A Tool System for Automatic Scheduling of Data Exchange in Real-Time Distributed Avionics Systems[1]

V.V. Balashov, V.A. Kostenko, R.L. Smeliansky **

** *Department of Computational Mathematics and Cybernetics, Moscow State University*

119992, 2nd educational building, MSU, Leninskie Gory, Moscow, Russia

Abstract

In this paper we address the problem of automatic scheduling of data exchange over a data bus with centralized arbitration. An example of such bus is the MIL STD-1553B multiplex data bus which is widely used in avionics systems. We present heuristic algorithms that perform scheduling with optimization of bus load and number of scheduled exchange jobs, and a tool system for automatic generation of data exchange schedules. An algorithm is also presented that suggests changes to requirements for data exchange over the bus. This algorithm is applicable if a schedule can not be constructed due to excessive requirements.

1. Introduction

Data buses with centralized arbitration (CA buses) are widely used in modern real-time distributed avionics systems. An example of such bus is the MIL STD-1553B multiplex data bus [1]. A number of terminal devices, including one bus controller, can be attached to the bus. Terminal devices can be sensors, computational nodes, data storage and display devices.

Static scheduling strategy is typically used to schedule data exchange over a CA bus. According to this strategy, scheduling is performed offline and the resulting schedule can not be changed in runtime. The schedule is executed by the bus controller. Due to high complexity of data flow over the bus (several hundreds of data words with different required frequencies of exchange for a typical MIL STD-1553B bus), manual construction of the schedule is infeasible. Intense data flow creates high bus loads, so an optimization process is required to put all data exchange jobs on schedule. In this paper we address the problem of automatic generation of a static schedule of data exchange over a CA bus. We present algorithms developed to solve this problem, and the tool system for automatic generation of data exchange schedules. If a data exchange schedule cannot be constructed due to excessive requirements for data exchange, the tool system suggests changes to the requirements in order to make schedule construction possible.

Section 2 provides a brief survey of related work in the area of interest. Section 3 introduces the main principles of the CA bus operation. Section 4 presents the structure of a data exchange schedule. In Section 5 the scheduling problem is decomposed into subproblems. Section 6 describes the algorithms developed to solve the problem. Section 7 presents the tool system for automatic generation of data exchange schedules, lists the key features and modules of the system.

2. Related work

There is a significant number of papers that mention applications of CA buses in avionics systems [6, 8]. For instance, [6] describes an X-ray all-sky monitor for use on the International Space Station (ISS), which is attached to

[1] This work is supported financially by the Russian Foundation for Basic Research under grant 04-01-00556, and performed under the contract to the International Science and Technology Center (ISTC), Moscow.

the MIL STD-1553B bus of the ISS. However, these papers provide little or no information on algorithms and on tool systems used to build the data exchange schedule, or on the structure of the schedule. Technical papers that present features of bus controller interface cards also give no such information.

A variety of approaches have been proposed for constructing a schedule of periodic computational tasks in avionics systems (for instance [7,9]); these approaches cannot be directly applied to schedule data exchange over a CA bus since they do not deal with specific constraints on the schedule, including constraints presented in Section 4. There are several approaches to changing properties (frequencies, durations) of tasks in order to make dynamic scheduling of tasks feasible [10, 11]. These approaches rely on analytical methods for schedulability assessment [5]. Such methods are not applicable in the area of static scheduling of data exchange if specific constraints on the schedule must be considered, since there are no known analytical methods for schedulability assessment that take into account realistic sets of specific constraints, e.g. those listed in Section 4.

3. Principles of CA bus operation

A number of terminal devices, including one bus controller, are typically connected to a CA bus. Each device can operate as a data source and/or a data receiver. The bus controller manages the data exchange and monitors the state of other terminal devices. Only the bus controller can initiate data exchange over the bus. Other terminal devices execute the commands issued by the controller. Data are exchanged accordingly to the command/response model (see, for example, [1]), asynchronously with execution of primary functions of the terminal devices.

Data is transferred over the bus in form of *messages* that consist of *words*. Each transfer of a message will be farther referred to as *data exchange job* (or *job*, for short). Messages can be transferred in two modes: either point-to-point or broadcast. In this paper we consider the following scheme of bus controller operation:

1. At system startup, a set of *job chains* is loaded to the bus controller. Job chain is a sequence of data exchange jobs that are to be executed one by one.
2. At the $k*L_{sc}$ point of time[2] (where L_{sc} is a given period), the controller starts execution of the k-th job chain, and executes it until the end. Execution of a job chain cannot be interrupted.
3. After completion of the execution of the job chain and until the $(k+1)*L_{sc}$ point of time, the controller can perform repeated execution of those jobs from the chain that did not succeed, e.g. due to external interference on the bus.

The time interval $[k*T; (k+1)* L_{sc})$ is referred to as *subcycle*, and L_{sc} is referred to as *subcycle length*. There may be no job chains in some subcycles.

There are other schemes of controller operation besides one described above. The algorithms proposed in this paper can be extended to other schemes, including aperiodic operation, scheme without job chains, etc.

4. Structure of a schedule of data exchange over a CA bus

Data exchange schedule is an ordered list of jobs with fixed start and finish times. Start and finish times of a job define the job's execution interval. Execution intervals of jobs in a schedule must not intersect. Data exchange schedule must be executed by the bus controller. According to the scheme of bus controller operation (see Section 3), jobs in a schedule are grouped into job chains, each of which corresponds to a specific subcycle.

Let us define *main cycle* of the schedule as the interval of time after which the schedule must be repeated exactly.

Let us consider a CA bus with a given set of terminal devices, and a given set of data words which must be transferred between these devices.

In this paper we consider the following sample set of inputs for scheduling of data exchange, farther referred to as *requirements for data exchange*:

1) for each word: source and destination devices, required frequency of exchange and, optionally, phase offsets;
2) length T_{MC} of the main cycle of the schedule;
3) technological constraints on the schedule, for instance:

[2] Here and farther we consider that time is measured in bus ticks.

a) maximum number of input and output messages[3] for each device;
b) maximum number of words in a message;
c) maximum allowed length of a job chain (i.e. number of jobs in the chain);
d) maximum allowed duration of a job chain (i.e. total execution time of jobs in the chain);
e) subcycle length;
f) fraction of the subcycle length, reserved in the end of the subcycle for repeated execution of unsuccessfully executed jobs.

For each word, a set of *deadline intervals* is defined as follows:

- if the word has no phase offsets: $[T*(i-1); T*i), i = 1,...,\lfloor T_{MC}/T \rfloor$, where T is the period of the word;
- if the word has offsets φ_1 and φ_2: $[\varphi_1 + T*(i-1); \varphi_2 + T*(i-1)), i = 1,...,\lfloor T_{MC}/T \rfloor$; both φ_1 and φ_2 must be non-negative, and the condition ($\varphi_1 < \varphi_2 \le T$) must hold.

Figure 1 shows a set of deadline intervals for a word with phase offsets (intervals are indicated by bold lines).

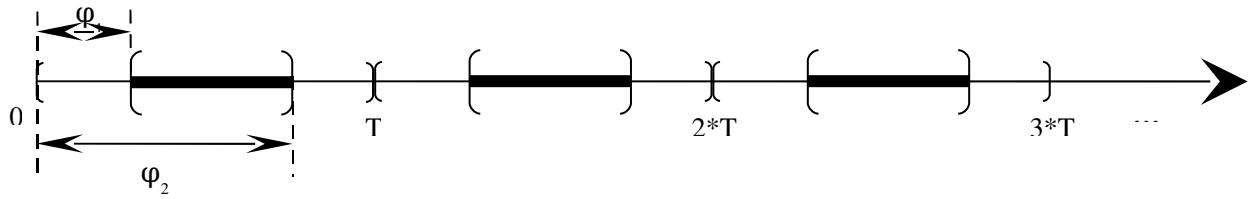

Figure 1. Deadline intervals for a word with phase offsets.

Each word must be transferred (as a part of some message) at least once within each of its deadline intervals. Figure 2 shows a fragment of a data exchange schedule containing two subcycles and two job chains.

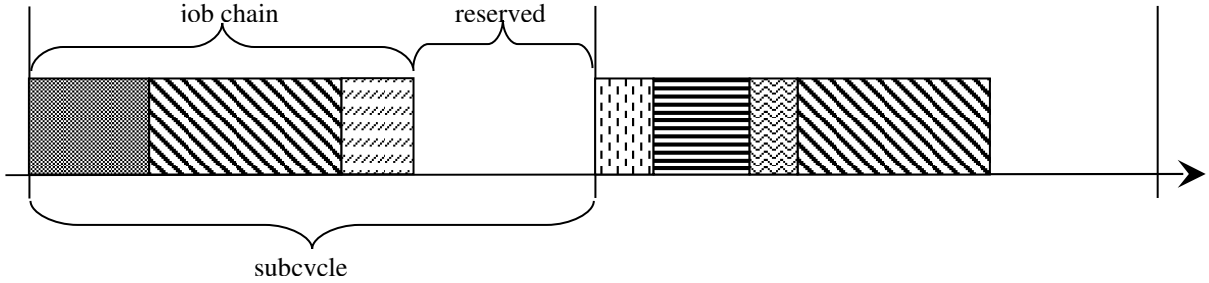

Figure 2. Fragment of a data exchange schedule.

5. The problem of scheduling of data exchange

Let us define *bus load for a given schedule* as T_{exch}/T_{MC}, where T_{exch} is the total duration of jobs within the schedule, and T_{MC} is the length of its main cycle.

The problem of scheduling of data exchange is stated as follows:

- *Input data*: requirements for data exchange over the bus, as given in Section 4;
- *Goal***:** to build a schedule of data exchange for the bus controller;
- *Criterion to be minimized*: bus load for the schedule;
- *Constraints*:
  - each word must be transferred at least once within each of its deadline intervals;
  - technological constraints must be met.

This problem can be decomposed into the following subproblems:

1. Packing data words into messages.

[3] Messages are considered different if they have different structure (are formed from different words).

Let us define *bus load for a given set of messages* as $\sum_{i=1}^{N}(c_i * f_i)$, where N is the number of messages, c_i is the duration of i-th message, and f_i is its required frequency of exchange.

This subproblem is stated as follows:

- *Input data*:
  - properties of data words (see item 1 of requirements for data exchange in the Section 4);
  - technological constraints:
    - maximum number of input and output messages for each device;
    - maximum number of words in message;
- *Goal***:** to allocate all given data words into messages;
- *Criterion to be minimized***:** bus load for the set of messages;
- *Constraints***:**
  - all words of a message must have same sender device;
  - technological constraints must be met.

It is allowed to pack words with different required frequencies into one message. In this case, the required frequency of exchange for the resulting message is a maximum of required frequencies for data words that form the message. This may lead to increase of the bus load in comparison to the case when several messages are formed, each containing words with equal frequencies.

In this paper we consider following technological constraints on messages [1]:

- message can contain up to 32 words;
- up to 30 input and 30 output messages can be defined for each terminal device, excluding the controller to which this constraint is not applicable.

The second constraint can be relaxed if a message is identified not only by the number of the input buffer (subaddress) it is received into, but also by an additional identifier packed into one of the data words. This allows receiving different messages into one input buffer.

2. Forming the set of data exchange jobs. For each message, an ordered set of data exchange jobs is constructed according to: (1) message's frequency; (2) length T_{MC} of the main cycle of the schedule.

For each job, a deadline interval is calculated, within which the execution of the job must be started and completed. Data for calculation of the deadline interval for a job are: (1) frequency and phase offsets of its message; (2) number of the job among jobs of the same message. Deadline intervals for jobs are calculated in the same way as deadline intervals for words, see Section 4.

3. Constructing the data exchange schedule. This subproblem is stated as follows:

- *Input data*:
  - set of jobs with deadline intervals;
  - technological constraints;
- *Goal***:** to schedule the given set of jobs;
- *Criterion to be maximized***:** number of scheduled jobs;
- *Constraints:*
  - each job must be executed within its deadline interval;
  - technological constraints must be met.

4. Generating suggestions for changing the requirements for data exchange, in case of *inconsistency* of the initial set of requirements. Requirements for data exchange are *inconsistent* if a given scheduling algorithm cannot construct a schedule that includes all data exchange jobs and meets all technological constraints. If an attempt to construct a schedule fails, the avionics system developer has to change values of some of the requirements (e.g. relax some technological constraints) in order to make the requirements consistent. If ranges of allowed variation and cost of variation are provided for some requirements, an automatic search for consistent values of these requirements can be performed.

The subproblem of generation of suggestions is stated as follows:

- *Input data*:
  - the algorithm used to schedule data exchange;
  - initial inconsistent values of requirements for data exchange;
  - ranges of allowed variation of requirements;
  - costs of variation of requirements (in form of cost multipliers);
- *Goal***:** to find a consistent set of requirements;
- *Criterion to be minimized***:** cost of variation of requirements from the initial inconsistent set;
- *Constraints***:** values of requirements must lie within given ranges.

By variation of a requirement we mean variation of its value; for instance, increasing the maximum length of a job chain. Examples of changeable requirements are: fraction of the subcycle length reserved for repeated exchanges; maximum length of a job chain. The cost of a specific change of a given requirement is calculated by multiplying the absolute difference between the initial value of the requirement and its changed value by the cost multiplier corresponding to the requirement. If it is difficult for the system's developer to choose appropriate costs of variation for the changeable requirements, equal costs can be assigned to all of them.

6. Essential algorithms used to solve the problem

In this section we present the general description of the essential algorithms proposed to solve the problem of scheduling of data exchange over a CA bus. Detailed description and performance analysis for these algorithms are given in the papers [3] (algorithm 6.1), [2] (algorithm 6.2), [4] (algorithm 6.3).

6.1 An algorithm for packing data words into messages

The algorithm proposed for packing data words into messages uses a greedy scheme for bus load minimization. The general scheme of the algorithm is as follows:

1. *Initialization*: pack each data word into a separate message and divide the resulting messages into groups. Messages with same pair <sender device, receiver device> are put into one group.
2. Merge messages with same frequency inside groups to form messages of maximum allowed length (full messages).
3. Merge remaining non-full messages inside groups. This step has two phases:
   i. Merge messages to minimize the bus load. The criterion for choosing message pairs for merge is *maximum decrease of bus load*. This phase ends when there is no message pair inside any group, after merging which the bus load decreases. Decrease of bus load on merging of messages M1 and M2 into message M12 is calculated as $(L(M1)+L(M2)-L(M12))$, where *L(M)* is the bus load by the message M.
   ii. Merge messages to eliminate violations of the constraint on maximum number of input and output messages for a terminal device. On this phase, bus load may be increased by some merges of messages with different frequencies, since the resulting message inherits the maximum frequency from the source messages. The criterion for choosing message pairs for merge is *minimum increase of bus load*; additionally, merges that reduce the violation of the constraint on *both* sender and receiver devices have higher priority.

The algorithm terminates when all of the following conditions are met:

i. steps 1) and 2) the are finished;
ii. constraint on maximum number of input and output messages is met for each terminal device;
iii. there are no possible merges that decrease the bus load.

The algorithm terminates after a finite number of merges, since the number of words is finite. Maximum complexity of the algorithm is $O(N_{words}^3)$, where N_{words} is the number of words. An extension of the algorithm was developed that supports generation of broadcast messages from words that have same sender and different receivers. Broadcast messages are generated before the beginning of step 3 of the basic algorithm. Only those broadcast messages are created that decrease the bus load. The remaining one-word messages are processed as in the basic algorithm.

6.2 An algorithm for constructing the data exchange schedule

A heuristic algorithm for constructing the data exchange schedule is proposed. The algorithm maximizes the number of jobs put on schedule without violation of deadlines or of technological constraints. The general scheme of the algorithm is as follows:

1. *Initialization*: order the jobs by ascending earliest possible finish time: $fin_i^* = s_i + t_i$, where s_i is the beginning of the deadline interval of the *i*-th job, and t_i is the duration of the job.
2. *Loop*:
   i. Choose the job *b* with minimum fin_i^*.
   ii. Determine the set *INT* of jobs, intersecting with *b* by deadline intervals and not yet put on schedule.
   iii. If $INT = \varnothing$, then put *b* on schedule.
   iv. If $INT \neq \varnothing$:
      a. If ($\forall d \in INT$ => *d* is incompatible[4] with *b*), then put *d* on schedule.
      b. If ($\exists d \in INT : d$ is compatible with *b*), then choose such compatible pair of jobs $(d1,d2) \in INT \times INT$ that has minimum total duration; put *d1* on schedule.
   v. Calculate the nearest time at which the next job can be scheduled.
   vi. Update the deadline intervals of jobs not yet put on schedule.

Technological constraints are taken in account on steps *v* and *vi* of the algorithm.

The algorithm terminates when there are no more jobs that can be put on schedule without violation of deadlines or technological constraints. An alternative version of step *iv* performs a limited enumeration of the job list to find the pair (*d1*, *d2*), considering no more than a given number of *N* jobs from the head of the job list. The algorithm can be tuned to the specifics of input data by choosing the depth of enumeration, or by applying a specific heuristic instead of limited enumeration. The algorithm terminates after a finite number of iterations, because the number of jobs is limited, and on every iteration one job is scheduled.

Maximum complexity of the algorithm is $O(N_{jobs} * \log_2(N_{jobs}) * C_{pair})$, where:

- N_{jobs} is the number of jobs;
- C_{pair} is the complexity of finding the (*d1, d2*) pair of jobs, see step *iv*.

6.3 An algorithm for generating suggestions for changing the requirements for data exchange

The algorithm proposed for generation of suggestions for changing the requirements for data exchange finds a consistent set of requirements and minimizes the cost of variation of requirements from the initial inconsistent set. The general scheme of the algorithm is as follows:

1. Take the initial inconsistent values of requirements as the current values.
2. Estimate consistency of the current values of requirements by scheduling the data exchange and checking the completeness of the schedule.
3. If the current values of requirements are consistent, STOP.
4. If all requirements are already changed to the limits, STOP.
5. Choose the requirement to be changed according to the heuristic "*minimum non-zero total cost of violation of the requirement by unscheduled jobs*" (considering only the requirements that are not yet changed to the limits).
6. Change (relax) the chosen requirement (e.g. increase the maximum allowed length of a job chain).
7. Go to step 2.

Termination of the algorithm on step 4 is unsuccessful, i.e. in this case the algorithm does not find a solution. This may take place due to following reasons:

[4] Two jobs are considered compatible if they can be put on schedule one after another.

1) absence of the solution in given ranges of allowed variation of requirements;
2) use of a heuristic that does not guarantee full traversal of the ranges of allowed variation (see step 5).

The algorithm terminates after a finite number of iterations, because:

- ranges of allowed variation of requirements for data exchange are finite;
- the number of requirements is finite;
- changes of requirements are monotonous (only relaxation of constraints is performed);
- a minimum step of change is given for each of the requirements.

Maximum complexity of the algorithm is: $N_{steps} * N_{reqs} * C_{sched}$, where:

- N_{steps} is the maximum number of steps after which a value of a requirement is changed to its limit;
- N_{reqs} is the number of changeable requirements;
- C_{sched} is the complexity of the scheduling algorithm.

7. Features and structure of the tool system for automatic scheduling of data exchange

The tool system for automatic scheduling of data exchange has following features:

1) automatic packing of words into messages with minimization of bus load;
2) automatic construction of a correct data exchange schedule that includes all data exchange jobs, and meets all technological constraints;
3) automatic generation of suggestions for changing the requirements for data exchange in case when the scheduler fails to construct a correct schedule;
4) support for manual correction of messages and schedules;
5) automatic generation of reports on the schedule and set of messages;

The tool system has a modular structure, which simplifies the adaptation of the system to concrete specifics of the target avionics system. Figure 3 shows the general structure of the tool system.

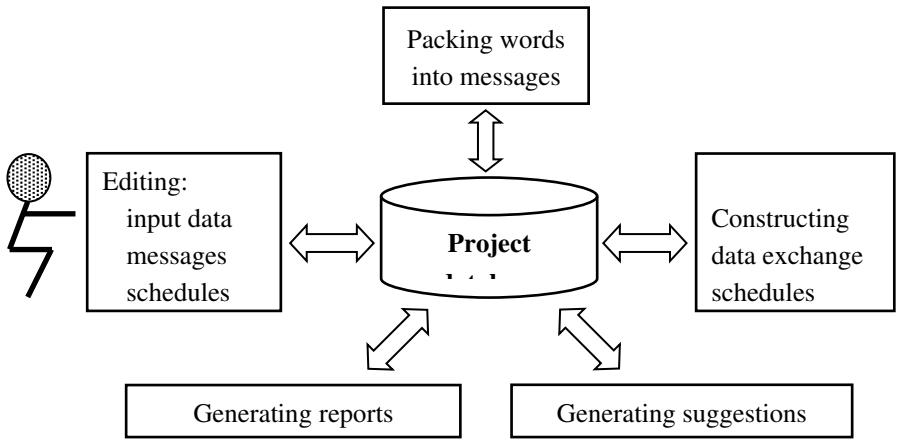

Figure 3. General structure of the tool system.

The project database is intended for storage of the following information:

1) input data;
2) sets of messages: message names and formats; sources and receivers; source and target subaddresses; words packed into the messages; required frequencies of exchange for the messages;
3) data exchange schedules: start and finish times for each scheduled job; sets of unscheduled jobs;
4) auxiliary data for generation of suggestions.

8. Conclusion

In this paper the problem of automatic scheduling of data exchange over a data bus with centralized arbitration was addressed. The algorithms developed to solve this problem were presented, along with a tool system for automatic generation of data exchange schedules. The tool system was successfully used for automatic generation of data exchange schedules for the MIL STD-1553B bus in real-time distributed avionics systems. Future work on the algorithms includes implementing support for new classes of technological constraints on the schedule.

References

[1] U.S. Standard MIL STD-1553B "Aircraft internal time division command/response multiplex data bus". U.S. Department of Defense, Washington D.C., 1978.

[2] V.A. Kostenko, E.S. Guryanov, "An algorithm for scheduling exchanges over a bus with centralized control and an analysis of its efficiency". *Programming and Computer Software*, Vol. 31, No. 6, 2005, pp. 340–346.

[3] S.V. Vavinov, "An algorithm for packing data words into messages for a multiplex bus with centralized control". In *Proc. of MCO-2005 Conference*, pp. 522-528, Moscow, 2005.

[4] V.V. Balashov, "Generating suggestions for changing the requirements for data exchange over a multiplex data bus". In *Proc. of PACO'2006 Conference*, pp. 422-437, Moscow, 2006.

[5] I.J. Bate, "Scheduling and timing analysis for safety-critical systems". Ph.D. thesis, Department of Computer Science, University of York, 1999.

[6] H. Negoro, M. Kohama, T. Mihara et al, "MAXI software system: photon event database". In *Proc. of Astronomical Data Analysis Software and Systems XIII Conference*, ASP Conference Series, Vol. 314, 2004.

[7] T. Sivanthi, U. Killat. "Static scheduling of periodic tasks in a decentralized real-time control system using an ILP". In *Proc. of IEEE International Workshop on Distributed, Parallel and Network Applications*, Fukuoka, Japan, 2005.

[8] G. Urban, H.-J. Kolinowitz, J. Peleska, "A Survivable Avionics System for Space Applications". In *Proc. of the Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing*, 1998.

[9] I. Bate and P. Emberson, "Design For Flexible And Scalable Avionics Systems". In *Proc. of IEEE Aerospace Conference*, 2005.

[10] Stewart D.B., Arora G. "A Tool for Analyzing and Fine Tuning the Real-Time Properties of an Embedded System". *IEEE Transactions on Software Engineering, Volume 29, Issue 4*, 2003.

[11] Park J., Ryu M., Hong S., Bello L. "Rapid performance re-engineering of distributed embedded systems via latency analysis and k-level diagonal search". *Journal of Parallel and Distributed Computing, Volume 66, Issue 1*, 2006.

This page has been purposedly left blank