Analysing Bang & Olufsen's BeoLink® Audio/Video System Using Coloured Petri Nets

Søren Christensen & Jens Bæk Jørgensen

Computer Science Department, University of Aarhus
Ny Munkegade, Bldg. 540, DK–8000 Aarhus C, Denmark
E-mail: {schristensen, jbj}@daimi.aau.dk

Abstract. Bang & Olufsen A/S (B&O) is a renowned manufacturer of audio and video products. Their BeoLink® (BeoLink) system distributes sound and vision throughout a home via a network. In this way, e.g., while doing the cooking in the kitchen, a person can remotely select and listen to a track from a CD loaded in the CD player situated in the living room. To resolve conflicts, synchronisation between various actions is needed, and is indeed taken care of by appropriate communication protocols.
The purpose of the project described in this paper was to test Coloured Petri Nets (CP-nets or CPN) as a way to improve B&O's methods for specification, validation, and verification of protocols. In the main experiment, an engineer from B&O used the Design/CPN tool to build a CPN model of vital parts of BeoLink, to validate its behaviour using simulations with a familiar graphical feedback, and to formally verify crucial properties using occurrence graphs (also known as state spaces and reachability graphs/trees). The latter activity demonstrated the applicability of occurrence graphs for timed CP-nets. Moreover, CPN was used to examine important aspects of a possible future revision of BeoLink, and to check compatibility between the new and the old version.
Based on the experiments reported in this paper, CPN has been included in the set of methods for specification, validation, and verification of future protocols at B&O.

Topics. System design and verification using nets; higher-level net models; computer tools for nets; experience with using nets, case studies; application of nets to protocols and embedded systems.

1 Introduction

The Danish company Bang & Olufsen A/S (B&O) has a long tradition for producing sophisticated audio and video products. B&O employs 230 developers. 50 of these produce software full-time. Part of the software development deals with communication protocols. This paper describes a project aimed at improving B&O's ways of specifying, validating, and verifying communication protocols. The project was carried out in cooperation between B&O and the CPN group, University of Aarhus, with participation of the authors of this paper.

A relatively new invention of B&O is the *BeoLink* concept. A home equipped with a BeoLink system has a central room, typically the living room, where audio/video sources such as a radio, a CD player, a cassette recorder, a TV, and a video recorder are located. The idea is that from the other rooms such as the kitchen or the children's room, the audio/video sources of the central room can be controlled and accessed remotely. In this way, there is no need to buy, e.g., two CD players. Figure 1 sketches a house with a BeoLink system, which can be viewed as a number of audio/video devices connected in a network. The figure is drawn in a Petri net-like style. In this example, there are four devices. Generally, a BeoLink system connects up to 16 devices.

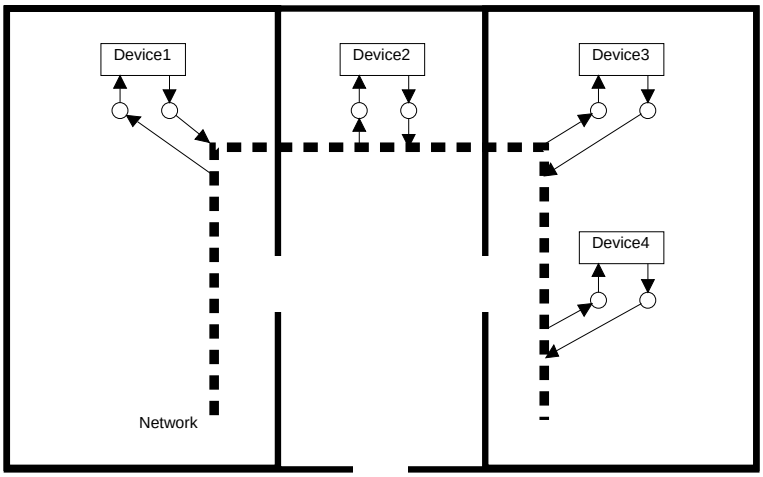

Fig. 1. A BeoLink system.

The BeoLink concept stipulates synchronisation. Assume, e.g., that a home has one CD player in the central room. If two persons being in separate rooms want to listen to different tracks at the same time, there is a conflict. A communication protocol is needed to resolve the situation sensibly and gracefully.

The rest of this paper is structured as follows: Section 2 provides an overview of the project. In Sect. 3, the considered vital part of BeoLink, the so-called lock management protocol, and the CPN model created are outlined. Section 4 introduces a new software library heavily used in this project for graphical feedback from simulations. Moreover, the section describes the approach to simulation and the validation results obtained. Section 5 covers how aspects of the CPN model were formally verified using occurrence graphs. Section 6 describes how CPN was used to test an idea for a future revision of BeoLink. In particular, compatibility between devices running the old and the new version of the considered protocol is investigated. Section 7 describes a supplementary CPN experiment

at B&O. In Sect. 8, the project described in this paper is compared with three other similar ones. The conclusions are drawn in Sect. 9.

2 Overview of the Project

The project began with intensive CPN training of three B&O engineers who were not familiar with Petri nets. They attended a six-day course given by the CPN group, with two days each week over three weeks. Practical application of CPN was emphasised, more than theoretical and mathematical aspects. The format was a mixture of lectures and practical exercises using textbook material from [8].

The first two days introduced the basic CPN concepts through small toy examples. Moreover, Design/CPN [10], the editing, simulation, and occurrence graph analysis tool for CP-nets to be used in the project was presented. About half the time was spent getting hands-on experience at the computer, modifying and simulating small models. Design/CPN uses the language CPN ML for declarations and net inscriptions. CPN ML is an extension of the functional programming language Standard ML [12]. The engineers were experienced C-programmers, but unacquainted with Standard ML. Therefore, some time was devoted to an elementary introduction to this language. Also, other examples on industrial CPN projects were presented to demonstrate the potential of the method.

The middle two days covered more advanced topics such as hierarchical CP-nets and CP-nets with time. During the last two days, models of systems chosen by the engineers were created and simulated. Being able to use the CPN method and tools on something that is well-known and relevant is an important constituent in a learning activity like this. Two separate CPN projects were initiated. The one that will be referred to as the *lock management project*, is the main subject of this paper. It deals with the lock management protocol of BeoLink to be described in Sect. 3. The other project is briefly discussed in Sect. 7.

In parallel with the CPN course for the B&O engineers, the CPN group learned about the BeoLink concept by reading technical documentation and getting demonstrations. Subsequently, the CPN group made a rough draft of a CPN model of the lock management protocol. The model was based upon a state-event matrix and a flow diagram, extracted from the existing description of the protocol. This draft model certainly eased the initial discussions with the B&O engineer who was going to carry out the lock management project. It was highly valuable to discuss the pros and cons of this concrete proposal.

After the CPN course came a period with close contact between the B&O engineers and the CPN group. The engineer responsible for the lock management project spent eight full days within one month at University of Aarhus creating the first parts of the CPN model. During these visits, he was working on his own most of the time. When needed, he was assisted by a person from the CPN group at the computer. It turned out to be an effective way of getting started with the

project. After this phase, the engineer was able to work quite independently, i.e., with one weekly meeting, in addition to contacts via telephone and e-mail.

The project was organised with a weekly visit by a person from the CPN group at B&O. This day provided an opportunity to discuss different ideas and to solve technical problems. In addition, there were monthly meetings involving several people, and with more general agendas. Models, simulation results etc were presented, reviewed, and discussed; and the directions to take now were decided.

In total, the project ran effectively for nine months in which the responsible B&O engineer spent 50% of his time on the lock management project. In chronological order, the focus of the project was on modelling, on simulation, and on occurrence graph analysis. Finally there was a phase with a mixture of all three activities, because a design of a new version of the protocol was undertaken.

Modelling and simulation were, except for the first month after the initial CPN course, done by the B&O engineer alone. In contrast, the CPN group participated closely in the entire occurrence graph analysis phase. This was caused by two reasons. First of all, the tool support for occurrence graph analysis of CP-nets was rather new. Therefore, there was not much experience to draw from regarding application of occurrence graph analysis to real-world CPN models. Thus it would be at least difficult, and most likely impossible, for the CPN group to provide appropriate guidance without being an active participant. Secondly, this project was seen as an opportunity to test the applied tool which is developed by the CPN group.

3 Lock Management in the BeoLink

This section first introduces the lock management protocol of BeoLink. The corresponding CPN model created is described next.

3.1 The Lock Management Protocol

The *lock management protocol* is a vital part of BeoLink used to grant exclusive access to various services. The purpose of the protocol is to prevent disorder, e.g., that track 11 is selected on a CD if two users simultaneously request track 1. The protocol manages a key which must be possessed by any device wanting to execute a function that can alter the audio/video distribution, or is depending on it. Examples of functions requiring the key are selection of audio/video source, e.g., switch on a radio or change from radio to CD player; and control of source, e.g., change of track on a CD. The device possessing the key is called the *lock manager*.

Communications in the protocol are initiated by users issuing events. It is apt to think of a user as a person operating a device, although this is a simplification: A person does not explicitly request a key. He or she does not even know that such a thing exists. Instead, e.g., the person wants to change track on a CD, and this prompts internal actions within a device resulting in communications in the

lock management protocol. In a similar simplified fashion, an event issued by a user can be thought of as a press of a button on a remote control.

Figure 2 depicts a typical communication sequence in the lock management protocol. Only one user is shown, `User1` who operates `Device1`. Say that `Device1` is a CD player, and that `User1` wants to change track. The event `key_wanted` is sent to `Device1`, which is not the lock manager. Therefore, `Device1` requests the key on the network by broadcasting a `REQUEST_KEY` *telegram* (telegram is B&O's preferred synonym for message). `Device3` is the lock manager and is ready to give away the key. When `Device3` sends the `KEY_TRANSFER` telegram to `Device1`, the key gets reserved. `Device1` is granted the key upon reception of the `KEY_TRANSFER` telegram, and sends a `NEW_LOCK_MANAGER` to `Device3` as an acknowledgement of a successful transfer. Finally, `User1` gets the event `key_ready`, and the change of track on the CD can take place.

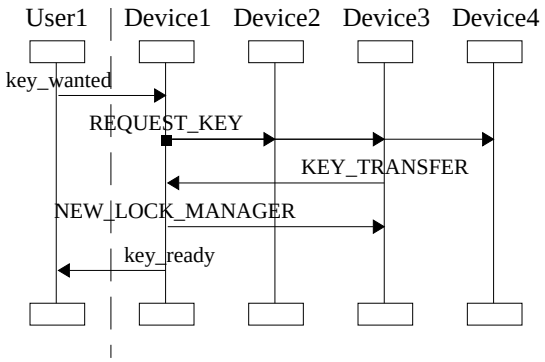

Fig. 2. A communication sequence in the protocol.

This is an example of a typical communication sequence, but obviously it is not a complete description of the protocol. Many cases must be dealt with: What if the lock manager is not ready to give away the key? What if the key is already reserved for a third device? What if the key is lost? Creating a formal executable model of the protocol like it is done in this project provides a sound basis for answering such questions.

When a BeoLink system starts from scratch, no key exists. The lock management protocol describes rules ensuring that a key gets generated in the initialisation phase. It is the obligation of the *power master* to do so. The power master is the device delivering electrical current to the data connection of a BeoLink system. Normally, there is exactly one power master. If there are more, pathological situations may appear, as we shall see in Sect. 4.2. Also, if the key is lost in a running BeoLink system, e.g., if the lock manager is turned off, the power master must ensure timely generation of a new key.

3.2 The Lock Management Model

The CPN model of the lock management protocol, *the lock management model*, consists of 13 *pages* (also known as subnets and modules). An overview is given in terms of the hierarchy page shown in Fig. 3 which has a node for each page.

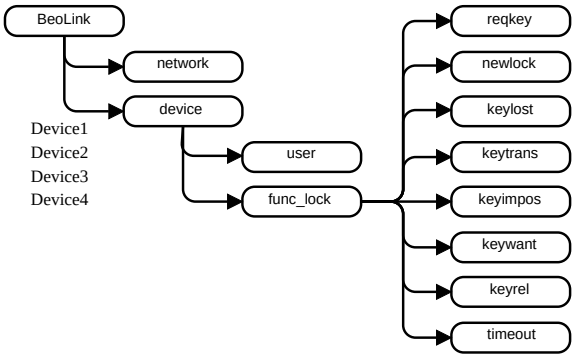

Fig. 3. The hierarchy page.

The topmost page `BeoLink` contains a high-level description of BeoLink and is already shown in Fig. 1 in Sect. 1. Here, the objects `Device1` to `Device4` and `Network` are *substitution transitions* meaning that their detailed behaviours are described on *subpages* (for presentation purposes, `Network` has a non-standard graphical appearance). Generally, in Fig. 3 an arc between two nodes indicates that the page of the source node contains a substitution transition described on the subpage of the destination node.

Thus a BeoLink system consists of a network and a number of devices. The `network` page models the primitives for sending, receiving, and broadcasting telegrams. In this presentation of the model, there are four *instances* of the `device` page corresponding to four devices in the considered BeoLink system. The `device` page models the selected aspects of a device, in particular `device` is connected to a description of the lock management protocol: The protocol is modelled by the page `func_lock` plus all its subpages. In addition, `device` is connected to the `user` page, which models the user of a device that sometimes wants the key. Each device has its own separate set of page instances. In total, for a BeoLink system with four devices, the model has 46 page instances.

The model contains colour sets, i.e., types, describing the telegrams. Also, there is a colour set `FL_STATE` (FL for function lock, a synonym used by B&O for the lock management protocol) used to represent the state of a device with respect to the key. The lock manager is either in state `KEY_FREE` or `KEY_USED`, depending on whether the key can be given away or not. A device which does not have the key and does not want it either, is in state `KEY_IDLE`. A device

which has requested the key, but has not got it yet, is in state `KEY_WAIT`. There are a number of other possible states used for various purposes.

To give an impression of the model, we will now describe two selected pages at a general level. Not all details are explained.

Page `func_loc`. An extract of the page `func_loc` is shown in Fig. 4. As mentioned earlier, this page is the topmost of the pages modelling the lock management protocol.

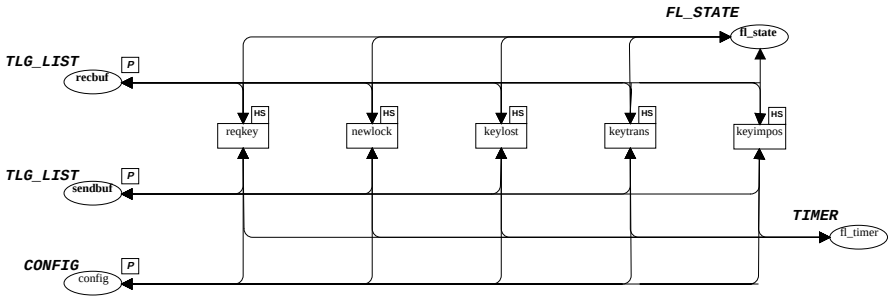

Fig. 4. Extract of the page `func_loc`.

All the transitions on `func_loc` are substitution transitions. Hence, there are no arc inscriptions in Fig. 4 — these details are hidden on subpages. The transitions correspond to the pages of the rightmost column of the hierarchy page in Fig. 3. Only five of the transitions are shown in Fig. 4. The shown transitions describe how incoming telegrams are handled. The transitions not shown correspond to the three bottommost pages in the rightmost column of Fig. 3. Two of these transitions deal with incoming events from users, the third transition handles timeouts.

In Fig. 4, when a telegram is available on the place `recbuf`, one of the actions represented by the transitions handling incoming telegrams can happen. Which action depends on the telegram. The action represented by such a transition may produce a telegram on the place `sendbuf`; may change the state of the device with respect to the key, i.e., change the marking of the place named `fl_state`; and it may read and/or set a timer, represented by the place `fl_timer`. The marking of the `config` place is used to make various choices. `config` contains configuration information for a device, e.g., saying whether a device is power master or not.

Page `reqkey`. Now we will describe the `reqkey` page shown in Fig. 5. It is subpage for the leftmost transition of Fig. 4.

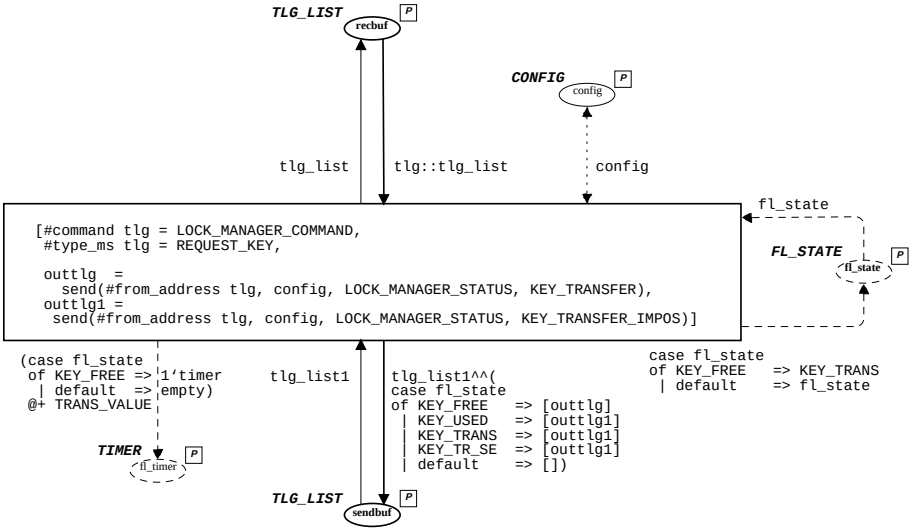

Fig. 5. The page `reqkey`.

There is only one transition on this page. The transition models the action taken when a `REQUEST_KEY` telegram is received by a device. In general, there are three kinds of possible responses: 1) If the receiving device is lock manager and is ready to give away the key, the response is a `KEY_TRANSFER` telegram sent to the requesting device. 2) If the receiving device is lock manager but is not ready to give away the key, the response is a `KEY_TRANSFER_IMPOS` telegram sent to the requesting device. 3) If the receiving device is not lock manager, it does not reply at all.

We now take a closer look at Fig. 5 and explain how the above rules are modelled.

A telegram is coming from the topmost place `recbuf`. To ensure that the first telegram received by a device is also handled first, the colour set of `recbuf` is the list type `TLG_LIST`. The reception of a telegram consists in extracting the first element of the list, i.e., taking a list matching the pattern `tlg::tlg_list` from `recbuf` and returning only `tlg_list`. The operator `::` is the basic list constructor putting its left argument (here, the telegram bound to the variable `tlg`) in front of its right argument (here, the list of telegrams bound to the variable `tlg_list`).

The first two equations of the (large) guard positioned in square brackets inside the transition check that the telegram is to be handled here. The commas in the guard are interpreted as logical conjunction. The first equation insists that the telegram concerns the lock management protocol — there are many other telegrams on the network used for other purposes. The second equation checks that it is indeed a `REQUEST_KEY` telegram.

The two remaining equations of the guard are used to construct the response to be sent on the network. `send` is a function returning a telegram with an appropriate sender and receiver composed from the given arguments. Whether the response is a `KEY_TRANSFER` telegram, a `KEY_TRANSFER_IMPOS` telegram, or nothing at all, can be seen from the guard and the `case` expression on the arc going to the bottommost place `send_buf`. The `case` expression evaluates to a list with either one or zero elements, which is concatenated to the list of telegrams already present on `sendbuf` using the operator `^^`. The marking of the rightmost place `fl_state` is used to determine the response, and may change as a consequence of the decision made.

If a key transfer is started, a timer is set: A token is put on the place `fl_timer`. This is done in order to be able to time out if an acknowledgement of a successful transfer from the recipient of the key does not arrive in due time.

The four other subpages of page `func_loc` for handling incoming telegrams have the same net structure as `reqkey`, but different guards and arc expressions. The three subpages for handling user events and time-outs are simpler.

The CPN model described in this section is easily extensible. This is an important property, because presently, only the lock management protocol of BeoLink is modelled. It is natural to consider inclusion of other functionalities, e.g., selection of audio/video source. Selection of a source can be added as an additional subpage for the `device` page. In this case, the hierarchy page (Fig. 3) would appear with another column of pages next to and similar to `func_loc` and its subpages. The only modification of existing pages would be addition of one substitution transition to the `device` page.

4 Simulation

This section first introduces the concept of message sequence charts that were used to provide graphical feedback from simulations in a way that is very familiar to the B&O engineers. The approach to simulation of the lock management model and the validation results obtained are described next.

4.1 Message Sequence Charts

When discussing behaviour of distributed systems, designers often draw charts illustrating the message passing between components of the system. Such *message sequence charts* [7] are very useful to capture normal behaviours, and also to discuss special cases. In fact, the B&O engineers used exactly this kind of charts when they explained the BeoLink concept and the lock management protocol for the CPN group in the beginning of this project. We already saw an example of a message sequence chart in this paper, Fig. 2 in Sect. 3.

The Design/CPN tool includes a recently implemented library [3] supporting automatic creation of message sequence charts as logs of simulations. Figure 6 shows an example of a message sequence chart from the lock management model.

There is a vertical bar for each of the four devices of the considered BeoLink system.

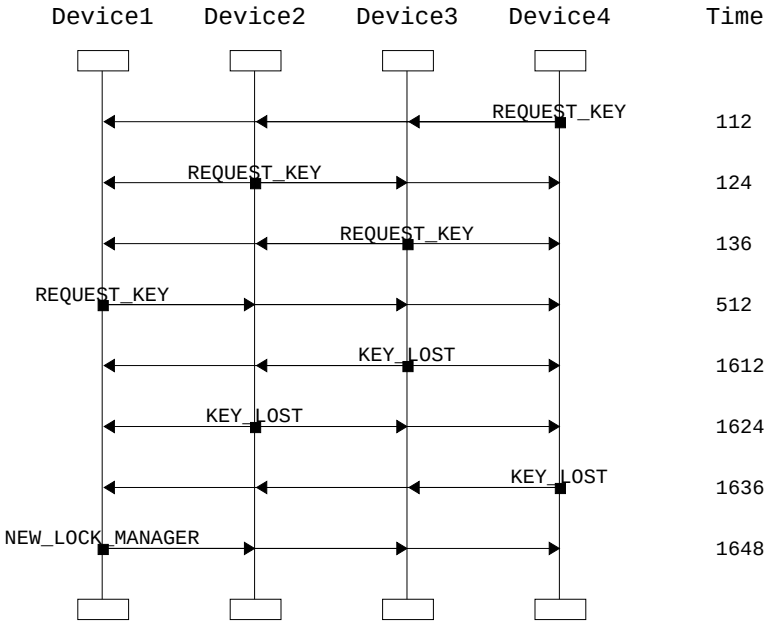

Fig. 6. A message sequence chart.

The lock management model is timed. Timed CP-nets are formally defined in [9], and will be explained in more detail in Sect. 5.1 of this paper. For now it suffices to say that there is a global clock, and it makes sense to speak about the time of an action. The message sequence chart of Fig. 6 displays the telegrams sent to the network and the time at which the sending happens. A horizontal arrow indicates the sender and the receiver. Multi-headed arrows are used for broadcasts.

Figure 6 shows a snapshot of the initialisation phase, a simulation in the time interval from 0 to 1648. A time unit in the CPN model corresponds to a millisecond in the protocol implementation. At time 112, `Device4` requests the key. Shortly after, the three other devices do the same. However, as mentioned in Sect. 3.1, no key exists in the BeoLink system when it starts from scratch. `Device3` is first to discover this — at time 1612, it does so by broadcasting a `KEY_LOST` telegram. Subsequently, both `Device2` and `Device4` also discover the missing key and take appropriate action by broadcasting `KEY_LOST` telegrams. At time 1648, the power master, `Device1`, finally notifies the other devices that it has generated a key by broadcasting a `NEW_LOCK_MANAGER` telegram.

4.2 Simulation Approach and Results

In the early modelling phase, interactive simulations were used intensively. In these kinds of simulations, the behaviour of the model is investigated on the level of the token game. Often, the B&O engineer and a person from the CPN group sat together at the computer and studied the effect of the individual transitions in detail in order to get the model right.

After about three months, the lock management model was considered complete. It was now instrumented to create and update message sequence charts. This consisted in attaching code segments, i.e., pieces of Standard ML code, to two transitions. A code segment is evaluated when its transition occurs. Quite often, like here, the purpose is to have a side effect such as an update of a drawing. From this point, the B&O engineer used almost exclusively automatic simulations to study the behaviour of the model. In an automatic simulation, the token game is not displayed and the feedback comes solely from the effects of code segments. Automatic simulation allows a large number of steps to be executed in a short time.

At a certain point in the simulation phase, the engineer was convinced that the model behaved correctly in the normal cases. His next objective was to validate that it was able to deal with a number of special cases.

A case where an extra key was implanted into the model was simulated. Two keys may appear in the real world if telegrams are lost, although this occurs only very rarely. However, the lock management protocol must be sufficiently robust to handle such a situation. It indeed turned out to be. Two keys are merged into one shortly after their appearance. When a lock manager receives a `NEW_LOCK_MANAGER` telegram, it immediately accepts that somebody else has a key and consequently abandons its own. The real-world effect of two simultaneously existing keys could be some noise in a set of loud speakers during a short time interval, e.g., sound from the radio and the CD player at the same time. Also, it could be that a press on a button of a remote control is lost.

A BeoLink system installed with a wrong kind of cable may have two power masters. This case was simulated. If the users of the two devices being power masters never want the key, an apparently never-ending sequence of regenerating a key and immediately abandoning it in each of the two power masters may take place. It is an actual livelock of the protocol. However, it is not critical, because it can only happen in an incorrectly installed system. On the other hand, B&O generally strives to make software robust, i.e., not relying too much on assumptions about proper functioning of the environment. Thus it would make sense to modify the protocol to be able to deal sensibly with the presence of two power masters as well. It would even be quite easy to do so.

Another observation from the simulation runs is that the protocol is highly dependent on the duration of the time periods before a time-out. If time is switched off in the model — this corresponds to every action happening instantly — the protocol malfunctions. Several keys appear, and no proper lock management is in effect.

Except for the last one, the observations above were as expected by the engineer. He found it very interesting to recognise the real-world behaviour of the protocol in the CPN model. He believes that if the model had existed before the implementation, thus allowing experiments with different solutions in an easy and cheap fashion, the protocol would certainly have profited from that.

5 Occurrence Graph Analysis

This section first contains a general discussion of occurrence graph analysis for timed CP-nets such as the model of the lock management protocol. The verification results obtained using the method are described next.

The tool applied was the Design/CPN Occurrence Graph Tool [4] offering functionalities to generate occurrence graphs, to draw them, and to do queries.

5.1 The Challenge of Applying Timed Occurrence Graphs

An *occurrence graph* [9] for a CP-net is a directed graph with a node for each reachable marking and an arc for each occurring binding element[1]. An arc is going from the node of the marking in which the associated binding element occurs to the node of the marking resulting from the occurrence.

The lock management model is timed: Each marking has an associated global time and some tokens carry time stamps determining when they are ready to take part in occurrence of transitions. Time stamps are taken into account when two markings are tested for equality in generation of the occurrence graph. The global time is as well. Two identical markings existing at different global times are considered different. Another way to put it is to say that the global time is part of the state. Because the global clock advances and the protocol is intended to run forever, the occurrence graph for the lock management model is expected to be acyclic and infinite. An occurrence graph for a timed CP-net is called a *timed occurrence graph*.

If timing aspects were suppressed in the model, there would be a chance of generating a full occurrence graph. It might very well be huge, but expectedly finite. Unfortunately, as was noted in Sect. 4.2, timing is essential. Without it, the protocol malfunctions. Thus the challenge of analysing the original timed model was faced. The natural question to ask now was: Is it at all possible to use timed occurrence graphs to derive any formal analysis in practice? As with simulations, negative results may be obtained. E.g., if one of the markings encountered during a generation is dead, certainly the considered CP-net has a dead marking — because it has just been found. Here however, the aim was to establish positive results, i.e., prove that something works as expected. To achieve this, before a generation was initiated, we had to carefully decide how to proceed. It was impossible to encounter all reachable markings. Thus the target was less ambitious and more specific.

[1] A *binding element* is a pair containing a transition and a binding. The binding assigns values to all variables of the transition.

The Design/CPN Occurrence Graph Tool provided help through its functionality of *branching options*. A branching option is a user-defined predicate taking a state as argument and determining whether successors are generated or not. A certain generation strategy can thus be enforced by writing an appropriate predicate.

Figure 7 gives an example of the use of a branching option. The model giving rise to the figure is a much simplified version of the lock management model only used for illustration purposes here.

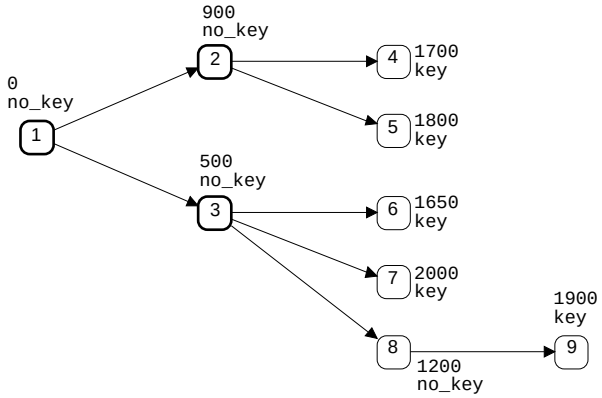

Fig. 7. Sketch of a partial timed occurrence graph.

The considered branching option says whether a marking has a key or not (`key`/`no_key`). In all the leaves, i.e., all states for which no successors have been generated, a key is present. Therefore, the generation has stopped and the graph is partial. We will return to this existence of a key issue for the real lock management model in Sect. 5.2.

In Fig. 7, next to each node is shown the associated global time[2]. Another example of a branching option could be a predicate saying whether the global time of a state is less than a certain value. This would make it possible to investigate all behaviours of a model up to a certain global time. If this value is, say, 1000, in Fig. 7, only the nodes 1, 2, and 3 shown with a thick border, would be processed during generation.

Having a branching option in effect is often not sufficient to yield a usable partial occurrence graph that can be stored in the memory of the computer available. In addition, typically the model itself must be adapted to make it tractable for occurrence graph analysis. In theory, the occurrence graph method

[2] The global clock is not necessarily advanced in consecutive steps. After occurrence of a step, the clock is set to the next moment at which something can happen, i.e., time leaps.

is pleasantly simple: Generate the graph and do the queries. In practice, this is often a deceit. Even for a system with a finite occurrence graph, because of the well-known state explosion problem, the graph is typically too big to be stored in the computer available. Thus a modeller working towards verification using occurrence graphs must initially consider modification and reconfiguration of the model so that there is at least a chance of succeeding.

A necessary modification of the lock management model was to introduce bounds on the buffers used for input and output between the devices and the network, e.g., the places `recbuf` and `sendbuf` on the page `reckey` shown in Fig. 5. These buffers were unbounded in the model used for simulation, and starting generation of an occurrence graph for this model would certainly not be wise. The time may be spent and the memory of the computer filled while the occurrence sequence, i.e., behaviour, where one device just produced a long sequence of telegrams in its output buffer is being recorded. I.e., there would be no guarantee at all that usable information was derived.

Generally, when trying to make a CPN model tractable for occurrence graph analysis, it is often beneficial to fix small values of possible system parameters. In the lock management model, the system parameter is the number of devices on the BeoLink system considered. However, as we shall see below, the (partial) occurrence graphs generated for the four devices system, which has been considered throughout this paper, are, with the chosen branching options, of a manageable size. Thus in this case, it was not needed to decrease the chosen system parameter.

5.2 Occurrence Graph Analysis Results

First, the part of the lock management protocol taking care that a key is generated when a BeoLink system starts from scratch was verified. It was formally proved that no matter what happens during this initialisation phase, the system always reaches a state where a key exists.

By means of the `key/no_key` branching option illustrated in Fig. 7, an occurrence graph for the entire initialisation phase was generated. It was of a very manageable size and generated within a short time. For the four devices system, there was 13,420 nodes and 41,962 arcs. To get more results than just for the four devices system, the B&O engineer augmented the model with a new flexible mechanism to turn devices on and off, and thus conveniently experiment with different numbers of devices. For each case, it was verified that the occurrence graph was acyclic, and that all terminal nodes corresponded to markings in which a key exists. This means that in a finite number of steps, eventually a marking where a key is present will be reached. Moreover, the minimum and maximum times to reach a marking in which the key is generated were found. For the four devices system, the minimum was 1600 and the maximum 2000. As mentioned previously, these measures are in milliseconds. Thus the analysis showed that it takes between 1.6 and 2.0 seconds before a key is generated. These times are in accordance with the times known from the real world, i.e., the actual times it

takes from power is switched on until a BeoLink system is ready. This indicates that reliable performance measures are indeed derivable from the CPN model.

The next natural step in the formal verification of the model was to try to verify that at all times, there is at most one key. Of course, it was not possible to do it the straightforward way because the occurrence graph was infinite. Instead, the lock management model was partially verified as follows: An occurrence graph containing all occurrence sequences, i.e., behaviours of the system, within one key transfer was generated. The initialisation phase had already been verified. Thus it was safe to initialise the model to already have a lock manager before the occurrence graph generation was started. A branching option was installed expressing that there is a new lock manager, i.e., the key is transferred to one of the other devices. It was interesting to check, e.g., that there were no dead markings, and that the bounds on places were as expected. For four devices, the graph had 2,578 nodes and 5,335 arcs. An intuitive appealing idea is to use this experiment as basis in a proof of total correctness using mathematical induction in the number of key transfers. Unfortunately, when the key is transferred once, the system may be in several different states. In fact, it turned out that there were too many cases to make the approach usable in practice. However, even though this experiment did not serve as a total verification of the protocol, the confidence in its correctness was strongly increased.

6 Next Revision of BeoLink

This section describes an experiment made alone by the engineer responsible for the lock management project. He wanted to use CPN to solve a recurring compatibility problem within the company. Each year, B&O releases a new series of products. Under the very reasonable assumption that the typical customer is not willing to replace his or her expensive audio/video system in total at one instant, new and old devices must be able to co-exist. A radio some years old, and a brand new TV installed by the dealer yesterday, must be able to function together in the same BeoLink system without causing communication problems.

Therefore the engineer wanted to design a possible future revision of the lock management protocol, and ensure that a BeoLink system with a mixture of devices running the new and the old protocol will have a proper lock management in effect. Without having an executable model of the protocol, a proposal for a new version, and in particular its compatibility with the old one, is cumbersome and expensive to try out. In practice, an implementation must be done before any testing can take place.

In the design of the new protocol, the concept of a power master is abandoned. Instead, there is now a *video master* and/or an *audio master*. The video master has the obligation and the right to generate a new key, immediately when it discovers that none exists. Under the same condition, the audio master may generate a new key, but only after some time period has elapsed. This asymmetry between the masters is introduced to ensure that only one key is generated. If they both respond immediately, there is a high risk of two keys appearing.

The CPN model of the new protocol was created by the engineer without support from the CPN group. At this stage, the engineer was fully capable of working with the CPN method and tools all by himself. It took only about two weeks (four weeks half-time) to design the new protocol, and to create and investigate the corresponding CPN model. In the original model, described in Sect. 3.2, new versions were created of four of the pages modelling the protocol, i.e., some of the subpages of the page `func_loc` (see Fig. 4). New substitution transitions were added accordingly to `func_loc`. A flexible configuration mechanism was created, making it possible to vary the mixture of new and old devices without changing the structure of the model itself. When the engineer presented the results of his efforts for the CPN group, the behaviour was conveyed using message sequence charts. Many simulation runs had been done, including detailed investigation of five cases corresponding to different configurations with new/old devices and presence of video/audio masters. The simulations confirmed that the new protocol is indeed sensible. In the first place, a BeoLink system with all new devices work as expected. Secondly, and equally important, new and old devices do co-exist without problems.

In addition to simulation, the model including a mixture of new and old devices was formally verified using occurrence graphs in the same way as the original model. The same five cases as considered using simulation, were investigated again. The work was done by the engineer alone. Being now experienced in this activity, it took only about half a day.

7 The CD Protocol — A Supplement

The lock management project was the main experiment in B&O's investigation of CPN. As an independent supplement, the *CD project* was carried out in parallel. Here, the *CD protocol* describing rules for communication between two processes internal to a so-called interactive CD was studied. The protocol is documented as prose text and adheres to the 7-layer OSI model. All other layers but 1, 2, and 7 are empty. The CD protocol is quite different from the lock management protocol.

In an intricate way, the CD protocol governs communication between two processes. These processes are not symmetrical. Reception of messages has priority over transmission of messages in one of the processes. This means that if this process is engaged in transmitting a message, and it detects that it has an incoming message waiting, it must immediately switch to receiving. The interrupted transmission must subsequently be resumed. The other process cannot be interrupted in any activity.

The aim of creating a CPN model was to investigate properties like data transmission reliability and collision handling. In comparison with the lock management project, only a small effort was invested. Nevertheless, it resulted in a sensible 9-page model. There was no attempt to do occurrence graph analysis. Simulation was done on the level of the token game. The project never got to a phase where a more appropriate way of representing the results of simulations

was implemented, such as using message sequence charts in the lock management project. Anyway, using simulation, the responsible B&O engineer discovered a bug in the design of the CD protocol. It was related to the situation where a switch from transmission to reception was forced. The bug did not cause the protocol to malfunction, but it did decrease its performance. It turned out that the programmer responsible for the implementation actually recognised this bug. But it had first been found late in the test phase, and it had been very convenient to have fixed it earlier.

According to the engineer who carried out the CD project, if the protocol was about to be implemented now, the CPN model would have provided a basis for writing better and simpler code.

In addition to investigating CPN for specification, validation, and verification at B&O, a small experiment was made to test the feasibility of CPN to document communication protocols. Presently, B&O has good methods for documenting the static parts of protocols, e.g., the involved telegrams/messages and states. However, better ways of documenting the dynamics of protocols are pursued as well. As an attempt, the CPN model of the CD protocol was connected with a textual description of the protocol via a hypermedia application [6] supporting organisation of information from many sources using cross references. This allowed easy navigation between the textual description of a part of the protocol and the corresponding location in the CPN model. In this way, the CPN model served as an integrated supplement to the existing documentation of the protocol.

8 Related Work

In this section, the project described in this paper, referred to as the *B&O project*, is compared with three other similar ones recently documented in the literature.

Philips Audio Protocol. In [1], the formal verification of an audio protocol from Philips is described. The protocol focuses on low-level data transmission, much like the CD protocol. It specifies rules for exchange of control messages between processes within a certain audio device. The model created is a timed automata which is verified using the model-checking tool `UPPAAL` [2]. The properties established are formulated as temporal logic formulas. As an example, it is proved that the receiver only receives messages that are actually transmitted.

[1] has many similarities with the B&O project. Both are attempts of research groups to test approaches and tools for systems specification and verification within a large company producing audio/video systems. A notable difference is that the modelling and verification described in [1] was carried out by the authors, i.e., by the research group. In the B&O project, most of the work was done by the B&O engineers. The CPN group was merely consultants. This was a deliberate choice which we see as a success. We learned that it is possible, within a reasonable time frame, to teach CPN including occurrence graph analysis to

engineers from the industry, enabling them to apply the method themselves. Another difference is that [1] focussed exclusively on formal verification. In addition to that, the B&O project included validation using simulation, an activity highly appreciated by the B&O engineers.

Modelling using CPN is quite similar to programming in a high-level language. This competence is obviously wide-spread in the industry, providing a good starting point for introducing CPN in other companies as well. With respect to formal verification using occurrence graphs as applied in the B&O project, the underlying mathematics is hidden for the user. The interface to generation, drawing, and querying of the Design/CPN Occurrence Graph Tool makes it possible to take advantage of the method without knowing the details of its theoretical foundation. We cannot judge how easy it is to understand the ideas of timed automata and temporal logic, although we do believe that it does require a rather strong mathematical background.

Alarm Systems. The small Danish company Dalcotech has successfully been using CPN for the design of a new alarm system as documented in [11]. This project and the B&O project were organised the same way: First came a phase with initial CPN training for the engineers, and initial model drafts made in close cooperation between the company and the CPN group. It continued with a long phase where the engineers worked relatively independently. Dalcotech and B&O share many conclusions on the CPN projects carried out. However, Dalcotech takes a more dramatic step and introduces CPN as a general development method for all future designs of alarm systems. B&O already uses SA/SD [13] as the main development method. A lot of resources have been invested in education of the B&O engineers, and SA/SD serves its purposes well. Thus B&O never had the intention to base all their development activities on CPN. Also, the Design/CPN tool lacks version control and proper support for multiple users working on the same project, obstructing optimal large-scale use in a company like B&O. Use of the current version of the tool, e.g., by 15 engineers working together on producing software for a new TV, would be awkward.

Network Gateways. Another recent industrial application of CPN and Design/CPN is described in [5]: The design of a gateway between radio networks and broadband integrated services digital networks within the Australian Defence Force. Of particular interest in this project is the fact that occurrence graphs were applied in an industrial environment. Quite similar to the B&O project, occurrence graphs were used to verify basic behaviours of a CPN model, not to do a total verification of the full model.

9 Conclusions

Towards the end of the project described in this paper, the involved B&O engineers wrote a status report for their managers. Here they summed up the goals

and the results obtained. The goal of using CPN was to improve the methods for specification, validation, and verification of protocols. The main conclusion of the status report is that the goals were met. Being able to create executable, formal models is recognised as a valuable basis for obtaining better results faster.

Many industrial modelling and simulation projects using Petri nets have been documented in the literature over the years. Thus, it is important to make clear what the contribution of this paper is for the research area of Petri nets. Did we learn anything new?

We learned that it is possible to convey the CPN method in a relatively short time to make it usable for industrial engineers in a way so that they are able to work independently. The hardest part for the B&O engineers was to cope with Standard ML. As experienced C-programmers, they were used to think in terms of an imperative language, and the shift to a functional language turned out to involve some difficulties. A lot of the consultancy work of the CPN group consisted in assisting with Standard ML tasks.

We learned that application-specific graphical feedback from simulations is highly important. The B&O engineer responsible for the lock management project relied heavily on the new support for message sequence charts in Design/CPN. It made his work both more pleasant and more efficient. Moreover, it enabled the engineer to discuss results of simulations with colleagues unfamiliar with CPN.

We learned that it is possible to introduce a formal analysis method, occurrence graphs, in an industrial company. It was demonstrated that timed occurrence graphs can prove really useful, also to derive positive formal analysis results.

B&O's interest in this project was explained by the fact that communication protocols are gaining increasing attention within the company. Today, BeoLink offers access to sound and vision throughout a home using traditional audio and video sources. B&O anticipates that its products will be used more and more to disseminate information in general in the home, e.g., via access to the Internet. Proper communication protocols are essential to achieve good results in this area. Based on the project described in this paper, B&O has concluded that CPN can be a useful aid in this process.

Acknowledgements

We thank B&O for being willing to participate in this project. In particular, we thank the involved B&O engineers: Niels Toft Sørensen was responsible for the lock management project, Kristian Lund for the CD project. Without their efforts and abilities, this project could not have happened.

We thank Kurt Jensen and Kim Halskov Madsen for valuable guidance, ideas, and comments during the project. Moreover, we thank Rikke Drewsen Andersen, Lars M. Kristensen, and Kjeld Høyer Mortensen for proof-reading and comments on the paper.

This work has been supported by grants from the Faculty of Science at University of Aarhus.

References

1. J. Bengtsson, W.O.D. Griffioen, K.J. Kristoffersen, K.G. Larsen, F. Larsson, P. Pettersson, and W. Yi. Verification of an Audio Protocol with Bus Collision Using `UPPALL`. In R. Alur and T. Henzinger, editors, *Proceedings of the 8th International Conference on Computer-Aided Verification, New Brunswick, New Jersey, USA*, volume 1102 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.
2. J. Bengtsson, K.G. Larsen, F. Larsson, P. Petterson, and W. Yi. `UPPALL` — A Tool Suite for Automatic Verification of Real-Time Systems. In *Proceedings of the 4th DIMACS Workshop on Verification and Control Hybrid Systems, New Brunswick, New Jersey, USA*, 1995. To appear in Lecture Notes in Computer Science, Springer-Verlag.
3. S. Christensen. *Design/CPN Message Sequence Charts Library Manual*. Computer Science Department, University of Aarhus, Denmark.
   Online: `http://www.daimi.aau.dk/designCPN/`.
4. S. Christensen, K. Jensen, and L.M. Kristensen. *Design/CPN Occurrence Graph Manual*. Computer Science Department, University of Aarhus, Denmark.
   Online: `http://www.daimi.aau.dk/designCPN/`.
5. D.J. Floreani, J. Billington, and A. Dadej. Designing and Verifying a Communications Gateway Using Coloured Petri Nets and Design/CPN. In J. Billington and W. Reisig, editors, *Proceedings of the 17th International Conference on Application and Theory of Petri Nets, Osaka, Japan*, volume 1091 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.
6. K. Grønbæk and R.H. Trigg. Design Issues for a Dexter-based Hypermedia System. *Communications of the ACM, Vol. 37, 2*, 1994.
7. International Telecommunication Union — Telecommunication Standardization Sector (ITU-T). ITU-T Recommendation Z.120: Message Sequence Chart, Geneva, Switzerland, 1993.
8. K. Jensen. *Coloured Petri Nets — Basic Concepts, Analysis Methods and Practical Use. Vol. 1, Basic Concepts*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1992.
9. K. Jensen. *Coloured Petri Nets — Basic Concepts, Analysis Methods and Practical Use. Vol. 2, Analysis Methods*. Monographs in Theoretical Computer Science. Springer-Verlag, 1994.
10. K. Jensen, S. Christensen, P. Huber, and M. Holla. *Design/CPN Reference Manual*. Computer Science Department, University of Aarhus, Denmark.
    Online: `http://www.daimi.aau.dk/designCPN/`.
11. J.L. Rasmussen and M. Singh. Designing a Security Systen by Means of Coloured Petri Nets. In J. Billington and W. Reisig, editors, *Proceedings of the 17th International Conference on Application and Theory of Petri Nets, Osaka, Japan*, volume 1091 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.
12. J.D. Ullman. *Elements of ML Programming*. Prentice-Hall, 1993.
13. E. Yourdan. *Modern Structured Analysis*. Prentice-Hall, 1989.