Formalising Nondeterministic Communication in Wireless Sensor Networks Using CSP

Sondre Ninive Andersen* , Asbjørn Engmark Espe† , Sverre Hendseth‡ , and Geir Mathisen§
Department of Engineering Cybernetics,
Norwegian University of Science and Technology, Trondheim, Norway
Email: {*sondre.n.andersen, †asbjorn.e.espe, ‡sverre.hendseth, §geir.mathisen}@ntnu.no

***Abstract*—Using communicating sequential processes (CSP), this paper presents a model for wireless sensor networks (WSNs) to be used for formal verification of communication reliability in mesh networks. Process models are derived for sensor nodes and communication links, introducing nondeterminism in order to capture the unreliability inherent in wireless communication. It is shown that a guarantee may be issued concerning the CSP model's worst-case performance in terms of packet corruption. This guarantee is substantiated by transformation of the model, employing a series of operations introduced to simplify the network while preserving worst-case performance. The end result is a formal proof of the entire network's worst-case reliability. As long as the nondeterminism of the communication links is modelled with care, the packet corruption rate through the network will be equal to or better than the worst-case performance of its most deterministic path.**

***Index Terms*—Communicating sequential processes, wireless sensor network, reliability, network communication.**

I. Introduction

Wireless sensor networks (WSNs) have become a ubiquitous part of our modern society. A wide range of infrastructure, machinery, processes, as well as natural structures are already monitored by, or will in the future be instrumented with such networks as part of maintenance scheduling or safety measures. In many cases, WSNs operate using a mesh topology, meaning that the nodes in the network also function as intermediate relays and help propagate data through the network—often in an ad hoc fashion [1]. Since distributed systems can be prone to nonintuitive failures, formal verification can be a useful approach to ensure robustness and reliability [2].

As a formal language, communicating sequential processes (CSP) was first introduced by Hoare [3] to model and describe the interaction of concurrent processes. The language was further developed by Roscoe [4], and has evolved to become a well-specified language for formal verification. In the literature, CSP has been applied in similar efforts to guarantee behaviours or performance goals. Jaskó and Simon [5] used the language to prove deadlock-free operation in a sensor network architecture, while Sakellariou *et al.* [6] incorporated CSP into programming languages for constraint programming, and employed it as a tool for distributed constraint satisfaction problems. Using timed CSP, Liu *et al.* [7] introduced the concept of Wireless Sensor Processes (WSP) to model contention-based wireless sensor networking systems. In later years, Steyn and Gruner [8] proposed an extension to the CSP language in the form of a new parallelism operator to facilitate the modelling of WSNs.

II. Model

A general mesh WSN may be modelled as a network of nodes connected by unreliable communication links. By formulating an algebraic model that fully accommodates the nondeterministic nature of the physical network, the implications of these unreliable communication links may be formally examined.

The following analysis considers a single packet's traversal from a source node, through a homogeneous network, to a sink node. An apt routing protocol running on top of the mesh WSN allows the construction of a directed acyclic graph (DAG) as an image of the network for the traversal of this single packet. The nodes in the graph correspond to physical nodes relaying the packet through the WSN, and the edges to physical links over which the packet was transmitted. Importantly, it is assumed that the routing protocol handles any conflict between packets, allowing the real-world case of simultaneous packet traversal to be considered as a superposition of single-packet cases.

By employing CSP to model this graph, certain guarantees can be issued concerning its worst-case reliability. The CSP formulation in the following splits the model into processes describing the behaviour of each node and each communication link in the graph. Pursuant to conventional naming for buffers [3], input channels are named $left$, and output channels $right$. The $COPY$-process introduced in [3] is also extensively used. As defined in (1), the process copies any input on its $left$-channel to its $right$-channel.

$$COPY = left?i \rightarrow right!i \rightarrow COPY \tag{1}$$

A. Node

Upon receiving a packet on one of its input links, a node in the graph will forward the packet on all of its output links. Since only the reliability of the traversal graph is considered, each node is permitted infinite memory capacity, thereby avoiding any synchronisation or deadlock issues. A node may then be modelled as

$$N(n,m) = N_{\mathrm{I}}(n) \gg N_{\mathrm{O}}(m) \tag{2a}$$
$$N_{\mathrm{I}}(n) = left?i : \mathbb{N}_n?p \rightarrow right!p \rightarrow N_{\mathrm{I}}(n) \tag{2b}$$
$$N_{\mathrm{O}}(m) = \Big|\Big|_{\{|left|\}}^{m}{}_{i=1} B^{\infty}[\![right.i/right]\!] \,, \tag{2c}$$

where n and m are the number of inputs and outputs, respectively, and B^{∞} is an empty, unbounded buffer as in [4].

Nodes may be classified based on the number of inputs and outputs they have, A single-input, single-output (SISO) node is a special case of (2),

$$\begin{aligned} N(1,1) &= N_{\text{SISO}}[\![^{left.1,\, right.1}/_{left,\, right}]\!] \\ &= N_{\text{I}}(1) \gg B^{\infty}[\![^{right.1}/_{right}]\!] \\ &= B^{\infty}[\![^{left.1,\, right.1}/_{left,\, right}]\!] \,, \end{aligned} \tag{3}$$

equivalent to a single unbounded buffer. To ease renaming later on, the naming $N_{\text{SISO}} = B^{\infty}$ is also introduced.

B. Link

Uncertain communication links are modelled as an unreliable variant of $COPY$, which can nondeterministically elect to corrupt c of every K packets delivered. This captures the behaviour of a real-world link with a probability of corruption equal to c/K. The confidence of the modelling can be brought to an arbitrarily high level by choosing a sufficiently large K. In a real-world network there are several reasons for which a transmission might fail, such as total loss of signal, single- or multi-bit faults, or other higher-level errors in the routing protocol. However, all failed transmissions are captured by this nondeterministic chance of corruption, assumed to be detectable but not correctable. The resulting model is shown in (4).

$$\begin{aligned} L^{i}_{j/c} &= left?p \to \Big(right!p \to L^{i+1}_{j/c} \\ &\qquad \sqcap\, right!\tilde{p} \to L^{i+1}_{j+1/c} \Big) && (i < K, j < c) && \text{(4a)} \\ L^{i}_{c/c} &= left?p \to right!p \to L^{i+1}_{c/c} && (i < K) && \text{(4b)} \\ L^{K}_{j/c} &= left?p \to \Big(right!p \to L^{1}_{0/c} \\ &\qquad \sqcap\, right!\tilde{p} \to L^{1}_{0/c} \Big) && (j < c) && \text{(4c)} \\ L^{K}_{c/c} &= left?p \to right!p \to L^{1}_{0/c} && && \text{(4d)} \end{aligned}$$

The initial state, $L^{1}_{0/c}$, represents a link that may either transmit each received packet p correctly, or—if $c > 0$—transmit a corrupted version, $\tilde{p}$. In the following, K is assumed to be a global horizon, i.e. defined for the whole graph. For readability, the shorthand $L(c) = L^{1}_{0/c}$ is introduced.

Do note that the relations

$$L(0) = COPY \tag{5}$$

$$L(c) \sqsupseteq L(c+1) \tag{6}$$

$$L(c_1) \gg L(c_2 + 1) = L(c_1 + 1) \gg L(c_2) \tag{7}$$

hold. (5) and (6) should be quite clear from the link definition. The argument for (7) is that both sides of the equation have equal traces and failures and are therefore equivalent. If values are momentarily disregarded, it is apparent that both chains behave like buffers of capacity 2, satisfying the specification

$$t \downarrow right \leq t \downarrow left \leq t \downarrow right + 2 \,, \tag{8}$$

for all traces t.

Both chains will always accept any packet p on their $left$ channels. For the $right$ channels, the failures of both the left hand side and right hand side are equal. Both chains can refuse to output both an uncorrupted packet and a corrupted packet until $c_1 + c_2 + 1$ packets have been corrupted by the chain. After this, both sides may only deliver uncorrupted packets until a total of K packets have been delivered, at which point the links reset to their initial state.

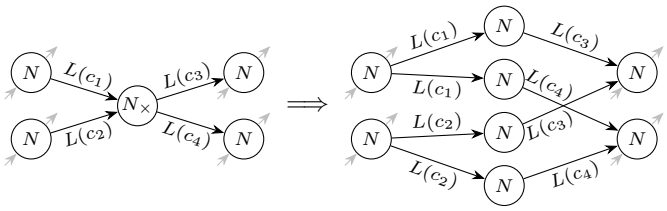

Fig. 1: The first operation, illustrated for $n = m = 2$, allows nodes to be split in order to form a set of disjoint paths form source to sink. The grey arrows indicate potential connections which are not relevant to the operation.

III. Method

Presented in the following is a method for reducing an arbitrary DAG of nodes and links into a trivial graph with a single link, exhibiting the same worst-case performance. The graph is assumed to be structured as described in the previous—i.e. that it represents a single packet's traversal through the physical network from a source node to a sink node. The method is composed of three basic operations which will be presented in turn.

A. Operation 1: Node splitting

Illustrated in Figure 1, the first operation consists of splitting a node with n inputs and m outputs into $n \times m$ nodes, one for each distinct path through the junction. The operation only considers the relevant subgraph, consisting of a set of n nodes with links feeding into a single junction node $N_{\times}$, from which links branch out again to a second set of m nodes. The input nodes are disjoint from the output nodes, as any nodes in both sets would cause a two-node cycle through $N_{\times}$, contradicting the assumption that the graph is acyclic.

Since all nodes are on the form $N_{\text{I}}(n) \gg N_{\text{O}}(m)$, it is simpler to only consider the latter part for the input nodes, and the former part for the output nodes. N_{O} consists of a set of unbounded buffers synchronising on the $left$ event, permitting all outputs not relevant to $N_{\times}$ to be discarded. The same holds for N_{I}, for which all inputs are independent. The initial subgraph G_1 may then be expressed as (9).

$$\begin{aligned} G_1 = \quad & \Big(\Big|\Big|\Big|_{i=1}^{n} N_{\text{O}}(1)[\![^{right}/_{right.1}]\!] \\ & \qquad \gg L(c_i)[\![^{right.i}/_{right}]\!] \Big) \\ & \gg N_{\times} \\ & \gg \Big(\Big|\Big|\Big|_{j=1}^{m} L(c'_j)[\![^{left.j}/_{left}]\!] \\ & \qquad \gg N_{\text{I}}(1)[\![^{left}/_{left.1}]\!] \Big) \end{aligned} \tag{9}$$

From the definition of refinement, $P \sqsubseteq Q \Leftrightarrow P = P \sqcap Q$, the two relations (10) and (11) must hold. Also note that

the worst-case performance in terms of packet corruption stays invariant for both expressions.

$$L(c) \gg \left(\Big|\Big|^{k}_{i=1} B^{\infty}[\![^{right.i}/_{right}]\!] \right)_{\{|left|\}}$$
$$\sqsupseteq \Big|\Big|^{k}_{i=1} L(c) \gg B^{\infty}[\![^{right.i}/_{right}]\!] \quad (10)$$
$$\{|left|\}$$

$$\begin{aligned}
&(\mu\, q.left?i : \mathbb{N}_k!j?p \to right!j!p \to q) \\
&\gg L(c)[\![^{left.j,right.j}/_{left,right}]\!] \\
&\sqsupseteq \left(\Big|\Big|\Big|^{k}_{i=1} L(c)[\![^{left.i.j,right.i.j}/_{left,right}]\!] \right) \\
&\gg (\mu\, q.left?i : \mathbb{N}_k!j?p \to right!j!p \to q) \quad (11)
\end{aligned}$$

The junction node $N_\times$ may be expanded as a set of $n \times m$ parallel unbounded buffers, and renaming may be introduced as shown in (12).

$$\begin{aligned}
N_\times &= N_{\mathrm{I}}(n) \gg N_{\mathrm{O}}(m) \\
&= \Big|\Big|\Big|^{n}_{i=1} \left(N_{\mathrm{I}}(1)[\![^{left.i}/_{left.1}]\!] \gg N_{\mathrm{O}}(m) \right) \\
&= \Big|\Big|^{n}_{i=1} \Big|\Big|^{m}_{j=1} B^{\infty}[\![^{right.j,left.i}/_{right,left}]\!] \\
&\quad \{|left|\}\,\{|left|\} \\
&= \Big|\Big|^{m}_{j=1} \left(\left(\Big|\Big|\Big|^{n}_{i=1} N_{\mathrm{SISO}}[\![^{right.i.j,left.j.i}/_{right,left}]\!] \right) \right. \\
&\quad \{|left|\} \\
&\quad \gg (\mu\, q.left?i : N_n!j?p \to right!j!p \to q) \\
&\quad \left. \right) [\![^{left}/_{left.j}]\!] \quad (12)
\end{aligned}$$

Subsequently, the links are extracted and (12) inserted into (9). The two refinement expressions (10) and (11) are then employed to obtain the following subgraph, in which all paths are disjoint.

$$\begin{aligned}
G_1 \sqsupseteq\ & \Big|\Big|\Big|^{n}_{i=1} \left(N_{\mathrm{O}}(m)[\![^{right.i}/_{right}]\!] \right) \gg \\
& \Big|\Big|\Big|^{n}_{i=1} \Big|\Big|\Big|^{m}_{j=1} \left(L(c_i)[\![^{left.i.j}/_{left}]\!] \right. \\
& \gg N_{\mathrm{SISO}} \gg L(c'_j)[\![^{right.j.i}/_{right}]\!] \Big) \\
& \gg \Big|\Big|\Big|^{m}_{j=1} \left(N_{\mathrm{I}}(n)[\![^{left.j}/_{left}]\!] \right) \quad (13)
\end{aligned}$$

It should be noted that (13) is refined by (9) and not necessarily an equivalent process. However, its worst-case behaviour is still retained, which is the sole invariant needed to proceed.

B. Operation 2: Serial merging

The second operation merges serial nodes and links, as illustrated in Figure 2. The relevant, initial subgraph can be formulated as

$$G_2 = B^{\infty} \gg L(c_1) \gg N_{\mathrm{SISO}} \gg L(c_2) \gg \cdots \gg L(c_n)\,, \quad (14)$$

where the leading unbounded buffer is the output buffer of the branch node.

Fig. 2: The second operation, illustrated for $n = 2$, merges sequential links and nodes into a single link. The grey arrows indicate potential connections which are not relevant to the operation.

The chain of links and nodes is to be reduced into a single link whose weight is some function of the weights of the individual links, as shown in (15).

$$G_2 \stackrel{?}{=} B^{\infty} \gg L(\hat{c})\ , \quad \hat{c} = f(c_1, \ldots, c_n) \quad (15)$$

To this end, three step laws are introduced, in which all nodes $N_{\mathrm{SISO}} = B^{\infty}$ as per (3):

$$L(c) \gg N_{\mathrm{SISO}} = N_{\mathrm{SISO}} \gg L(c) \quad (16)$$
$$N_{\mathrm{SISO}} \gg N_{\mathrm{SISO}} = N_{\mathrm{SISO}} \quad (17)$$
$$N_{\mathrm{SISO}} \gg L(c_1) \gg L(c_2) = N_{\mathrm{SISO}} \gg L(c_1 + c_2) \quad (18)$$

The first step law, (16), is self-explanatory; moving the buffering before or after a nondeterministic link does not alter the chain's behaviour. The second law, (17), simply states that an unbounded buffer chained with an unbounded buffer is an unbounded buffer, as per the buffer laws in [4]. The third law, (18), is a consequence of (7) combined with the buffer laws in [4]. First, (7) is used repeatedly until the first link in the chain becomes $L(0) = COPY$, at which point the buffer laws are employed to merge this link with the unbounded buffer.

Utilising (16)–(18), (14) may be expressed on the form (15). Initially, (16) is repeatedly employed to group all links and all nodes.

$$\begin{aligned}
G_2 = B^{\infty} &\gg N_{\mathrm{SISO}} \gg \cdots \gg N_{\mathrm{SISO}} \\
&\gg L(c_1) \gg \cdots \gg L(c_n) \quad (19)
\end{aligned}$$

At this stage, the nodes are merged by (17), and subsequently the links by (18). This results in

$$G_2 = B^{\infty} \gg L(\hat{c}) \quad (20)$$
$$\hat{c} = c_1 + \cdots + c_n\ , \quad (21)$$

where the initial buffer is—as in (14)—the output buffer of the branching node.

C. Operation 3: Parallel merging

The third operation takes a graph composed of two terminal nodes connected by n parallel links, and reduces it by merging all the parallel links into a single link, as illustrated in Figure 3. It is assumed that the terminal nodes have no other connections.

The initial graph is defined as

$$\begin{aligned}
&N(1,n) \gg \\
&\Big|\Big|\Big|^{n}_{i=1} L(c_i)[\![^{left.i,right.i}/_{left,right}]\!] \\
&\gg N(n,1)\ , \quad (22)
\end{aligned}$$

and the goal in this stage is to reduce this parallel connection of links into a single link whose weight is some function of the weights of the individual links,

$$L(\hat{c})\ , \quad \hat{c} = f(c_1, \ldots, c_n)\ . \quad (23)$$

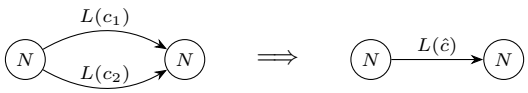

Fig. 3: The third operation, illustrated for $n = 2$, merges one or more parallel links into a single link.

It is essential to also consider that, as a result of the parallel links, the sink node will receive n possibly corrupted duplicates of each packet sent from the source node. The important metric is the graph's worst-case behaviour, i.e. the maximum number of packets that may be corrupted between two nodes. Based on the modelling assumption that corruption of packets can be detected, a filtering process $F(n)$ is introduced and defined in (24), which forwards an uncorrupted packet among the n duplicates if there is one, or otherwise an arbitrary packet. The predicate $\checkmark(p)$ represents error detection, and is true if and only if p is not corrupt.

$$F(n) = left?p \rightarrow F_p^n(n-1) \tag{24a}$$

$$F_p^n(i) = left?q \rightarrow \big(F_q^n(i-1) \not\in \checkmark(q) \not\ni F_p^n(i-1) \big) \quad (0 < i < n) \tag{24b}$$

$$F_p^n(0) = right!p \rightarrow F(n) \tag{24c}$$

Appending the filtering process to the graph leads to (25).

$$\begin{aligned} G_3 = N(1,n) \gg & \Big|\Big|\Big|_{i=1}^{n} L(c_i)[\![^{left.i,\, right.i}/_{left,\, right}]\!] \\ & \gg N(n,1) \gg F(n)[\![^{left.1}/_{left}]\!] \end{aligned} \tag{25}$$

This graph will to the best of its ability output a single uncorrupted packet for each inputted packet. The introduction of the filtering process means that the parallel connection of links may never perform worse than the worst-case performance of its best link. In other words, all link coefficients c_i may be assigned the value $\hat{c} = \min(c_1, \ldots, c_n)$ without altering the graph's behaviour. Indeed, since this filtering process essentially "hides" all worse-performing links, all parallel links may be replaced with a single link $L(\hat{c})$. It follows that the filtering process $F(1) = COPY$ becomes redundant resulting in the reduced graph (26).

$$G_3 = N_{\text{SISO}} \gg L(\hat{c}) \gg N_{\text{SISO}} \tag{26}$$

$$\hat{c} = \min(c_1, \ldots, c_n) \ , \tag{27}$$

D. Transforming the DAG

Using these three operations, the graph representation of the single packet's traversal can now be transformed without affecting the worst-case rate of corruption. First, the graph is stepped through through from the source node, in a breadth-first order, and Operation 1 is applied to each node. The result is a new graph in which every path is divorced from every other, with the worst-case reliability unaffected. Next, each path is merged into a single link using Operation 2. This results in an equivalent graph consisting only of the source node and the sink node, connected by a number of links. Finally, the third operation is applied, resulting in the final graph in which a single link connects the source node and the sink node.

The functions (21) and (27) can be used to derive the coefficient C of the single remaining link in the transformed graph as equal to the smallest sum of coefficients along the paths through the original DAG. This leads to a guarantee that, if the number of packets sent a satisfies $C < a \leq K$, then at least one packet will reach its destination uncorrupted.

IV. Conclusions

Introduced in this paper is a formal model to be employed in analysis of the reliability of communication in mesh WSNs. Initially, a DAG was constructed to capture how a single packet propagates through the network from a source node to a sink node. Based on this graph description, CSP was employed to create a model of the nondeterministic behaviour of nodes and communication links.

It was shown that this DAG could be reduced to a single link connecting the endpoints, and formally proven that this transformation does not alter the worst-case behaviour of the graph. The result is that the reliability of the single link in the graph reduction represents the worst-case performance of the original network, and it was therefore possible to define a lower bound on the number of packets that must be sent in order to guarantee that at least one packet arrives uncorrupted.

It is important to note that the model is only an accurate representation of the physical network to a certain degree; the validity of the guarantees are limited by the confidence level of the modelling. The model parameters must therefore be chosen with care to sufficiently reflect the real-world nondeterminism. As part of future work, this model and CSP in general has the potential to be employed for other formal verification purposes, such as proving deadlock-free operation.

References

[1] I. F. Akyìldìz, W. Su, Y. Sankarasubramaniam, and E. Çayırcı, "Wireless sensor networks: A survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[2] D. He, L. Cui, H. Huang, and M. Ma, "Design and Verification of Enhanced Secure Localization Scheme in Wireless Sensor Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 7, pp. 1050–1058, Jul. 2009.

[3] C. A. R. Hoare, *Communicating Sequential Processes*. Prentice Hall, 1985.

[4] A. W. Roscoe, *The Theory and Practice of Concurrency*. Prentice Hall, 2005.

[5] S. Jaskó and G. Simon, "CSP-Based Sensor Network Architecture for Reconfigurable Measurement Systems," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 6, pp. 2104–2117, Mar. 2011.

[6] I. Sakellariou, I. Vlahavas, I. Futo, Z. Pasztor, and J. Szeredi, "Communicating sequential processes for distributed constraint satisfaction," *Information Sciences*, vol. 176, no. 5, pp. 490–521, Mar. 2006.

[7] S. Liu, X. Wu, Q. Li, H. Zhu, and Q. Wang, "Formal Approaches to Wireless Sensor Networks," in *2011 Fifth International Conference on Secure Software Integration and Reliability Improvement*, Jeju, South Korea, 2011, pp. 11–18.

[8] T. J. Steyn and S. Gruner, "A New Optional Parallelism Operator in CSP for Wireless Sensor Networks," in *Proceedings of the South African Institute of Computer Scientists and Information Technologists*, Thaba 'Nchu, South Africa, 2017, pp. 1–8.