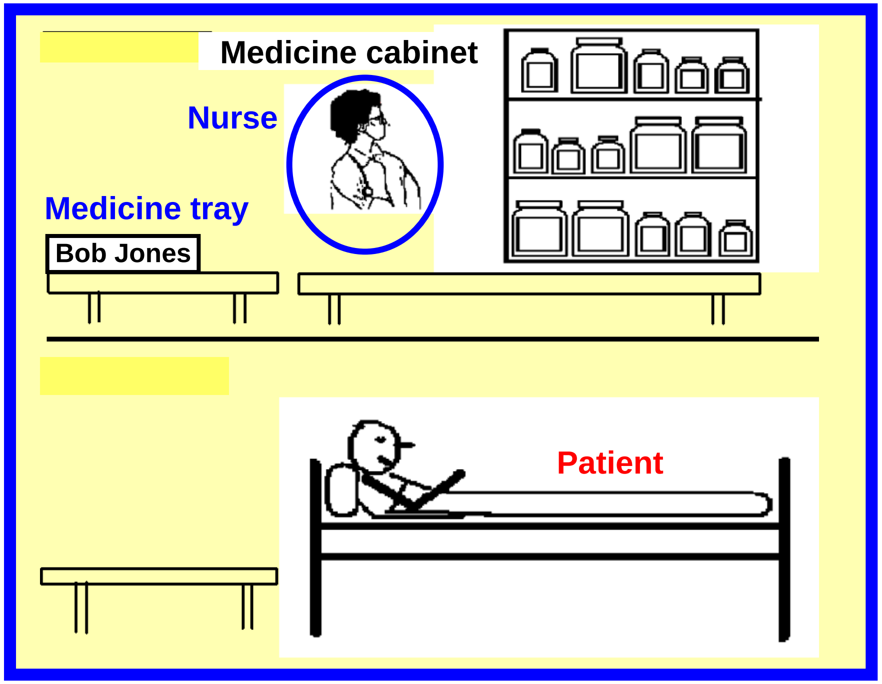
# Coloured Petri Nets

## Modelling and Validation of Concurrent Systems

# Chapter 14: Industrial Applications

**Kurt Jensen &**
**Lars Michael Kristensen**

**{kjensen,lmkristensen}**
**@cs.au.dk**



# Industrial projects

- We present four projects where CP-nets and their supporting computer tools have been used for system development in an industrial context.
- The projects illustrate that CP-nets can be used in many different phases of system development – ranging from requirement specification to design, validation, and implementation.
- The CPN models have been constructed in joint projects between our research group at Aarhus University and industrial partners.
- More than 100 examples of documented industrial projects can be found at:

  www.cs.au.dk/CPnets/industrialex

# The four projects

- Design of an edge router discovery protocol for mobile ad-hoc networks (with Ericsson Telebit).

- Specification of business processes and requirements engineering for a new IT system (with Systematic Software Engineering and Aarhus County Hospital).

- Design of the BeoLink system (with Bang & Olufsen).

- Development of a planning tool for the Australian Defence (with Australian Defence Science and Technology Organisation).

# Industrial project: Protocol design at Ericsson Telebit

- Design of an Edge Router Discovery Protocol (ERDP) for mobile ad-hoc networks.

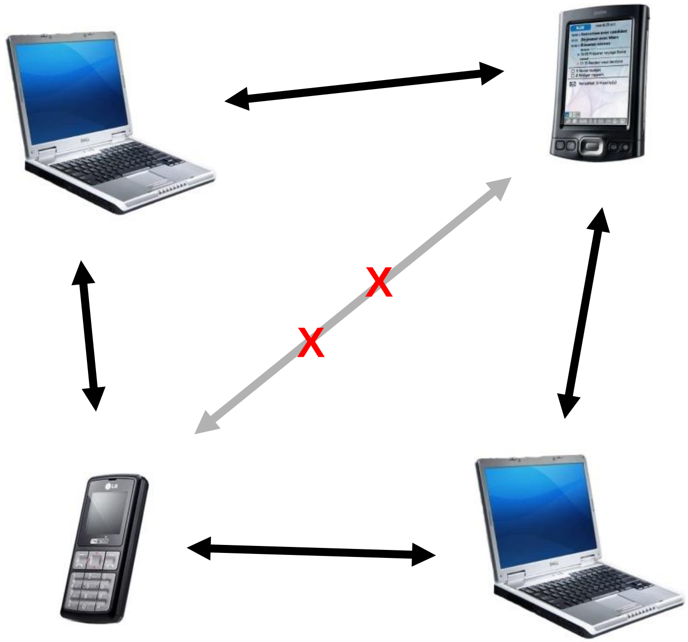
- A CPN model was constructed constituting a formal executable specification of the ERDP protocol.
- Simulation and message sequence charts were used for initial investigations of the protocol's behaviour.
- State space analysis was applied to conduct a formal verification of key properties of ERDP.

- Both the modelling, simulation, and subsequent state space analysis helped in identifying several omissions and errors in the design – demonstrating the benefits of using formal techniques in a protocol design process.

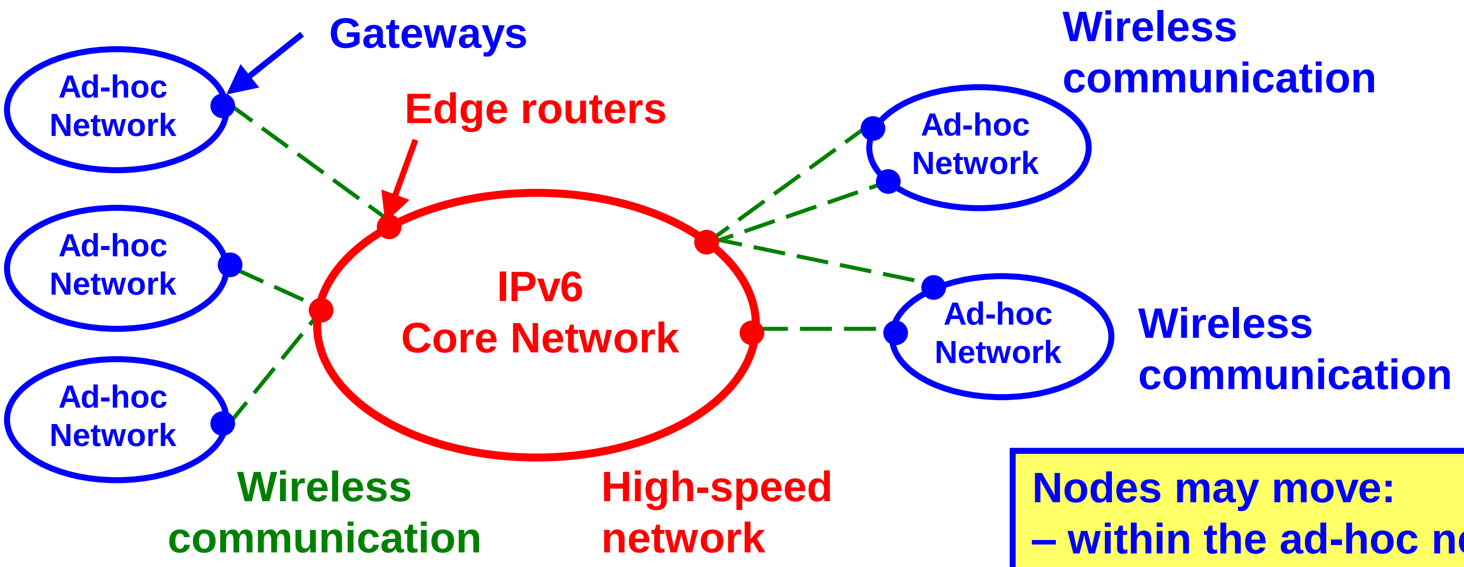
# Mobile ad-hoc network

- Collection of mobile nodes (devices), such as laptops, tablets, and mobile phones, capable of establishing a communication infrastructure for their common use.

- The nodes in an ad-hoc network operate:
  - in a fully self-configuring and distributed manner,
  - without any pre-existing communication infrastructure (such as designated base stations and routers).



# Network architecture


Gateways

Edge routers

Wireless communication

Ad-hoc Network

Ad-hoc Network

Ad-hoc Network

Ad-hoc Network

Ad-hoc Network

IPv6 Core Network

Wireless communication

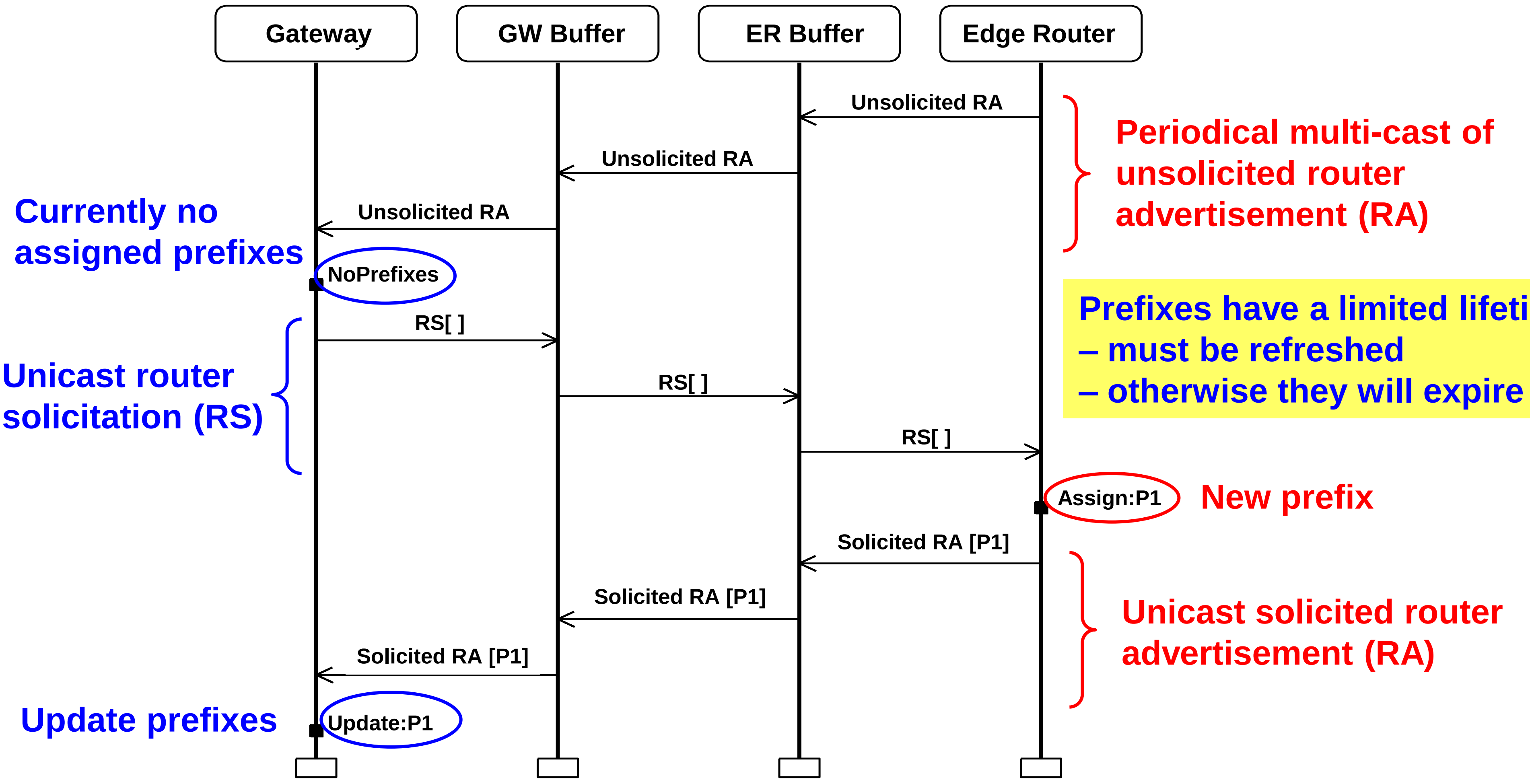
Wireless communication

High-speed network

**Nodes may move:**
**– within the ad-hoc networks**
**– from one ad-hoc network to another**

- ERDP supports:
  - gateways in discovering edge routers,
  - edge routers in configuring gateways with a globally routable IPv6 address prefix.

# Configuration of a gateway



Periodical multi-cast of unsolicited router advertisement (RA)

Currently no assigned prefixes

Unicast router solicitation (RS)

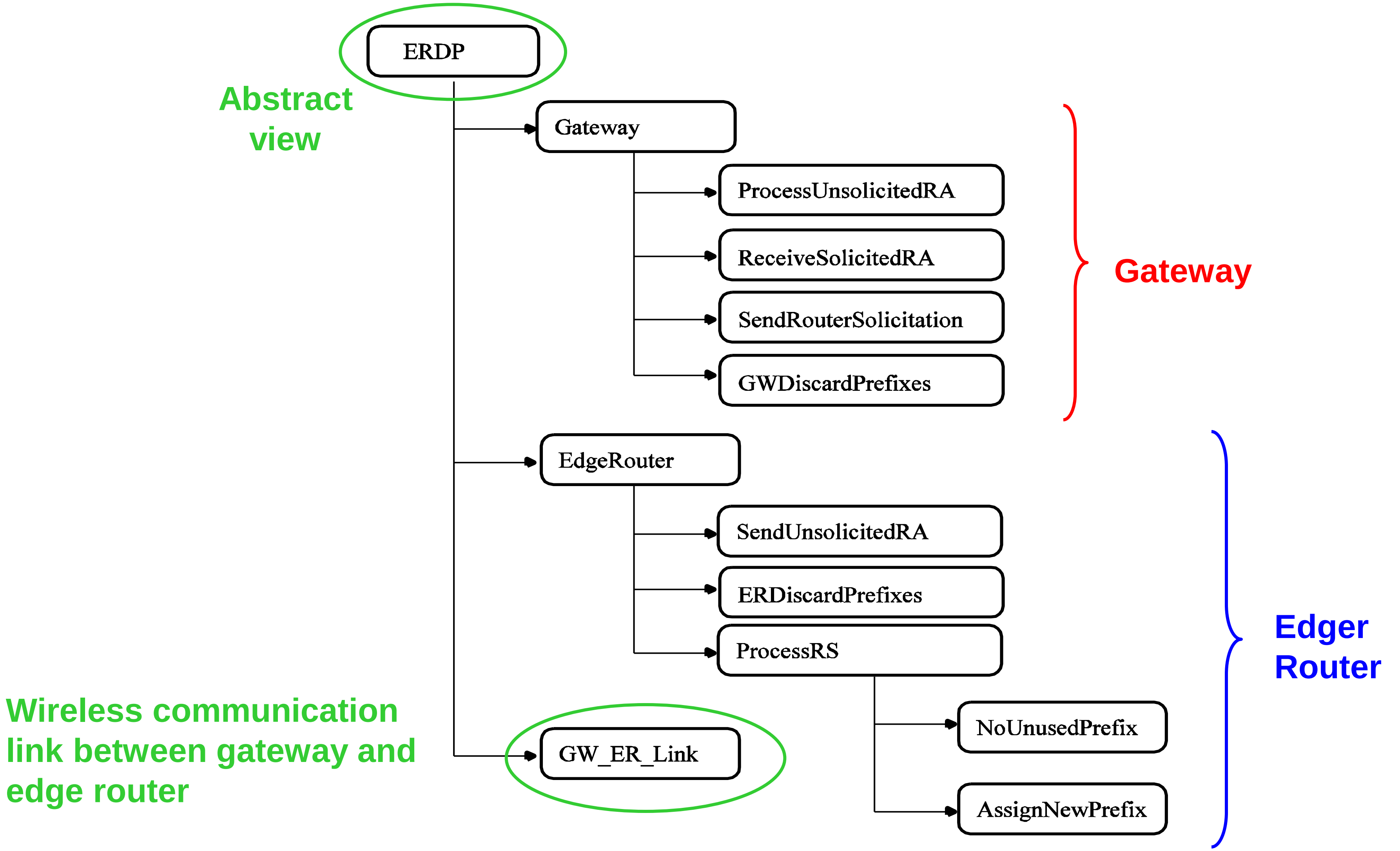
New prefix

Unicast solicited router advertisement (RA)

Update prefixes

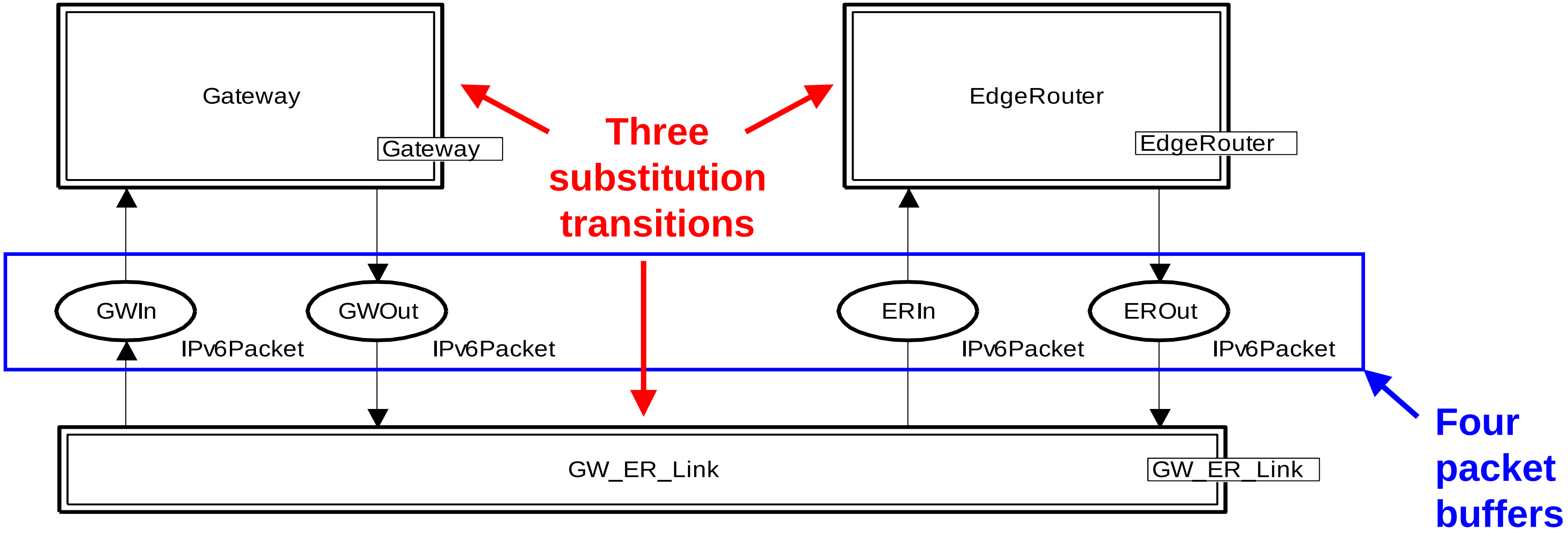
Prefixes have a limited lifetime
– must be refreshed
– otherwise they will expire

# Module hierarchy for ERDP model



# Demo in CPN Tools

# ERDP module (most abstract view )



Three substitution transitions

Four packet buffers

- The colour set IPv6Pakcet is used to model packets sent between the gateway and the edge router.

# Colour sets for router solicitations

- IPv6 addresses:

```
colset IPv6Addr = string;
```

**IPv6 addresses are modelled as strings**

**This makes it possible to use both mnemonic names and standard hexadecimal notation**

- Router solicitations:

```
colset RSOption = union
                    RS_SrcLinkAddr : NDLinkAddrOption +
                    RS_PrefixInformation : NDPrefixInfoOption;

colset RSOptions = list RSOption;


colset RouterSolicitation = record Options : RSOptions *
                                   NU : NOTMOD;
```

**Protocol fields that do not affect the operation of ERDP are modelled using the colour set NOTMOD containing the single dummy value notmod**

# Colour sets for router advertisements

```
colset RAOption = union
                    RA_SrcLinkAddr : NDLinkAddrOption +
                    RA_MTU : NDMTUOption +
                    RA_PrefixInformation : NDPrefixInfoOption;
colset RAOptions = list RAOption;


colset RouterAdvertisement = record CurHopLimit : UInt8 *    } INT
                                              M : Bit *       } Bool
                                              O : Bit *
                                 RouterLifetime : UInt16 *
                                  ReachableTime : UInt32 *   } INT
                                   RetransTimer : UInt32 *
                                        Options : RAOptions;
```

# Colour sets for ICMP packets

- ERDP is based on the IPv6 Neighbour Discovery Protocol (NDP) which uses Internet Control Message Protocol (ICMP) packets.

```
colset ICMPBody = union RS : RouterSolicitation +
                        RA : RouterAdvertisement;

colset ICMPMessage = record    Type : UInt8 *
                               Code : UInt8 *
                            Message : ICMPBody;
```

**Two kinds of messages**

**Body**

# Colour sets for IPv6 packets

```
colset IPv6Payload = union ICMP : ICMPMessage;

colset IPv6Header = record Version : Bit4 *
                           TrafficClass  : NOTMOD *
                           Flowlabel     : NOTMOD *
                           PayloadLength : NOTMOD *
                           NextHeader    : Bit8 *
                           HopLimit      : Bit8 *
                           SourceAddress : IPv6Addr *
                           DestAddress   : IPv6Addr;

colset IPv6Packet = record Header     : IPv6Header *
                           ExtHeaders : NOTMOD *
                           Payload    : IPv6Payload;
```

Payload

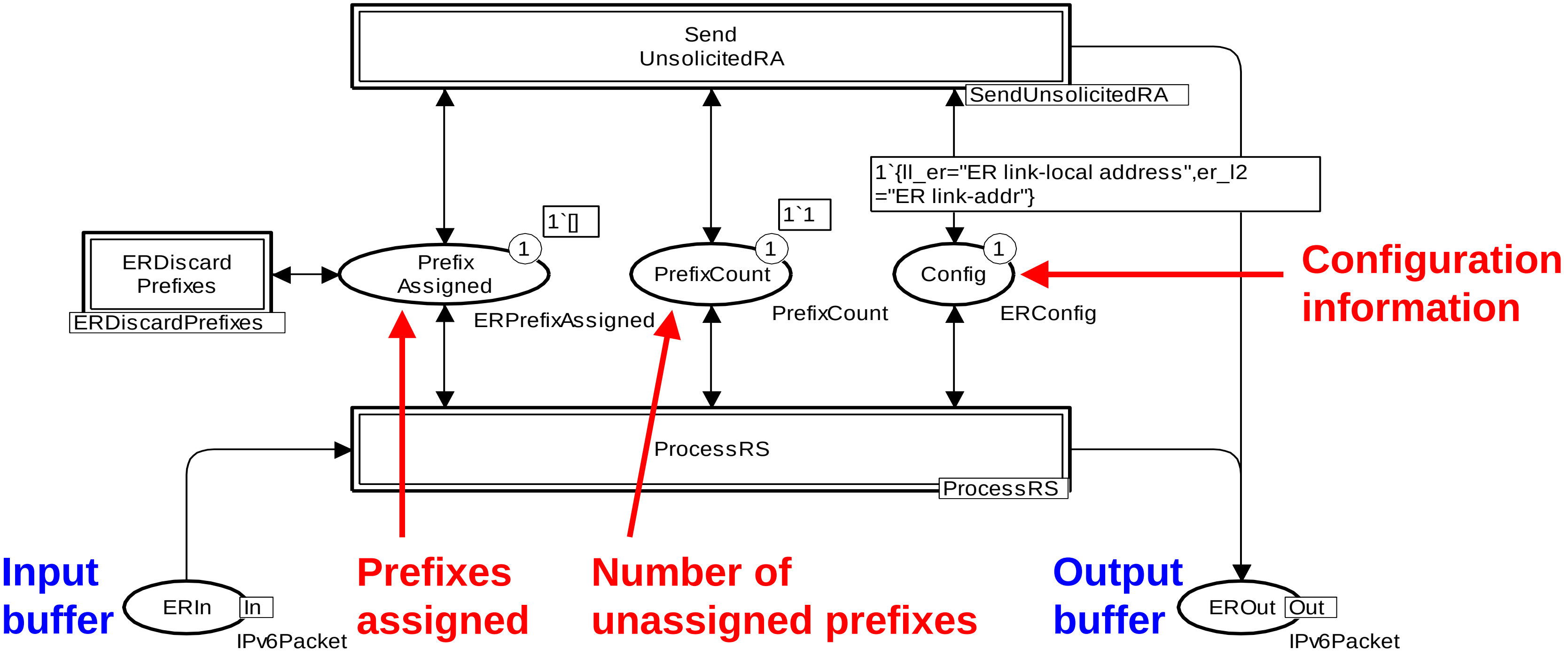
Header

Header

Payload

# EdgeRouter module

- Three substitution transitions:
  - Multi-cast of periodic unsolicited Router Advertisements.
  - Reception and processing of unicast Router Solicitations.
  - Expiration of prefixes on the edge router side.



Configuration information

Input buffer

Prefixes assigned

Number of unassigned prefixes

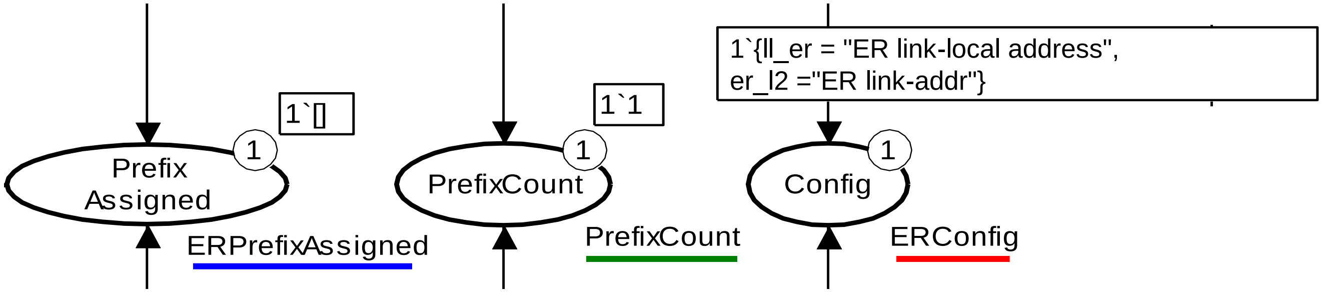
Output buffer

# Colour sets for EdgeRouter module

```
colset ERPrefixEntry    = product IPv6Addr * IPv6Prefix;
colset ERPrefixAssigned = list ERPrefixEntry;

colset PrefixCount = int;

colset LinkAddr = string;
colset ERConfig = record
                    ll_er : IPv6Addr *   (* link-local address  *)
                    er_l2 : LinkAddr;    (* link-addr (layer 2) *)
```



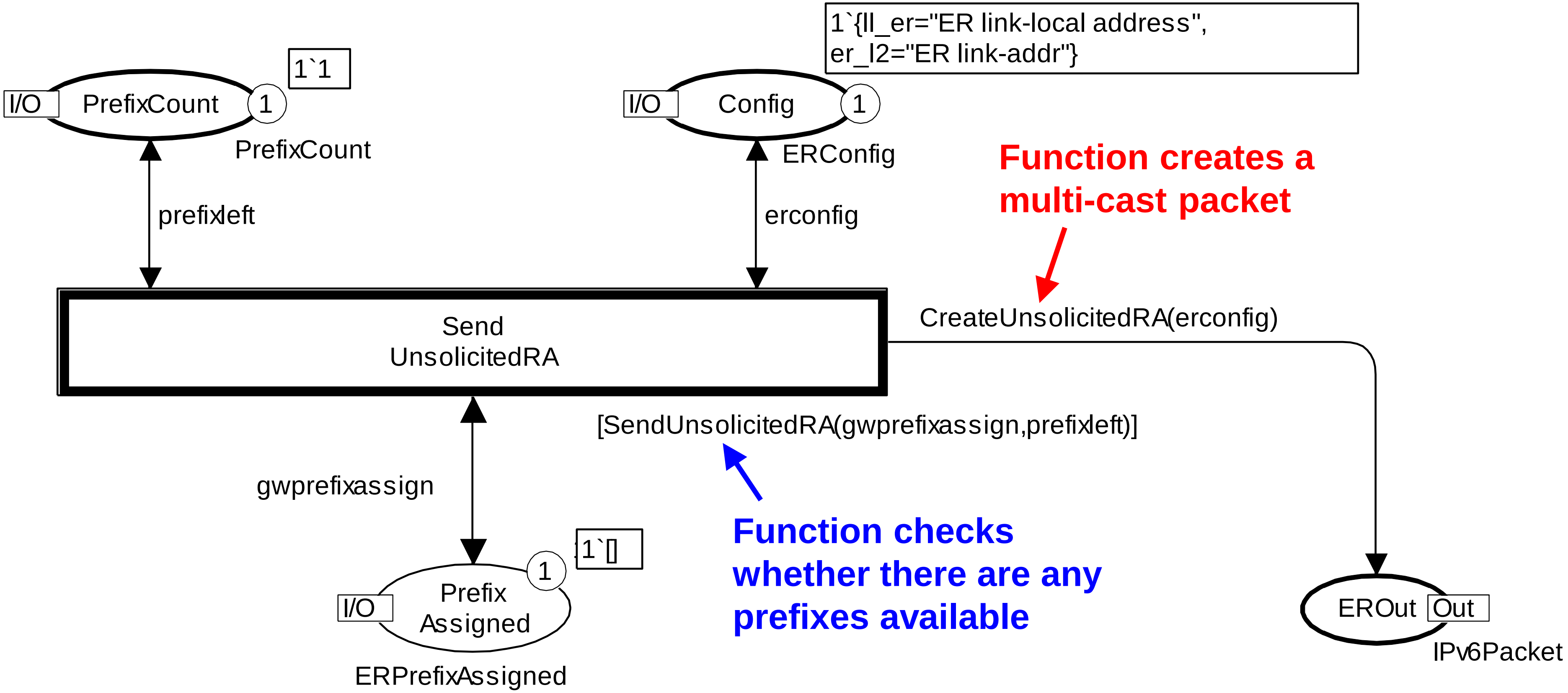
**Initial marking:**

**No prefixes assigned**

**One unassigned prefix**

**Link-local address**
**Link address**

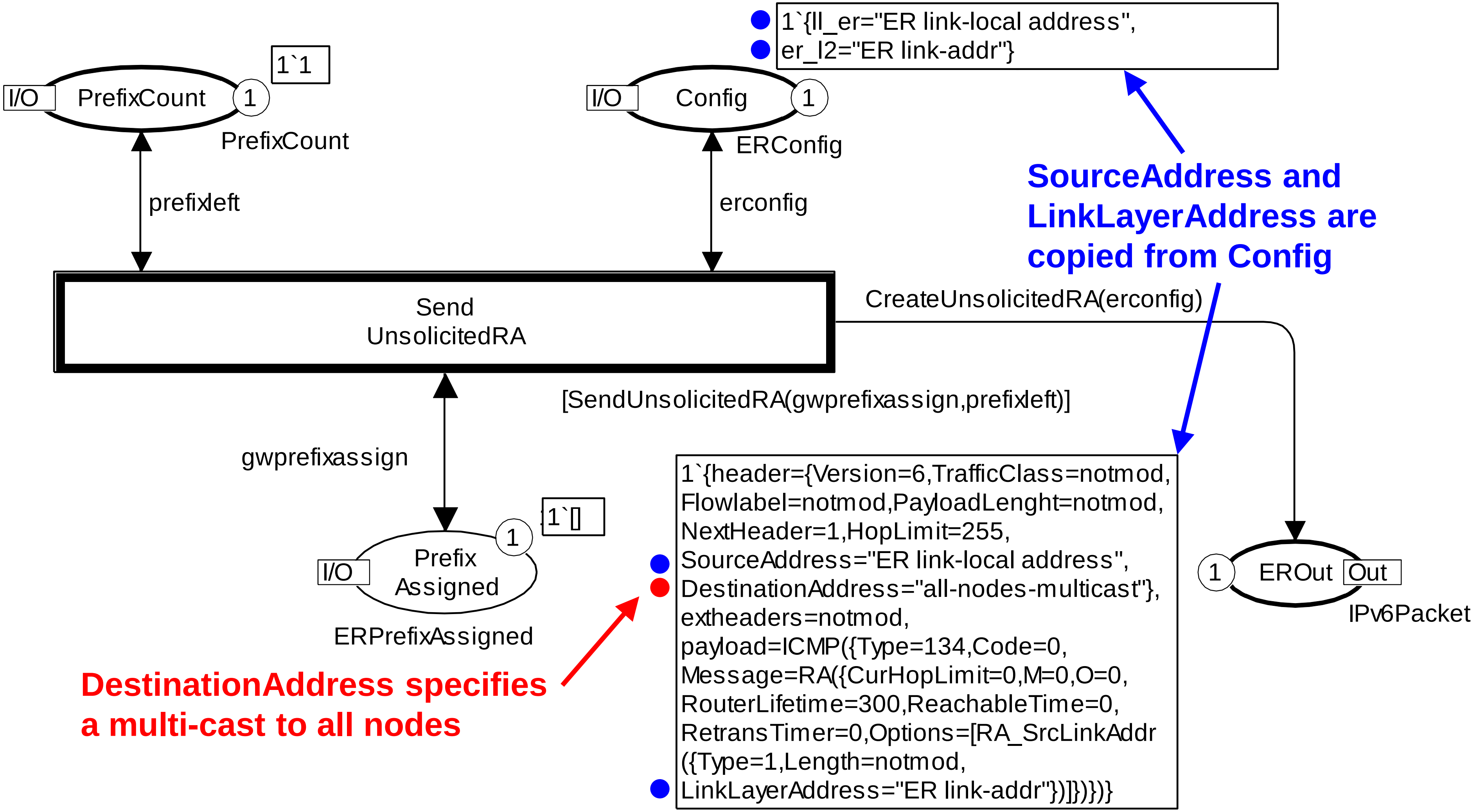
# SendUnsolicitedRA module



Function creates a multi-cast packet

Function checks whether there are any prefixes available

# Module after occurrence of transition



● 1`{ll_er="ER link-local address",
● er_l2="ER link-addr"}

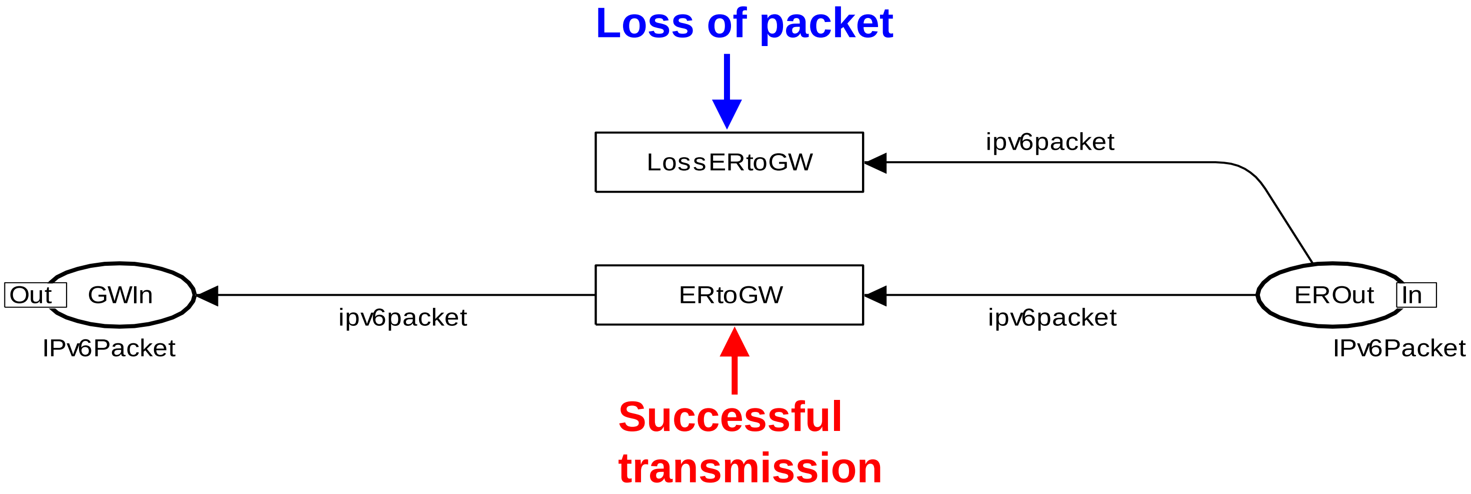
1`{header={Version=6,TrafficClass=notmod,
Flowlabel=notmod,PayloadLenght=notmod,
NextHeader=1,HopLimit=255,
● SourceAddress="ER link-local address",
● DestinationAddress="all-nodes-multicast"},
extheaders=notmod,
payload=ICMP({Type=134,Code=0,
Message=RA({CurHopLimit=0,M=0,O=0,
RouterLifetime=300,ReachableTime=0,
RetransTimer=0,Options=[RA_SrcLinkAddr
({Type=1,Length=notmod,
● LinkLayerAddress="ER link-addr"})]})})}

**SourceAddress and LinkLayerAddress are copied from Config**

**DestinationAddress specifies a multi-cast to all nodes**

# GW_ER_Link module

- Models the wireless communication link between edge router and gateway.



**Loss of packet**

**Successful transmission**

- Transmission in the other direction is modelled analogously (not shown).

# Development of CPN model

- The CPN model was developed:
  - in cooperation with protocol engineers at Ericsson Telebit,
  - in parallel with the development of the ERDP specification.

- 70 man-hours were spent on CPN modelling.

- Protocol developers were given a 6 hour course on the CPN modelling language.

- This enabled them to read and interpret the CPN models – which were used as basis for discussions of the protocol design.

# First round

- Development started with the creation of an initial ERDP specification (in natural language).
- Based on this, a first version of the CPN model was created.
- While creating the initial CPN model and discussing it (in Review 1) the group identified a number of:
  - design errors,
  - incompletenesses and ambiguities in the specification,
  - ideas for simplifications and improvements of the protocol design.

# Second and third round

- Based on the discoveries in Review 1, the ERDP specification and the CPN model were revised and extended.
- Review 2 identified a number of new issues to be resolved.

- Once more, the ERDP specification and the CPN model were revised and extended.
- In review 3, no further problems were discovered.

- Message sequence charts (MSCs) integrated with simulation was used to investigate the detailed behaviour of ERDP.
- This presented the operation of the protocol in a form which was well-known to the protocol developers.

# Problems identified

- The following number of issues were identified during:
  - construction of the CPN model,
  - single step execution of the CPN model,
  - discussions of the CPN model among the project group members.

| Category | Rev 1 | Rev 2 | Total |
|---|---|---|---|
| Errors in protocol specification/operation | 2 | 7 | 9 |
| Incompleteness and ambiguity in specification | 3 | 6 | 9 |
| Simplifications of protocol operation | 2 | 0 | 2 |
| Additions to the protocol operation | 4 | 0 | 4 |
| Total | 11 | 13 | 24 |

# Two complementary descriptions

- We used an iterative process involving a:
  - Conventional natural language specification.
  - CPN model.
- Both are required:
  - The implementers of the protocol are unlikely to be familiar with CP-nets.
  - Important parts of the ERDP specification are not reflected in the CPN model (such as the layout of packets).
- Construction of CPN models was a thorough and systematic way to review the protocol design.
- Effective way of integrating CPN technology into the development of a protocol.

# State space analysis

- State space analysis was pursued after the three iterations of modelling described above.
- The purpose was to conduct a more thorough investigation of the operation of ERDP, including verification of its key properties.

- The first step was to obtain a finite state space.
- The CPN model above can have an arbitrary number of tokens on the packet buffers.
- As an example, the edge router may send an arbitrary number of unsolicited router advertisements.

# Finite state space

- An upper integer bound of 1 was imposed on each of the four packet buffers (GWIn, GWOut, ERIn, and EROut).
- This also prevents overtaking among the packets transmitted across the wireless link.
- Furthermore, the number of tokens simultaneously on the four packet buffers was limited to 2.

- We used the branching options in the CPN state space tool to prevent the processing of enabled transitions which would violate the limitations.
- Alternatively, we could have modified the CPN model.

# Key properties to verify

- The key property of ERDP is the proper configuration of the gateway with prefixes:
- For a given prefix and state where the gateway has not yet been configured with that prefix, the protocol must be able to configure the gateway with the prefix.
- The edge router and the gateway should be consistently configured, i.e., the assignment of a prefix must be recorded in both entities.

# One prefix, no loss, no expiration

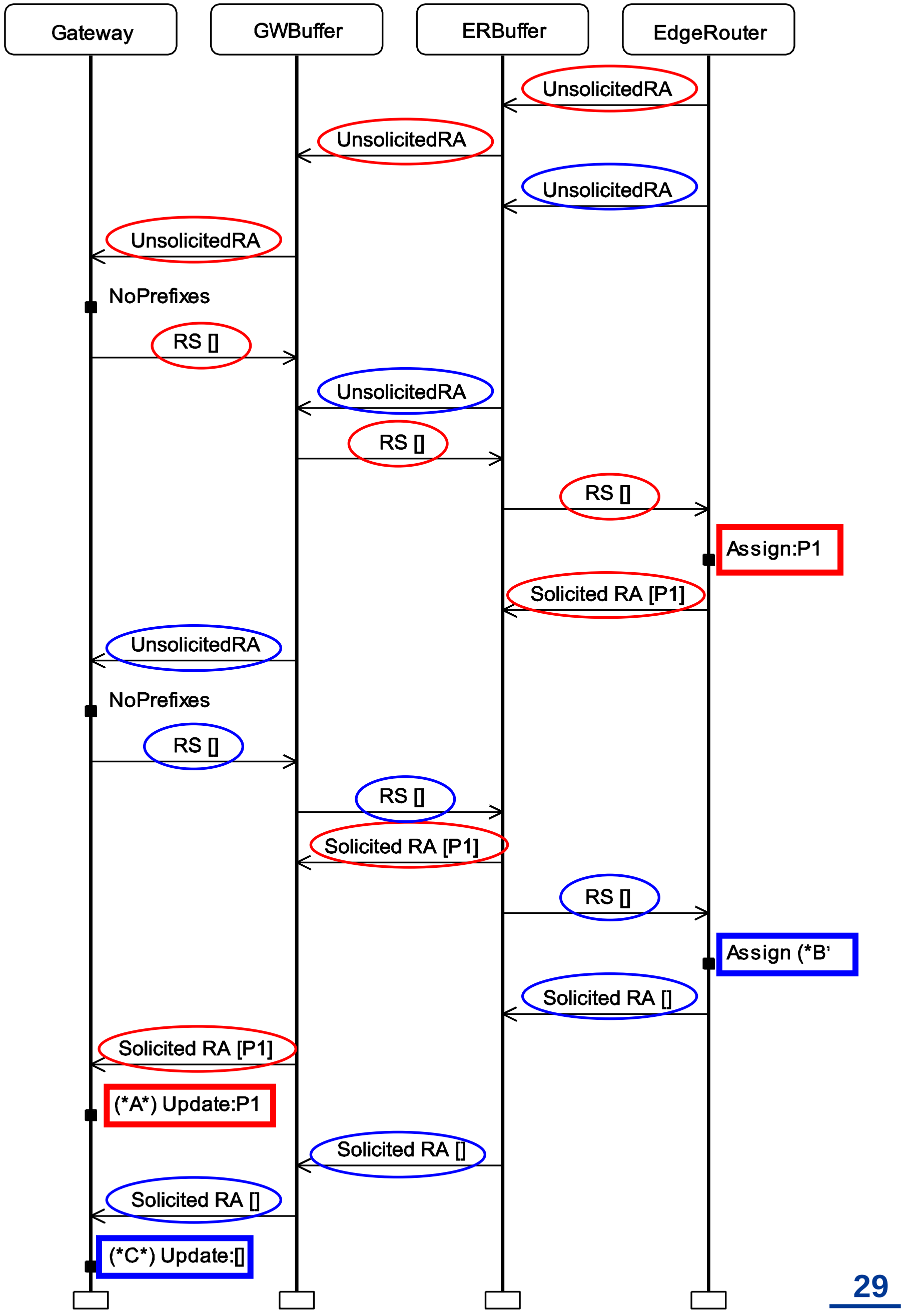
- State space: 46 nodes and 65 arcs.
- SCC-graph: 36 nodes and 48 arcs.
- A single dead marking.

- Inspection shows that the dead marking is inconsistently configured.
  - The edge router has assigned a prefix to the gateway.
  - BUT, the gateway is not configured with the prefix.

- To locate the problem, query functions in the state space tool were used to obtain a shortest counter example leading from the initial marking to the dead marking.
- The error-trace was visualised by means of a message sequence chart.

# MSC for error trace

- The edge router sends two unsolicited RAs.
- The first one gets through and we obtain a consistent configuration with prefix P1.
- When the second reaches the edge router there are no unassigned prefixes available.
- A Solicited RA with the an empty list of prefixes is sent.
- The gateway updates its prefixes to be the empty list.



# One prefix, no loss, no expiration (rev)

- To fix the error, the protocol was modified such that the edge router always replies with the list of all prefixes that it has currently assigned to the gateway.
- State space: 34 nodes and 49 arcs.
- No dead markings.
- 11 home markings (constituting a single terminal SCC).
- Inspection shows that all home markings are consistently configured with the prefix. Hence we conclude:
  - It is always possible to reach a consistently configured state for the prefix.
  - When such a state has been reached, the protocol entities will remain consistently configured.

# One prefix, no loss, no expiration (rev)

- To verify that a consistently configured state will eventually be reached, it was checked that the single terminal SCC was the only non-trivial SCC.

- The protocol is not supposed to terminate.
- When the gateway is configured with a prefix it may (at any time) send a router solicitation back to the edge router to have its prefixes refreshed.
- Since we ignore expiration of prefixes, the edge router will always refresh the prefix.

# More prefixes, no loss, no expiration

- With more than one prefix, the edge router may (at any time) decide not to configure the gateway with additional prefixes.
- Hence, there is no guarantee to reach a state where all prefixes are configured.

- Instead it was verified that there is a single terminal SCC – where all markings are consistently configured for all prefixes.
- It is always possible to reach such a marking, after which all protocol entities will remain consistently configured.

- It was also checked that all markings in each non-trivial SCC represent states consistently configured for a subset of the prefixes.

# One prefix, loss, no expiration

- The next step was to allow packet loss on the wireless link between the edge routers and the gateway.

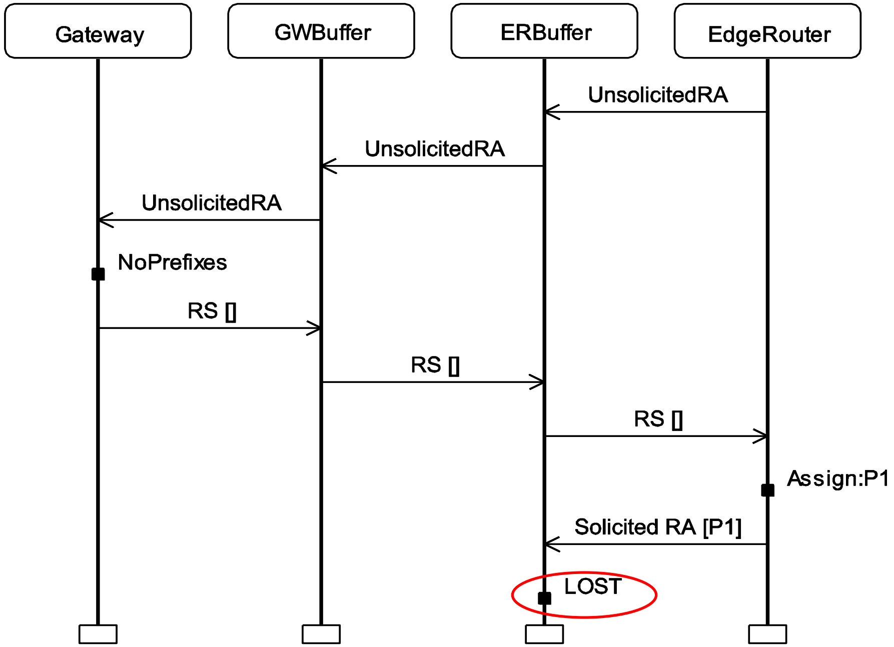
- State space: 40 nodes and 81 arcs.
- SCC-graph: 36 nodes and 48 arcs.

- A single dead marking (representing an undesired terminal state with inconsistent configuration).

- To locate the problem, an error trace was found and visualised by means of a message sequence chart.

# MSC for error trace

- The solicited RA containing the prefix is lost.
- The edge router has assigned its last prefix and is no longer sending any unsolicited RAs.
- There are no timeouts to trigger retransmission of the prefix to the gateway.



- The problem was fixed by ensuring that the edge router will resend an unsolicited RA to the gateway as long as it has prefixes assigned to the gateway.

# One prefix, loss, no expiration (rev)

- State space: 68 nodes and 160 arcs.
- No dead markings and no home markings.

- Two terminal SCCs each containing 20 markings.
  - In one of them, all markings are consistently configured (with the single prefix P1).
  - In the other, all markings are inconsistently configured.

- An error trace was obtained, the problem fixed and a new state space produced.

- This time there was only one terminal SCC (containing 20 consistently configured markings).

# One prefix, loss, no expiration (rev)

- With packet loss there is no guarantee that the two protocol entities will eventually be consistently configured – since any number of packets can be lost on the wireless link.

- Each of the non-trivial SCCs were inspected using a user-defined query to investigate the circumstances under which the protocol entities would not eventually be consistently configured.

- It was concluded that the absence of reaching a consistently configured state is due to packet loss and nothing else. If only finitely many packets are lost, a consistently configured state for some prefix will eventually be reached.

# One prefix, loss, expiration

- State space: 173 nodes and 513 arcs.
- A single dead marking where:
  - The edge router has no further prefixes to distribute and no prefixes recorded for the gateway.
  - The gateway is not configured with any prefix.
- This dead marking is OK – as we expect prefixes to eventually expire.
- The single dead marking was also a home marking. The protocol can always enter the expected terminal state.

# One prefix, loss, expiration

- When prefixes can expire there is no guarantee to reach a consistently configured state.
- This is because a prefix may expire in the edge router before the gateway has been configured with it.
- We can prove that for any state where a prefix still is available in the edge router, it is possible to reach a consistently configured state with this prefix.

# Iteration from simple to more complex

- Good idea to begin state space analysis from the simplest possible configuration and then gradually lift the assumptions.
- As the assumptions are relaxed, the size of the state spaces grows.

- For the ERDP protocol we did not encounter state explosion.
- The key properties could be verified for the number of prefixes that are envisioned to appear in practice.

# Statistics for state space analysis

| \|P\| | No loss/No expire | | Loss/No expire | | Loss/Expire | |
|---|---|---|---|---|---|---|
| 1 | 34 | 49 | 68 | 160 | 173 | 531 |
| 2 | 72 | 121 | 172 | 425 | 714 | 2,404 |
| 3 | 110 | 193 | 337 | 851 | 2,147 | 7,562 |
| 4 | 148 | 265 | 582 | 1,489 | 5,390 | 19,516 |
| 5 | 186 | 337 | 926 | 2,390 | 11,907 | 43,976 |
| 6 | 224 | 409 | 1,388 | 3,605 | 23,905 | 89,654 |
| 7 | 262 | 481 | 1,987 | 5,185 | 44,450 | 169,169 |
| 8 | 300 | 553 | 2,742 | 7,181 | 78,211 | 300,072 |
| 9 | 338 | 625 | 3,672 | 9,644 | 130,732 | 505,992 |
| 10 | 376 | 697 | 4,796 | 12,625 | 209,732 | 817,903 |

- When a state space has been generated, the verification of the key properties can be done in a few seconds.

# State spaces cover all cases

- Without state space analysis, the inconsistent configurations would probably not have been discovered until a first implementation of ERDP was operational.
- To discover these problems you need to consider subtle execution sequences of the protocol, and there are too many of these to do it manually.
- The state space analysis covers all execution sequences in a systematic way.

# Conclusions from ERDP project

- The application of CPN technology in the development of ERDP was successful.
- The CPN modelling language and computer tools were powerful enough to handle a real-world communication protocol and could easily be integrated in the conventional protocol development process.
- Modelling, simulation and state space analysis identified several non-trivial design problems which otherwise might not have been discovered until implementation/test/deployment.
- Only 100 man-hours were used for CPN modelling and analysis. This is a relatively small investment compared to the many problems that were identified and resolved early in the development.

# Second industrial project: Requirements engineering at Systematic

- Specification of workflows (business processes) at Aarhus County Hospital and their support by a new Pervasive Health Care IT System.

- Behavioural visualisation driven by a CPN model was used to engineer requirements through discussions with nurses and doctors who were not familiar with the CPN modelling language.

- This provided valuable input for the system requirements.

# Pervasive health care system

- The aim of the Pervasive Health Care System (PHCS) is to improve the Electronic Patient Record (EPR) deployed at hospitals in Aarhus, Denmark.

- EPR is a huge IT system with a budget of approximately 15 million US dollars. It will eventually have 8,000-10,000 users.

- EPR solves obvious problems with paper-based patient records such as:
  - being not always up-to-date,
  - only present in one location at a time,
  - misplaced and sometimes even lost.

# Electronic patient record

- The EPR system is based on desktop PCs. This induces at least two problems for the users:
- **Immobility**
  An electronic patient record accessed only from desktop PCs is difficult/impossible to transport.
- **Time-consuming login and navigation**
  EPR requires login (to ensure data security), and to use the system for clinical work, a user must navigate (to find a specific document for a given patient).
- The problems are aggravated because nurses and doctors often are:
  - Away from their offices and hence their PCs.
  - Interrupted during their work.

# Possible solutions

- In the ideal world, users should have access to the IT system wherever they need it, and it should be easy to resume an interrupted work process.

- Use of personal digital assistants (PDAs) is a possible solution to the immobility problem.
- Unfortunately, this creates new problems due to the small screens and limited memory.
- Moreover it does not fully solve the time-consuming login and navigation problems.

- PHCS is a more ambitious solution which takes advantage of the possibilities of modern pervasive computing.

# Basic principles of PHCS

- The system is **context-aware**.
  Nurses, patients, beds, medicine trays, and other items are equipped with radio frequency identity (RFID) tags, enabling the presence of such items to be detected automatically by nearby computers.

- The system makes **qualified guesses**.
  As an example, detection of a nurse or medicine tray may result in automatic generation of buttons in the task-bar of a computer (for easy navigation).

- The system is **non-intrusive**.
  It does not interfere or interrupt hospital work processes in an undesired way. As an example, nurses may use the buttons in a task bar (by clicking on them), but they may also completely ignore the buttons.

# User interface (simplified)

Nurse Jane Brown is present in the medicine room

The medicine tray for patient, Tom Smith, stands close to the computer.

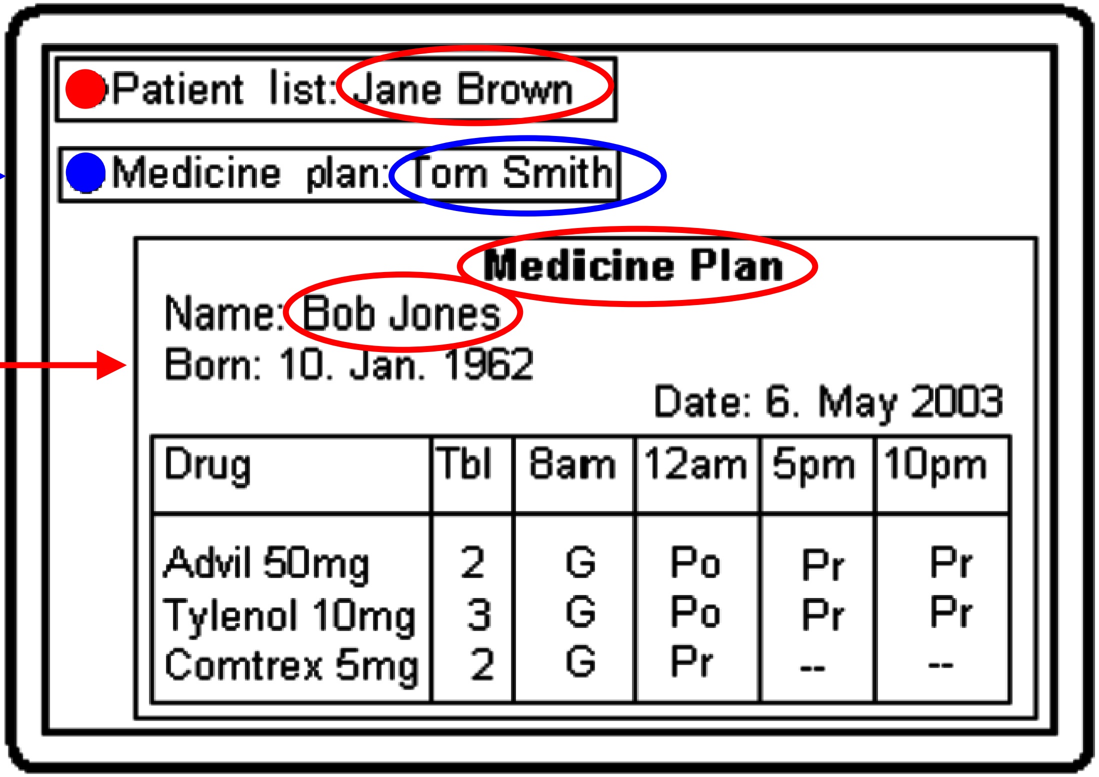
Jane Brown is pouring medicine for patient Bob Jones to be given at 12 a.m.

The medicine plan shows which medicine has been:

– prescribed (Pr),

– poured (Po),

– given (G).



● Patient list: Jane Brown

● Medicine plan: Tom Smith

**Medicine Plan**

Name: Bob Jones

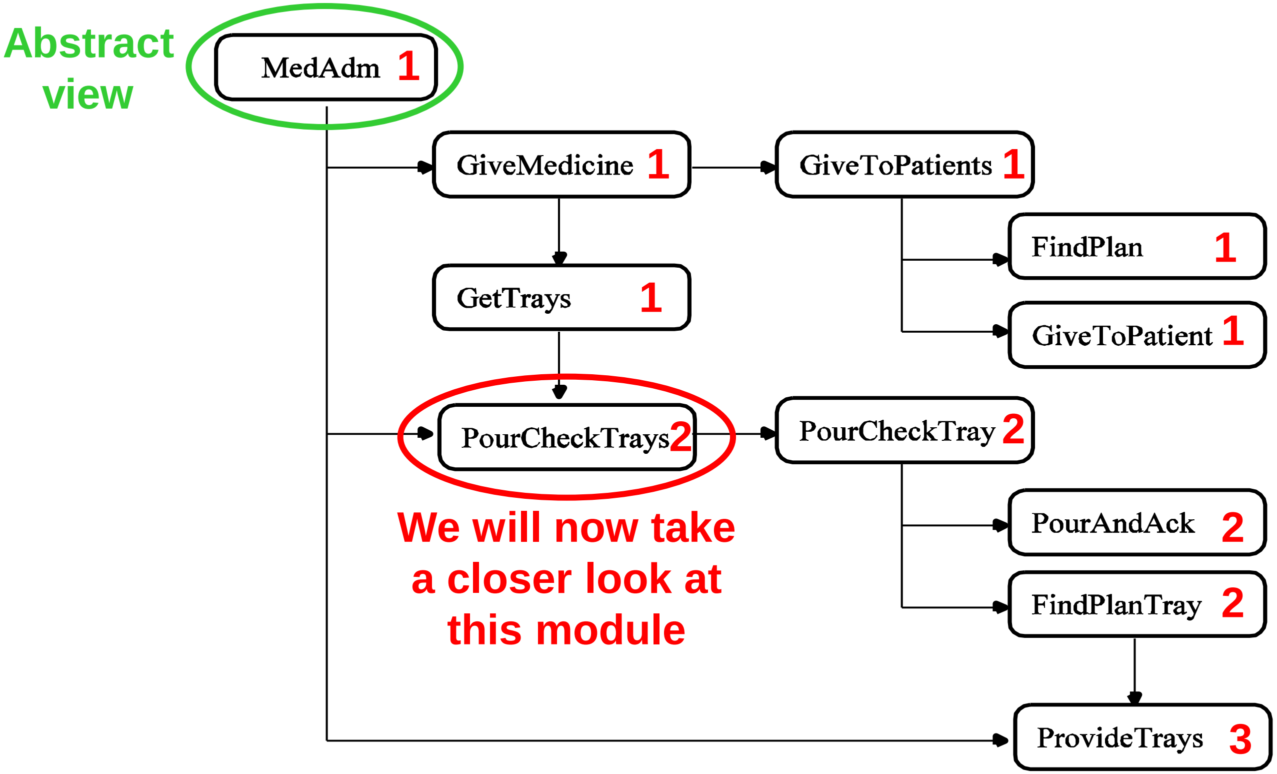
Born: 10. Jan. 1962

Date: 6. May 2003

| Drug | Tbl | 8am | 12am | 5pm | 10pm |
|---|---|---|---|---|---|
| Advil 50mg | 2 | G | Po | Pr | Pr |
| Tylenol 10mg | 3 | G | Po | Pr | Pr |
| Comtrex 5mg | 2 | G | Pr | -- | -- |

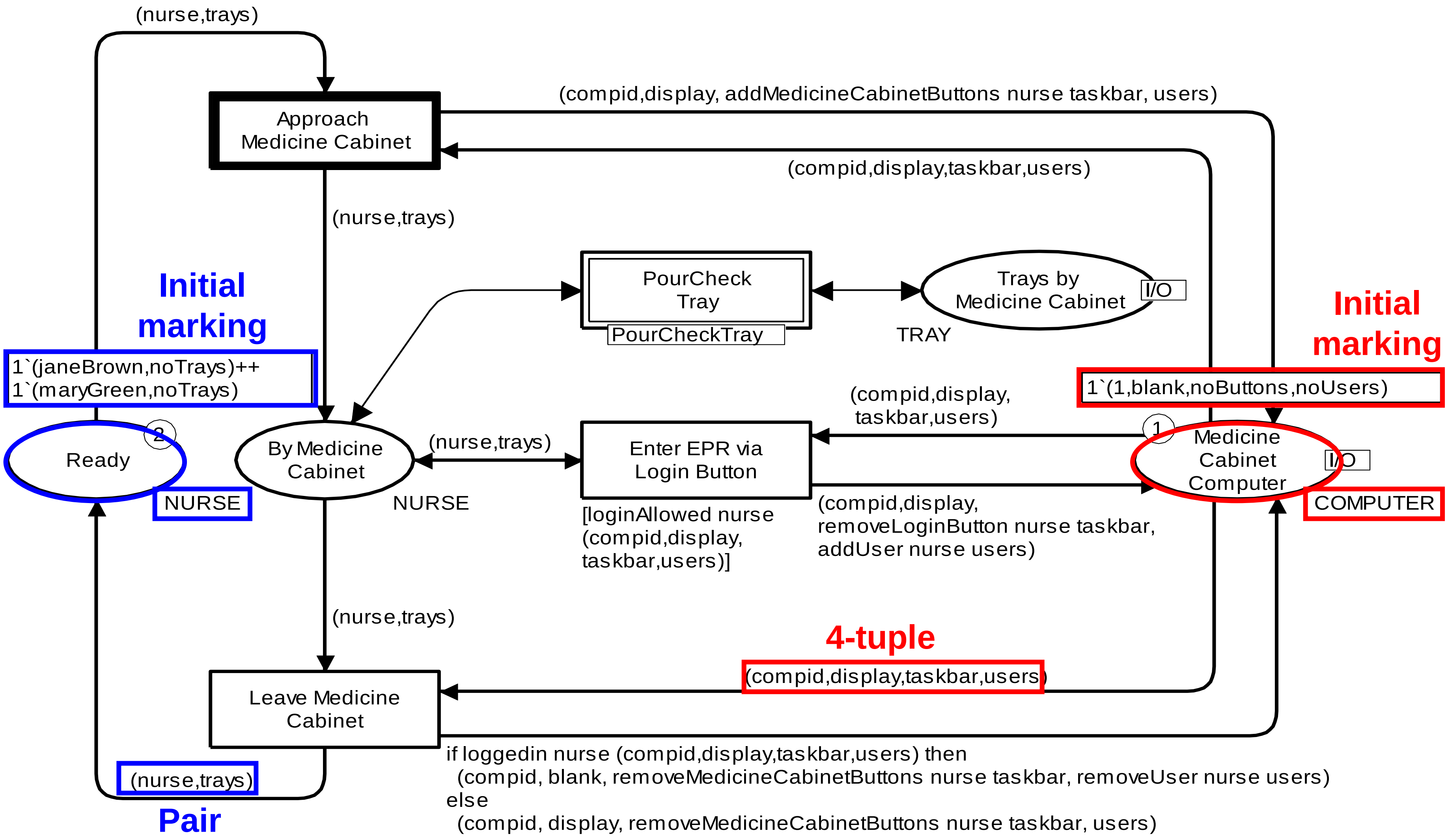
# Module hierarchy for PHCS model

- CPN model focus on the medicine administration work process.



- Complex graph structure.
- How many instances do we have of each module?

# PourCheckTrays module



(nurse,trays)

Approach Medicine Cabinet

(compid,display, addMedicineCabinetButtons nurse taskbar, users)

(compid,display,taskbar,users)

(nurse,trays)

PourCheck Tray

PourCheckTray

Trays by Medicine Cabinet

I/O

TRAY

Initial marking

1`(janeBrown,noTrays)++
1`(maryGreen,noTrays)

Initial marking

1`(1,blank,noButtons,noUsers)

(compid,display,
taskbar,users)

2

Ready

NURSE

By Medicine Cabinet

NURSE

(nurse,trays)

Enter EPR via Login Button

[loginAllowed nurse
(compid,display,
taskbar,users)]

1

Medicine Cabinet Computer

I/O

COMPUTER

(compid,display,
removeLoginButton nurse taskbar,
addUser nurse users)

(nurse,trays)

4-tuple

(compid,display,taskbar,users)

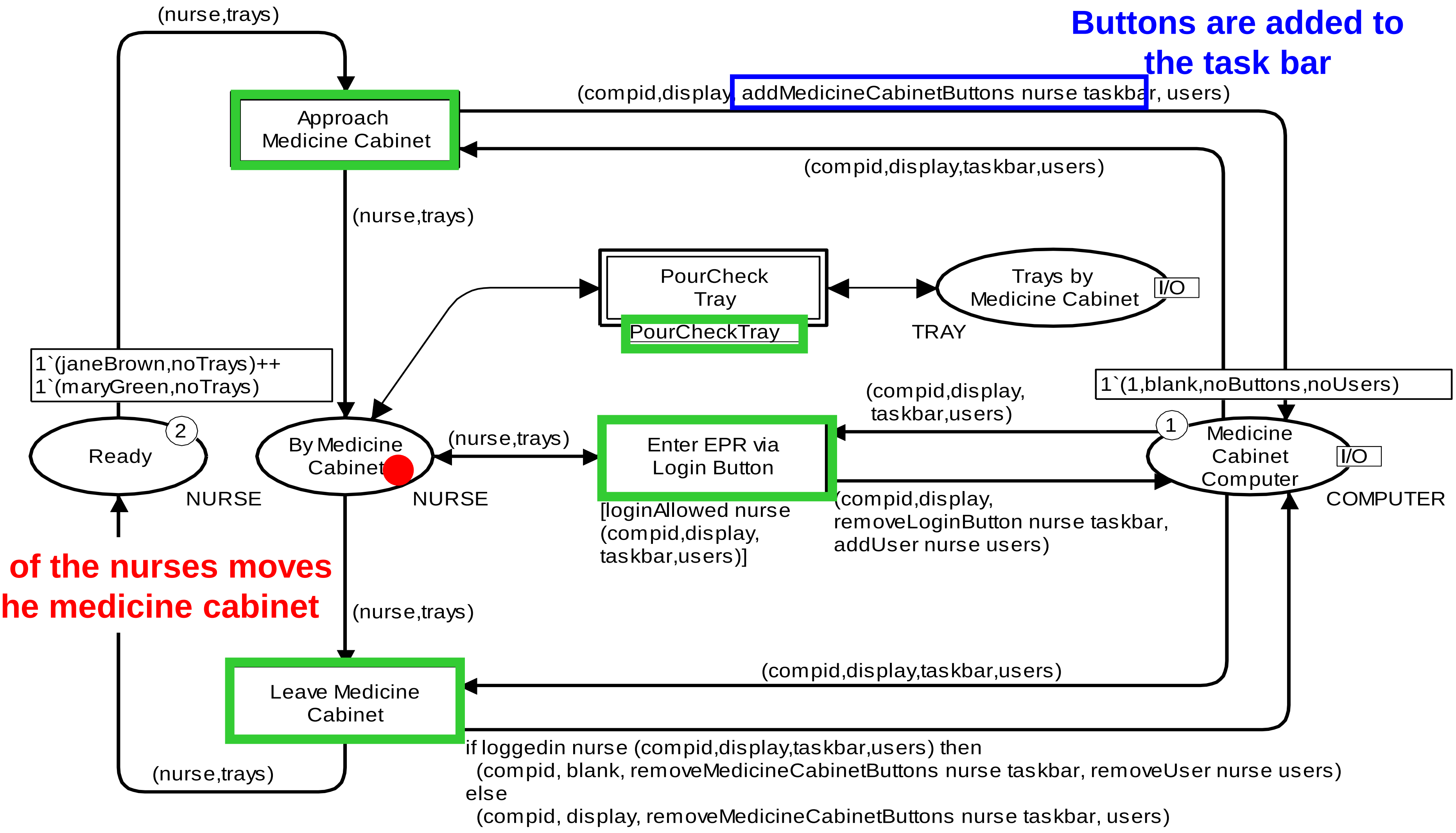
Leave Medicine Cabinet

if loggedin nurse (compid,display,taskbar,users) then
  (compid, blank, removeMedicineCabinetButtons nurse taskbar, removeUser nurse users)
else
  (compid, display, removeMedicineCabinetButtons nurse taskbar, users)

(nurse,trays)

Pair

# ApproachMedicineCabinet transition

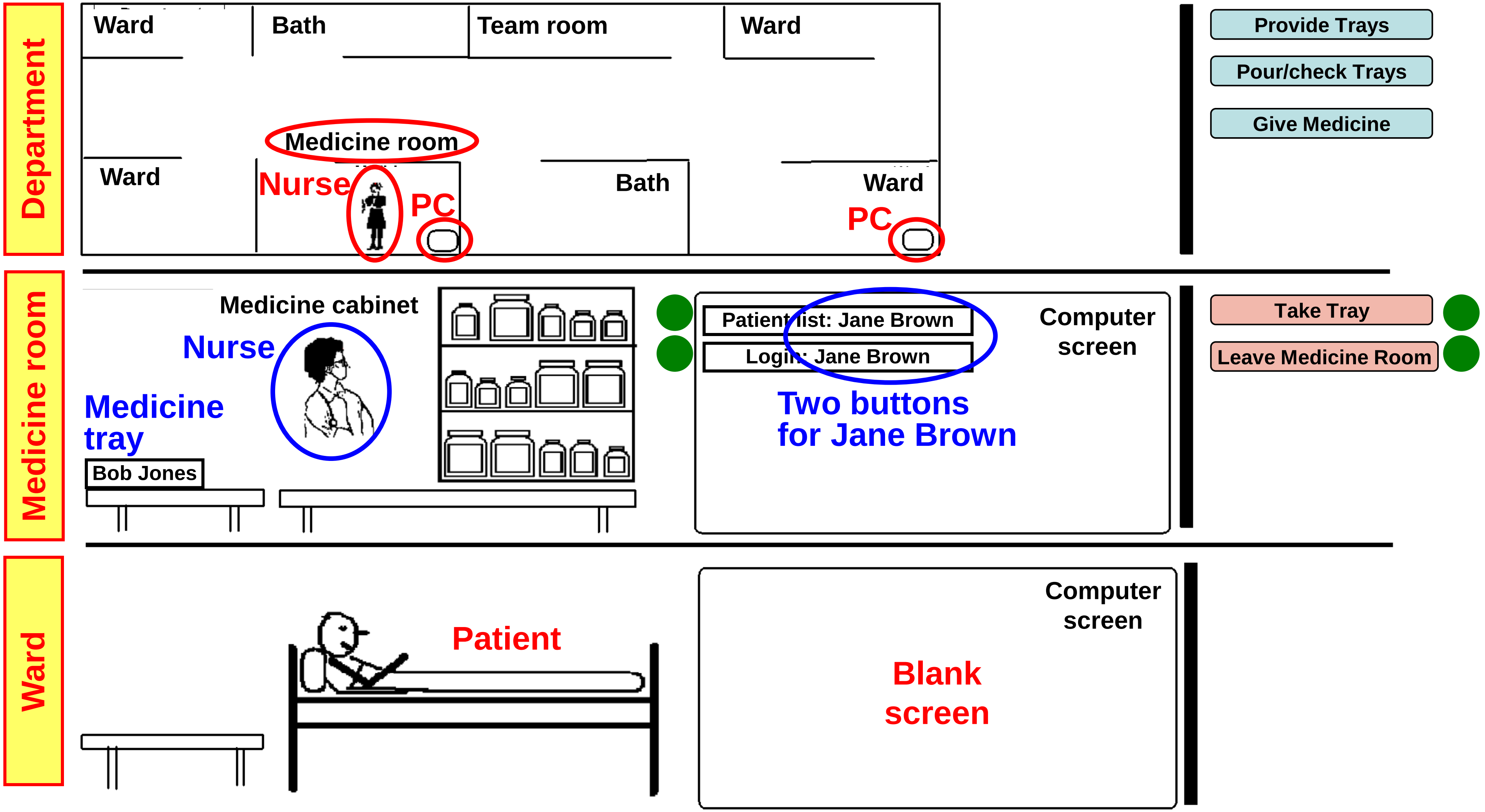

(nurse,trays)
Approach Medicine Cabinet
(compid,display, addMedicineCabinetButtons nurse taskbar, users)
(compid,display,taskbar,users)
(nurse,trays)
PourCheck Tray
PourCheckTray
Trays by Medicine Cabinet
I/O
TRAY
1`(janeBrown,noTrays)++
1`(maryGreen,noTrays)
(compid,display,
taskbar,users)
1`(1,blank,noButtons,noUsers)
2
Ready
NURSE
By Medicine Cabinet
NURSE
(nurse,trays)
Enter EPR via Login Button
1
Medicine Cabinet Computer
I/O
COMPUTER
[loginAllowed nurse
(compid,display,
taskbar,users)]
(compid,display,
removeLoginButton nurse taskbar,
addUser nurse users)
(nurse,trays)
(compid,display,taskbar,users)
Leave Medicine Cabinet
(nurse,trays)
if loggedin nurse (compid,display,taskbar,users) then
(compid, blank, removeMedicineCabinetButtons nurse taskbar, removeUser nurse users)
else
(compid, display, removeMedicineCabinetButtons nurse taskbar, users)

**Buttons are added to the task bar**

**One of the nurses moves to the medicine cabinet**

# Interaction graphics

User has four choices (corresponding to four enabled transitions in the CPN model)



# Project organisation

- The PHCS project started with:
  - Domain analysis in the form of ethnographic field work.
  - A series of vision workshops with participation of nurses, doctors, computer scientists, and an anthropologist.
- An outcome of this was natural-language descriptions of work processes and their proposed computer support.
- The first version of the CPN model was based on these prose descriptions.
- The CPN model and the interaction graphics were extended and modified in a number of iterations – each version based on feedback on the previous version.
- The interaction graphics allowed discussions in evaluation workshops with participation of nurses.

# CPN model with interaction graphics

- The CPN model and the interaction graphics were effective for:
- Specification of requirements.
  Requirements are specified by net-inscriptions of transitions modelling the manipulation of the involved computers.
- Analysis of requirements.
  Supported through trial-and-error simulations with visualisation of various scenarios for the envisioned work process.
- Discovery of new requirements.
  Interaction with the CPN model (which describe multiple scenarios) raised many new ideas and questions.
- Negotiation of requirements.
  CPN model constitutes a formal and unambiguous description of the requirements.

# Examples of requirements

- From transitions in the PourCheck module we get the following requirements:

- When a nurse enters the medicine room, the medicine computer must add a login button and a patient list button for the nurse (transition ApproachMedicineCabinet).

- When a logged-in nurse leaves the medicine room, the medicine computer must return to a blank display, remove her buttons from the task-bar, and log her out (transition LeaveMedicineCabinet).

- When a nurse clicks her login button, she must be added as a user of EPR, and the login button must be removed from the task-bar of the computer (transition EnterEPRviaLoginButton).

# Examples of analysis questions

- What happens if two nurses are both close to the medicine computer?
- The computer generates login buttons and patient list buttons for both of them.

- What happens when a nurse with several medicine trays approaches a bed?
- Only a medicine plan button for the patient in the bed is generated.

- Is it possible for a nurse to acknowledge pouring of medicine while another nurse simultaneously acknowledges giving of medicine for the same patient?
- No, that would require a more fine-grained concurrency control of the patient records.

# Revision of CPN model

- Questions and answers gave proposals/requests for changes to be made to the CPN model.

- An example:
  - In an early version of the CPN model, the leaving of any nurse from the medicine room resulted in the computer display being blanked off.
  - To be compliant with the non-intrusive design principle, the leaving of a nurse who is not logged in, should not disturb another nurse in the room working at the computer.
  - Hence the CPN model had to be changed accordingly.

# Negotiation of requirements

- In large projects, negotiation about requirements inevitably takes place during the project and may have strong economical consequences (since they are an essential part of a legal contract between the involved parties).
- It is important to be able to determine the requirements included in the initial agreement.
- If the parties agree that medicine administration should be supported and that the formal and unambiguous CPN model is the authoritative description, many disagreements can quickly be settled.

# Conclusions from PHCS project

- CPN models are able to support requirements engineering.
- The CPN model and the visualisation graphics was built "on top" of prose descriptions (of work processes and the intended computer support) – consolidated as a set of UML use cases.
- The stakeholders of PHCS were already familiar with these UML use cases via earlier work on EPR.
- The interaction graphics enabled users like nurses and doctors to be actively engaged in specification analysis – increasing the probability that a system is built that fits the future users' work processes.

# Third industrial project: Embedded system at Bang & Olufsen

- Concerned with the design and analysis of the BeoLink system which distributes audio and video sources (such as radios, CD/DVD players, and TVs) to different rooms via a dedicated network.

- A timed CPN model was developed for the lock management subsystem which is responsible for the basic synchronisation of devices in the BeoLink system.

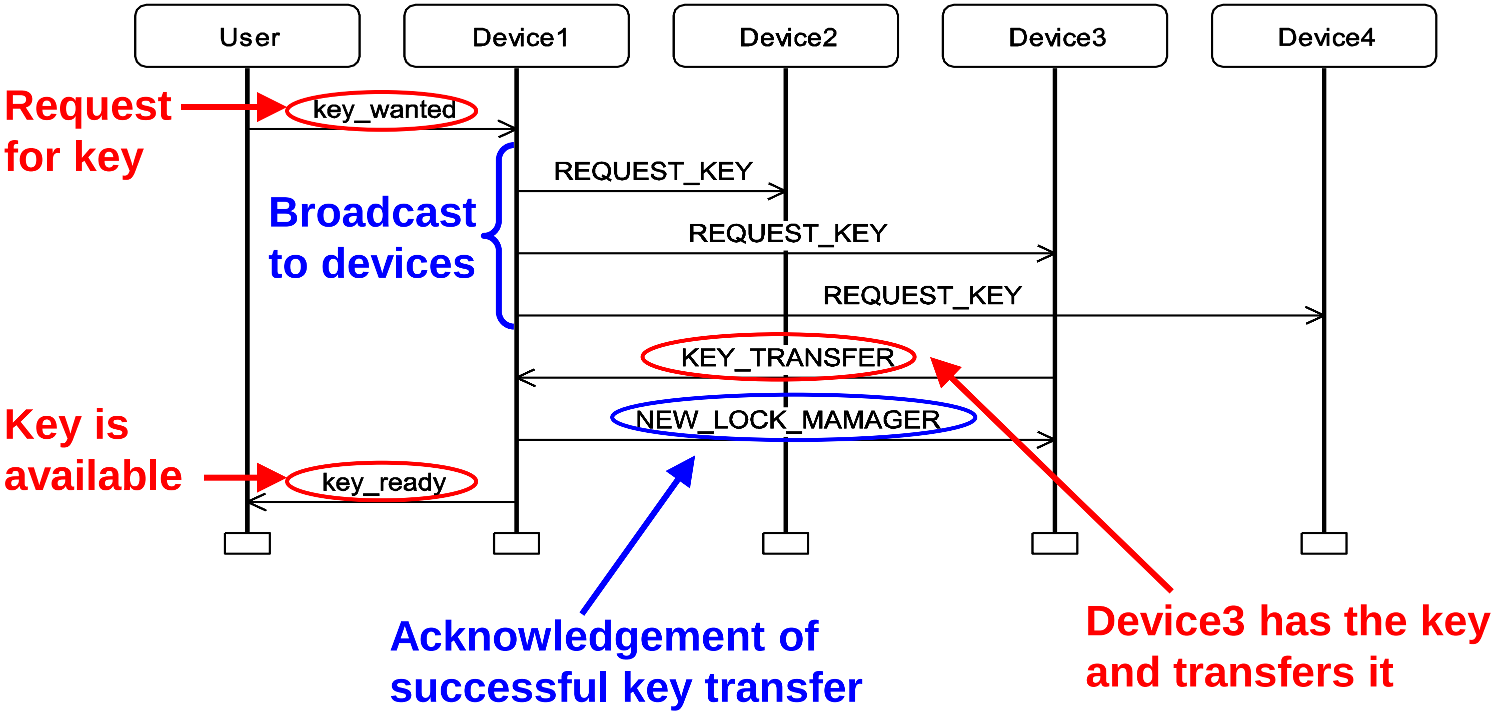
- State spaces (including a number of advanced state space methods) were used to verify the lock management system.

# Lock management protocol

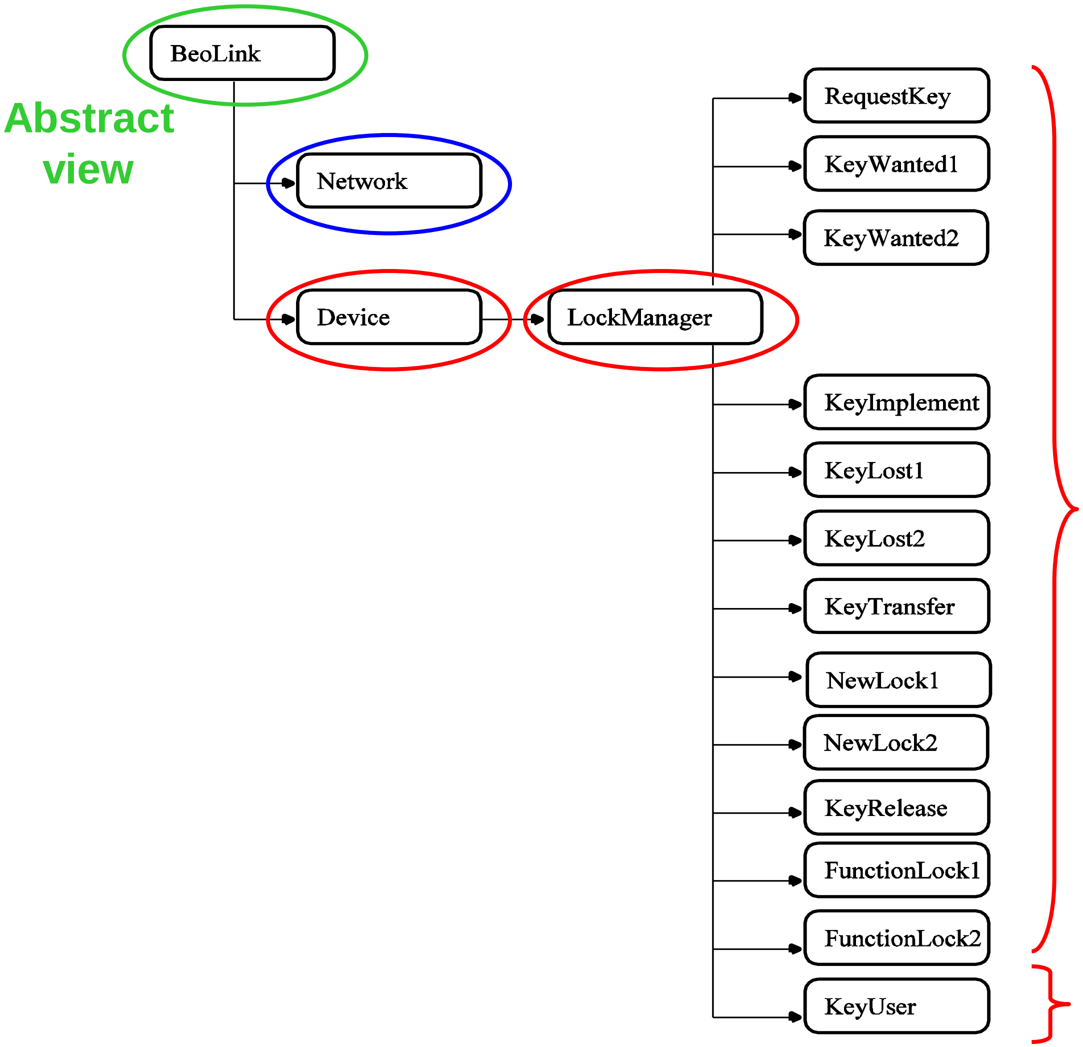
- The protocol is used to grant devices exclusive access to services in the system, such as being able to use the loud speakers.
- To access services in the system, a device is required to possess a key.
- When the system is switched on, exactly one key must be generated by the devices in the system and this must happen within 2 seconds.
- Special devices in the system called audio and video masters are responsible for generating the key.

# MSC for a typical scenario

- A user wish to change CD track on Device1.



# Module hierarchy for BeoLink model

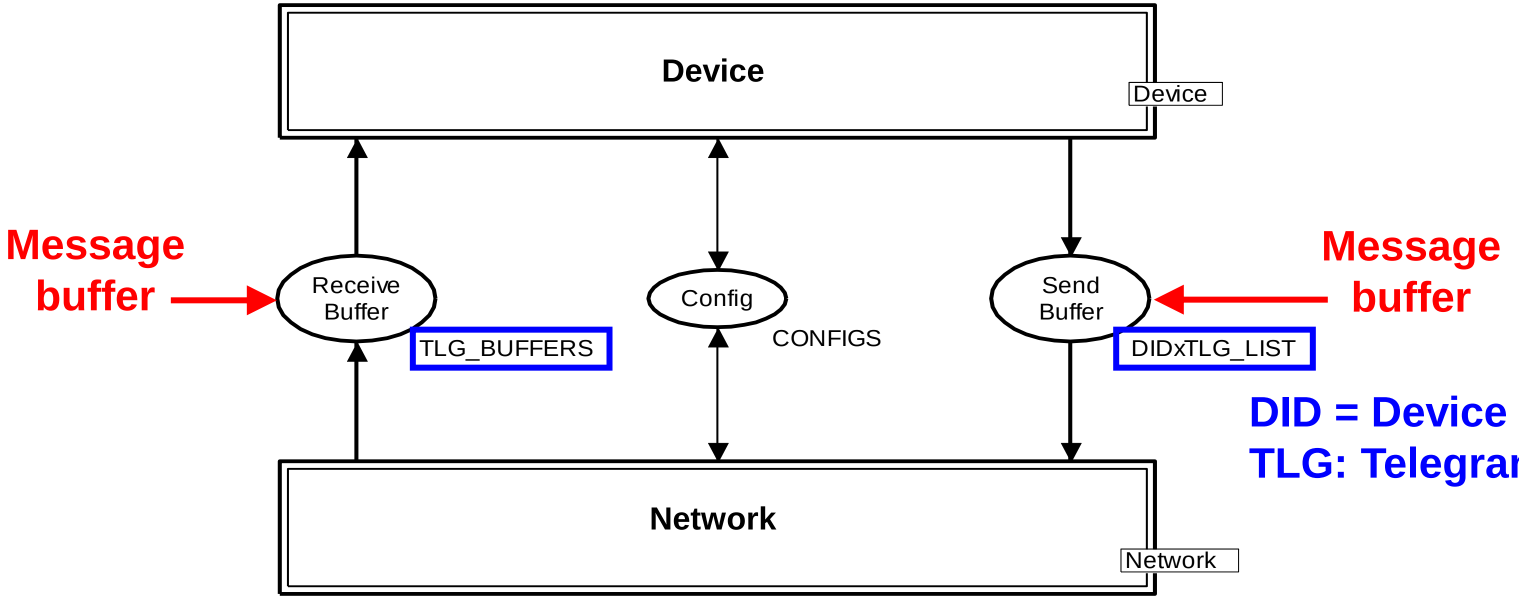


Different functional blocks in the lock management protocol

Behaviour of device in the lock management protocol

# BeoLink module (most abstract view)

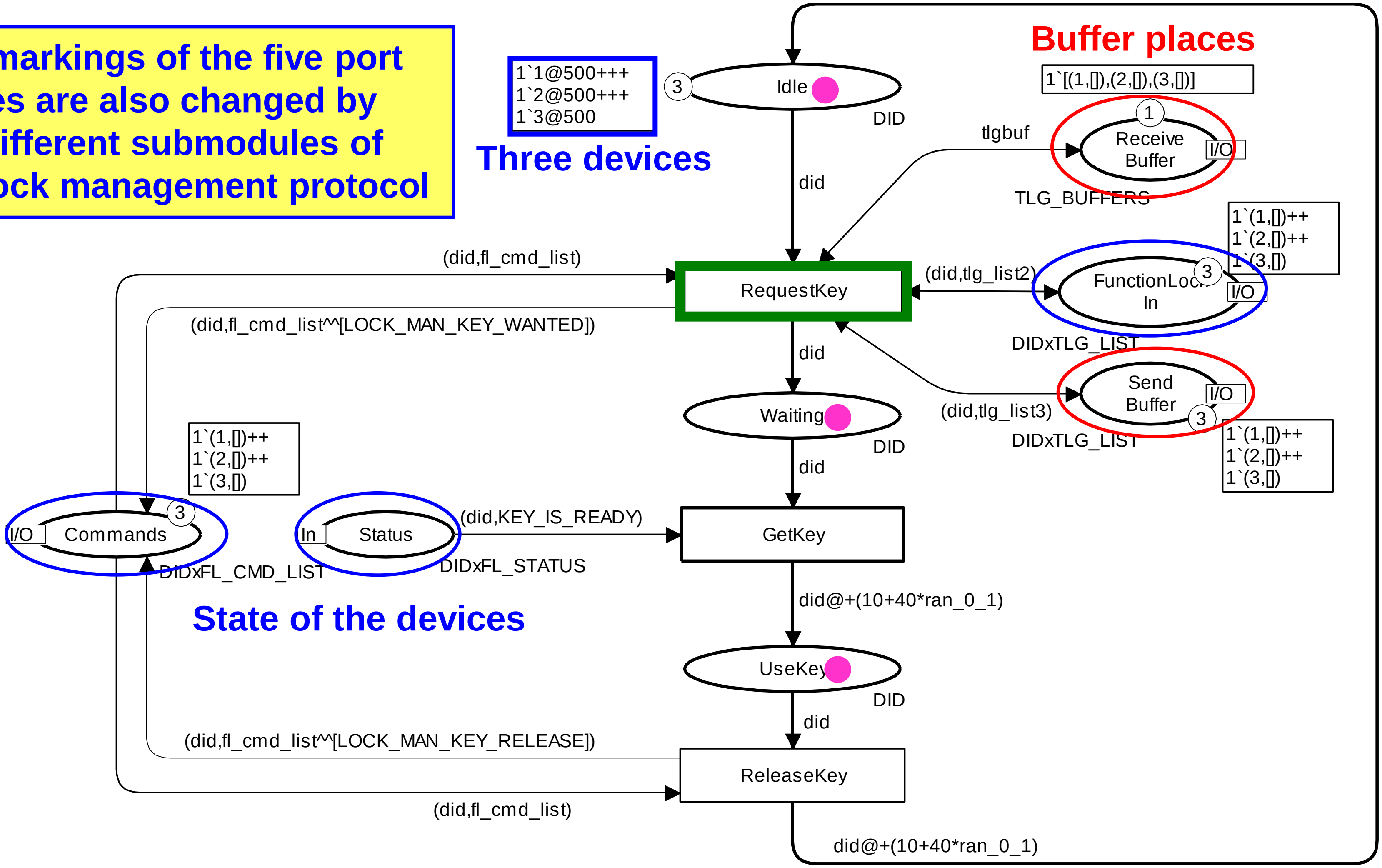
- The CPN model provides a folded representation of the devices by encoding the identity of devices in the token colours (as known from the protocol with multiple receivers).
- This makes the CPN model parametric and able to represent any number of devices.



DID = Device Identity
TLG: Telegram = Message

# KeyUser module (initial marking)

The markings of the five port places are also changed by the different submodules of the lock management protocol



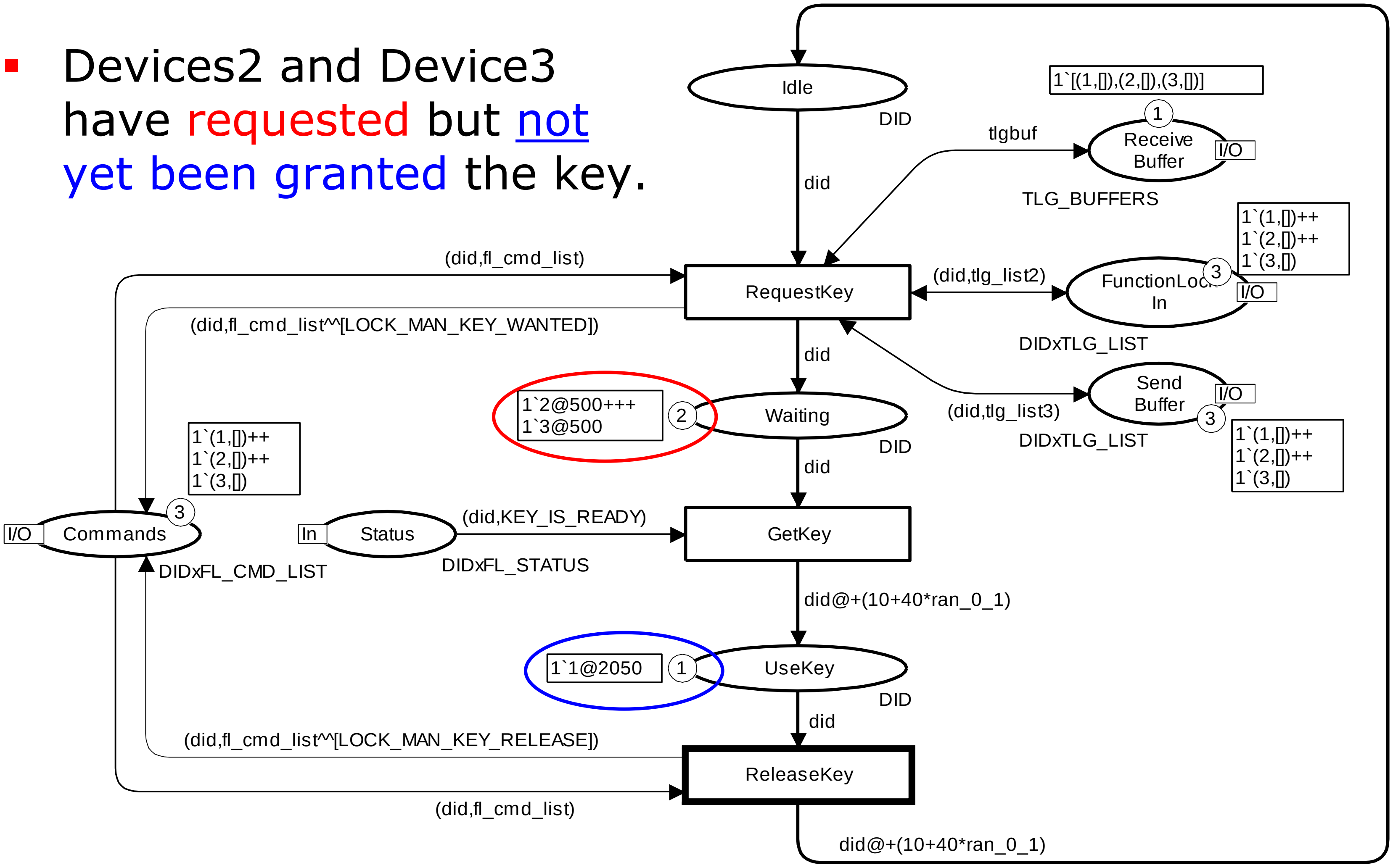
Three devices

Buffer places

State of the devices

# Device1 has obtained the key

- Devices2 and Device3 have requested but not yet been granted the key.



# Validation and verification

- First the CPN model was validated by means of simulation.
- Then state spaces were used to verify the following properties of the protocol:

- **Key generation**
  When the system is booted, a key is generated within 2.0 seconds.

- **Mutual exclusion**
  At any time at most one key exists.

- **Key access**
  Any given device always has the possibility of obtaining the key.

# State space for BeoLink model

- The state space is infinite because:
  - The system contains cycles.
  - We have an absolute notion of time (in the global clock and time stamps).

- As an example, consider the marking obtained when all devices have had the key once (and are back to Idle).

- This marking is similar to the initial marking but have different time stamps and a different value for the global clock.

- The two markings are different and they are represented by different state space nodes.

# Verification of key generation

- **Key generation**
  When the system is booted, a key is generated within 2.0 seconds.

- The property was verified by constructing a partial state space obtained by not generating successors for markings where:
  - the key had been generated, or
  - the model time had passed two seconds.

- It was then checked that in all markings for which successor markings had not been generated, a key was present in the system.

# State space for initialisation phase

- To save memory the arcs in the state space were not stored – since they are not needed for verifying the key generation property.

**One audio master**
**Total of 3-5 devices**

**One video master**
**Total of 3-5 devices**

| Config | Nodes |
|---|---:|
| AM:3 | 1,839 |
| AM:4 | 22,675 |
| AM:5 | 282,399 |
| VM:3 | 1,130 |
| VM:4 | 13,421 |
| VM:5 | 164,170 |

# State space for full BeoLink system

- Obtained by using the time equivalence method, which factors out the absolute notion of time.
- Whenever the underlying untimed CPN model is finite, the method yields a finite state space for the timed CPN model.

**One audio master**
**Total of 2-3 devices**

**One video master**
**Total of 2-3 devices**

| Config | Nodes |
|---|---:|
| AM:2 | 22,675 |
| AM:3 | 282,399 |
| VM:2 | 13,421 |
| VM:3 | 164,170 |

# Verification of the other properties

- Using the condensed state space it is now possible to verify the remaining two properties.

- **Mutual exclusion**
  At any time at most one key exists.

- We use PredAllNodes to verify that place UseKey never has more than one token.

- **Key access**
  Any given device always has the possibility of obtaining the key.

- We use HomePred to verify that it is always possible to reach a marking in which the device is represented by a token on place UseKey.

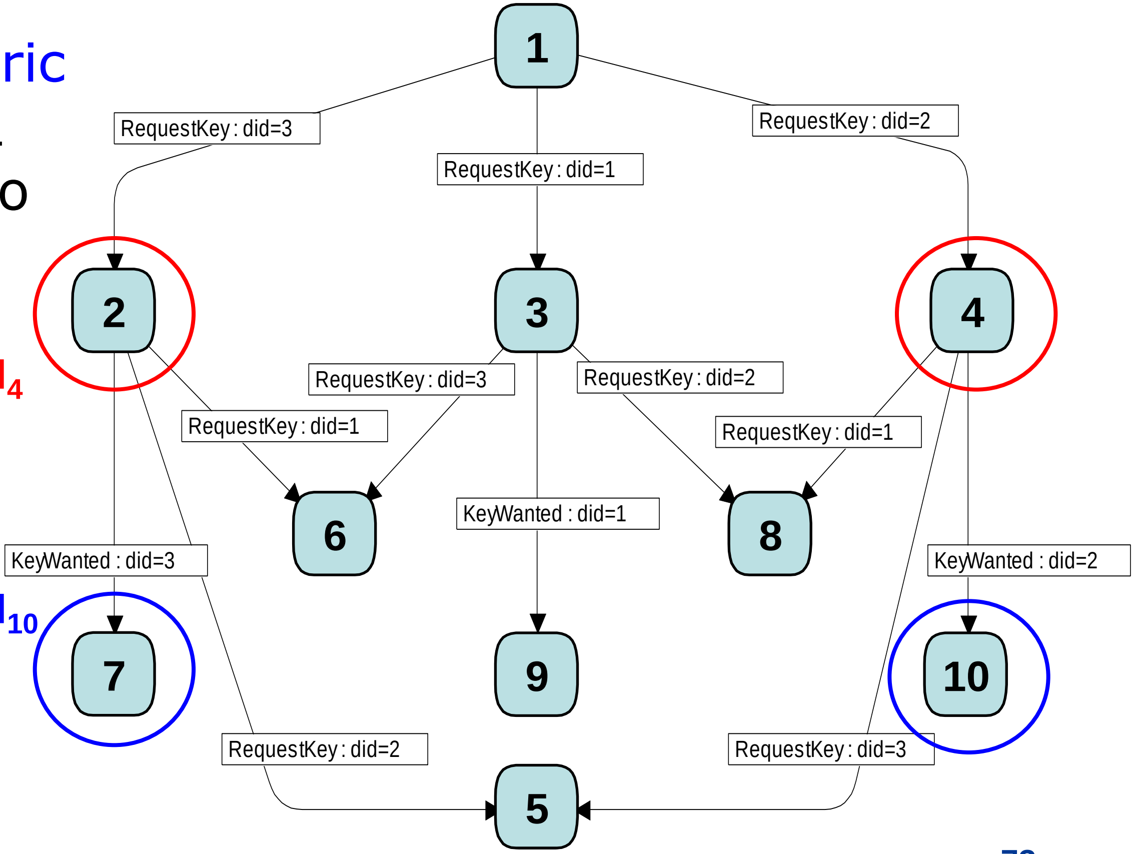
# Symmetry method

- To be able to generate state spaces for more devices we applied the symmetry method representing symmetric markings and symmetric binding elements by means of equivalence classes.

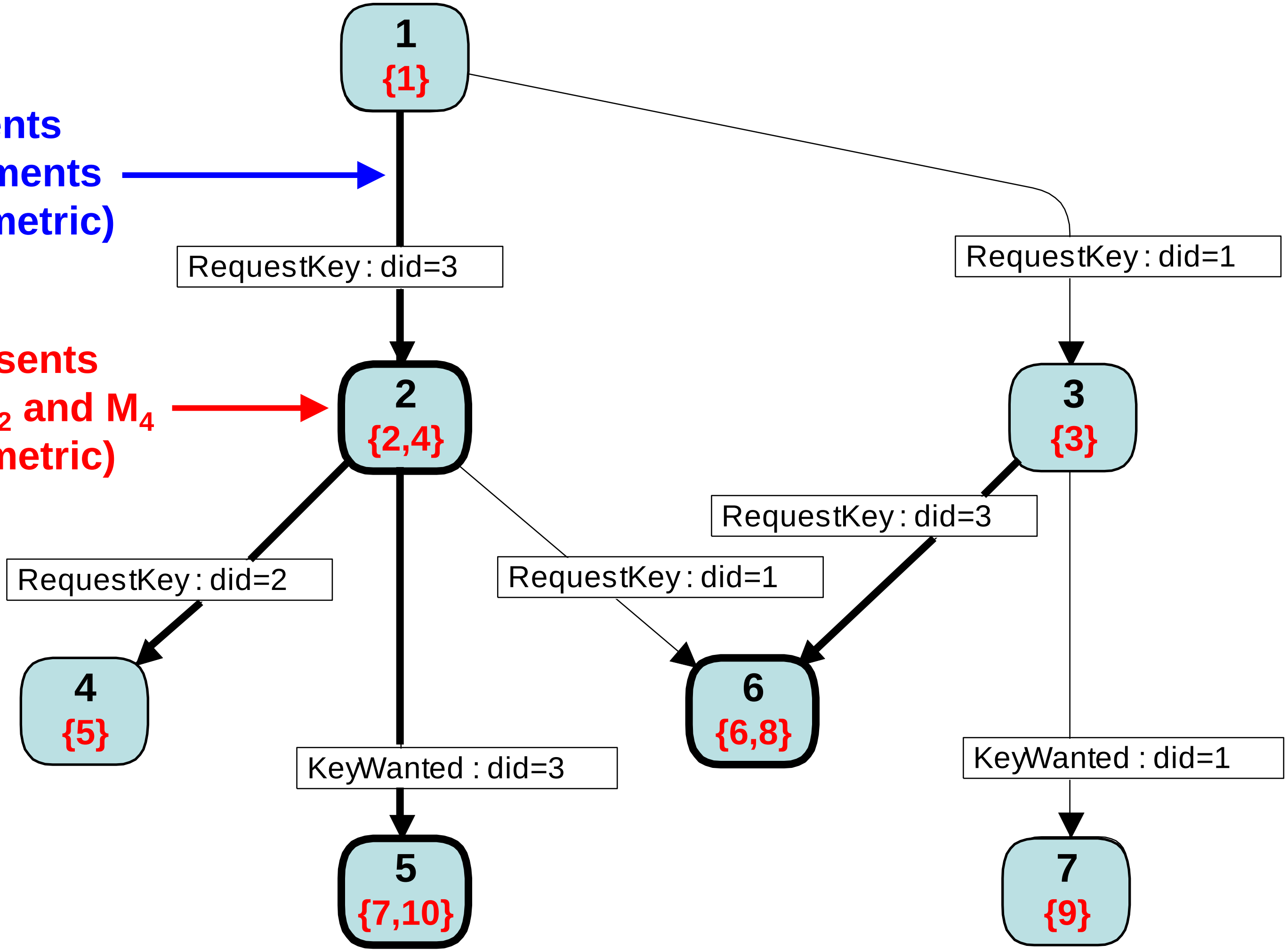
- All devices are symmetric (except for the Device1 which is the audio/video master).

**$M_2$ can be obtained from $M_4$ by swapping the identities of Device2 and Device3**

**$M_7$ can be obtained from $M_{10}$ by swapping the identities of Device2 and Device3**



# State space for symmetry method



This arc represents two binding elements (which are symmetric)

This node represents two markings $M_2$ and $M_4$ (which are symmetric)

# Symmetry method for initialisation

| Config | State Space Nodes | Symmetry Nodes | Node Ratio | Time Ratio | (n-1)! |
|---|---:|---:|---:|---:|---:|
| AM:3 | 1,839 | 968 | 1.9 | 1.0 | 2 |
| AM:4 | 22,675 | 4,361 | 5.2 | 2.5 | 6 |
| AM:5 | 282,399 | 15,865 | 17.8 | 10.0 | 24 |
| AM:6 | 3,417,719 | 47,867 | 71.4 | — | 120 |
| VM:3 | 1,130 | 594 | 1.9 | 1.0 | 2 |
| VM:4 | 13,421 | 2,631 | 5.1 | 2.5 | 6 |
| VM:5 | 164,170 | 9,328 | 17.6 | 10.0 | 24 |
| VM:6 | 1,967,159 | 27,551 | 71.4 | — | 120 |
| VM:7 | 22,892,208 | 68,683 | 333,3 | — | 720 |

# Symmetry method for full system

| Config | Time Equiv Nodes | Sym+TimeEquiv Nodes | Node Ratio | Time Ratio | (n-1)! |
|---|---|---|---|---|---|
| AM:3 | 27,246 | 13,650 | 1.9 | 2.0 | 2 |
| AM:4 | 12,422,637 | 2,074,580 | 5.9 | — | 6 |
| VM:3 | 10,713 | 5,420 | 2.0 | 2.0 | 2 |
| VM:4 | 3,557,441 | 594,092 | 6.0 | — | 6 |

# Sweep-line method

- Next we used the sweep-line method in which we have a progress measure.
- This allows us to explore all reachable markings, while only storing small fragments of the state space in memory at a time – thereby reducing peak memory usage.
- The sweep-line method is aimed at on-the-fly verification of safety properties, e.g., determining whether a reachable marking exists satisfying a given state predicate.
- Hence, it can be used to verify key generation and mutual exclusion but not key access.

# Progress measure

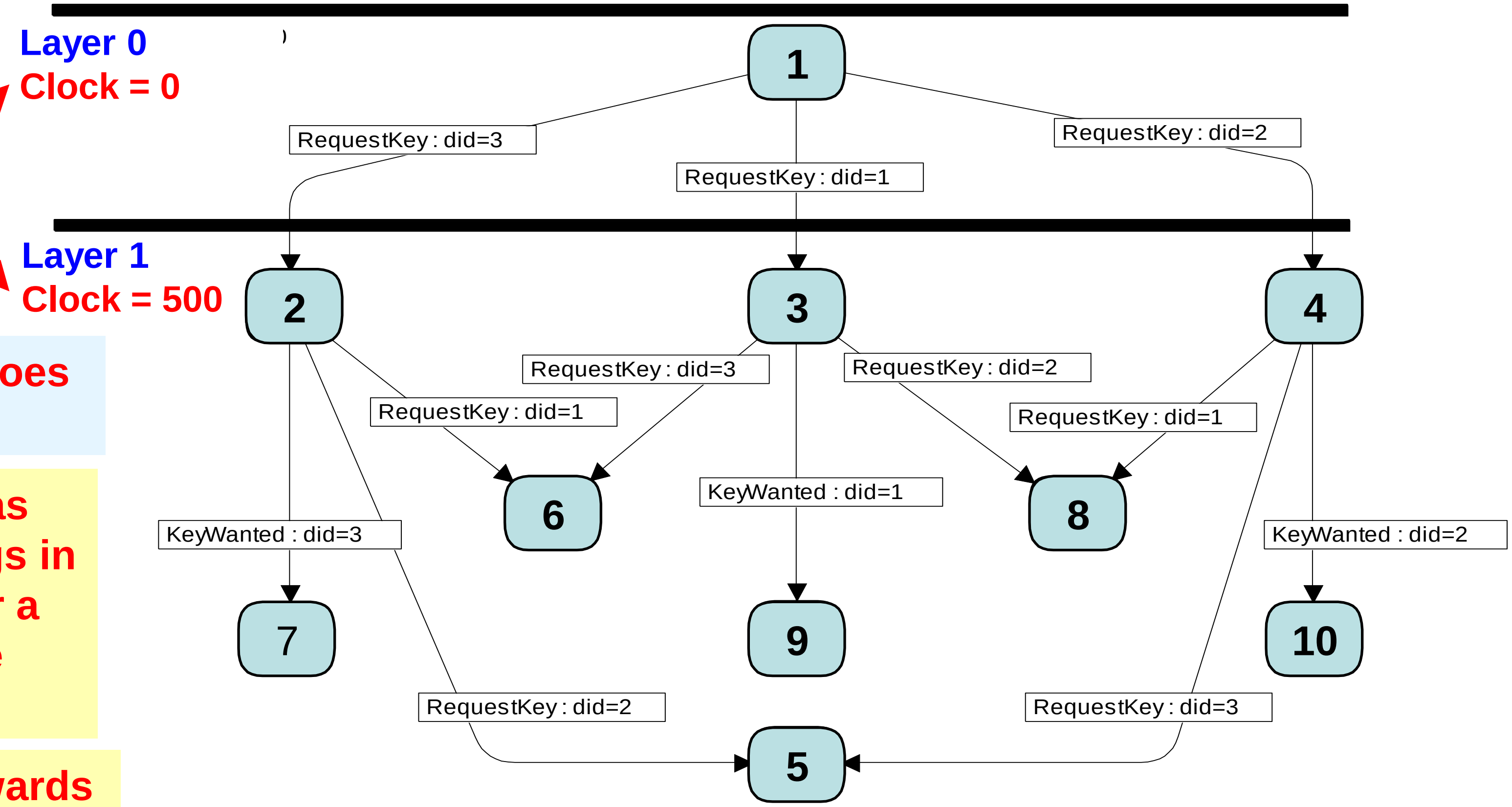
- We use the global clock as progress measure.

All markings in a layer have the same clock value

The clock never goes backwards

Each marking has successor markings in the same layer or a layer with more progress

All arcs go downwards or horizontal



# Sweep-line method for initialisation

| Config | State Space Nodes | Sweep-line Peak nodes | Node Ratio | Time Ratio |
|---|---|---|---|---|
| AM:3 | 1,839 | 1,839 | 1.0 | 1.0 |
| AM:4 | 22,675 | 5,169 | 4.4 | 1.2 |
| AM:5 | 282,399 | 35,017 | 8.1 | 2.5 |
| VM:3 | 1,130 | 1,130 | 1.0 | 1.0 |
| VM:4 | 13,421 | 5,167 | 2.6 | 0.9 |
| VM:5 | 164,170 | 34,968 | 4.7 | 2.2 |

# Sweep-line method for full system

- To apply the sweep-line method for the full BeoLink system we need to combine it with the time equivalence method (otherwise the state space will be infinite).

- The use of the time equivalence method implies that the global clock becomes zero in all markings (and hence we cannot use the clock as progress measure).

- It is however possible to define a progress measure based on the control flow of the devices and use this with the generalised sweep-line method in which some regress arcs (backwards arcs) are allowed.

# Sweep-line method for full system

| Config | Time Equiv<br>Nodes | Sweep-line + Time Equiv<br>Nodes Explored | <br>Peak Nodes | Node<br>Ratio | Time<br>Ratio |
|---|---|---|---|---|---|
| AM:2 | 346 | 355 | 65 | 5.3 | 0.5 |
| AM:3 | 27,246 | 28,363 | 2,643 | 10.3 | 0.3 |
| VM:2 | 274 | 283 | 41 | 6.7 | 0.5 |
| VM:3 | 10,713 | 11,388 | 1,039 | 10.3 | 0.5 |

**The time penalty was due to an inefficient implementation of deletion of states in the sweep-line library.**

**A more efficient algorithm has now been developed**

# Combination of advanced state space methods

- Above we have seen that it is possible to combine time condensed state spaces with both the symmetry method and the sweep-line method.
- It is also possible to use the symmetry method and the sweep-line method together.
- In all the combinations we get a better reduction than when one method is used in isolation.

# Conclusions from BeoLink project

- CP-nets can be used to model and validate a real-time system (in which the correctness depends on timing information).
- The construction of the CPN model was done in close cooperation with engineers at Bang & Olufsen.
- The engineers were given a four day course on CP-nets enabling them to construct large parts of the CPN model.
- Using advanced state space methods, we could verify larger configurations (sometimes all configurations that are expected to appear in practice).
- The advanced state space methods can be combined to get better reduction than either method in isolation.

# Fourth industrial project: Scheduling at Australian defence

- Development of a scheduling tool (called COAST).
- CPN modelling was used to conceptualise and formalise the planning domain to be supported by the tool.
- A CPN model was extracted in executable form from CPN Tools and embedded into the COAST server together with a number of tailored state space analysis algorithms.
- We bridged the gap between the design (specified as a CPN model) and the implementation of the system.

# Plans and task schedules

- A plan (also called a course of action) is a set of tasks.
- The COAST tool supports development and analysis of military plans and their task schedules.
- CPN is used to model execution of tasks according to their pre- and postconditions, the imposed synchronisations, and the available resources.
- Possible task schedules are obtained by generating a state space and extracting paths from it.

# Planning framework

- Tasks are the basic units in a plan and have associated preconditions, effects, resources.
- Conditions are used to describe the logical dependencies between tasks via preconditions and effects.
- Resources (such as planes, ships, and personnel) are used by tasks during their execution. They may be available only at certain times and may be lost.
- Synchronisations tell that a set of tasks must begin or end simultaneously or that there has to be a specific amount of time between them.

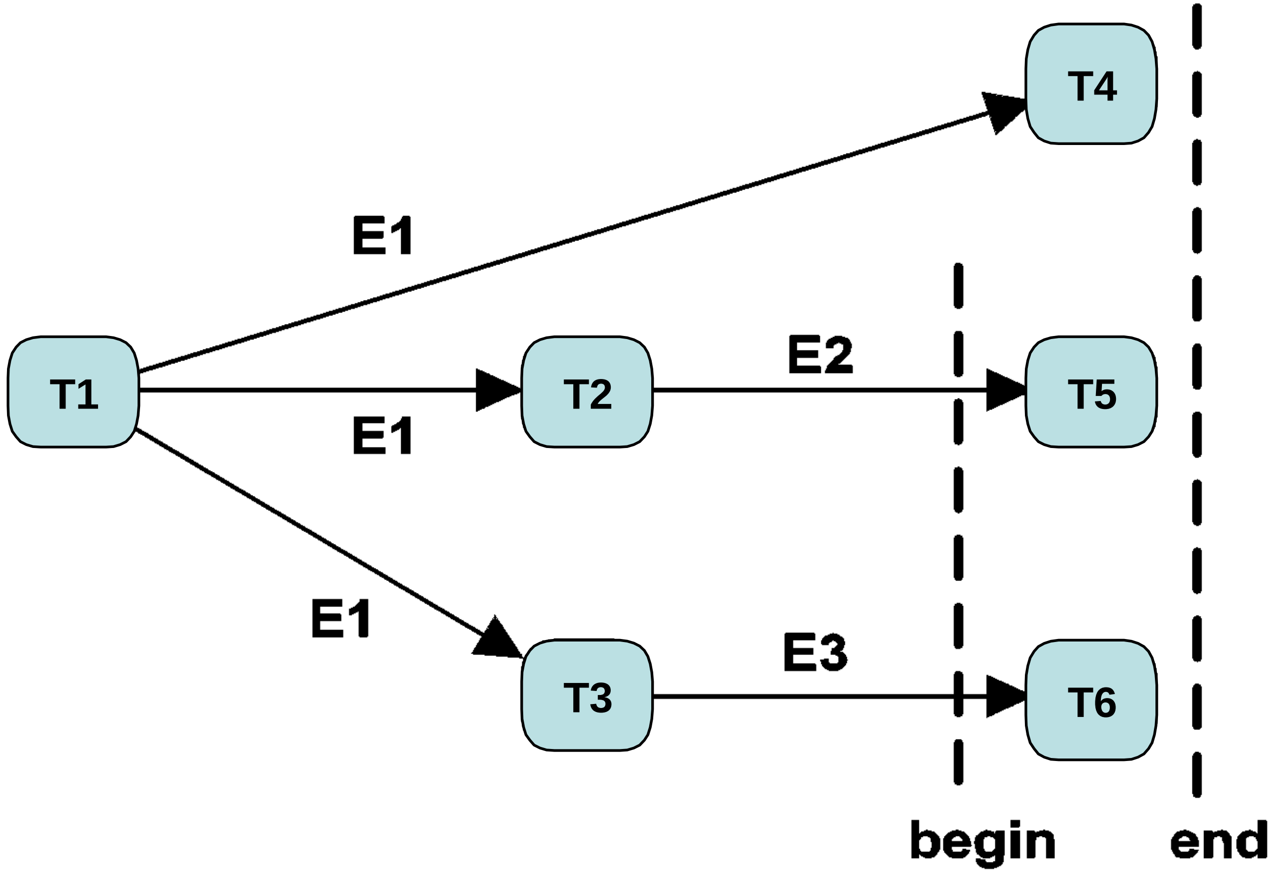
# Plan represented as a table

| Task | Preconditions | Effects | Resources | Duration |
|---|---|---|---|---|
| T1 | - | E1 | 4`R1 | 2 |
| T2 | E1 | E2 | 2`R2 ++ 2`R3 | 4 |
| T3 | E1 | E3 | 2`R2 ++ 2`R3 | 7 |
| T4 | E1 | E4 | 1`R2 ++ 1`R3 | - |
| T5 | E2 | E5 | 1`R4 | 7 |
| T6 | E3 | E6 | 1`R5 | 7 |

```
Available resources:
4`R1 ++ 3`R2 ++ 3`R3 ++ 1`R4 ++ 1`R5
```

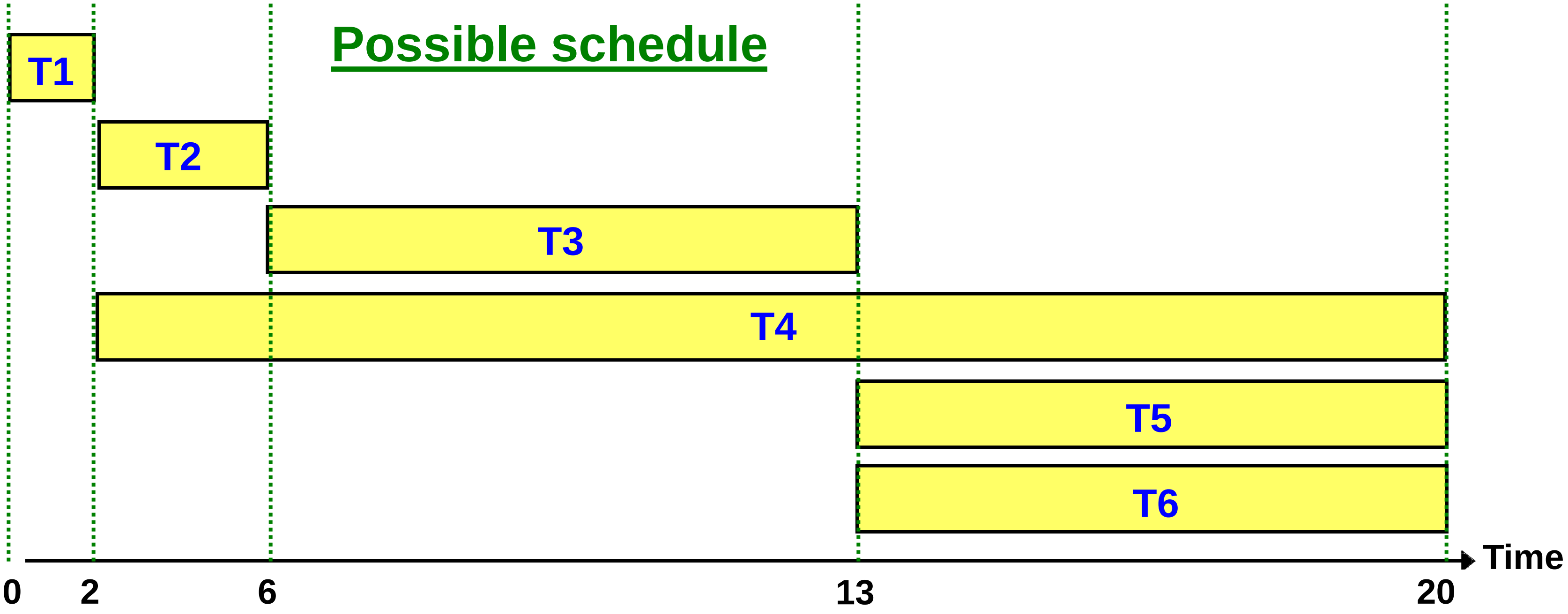
```
{T5,T6} are begin-synchronised
{T4,T5,T6} are end-synchronised.
```

# Graphical representation of plan



# Scheduling of tasks

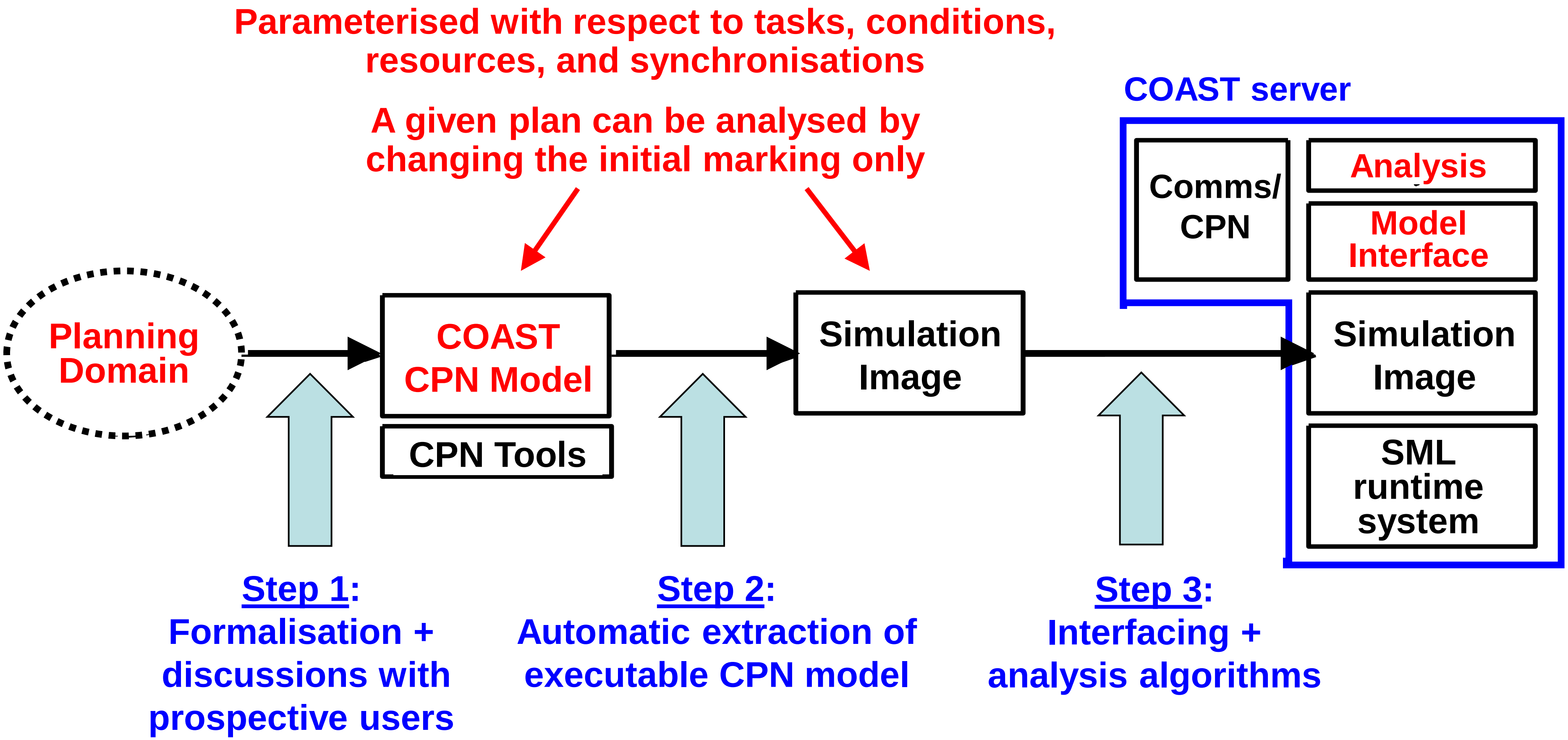
- We want to calculate the possible task schedules – the ways in which the set of tasks can be sequenced.
- Each schedule must respect effects and preconditions, available resources, and synchronisation constraints.



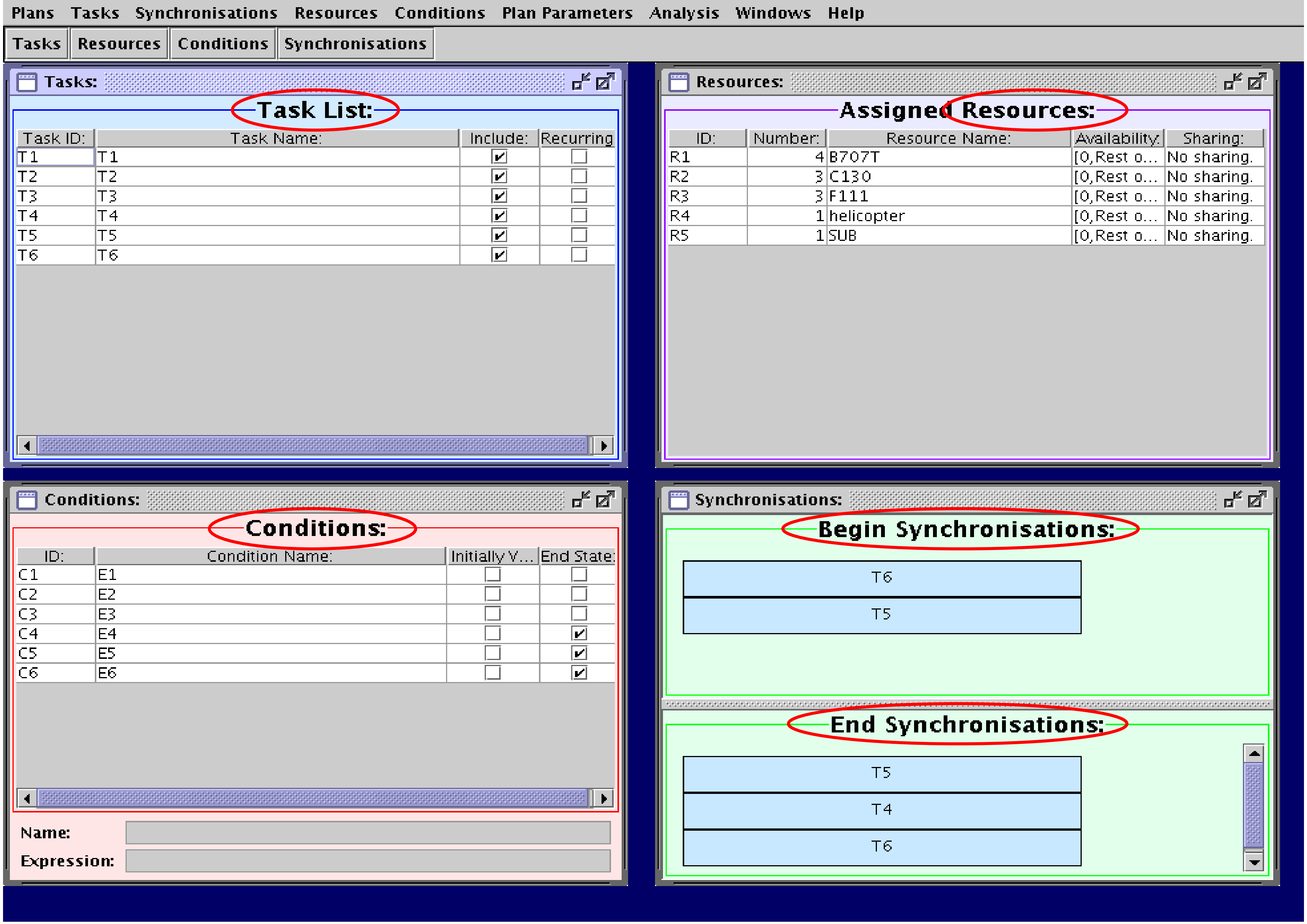
# Architecture of scheduling tool

- Based on a client-server architecture.
- The client offers a domain-specific graphical user interface for the specification of plans including their resources, conditions, and synchronisations.
- To analyse a plan, the client invokes analysis algorithms in the server (computing task schedules).
- The server also supports exploration and debugging of plans (when analysis shows that no task schedules exists).
- Communication between the client and the server is based on remote procedure calls in the Comms/CPN library.

# Construction of COAST server


Parameterised with respect to tasks, conditions, resources, and synchronisations
A given plan can be analysed by changing the initial marking only
COAST server
Comms/ CPN
Analysis
Model Interface
Planning Domain
COAST CPN Model
CPN Tools
Simulation Image
Simulation Image
SML runtime system
Step 1: Formalisation + discussions with prospective users
Step 2: Automatic extraction of executable CPN model
Step 3: Interfacing + analysis algorithms

# Editing a plan


Plans Tasks Synchronisations Resources Conditions Plan Parameters Analysis Windows Help
Tasks Resources Conditions Synchronisations

Tasks:

Task List:

| Task ID: | Task Name: | Include: | Recurring |
|---|---|---|---|
| T1 | T1 | ☑ | □ |
| T2 | T2 | ☑ | □ |
| T3 | T3 | ☑ | □ |
| T4 | T4 | ☑ | □ |
| T5 | T5 | ☑ | □ |
| T6 | T6 | ☑ | □ |

Resources:

Assigned Resources:

| ID: | Number: | Resource Name: | Availability: | Sharing: |
|---|---|---|---|---|
| R1 | 4 | B707T | [0,Rest o... | No sharing. |
| R2 | 3 | C130 | [0,Rest o... | No sharing. |
| R3 | 3 | F111 | [0,Rest o... | No sharing. |
| R4 | 1 | helicopter | [0,Rest o... | No sharing. |
| R5 | 1 | SUB | [0,Rest o... | No sharing. |

Conditions:

Conditions:

| ID: | Condition Name: | Initially V... | End State: |
|---|---|---|---|
| C1 | E1 | □ | □ |
| C2 | E2 | □ | □ |
| C3 | E3 | □ | □ |
| C4 | E4 | □ | ☑ |
| C5 | E5 | □ | ☑ |
| C6 | E6 | □ | ☑ |

Name:

Expression:

Synchronisations:

Begin Synchronisations:

T6

T5

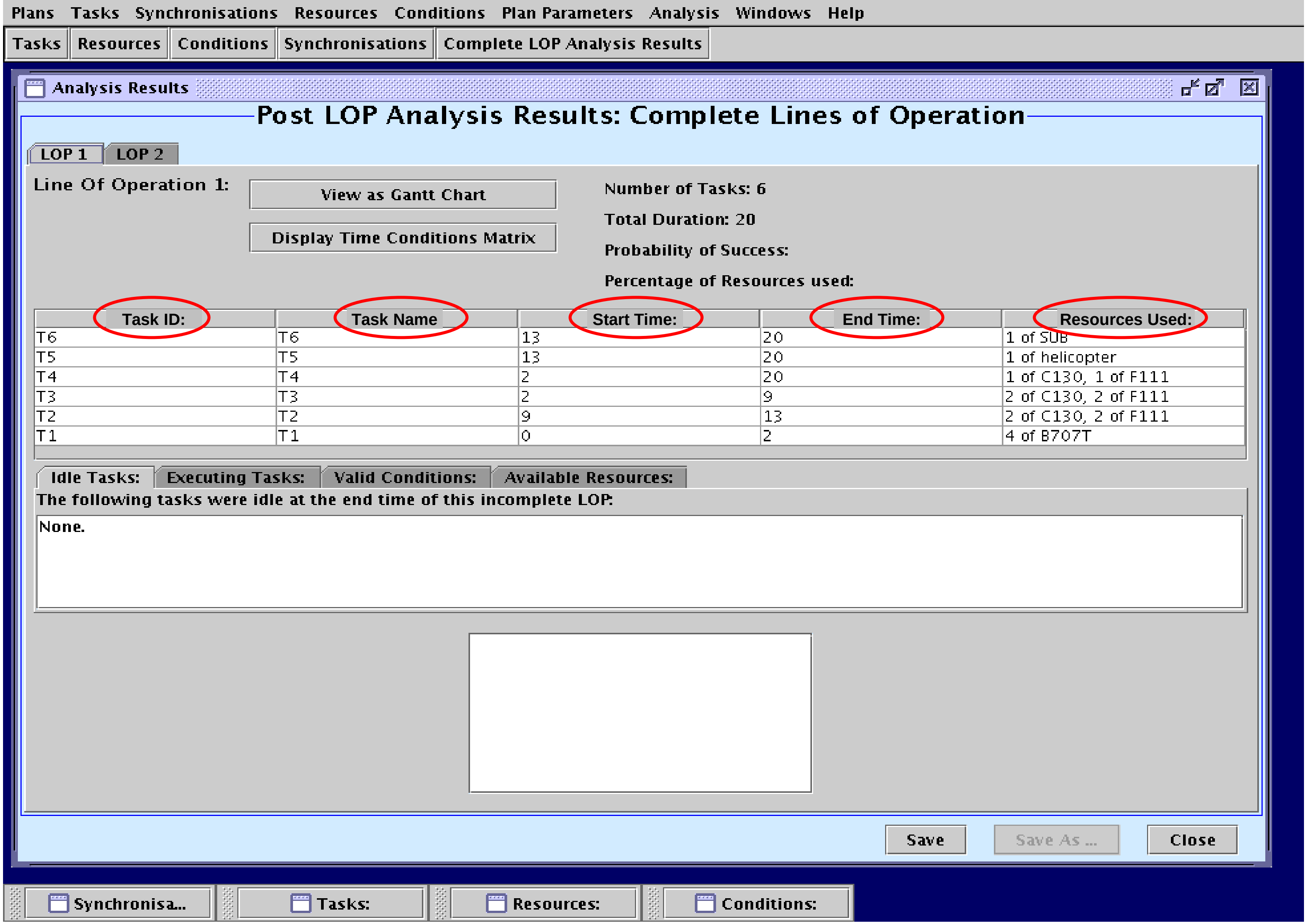
End Synchronisations:

T5

T4

T6

# Analysing a plan


Plans Tasks Synchronisations Resources Conditions Plan Parameters Analysis Windows Help
Tasks Resources Conditions Synchronisations Complete LOP Analysis Results
Analysis Results
LOP 1
LOP 2
View as Gantt Chart
Display Time Conditions Matrix
Idle Tasks:
Executing Tasks:
Valid Conditions:
Available Resources:
Save
Save As ...
Close
Synchronisa...
Tasks:
Resources:
Conditions:

## Post LOP Analysis Results: Complete Lines of Operation

Line Of Operation 1:

Number of Tasks: 6

Total Duration: 20

Probability of Success:

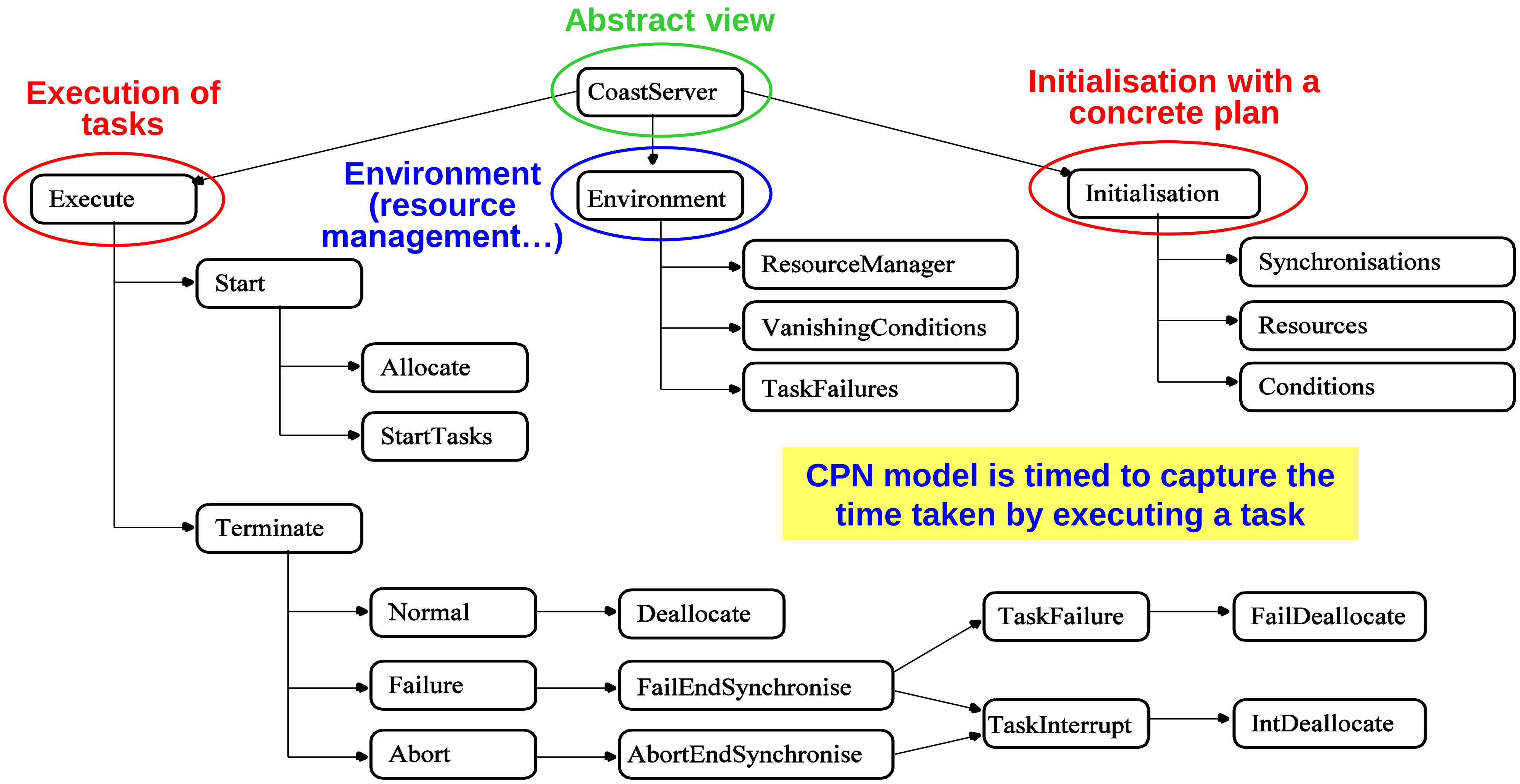
Percentage of Resources used:

| Task ID: | Task Name | Start Time: | End Time: | Resources Used: |
|---|---|---|---|---|
| T6 | T6 | 13 | 20 | 1 of SUB |
| T5 | T5 | 13 | 20 | 1 of helicopter |
| T4 | T4 | 2 | 20 | 1 of C130, 1 of F111 |
| T3 | T3 | 2 | 9 | 2 of C130, 2 of F111 |
| T2 | T2 | 9 | 13 | 2 of C130, 2 of F111 |
| T1 | T1 | 0 | 2 | 4 of B707T |

The following tasks were idle at the end time of this incomplete LOP:

None.

# Module hierarchy for COAST model



CPN model is timed to capture the time taken by executing a task

# Colour sets for conditions + resources

```
colset Condition  = product STRING * BOOL;
colset Conditions = list Condition;
```

```
colset Resource = product INT * STRING;
colset ResourceList = list Resource;

colset AvailSpecification = union INT : INTxINT + FROM : INT;
colset Availability  = list AvailSpecification;

colset ResourcexAvailability = product Resource * Availability;
colset ResourceSpecification = list ResourcexAvailability;

colset Resources = union IDLE : ResourceSpecification
                       + LOST : ResourceSpecification;
```

# Colour sets for tasks + synchronisations

```
colset Task = record
    name : STRING *
    duration : Duration *
    normalpreconditions         : Conditions *
    vanishingpreconditions      : Conditions *
    sustainingpreconditions     : Conditions *
    terminationpreconditions    : Conditions *
    instanteffects              : Conditions *
    posteffects                 : Conditions *
    sustainingeffect            : Conditions *
    startresources : ResourceList *
    resourceloss   : ResourceList;
```

Name

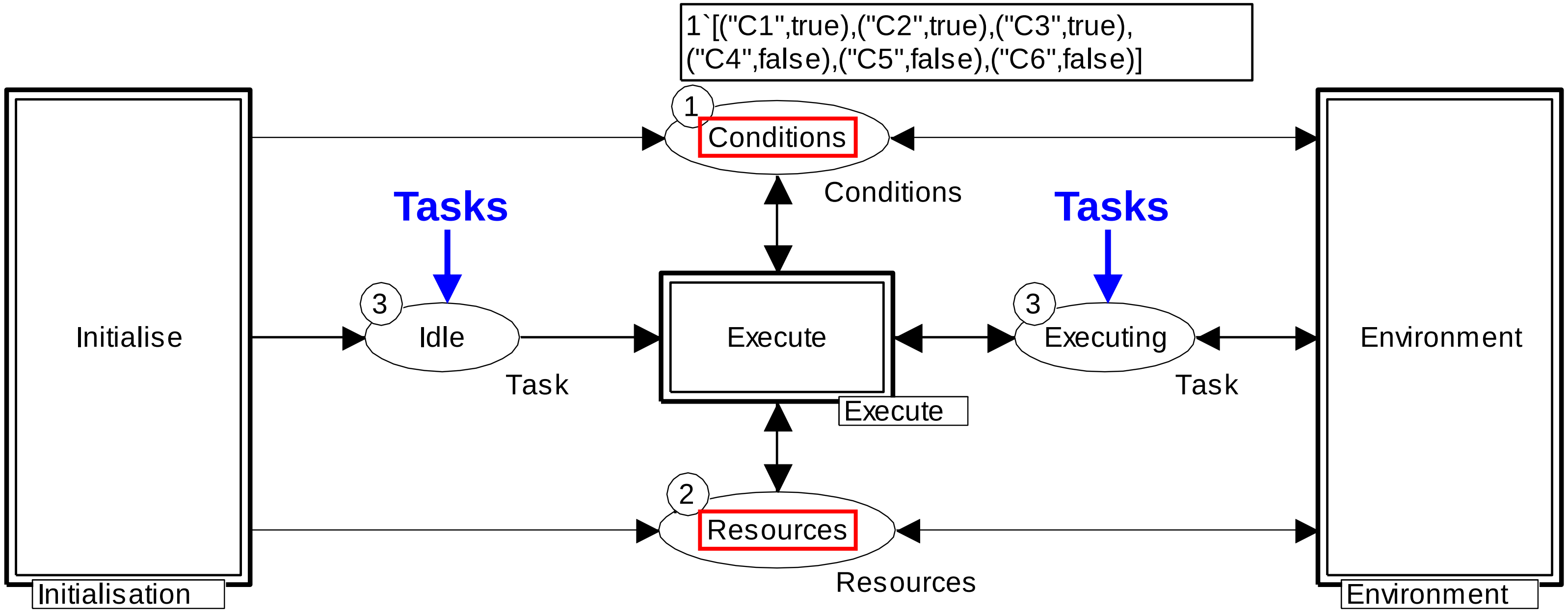
Duration

Different kinds of conditions

Resources

```
colset BeginSynchronisation = list Task;
colset EndSynchronisation   = list Task;
```

# CoastServer module (abstract view)



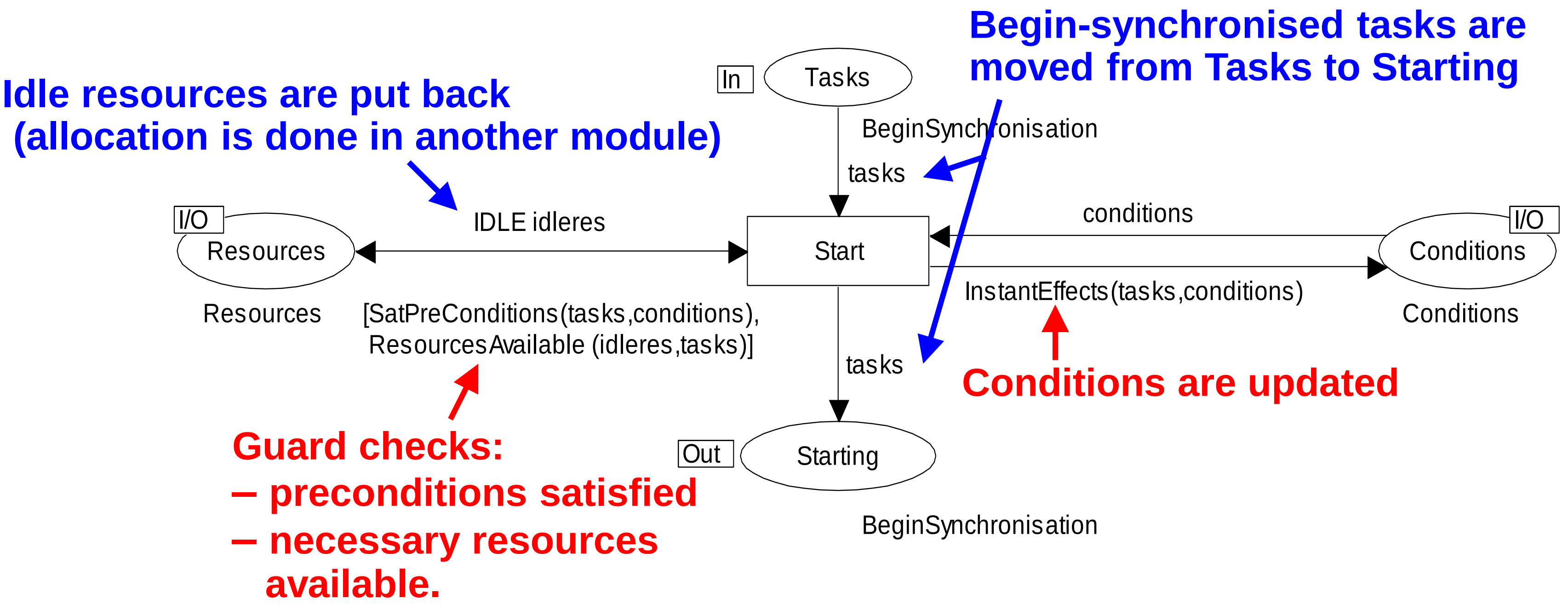
**Two tokens:**

**– a list of idle resources**

**– a list describing lost resources**

# Allocate module (part of Execute)

- The module represents the start of a set of begin-synchronised tasks.



Begin-synchronised tasks are moved from Tasks to Starting

Idle resources are put back (allocation is done in another module)

Conditions are updated

Guard checks:
– preconditions satisfied
– necessary resources available.

- Other modules model task execution and their effect on conditions and resources. They have a similar complexity.

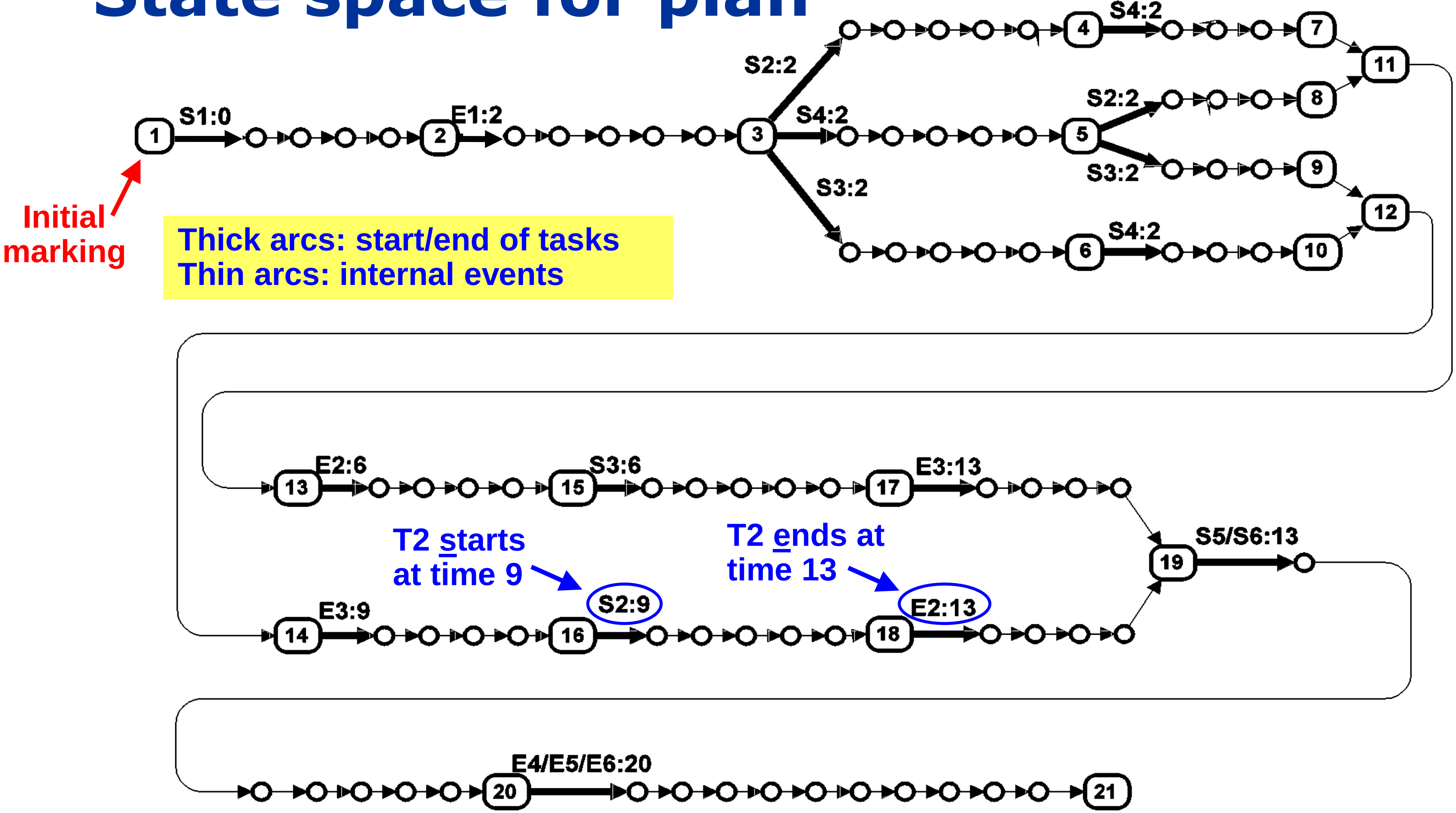
# Generation of task schedules

- We generate the state space for the plan to be analysed (successors are not generated for states that qualify as desired end-states).

- Then the task schedules are computed from paths in the state space and divided into two classes:
  - Complete schedules leading to desired end-states.
  - Incomplete schedules leading to undesired end-states (dead markings not satisfying the specification of the plan).

- Users can investigate incomplete schedules by means of a set of queries – allowing the planner to identify errors and inconsistencies in the plan.

# State space for plan



Initial marking

Thick arcs: start/end of tasks
Thin arcs: internal events

T2 starts at time 9

T2 ends at time 13

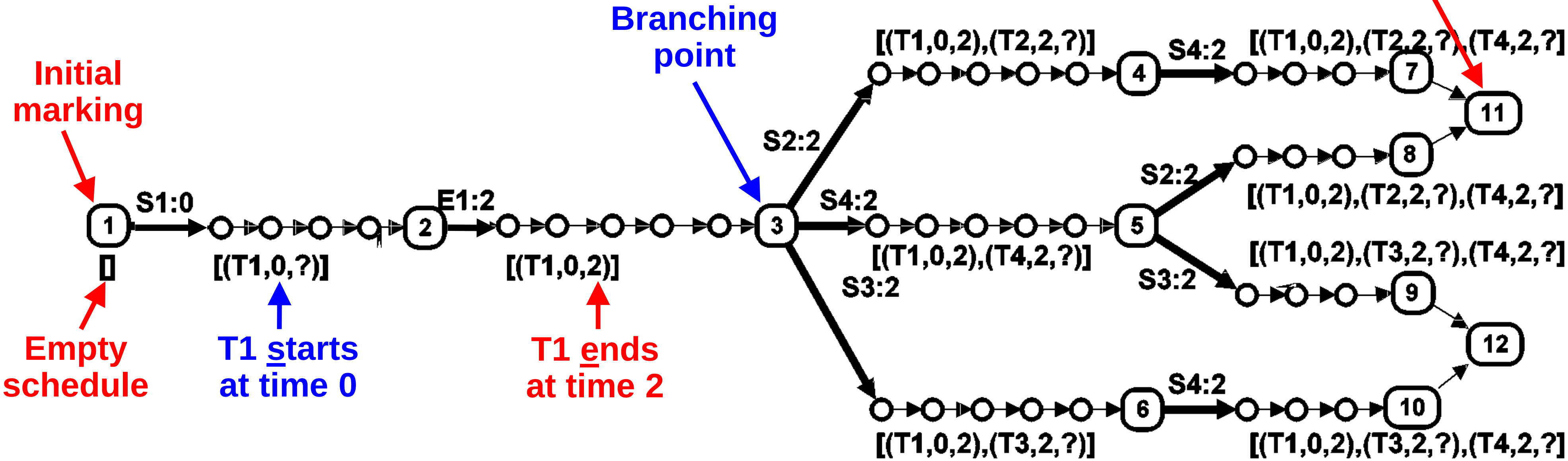
# Algorithm for generation of schedules

- Generated by a breadth-first traversal of the state space starting from the initial marking.

- For each marking we compute the schedules leading to it (from the schedules of its predecessors).

- The algorithm exploits that:
  - The state space of a plan is acyclic.
  - All paths leading to a given marking in the state space have the same length.

# Algorithm



Merging of two schedules from node 7 and node 8 (with consistent start/end times)

Initial marking

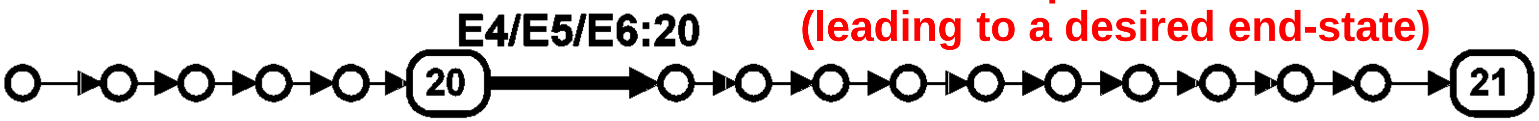
Branching point

Empty schedule

T1 starts at time 0

T1 ends at time 2

Many steps later we obtain:



Two complete schedules (leading to a desired end-state)

[(T1,0,2),(T2,2,6),(T4,2,20),(T3,6,13),(T5,13,20),(T6,13,20)]

[(T1,0,2),(T4,2,20),(T3,2,9),(T2,9,13),(T5,13,20),(T6,13,20)]

# Planning problems

- Typical planning problems consist of 15-25 tasks resulting in state spaces with:
  - 10,000-20,000 nodes.
  - 25,000-35,000 arcs.

- The state spaces are relatively small because the conditions, resources, and synchronisations limit the possible orders in which tasks can be executed.

# Conclusions from COAST project

- CPN modelling was used in the development and specification of the planning framework.
- The CPN model was used to implement the COAST server (closing the gap between design and implementation).
- State spaces were used to compute and analyse schedules.
- The project demonstrates the value of having a full programming language environment in the form of the Standard ML compiler integrated in CPN Tools.

# Questions