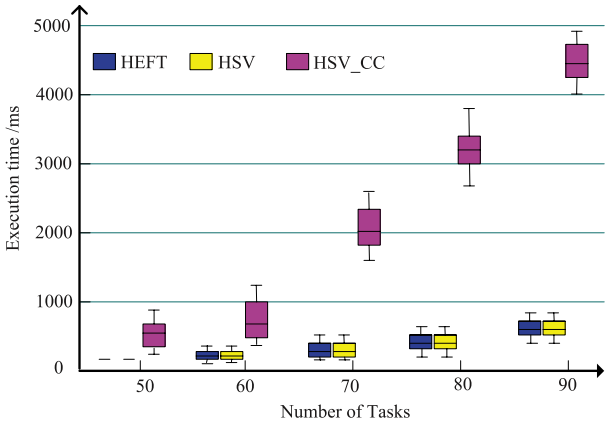

Fig. 11. Execution time for different numbers of tasks.

Experiment 5. We observe the scheduling times of the static algorithms themselves for varying number of tasks respectively. Notice that this experiment is to calculate the static scheduling times of algorithms themselves, not the actual running times. We also take the values of 1000 experimental results, as shown in Fig. 11. It is observed that there are no obvious gaps between the HSV algorithm and the HEFT algorithm on the execution times. The results mean that the task ordering criteria and the processor selection criteria of the HSV algorithm are not critical for runtime performance. However, this experiment shows that the HSV_CC algorithm needs more execution time compared with the HSV algorithm and the HEFT algorithm since it includes the synchronized scheduling with tasks and messages for communication contention.

5.4. Automobile electronic systems environments

Experiment 6. Automobile electronic systems are typical heterogeneous networked embedded systems, where computing and networking are both heterogeneous and deeply jointed. We have analyzed the WCRT for different numbers of messages in real automobile electronic environments by using the HSV and HSV_CC algorithms. Currently, CAN is the most widespread networking standard in automotive industries. CAN exhibits very predictable behavior, making it ideally suited for real-time distributed systems

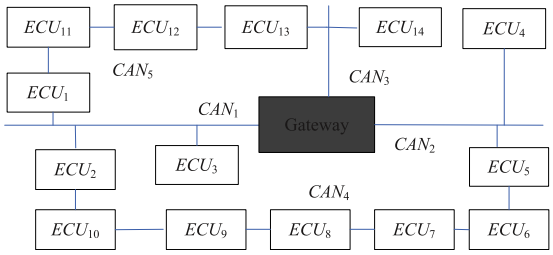

Fig. 12. Topology of HS-CAN/LS-CAN heterogeneous networks.

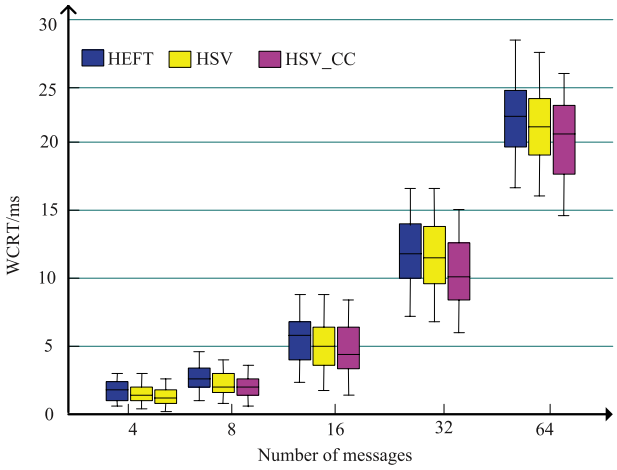

Fig. 13. WCRT for different numbers of messages.

due to its non-destructive and strictly deterministic medium arbitration. Therefore, we use CAN networks to do the experiment [20]. We employ the real message set based on HS-CAN/LS-CAN heterogeneous networks, which consist of two types of 500 Kbps for HS-CAN and 250 Kbps for LS-CAN respectively.

The message set contains total of 128 tasks (not including gateway tasks) and 64 messages, and is assigned in 14 ECUs. The architecture platform of our target system consists of 14 ECUs. All ECUs have different computation power, which is actually true in real systems. Fig. 12 shows the above described topology of HS-CAN/LS-CAN heterogeneous networks. CAN_1, CAN_2, and CAN_3 are configured with bus topologies, CAN_4 and CAN_5 are configured with ring topologies. The numbers of messages are changed in the range of 4–64. Accordingly, the numbers of tasks are changed in the range of 8–128. Values of 100 experimental results are also randomly selected from the sample space.

As shown in Fig. 13, the results show that the HSV_CC algorithm has shorter end-to-end WCRT and outperforms the HSV algorithm by 20%.

6. Conclusions

This paper proposes two novel and heterogeneity-driven DAG scheduling algorithms for heterogeneous networked embedded systems. For the problem that the task prioritizing and processor selection of existing efficient DAG task list scheduling algorithms can be improved, we employ HPRV for task ordering, and HSV for processor selection to propose the HSV algorithm based on the classic model for accurate and efficient schedules. For the problems that current DAG task and message scheduling algorithms do not fully consider the heterogeneity of communication, and synchronization of tasks and messages in communication contention environments, we propose the HSV_CC algorithm based on the communication contention model to get minimum end-to-end

WCRT on heterogeneous network embedded systems. The benchmark and experiments show that the HSV and HSV_CC algorithms can create accurate and efficient schedules compared with other algorithms. Gateways are only for message forwarding with cut-through communication scheme in this paper, we will consider the store-and-forward routing scheme, communication heterogeneity with different communication protocols, and communication congestion in our future work.

Acknowledgments

The authors thank the anonymous reviewers for their helpful remarks. This work was partially supported by the National High-Tech Research and Development Plan of China under Grant No. 2012AA01A301-01, the Key Program of National Natural Science Foundation of China under Grant No. 61432005, and the National Natural Science Foundation of China under Grant Nos. 61173036 and 61370095.

Appendix

Abbreviations

DAG = Directed Acyclic Graph
WCRT = Worst-Case Response Time
HSV = Heterogeneous Selection Value
HSV_CC = Heterogeneous Selection Value on Communication Contention
ECU = Electronic Control Unit
CAN = Controller Area Network
MOST = Media Oriented System Transport
ACC = Adaptive Cruise Control
EFT = Earliest Finish Time
NoC = Network-on-Chip
CSMA/CR = Carrier Sense Multiple Access/Collision Resolution
TDMA = Time Division Multiple Access
CSMA/CD = Carrier Sense Multiple Access with Collision Detection
HS-CAN/LS-CAN = High-Speed-CAN/Low-Speed-CAN
HEFT = Heterogeneous Earliest Finish Time
CPOP = Critical Path On a Processor
CEFT = Constrained Earliest Finish Time
APN = Arbitrary Processor Network
BTN = Binary Tree Network
HCNF = Heterogeneous Critical Node First
HCPFD = Heterogeneous Critical Parents with Fast Duplicator
HPRV = Heterogeneous Priority Rank Value
EST = Earliest Start Time
LDET = Longest Distance Exit Time
DFS = Depth-First Search
BFS = Breadth-First Search
LFT = Link Finish Time
LST = Link Start Time
SLR = Schedule Length Ratio
TGFF = Task Graphs For Free
CCR = Communication to Computation Ratio

References

[1] K. Albers, F. Bodmann, F. Slomka, Hierarchical event streams and event dependency graphs: A new computational model for embedded real-time systems, in: Real-Time Systems, 2006. 18th Euromicro Conference on, IEEE, 2006, 10pp.
[2] E. Azketa, J.J. Gutiérrez, J.C. Palencia, M.G. Harbour, L. Almeida, M. Marcos, Schedulability analysis of multi-packet messages in segmented can, in: Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on, IEEE, 2012, pp. 1–8.
[3] R. Bajaj, D.P. Agrawal, Improving scheduling of tasks in a heterogeneous environment, IEEE Trans. Parallel Distrib. Syst. 15 (2) (2004) 107–118.
[4] S. Bansal, P. Kumar, K. Singh, Dealing with heterogeneity through limited duplication for scheduling precedence constrained task graphs, J. Parallel Distrib. Comput. 65 (4) (2005) 479–491.
[5] S. Baskiyar, P.C. SaiRanga, Scheduling directed a-cyclic task graphs on heterogeneous network of workstations to minimize schedule length, in: Parallel Processing Workshops, 2003. Proceedings. 2003 International Conference on, IEEE, 2003, pp. 97–103.
[6] C. Buckl, A. Camek, G. Kainz, C. Simon, L. Mercep, H. Stahle, A. Knoll, The software car: Building ict architectures for future electric vehicles, in: Electric Vehicle Conference (IEVC), 2012 IEEE International, IEEE, 2012, pp. 1–8.
[7] P. Choudhury, P. Chakrabarti, R. Kumar, Online scheduling of dynamic task graphs with communication and contention for multiprocessors, IEEE Trans. Parallel Distrib. Syst. 23 (1) (2012) 126–133.
[8] D.E. Culler, R.M. Karp, D. Patterson, A. Sahay, E.E. Santos, K.E. Schauser, R. Subramonian, T. von Eicken, Logp: A practical model of parallel computation, Commun. ACM 39 (11) (1996) 78–85.
[9] R.P. Dick, D.L. Rhodes, W. Wolf, Tgff: task graphs for free, in: Proceedings of the 6th international workshop on Hardware/software codesign, IEEE Comput. Soc. (1998) 97–101.
[10] S. Fürst, Challenges in the design of automotive software, in: Proceedings of the Conference on Design, Automation and Test in Europe, European Design and Automation Association, 2010, pp. 256–258.
[11] A. Gerasoulis, T. Yang, A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors, J. Parallel Distrib. Comput. 16 (4) (1992) 276–291.
[12] T. Hagras, J. Janeček, A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems, Parallel Comput. 31 (7) (2005) 653–670.
[13] P. Heinrich, C. Prehofer, Network-wide energy optimization for adaptive embedded systems, ACM SIGBED Rev. 10 (1) (2013) 33–36.
[14] F. Ino, N. Fujimoto, K. Hagihara, Loggps: a parallel computational model for synchronization analysis, in: ACM SIGPLAN Notices, vol. 36, ACM, 2001, pp. 133–142.
[15] J.-P. Katoen, T. Noll, H. Wu, T. Santen, D. Seifert, Model-based energy optimization of automotive control systems, in: Proceedings of the Conference on Design, Automation and Test in Europe, EDA Consortium, 2013, pp. 761–766.
[16] M.A. Khan, Scheduling for heterogeneous systems using constrained critical paths, Parallel Comput. 38 (4) (2012) 175–193.
[17] S.C. Kim, S. Lee, J. Hahm, Push-pull: Deterministic search-based dag scheduling for heterogeneous cluster systems, IEEE Trans. Parallel Distrib. Syst. 18 (11) (2007) 1489–1502.
[18] H. Zeng, M. Di Natale, A. Ghosal, A. Sangiovanni-Vincentelli, Schedule optimization of time-triggered systems communicating over the flexray static segment, IEEE Trans. Indus. Inform. 7 (1) (2011) 1–17.
[19] H. Zeng, M. Di Natale, P. Giusto, A. Sangiovanni-Vincentelli, Stochastic analysis of can-based real-time automotive systems, IEEE Trans. Indus. Inform. 5 (4) (2009) 388–401.
[20] M. Sojka, P. Písa, O. Spinka, Z. Hanzálek, Measurement automation and result processing in timing analysis of a linux-based can-to-can gateway, in: Intelligent Data Acquisition and Advanced Computing Systems, IDAACS, 2011 IEEE 6th International Conference on, vol. 2, IEEE, 2011, pp. 963–968.
[21] E.G. Schmidt, K. Schmidt, Message scheduling for the flexray protocol: The dynamic segment, IEEE Trans. Vech. Technol. 58 (5) (2009) 2160–2169.
[22] K. Schmidt, E.G. Schmidt, A longest-path problem for evaluating the worst-case packet delay of switched ethernet, in: Industrial Embedded Systems (SIES), 2010 International Symposium on, IEEE, 2010, pp. 205–208.
[23] S.-H. Seo, J.-H. Kim, S.-H. Hwang, K.H. Kwon, J.W. Jeon, A reliable gateway for in-vehicle networks based on lin, can, and flexray, ACM Tran. Embed. Comput. Syst. 11 (1) (2012) 7.
[24] O. Sinnen, L.A. Sousa, Communication contention in task scheduling, IEEE Trans. Parallel Distrib. Syst. 16 (6) (2005) 503–515.
[25] O. Sinnen, L. Sousa, List scheduling: extension for contention awareness and evaluation of node priorities for heterogeneous cluster architectures, Parallel Comput. 30 (1) (2004) 81–101.
[26] O. Sinnen, L. Sousa, Experimental evaluation of task scheduling accuracy: Implications for the scheduling model, IEICE Trans. Inform. Syst. 86 (9) (2003) 1620–1627.
[27] O. Sinnen, L.A. Sousa, F.E. Sandnes, Toward a realistic task scheduling model, IEEE Trans. Parallel Distrib. Syst. 17 (3) (2006) 263–275.
[28] O. Sinnen, A. To, M. Kaur, Contention-aware scheduling with task duplication, J. Parallel Distrib. Comput. 71 (1) (2011) 77–86.
[29] J.A. Stankovic, Strategic directions in real-time and embedded systems, ACM Comput. Surv. 28 (4) (1996) 751–763.
[30] T. Steinbach, F. Korf, T.C. Schmidt, Comparing time-triggered ethernet with flexray: An evaluation of competing approaches to real-time for in-vehicle networks, in: Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on, IEEE, 2010, pp. 199–202.
[31] X. Tang, K. Li, G. Liao, K. Fang, F. Wu, A stochastic scheduling algorithm for precedence constrained tasks on grid, Future Gener. Comput. Syst. 27 (8) (2011) 1083–1091.
[32] X. Tang, K. Li, G. Liao, R. Li, List scheduling with duplication for heterogeneous computing systems, J. Parallel Distrib. Comput. 70 (4) (2010) 323–329.
[33] X. Tang, K. Li, R. Li, B. Veeravalli, Reliability-aware scheduling strategy for heterogeneous distributed computing systems, J. Parallel Distrib. Comput. 70 (9) (2010) 941–952.
[34] X. Tang, K. Li, D. Padua, Communication contention in apn list scheduling algorithm, Science China Ser. F Inform. Sci. 52 (1) (2009) 59–69.

[35] H. Topcuoglu, S. Hariri, M.-y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.
[36] X. Yong, Z. Gang, C. Yang, R. Kurachi, H. Takada, L. Renfa, Worst case response time analysis for messages in controller area network with gateway, IEICE Trans. Inform. Syst. 96 (7) (2013) 1467–1477.
[37] M. Zeller, C. Prehofer, G. Weiss, D. Eilers, R. Knorr, Towards self-adaptation in real-time, networked systems: Efficient solving of system constraints for automotive embedded systems, in: Self-Adaptive and Self-Organizing Systems (SASO), 2011 Fifth IEEE International Conference on, IEEE, 2011, pp. 79–88.
[38] Q. Zhu, H. Zeng, W. Zheng, M.D. Natale, A. Sangiovanni-Vincentelli, Optimization of task allocation and priority assignment in hard real-time distributed systems, ACM Trans. Embed. Comput. Syst. (TECS) 11 (4) (2012) 85.

Guoqi Xie received his Ph.D. degree in computer science from Hunan University, China, in 2014. He is an assistant professor of Hunan University, China, and a postdoctoral researcher of Nagoya University, Japan. His major research includes parallel and distributed computing, embedded and network computing, and software engineering and methodology.

Renfa Li is a full professor of computer science and electronic engineering, and the dean of College of Computer Science and Electronic Engineering, Hunan University, China. He is the director of the Key Laboratory for Embedded and Network Computing of Hunan Province, China. He is also an expert committee member of National Supercomputing Center in Changsha, China. His major research includes parallel and distributed computing, embedded and network computing, and cyber physical systems. He is a senior member of IEEE, and a senior member of ACM.

Keqin Li is a SUNY distinguished professor of computer science and an Intellectual Ventures endowed visiting chair professor at Tsinghua University, China. His research interests are mainly in design and analysis of algorithms, parallel and distributed computing, and computer networking. He has nearly 300 research publications. Currently, he is on the editorial board of IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, Journal of Parallel and Distributed Computing, International Journal of Parallel, Emergent and Distributed Systems, International Journal of High Performance Computing and Networking, and Optimization Letters. He is an IEEE Fellow.