6th IEEE International Conference on Intelligent Engineering Systems (INES),
May 2001, Opatija, Croatia

Achieving Dependability in Time-Triggered Networks by Sensor Fusion

Wilfried Elmenreich and Philipp Peti
Institut für Technische Informatik
Technische Universität Wien
Karlsplatz 13, Vienna, Austria
{wil,philipp}@vmars.tuwien.ac.at

Abstract — **This paper presents a system architecture that supports the development of dependable control applications by sensor fusion algorithms.**

Each measurement is represented as a compound of a name, a measurement instant, the measured value, and a confidence marker indicating the reliability and preciseness. Such sensor observations are processed by a network of fusion nodes in order to get data which is more dependable. The sensor fusion uses a probability model of sensor readings where the expected variance of a measurement corresponds directly to its confidence. Besides the fusion of different values the paper presents also approaches for fusing observations taken at different instants.

The presented approach has several advantages: First, it supports a modular system development, because the fusion algorithms are implemented only with respect to the interface specification and thus independent of the actual control application. This enables separate development and testing of subsystems that finally join together to form the whole application. Second, it supports the extension of existing applications in order to transform a real-time system that does not tolerate sensor faults into one that does. We use an analysis of the worst-case execution time of the resulting application. Given that the necessary timing constraints can be satisfied, the modified application will show the same temporal behavior as the original application.

1 Introduction

Computer systems that interact with its environment via sensors and actuators often require a certain degree of dependability and real-time capabilities (i. e. for closed control loops). Typically there are two starting-points to achieve these requirements: the system architecture level and the application level (cf. [1]). An example for a solution at architectural level is a time-triggered system [2] with replicated nodes. On the other hand, an existing application, that does, in its initial form, not fulfil given requirements, can be extended by extra hardware and additional communication and computation to achieve the necessary degree of dependability.

The first approach is usually more concise, but can be less efficient, especially if dependability is required selectively only in some cases or if a degraded level of service in case of failure is desired.

A solution at the application level can be more efficient, by integrating knowledge about the controlled process into the application program. However, this approach leads to increased design effort and application complexity. Thus, this approach promises lower hardware costs but higher software expenses.

Although hardware costs have dropped dramatically over the past decades, there are still fields of application where extra hardware can be critical because of constraints of weight or power consumption.

It is the objective of this paper to provide a system architecture that supports selective dependability by sensor fusion algorithms. These algorithms can be applied to existing systems without changing the communication timing of the prior services. The presented framework supports composable development by implementing small, controllable subsystems that can be tested separately and finally join together to form the final system (cf. [3]).

The remainder of the paper is organized as follows: The following Section 2 describes existing approaches found in the literature. Section 3 describes the architectural framework supporting operations with selective dependability. Section 4 examines some algorithms for the proposed operations. Section 5 sketches the integration of the proposed functions into the time-triggered fieldbus network TTP/A. The paper is concluded in Section 6

2 Related Work

A scheme for dependability assurance in digital systems is presented by Behrooz Parhami in [1]. The proposed approach attaches dependability tags to each data object and updates this tags according to operations performed on these data objects.

Another idea that contributed to the work in this paper is given by sensor validation for fieldbus nodes. So-called self-validating sensors are able to provide a standardized digital signal and generate diagnostic information. In the Oxford SEVA system [4] each measurement is delivered as a validated value together with the validated uncertainty and a measurement value status.

Finally if there are multiple measurements of a property in a technical process, the question arrives, how to further process these measurements until the data supports the required level of dependability. We assume that the taken measurements will have some degree of redundancy. Thus dependability can be achieved by performing voting or fusing algorithms.

When the replicated values are all identical, if correct, it is possible to implement fault-tolerant units by grouping identical units with a voter. The basic idea of such fault-tolerant units has already been proposed by John von Neumann [5]. A different approach [6] employs fail-silent units that produces either correct results or, in case of failure, no results at all.

Fusion of sensor information [7] is similar to voting by producing one output value from many related input data. However, fusion systems have the advantage, that the inputs to the fusion process are not required to by well-defined replicas and thus, the

values to be processed are not required to be identical or even close [8].

The sensor fusion problem addressed in this paper will be: given a set of n sensors with continuous output values, all with a limited accuracy and some of them delivering faulty messages, what is the smallest range of values where we can expect to find the correct value?

If the number of faulty sensors can be guaranteed to be at most t faulty sensors, the problem can be solved by fault-tolerant sensor averaging [9]. This approach uses a search algorithm to find all regions where $n-t$ sensor readings intersect.

3 Selective Dependability Framework

A sensor can be seen as a small window enabling us to get a view of a property of a technical process we are interested in. While a process is generally a continuous phenomenon in time and value, a sensor merely provides a part of the whole picture. Often the output of a sensor is reduced to the value it provides. In this paper we will take the following properties of a sensor measurement into account:

Value: The output value of the sensor. In general this value might be discrete or continuous, in our paper we will assume, that all values are given in a digital representation.

Context: What property of the process was measured? What units are used? Which sensor provided the measurement? The context of a measurement is often used implicity in the way the system process the value. Usually the sensor context is static.

Instant: *When* was the value observed? In a real-time system the instant of a measurement has similar importance than the value itself.

Reliability: Reliability values are created from self validating sensors or derived by comparing multiple measurements of the same property.

Impreciseness: Usually the impreciseness is a priori defined in the sensor's data sheet. However there are some scenarios, that afford dynamic impreciseness: Switch of metering ranges, sensor deprivation in a multi-sensor system, aging effects, etc. Note, that in real-time systems impreciseness can affect time as well as the value.

We will introduce the term *observation* for a compound of

$$<\text{entity name, instant } t\text{, value } x\text{, confidence } c>$$

The entity name defines the measurement context, instant and value map to the above described measurement value and measurement instant. Confidence is a value that expresses the estimated reliability and impreciseness of instant and value. The confidence measure will be introduced as a value between min_{conf} and max_{conf}, where min_{conf} is defined to be lowest confidence and max_{conf} is the highest confidence. The confidence value will be initially introduced by the sensor. A self-validating sensor performs self-diagnosis and can yield an appropriate confidence value. If a sensor has not the ability

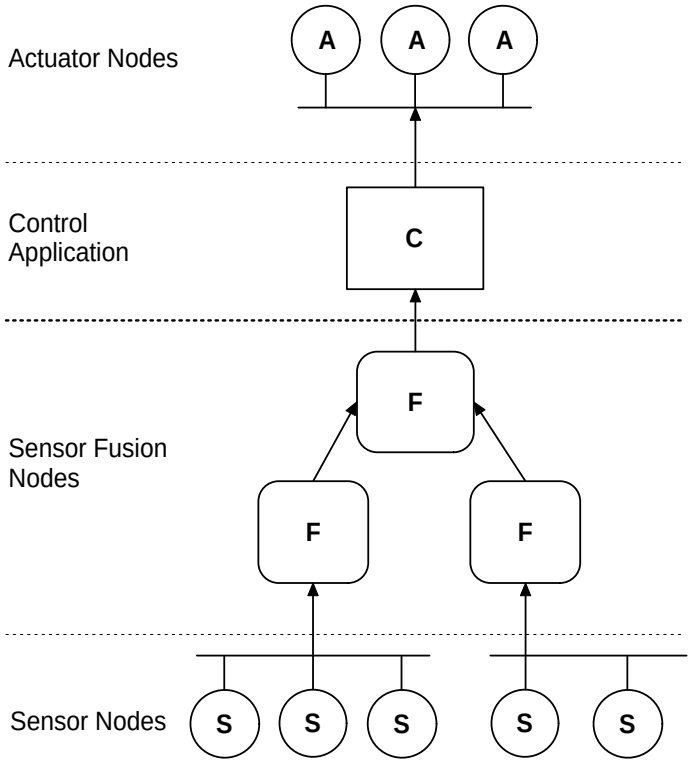

Fig. 1: Data Flow: Sensors ⇒ Fusion nodes ⇒ Control application ⇒ Actuators

to validate itself, the confidence value will be set to a standard value according to the a priori estimated sensor's average reliability.

The abstract network consists of nodes for sensors, actuators, and nodes containing fusion processing and control application. Several nodes of the abstract network can be clustered on one physical fieldbus node. The arrows in Fig. 1 depict the data flow in the network:

A fusion node (see Fig. 2) processes at least one observation as input and produces an observation as output. Besides the confidence value in the input observations, each input is assigned a *local confidence value*. Both confidence values are combined in a gate to form a new confidence for the observation. The input observations are then combined by sensor fusion algorithms. These algorithms can either produce an enhanced observation of the properties observed by the single sensors or may also produce a derived property, e. g. an acceleration value from speed measurements or the slippage by comparing rotation speed measurements. A fusion node produces an output observation that may differ in value, instant, and confidence from the input observation. The output observation will always have assigned a new name.

Fusion nodes can be cascaded as depicted in Fig. 1. Since the underlying transducer network produces digital data, it is possible to use the same sensor observation multiple in different fusion nodes. Usually a fusion node reduces data and rises dependability – thus, the confidence of the resulting observation will be higher than the confidences of the input observations. However in some cases the confidence might even be decreased, in case of several contradicting inputs.

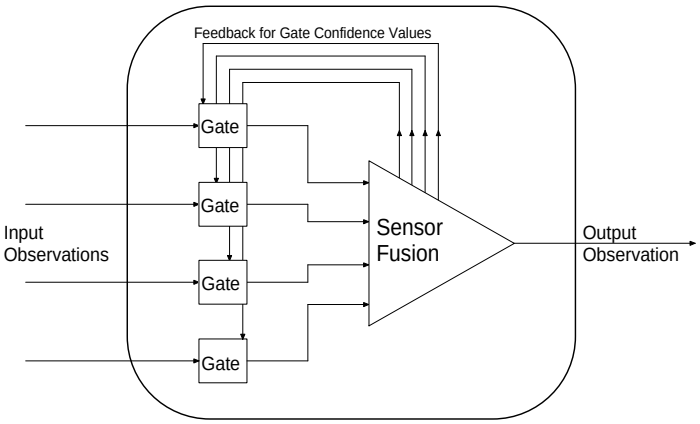

Fig. 2: Fusion Node

The control application uses at least one observation as input. It depends on the implementation of the control application if it uses the assigned confidence value when making a control decision.

4 Fusion Algorithms

Fusing Measurement Values

In comparison to the approach of searching intersecting regions (Marzullo [9] and Jayasimha [10]) we will choose a probabilistic sensor representation that encompasses the sensor's failure modes and then fuse measurements with respect to this model.

A sensor output will be given in the format of an observation. The entity name is the sensor's identification, the *value* will be produced by the measurement standardized to the predetermined units. The *instant* is determined by the measurement instant standardized to the predetermined time units. It is important to use a reference time that is common knowledge to all other nodes in order to make observation instants comparable. The confidence marker will be a transformed value of an estimator of the sensor value's variance $Var[S]$. We use the variance, because it is independent of the probability density function. From the probability density function and the variance, the probability of the measurement value being inside a given interval around the real value can be derived.

We assume the incoming observations to be taken from a continuous entity. The measurement values are fused by using a weighted average with the reciprocal variance values as weights.

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i \cdot \frac{1}{Var[S_i]}}{\sum_{i=1}^{n} \frac{1}{Var[S_i]}} \tag{1}$$

The variance values are derived from the confidence values (see Fig. 3).

A possible extension would be a fault tolerate averaging algorithm. This algorithm will remove the k largest and the k smallest data values and then perform the weighted

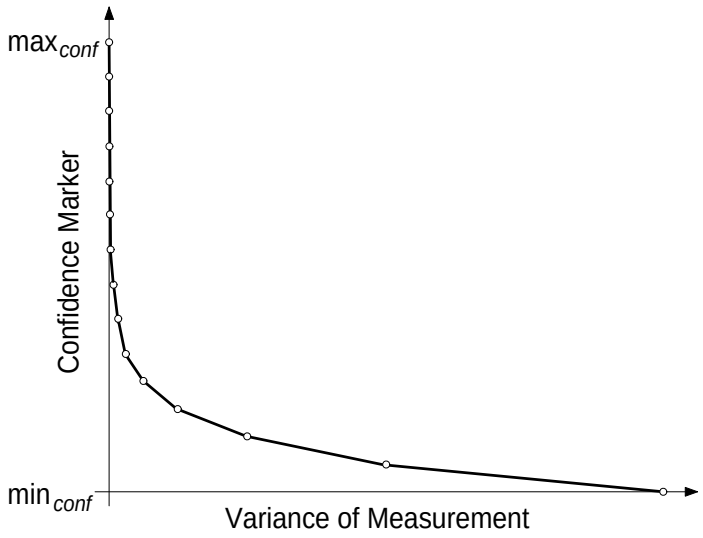

Fig. 3: Conversion function for confidence/variance values (based on an exponential function)

average as described above. Thus, at least k faulty measurements can be tolerated.

However the fault-tolerant averaging has two disadvantages in our application context:

- There is a tradeoff between the number of faults to be tolerated and the performance of the sensor fusion algorithm since our sensor fusion algorithm performs the better the more inputs are given.
- Finding the most distorting values among weighted values is sumptuous – first the average value $\bar{x}$ has to be calculated, then the "largest" value can be derived by finding a maximum respective minimum of $(\bar{x} \cdot \frac{1}{Var[S_i]} - x_i \cdot \frac{1}{Var[S_i]})$ for all i input values.

Some applications (e. g. classification) afford to select one observation out of a set of input observations. In this case voting algorithms [11] will be used to choose the appropriate output. When *exact* voting is requested, the confidence values of identical values are summarized in order to form a new confidence value. Then the value with the highest confidence is selected (see Fig. 4).

Obtaining Output Confidence

As we did by the fusion of the measurement values we will use statistical methods to generate the confidence value of the output observation. Using equation 1 to obtain a fused value, the following equation will yield the variance of the respective output observation:

$$Var[S_O] = \frac{1}{\sum_{i=1}^{n} \frac{1}{Var[S_i]}} \tag{2}$$

The confidence marker of the output observation is then approximated according to Fig. 3.

Fusing Observations of Different Instants

Most control algorithms need periodical updates of control values where the update times must have low jitter [12]. However if two or more observations of the same property taken at different instants have to be fused, there are different ways to determine the observation instant of the a fused value:

Drop old values: Keep only the observations with the most recent timestamp and discard the other. The fusion algorithm is then performed with value and confidence marker as described above. This method is applicable to time-triggered systems using a sparse time base, because in such systems measurements can by made exact synchronously. However if observations can take place at any instant of a fine-grained time line it is likely that all but one observations are discarded.

Extrapolate old values: If a history of values and instants is available for each input observation, old input observations can be transformed to refer to another instant by extrapolation. Thus, all observations with an instant prior to the instant of the most recent observation will be extrapolated to the newest instant. However such an algorithm could be critical if the output observation is used in a feedback loop.

Average instant: The instant of the output observation is given the average value over all input instants according to their weight:

$$\bar{t} = (\sum_{i=1}^{n} t_i \cdot \frac{1}{Var[S_i]}) / (\sum_{i=1}^{n} \frac{1}{Var[S_i]}) \tag{3}$$

5 Application in a Fieldbus Network

A fieldbus network connects transducers (sensors and actuators) to a control system. While in former approaches transducers where point-to-point connected and instrumented by analog signals, actual fieldbus systems use a digital communication medium to interconnect the transducers with the control system. We decided to insert the sensor fusion processing between the transducer nodes and the control system. Since

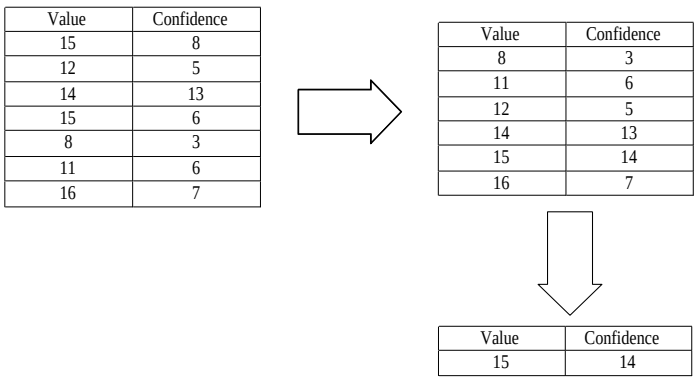

Fig. 4: Exact Voting with Confidence Values

future fieldbus nodes are equipped with a local microcontroller for signal conditioning, it would be possible to implement the local sensor signal processing directly at the sensor. Conceptually, the sensor fusion will be strictly separated from the control application in order to avoid increasing complexity for the control application [13].

As a case study we used the time-triggered master-slave fieldbus protocol TTP/A.

TTP/A is designed for predictable real-time communication in non-critical applications in the automotive and automation sector. The protocol [14] uses a time division multiple access (TDMA) bus arbitration scheme, which mets timing requirements for typical sensor fusion algorithms [12].

It is possible to address up to 254 nodes on a bus. One single node is the active master. This master provides the time base for a synchronized global time among all slave nodes. The communication is organized into rounds. A round consists of several slots. A slot is a unit for transmission of one byte of data. Data bytes are transmitted in a standard UART format. Each communication round is started by the master with a so-called fireworks byte. The fireworks byte defines the type of round.

A TTP/A round (see Fig. 5) consists of a configuration dependent number of slots and an assigned sender node for each slot. The configuration of a round is defined in the RODL (ROund Descriptor List). The RODL defines which node transmits in a certain slot, the semantics of each individual slot, and the receiving nodes of a slot. RODLs must be configured in the slave nodes prior to the execution of the corresponding round.

The TTP/A protocol [14] offers a unique addressing scheme for all relevant data of a node like communication schedules, calibration data, and I/O properties. This addressing scheme is called Interface File System (IFS) [15]. The IFS provides a universal interface to the TTP/A network for configuration and maintenance tools as well as for applications running local on a node. The IFS is structured in a record-oriented format. The smallest addressable unit is a record of 4 bytes. All nodes contain several files with a number of records that can contain information for automatic configuration.

TTP/A supports data types for 8 bit and 12 bit digitized analogue data. It is possible to assign a measurement a four-bit confidence marker referring to the quality of the sensor observation. These confidence markers will be used to represent the dependability of an observation as described in Section 3 Due to the fact, that the whole communication and computation is time-triggered with respect to a global time, the observation instant of each observation is known a priori to all nodes and may not be transmitted. The observation context is defined by its address in the IFS.

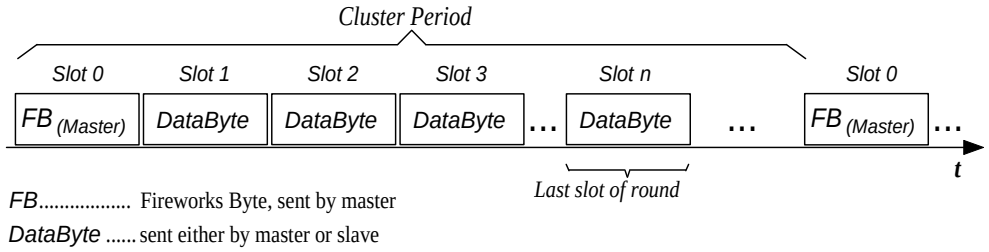

Fig. 5: TTP/A Communication

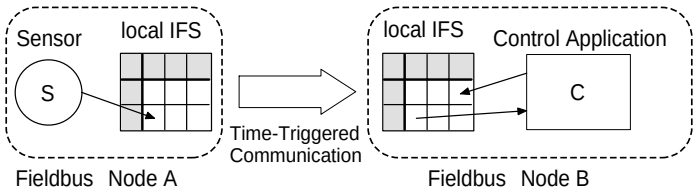

Fig. 6: Data Processing in TTP/A

The IFS acts as a temporal firewall [16] between communicating entities. Fig. 6 depicts a sample communication in TTP/A. Node *A* contains a sensor and a local IFS, node *B* contains the control application. The sensor performs a measurement at instant t_{m1}. Later at time $t_{t1} > t_{m1}$ the protocol transports the data from the local IFS of node *A* to the local IFS of node *B*. The data arrives at time t_{a1} at node *B*. The control application can now use the sensor data until it is overwritten with a new value at time t_{a2}. Usually the control application has to hold a deadline t_{c1} to output a control signal. The output of the control application is also written to the IFS and further transported at time t_{c1} by the time-triggered protocol to the actuators of the system. Fig. 8 shows the instants of communication in a timing diagram. The establishment of the temporal firewall is, that the task of the control application may be scheduled any time between t_{a1} and $min(t_{a2}, t_{c1})$ as long as the result is written to the IFS until the end of this duration.

We will now use the knowledge about the timing together with the IFS concept to introduce sensor fusion into an existing TTP/A application. In our implementation, the fusion algorithms will be implemented on the same fieldbus node as the control application.

Fig. 7 depicts an extension of the above described application with a fusion node. The fusion node is a virtual node implemented together with the control application in fieldbus node B. The incoming data is preprocessed by a fusion node until it is feeded into the control application.

Taking the timing constraints from the above example we now arrive at the constraint in equation 4:

$$WCET_{fusion} + WCET_{control} \leq min(t_{a2}, t_{c1}) - t_{a1} \tag{4}$$

where $WCET_{fusion}$ is the worst case execution time of the fusion process and $WCET_{control}$

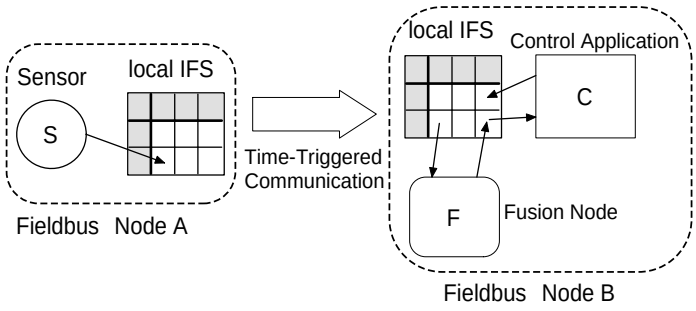

Fig. 7: Data processing with fusion processor inserted

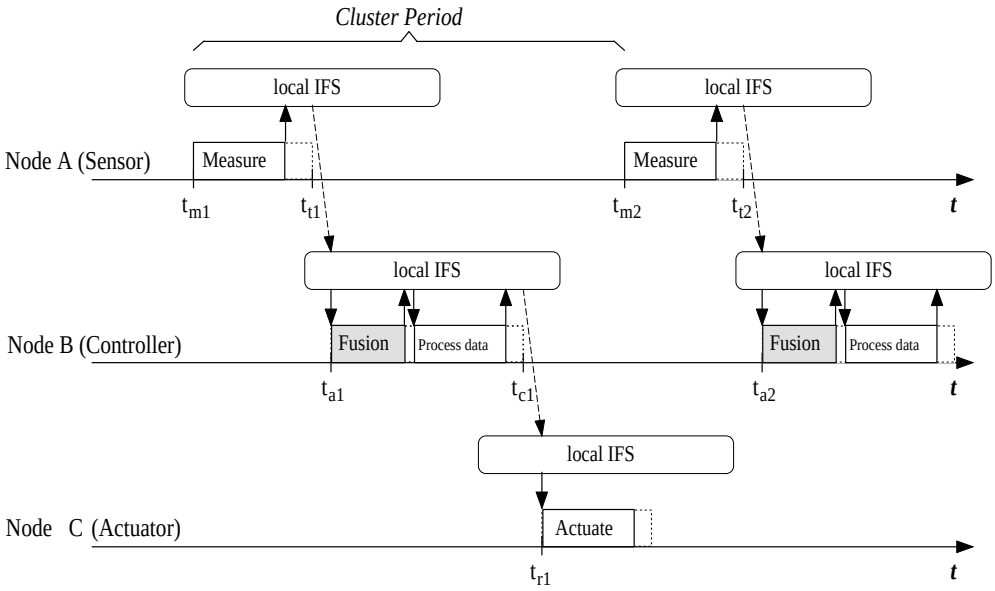

Fig. 8: Communication and computation instants among three nodes

is the worst case execution time of the control application.

To obtain the worst case execution time we plan to use a tool for static WCET analysis like CALC_WCET_167 [17]. The tool uses a graph-based approach [18] to derive a safe upper bound for the execution of a code piece written in WCETC. The language is an extension to C, that needs some annotations for loop bounds, etc.

Given that the necessary timing constraints can be satisfied, the sensor fusion process is transparent to the control application process. The main advantages of this architecture are:

No changing of the network configuration: Existing TTP/A networks can be used to host sensor fusion algorithms. The system behavior will thus be improved without adding extra nodes. This eases the installation of sensor fusion methods significantly since no adaption of existing configuration data (RODL files) is needed.

Reuse of existing control applications: The presented approach can be used to transform an application that does not tolerate sensor faults into one that does. If the WCET analysis guarantees the timing constraint (Eq. 4) the modified application will show the same temporal behavior as the original application.

Fusion algorithm is independent of control application: The strict separation of the fusion process and the control application allows a modular system development. Thus small, controllable subsystems can be implemented and tested separately and finally joined together to form the final system.

Use of generic algorithms: The fusion algorithm is independent of the application, thus generic fusion algorithms can be used for a variety of applications.

There is even the possibility to enhance an existing networks without extensions to hardware and under reuse of existing software. However this property cannot be guaranteed generally. If a system is already so compact, that there is no redundancy in sensor measurements and no space on the fieldbus controller to host the sensor fusion task, extensions are necessary anyway. However many applications provide some redundancy or free resources. For that cases, our approach allows a smooth integration of the additional functionality.

6 Conclusion

We presented an architecture for processing sensor measurements with respect to their dependability. Each measurement is represented as a compound of a name, a measurement instant, the measured value, and a confidence marker indicating the reliability and preciseness. Sensor observations are processed by a network of fusion nodes in order to get data which higher confidence. The confidence of each measurement is attached to the transmitted data in form of the confidence marker. The sensor fusion uses a probability model of sensor readings where the expected variance of a measurement corresponds directly to its confidence. Besides the fusion of different values the paper presents also approaches for fusion observations of different instants.

We examined methods for integration of the presented sensor fusion algorithm into existing application in order to transform a real-time application that does not tolerate sensor faults into one that does. An important point is the stability of the timing behavior after the integration which is supported by a time-triggered communication system featuring a temporal firewall at each communication action. An analysis of the worst-case execution time of the resulting application is used to verify the timing constraints of the original application. When the necessary timing constraints can be satisfied, the modified application will show the same temporal behavior as the original application.

Since our approach allows the joining of fusion nodes with existing tasks into existing hardware nodes, resources in existing TTP/A networks can be used to host systems with improved behavior.

Acknowledgments

We would like to give special thanks to our colleague Wolfgang Haidinger who acted as mathematical advisor when we were at our wits' end. This work was supported in part by the Austrian Ministry of Science, project TTSB and by the European IST project DSoS under contract No IST-1999-11585.

References

[1] B. Parhami. *A Data-Driven Dependability Assurance Scheme with Applications to Data and Design Diversity*, volume 4. Springer Verlag, Vienna, 1991.

[2] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[3] A. Krüger. *Interface Design for Time-Triggered Real-Time System Architectures*. PhD thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, April 1997.

[4] M. Henry. Sensor validation and fieldbus. *Computing & Control Engineering Journal*, 6(6):263–269, Dec. 1995.

[5] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 43–98. Princeton University Press, 1956.

[6] H. Kopetz, H. Kantz, G. Grünsteidl, P. Puschner, and J. Reisinger. Tolerating transient faults in MARS. *20th. Symposium on Fault Tolerant Computing, Newcastle upon Tyne, UK*, June 1990.

[7] E. Waltz and J. Llinas. *Multisensor Data Fusion*. Artech House, Norwood, Massachusetts, 1990.

[8] D. E. Bakken, Z. Zhan, C. C. Jones, and D. A. Karr. Middleware support for voting and data fusion. In *Proeedings of the International Conference on Dependable Systems and Networks*, pages 453–462, 2001.

[9] K. Marzullo. Tolerating failures of continuous-valued sensors. *ACM Transactions on Computer Systems*, 8(4):284–304, Nov. 1990.

[10] D. N. Jayasimha. Fault tolerance in a multisensor environment. *Proceedings of the 13th Symposium on Reliable Distributed Systems*, pages 2–11, 2001.

[11] B. Parhami. Voting networks. *IEEE Transactions on Reliability*, 40:380–394, August 1991.

[12] W. Elmenreich and S. Pitzek. Using sensor fusion in a time-triggered network. In *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society, Denver, Colorado*, volume 1, pages 369–374, Nov.-Dec. 2001.

[13] W. Elmenreich and S. Pitzek. The time-triggered sensor fusion model. *Proceedings of the 5th IEEE International Conference on Intelligent Engineering Systems, Helsinki, Finland*, pages 297–300, Sept. 2001.

[14] H. Kopetz et al. Specification of the TTP/A protocol. Technical report, Technische Universität Wien, Institut für Technische Informatik, March 2000. Available at http://www.ttpforum.org.

[15] H. Kopetz, M. Holzmann, and W. Elmenreich. A universal smart transducer interface: TTP/A. *International Journal of Computer System Science & Engineering*, 16(2), March 2001.

[16] H. Kopetz and R. Nossal. Temporal firewalls in large distributed real-time systems. *Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97)*, 1997.

[17] R. Kirner. *WCET Analysis Framework based on WCETC*. Technische Universität Wien, Institut für Technische Informatik. Available at http://www.wcet.at.

[18] P. Puschner and A. Schedl. Computing maximum task execution times – a graph-based approach. *Journal of Real-Time Systems*, 13(1), July 1997.