Modelling the NEO Distributed Storage Protocol with Coloured Petri Nets

The NEOPPOD project

Anna Dedova

LIPN - Université Paris 13

September 30th, 2011

Outline

Introduction

The NEO Protocol

Reverse-engineering

Protocol

Modelling Method: Grounding

Project structure

Key elements

Interactions and auxiliary elements

Module structure

Modelling Method: Code Analysis

Data structures to colour domains

States to places

Events to transitions

Transforms to arcs

Conditions to guards

Summary

Outline

The NEO Protocol

- part of enterprise resource planning system ERP5
- based on the object data base ZODB
- treatment of large highly-distributed data bases
- sensitive data, the loss is critical

Quality control

Testing

- unit tests checking individual methods behaviour
- functional tests checking nodes and cluster behaviour
- standard ZODB test suites

Formal methods

- model-checking
  - modelling
  - verification

Reverse-engineering

Modelling approaches

1. from informal description of a problem
2. from detailed specification of a system
3. from the source code

Abstraction level

- too low, the model contains too many details
  $\rightarrow$ the model is huge
  $\rightarrow$ no means to analyse it
  $\rightarrow$ it is useless
- too high, there are too many hypothesis and assumptions
  $\rightarrow$ nothing is left worth checking

Outline

Protocol Characteristics

Features

- supposed to be used in banking and e-government
- tolerance to faults
- replication of data
- cluster of nodes of different types

Scale

- 1 to 10 master nodes
- 100 to 10 000 storage nodes
- unlimited number of clients

Model Size

Source code

- high level object oriented
- 30 500 lines in 341 files
- no specifications
- comments

Model size

	Election	Bootstrap
places	28	98
transitions	56	60
message types	6	20
SML functions	8	31
nodes in state space	78 (2 MN, no faults) 329 (2 MN, with faults)	18 000 (for 4 SN)

Outline

Grounding steps

1. Understand structure
2. Choose key elements
3. Find interactions
4. Find auxiliary elements
5. Divide into modules

1. Understand Structure

Protocol Architecture

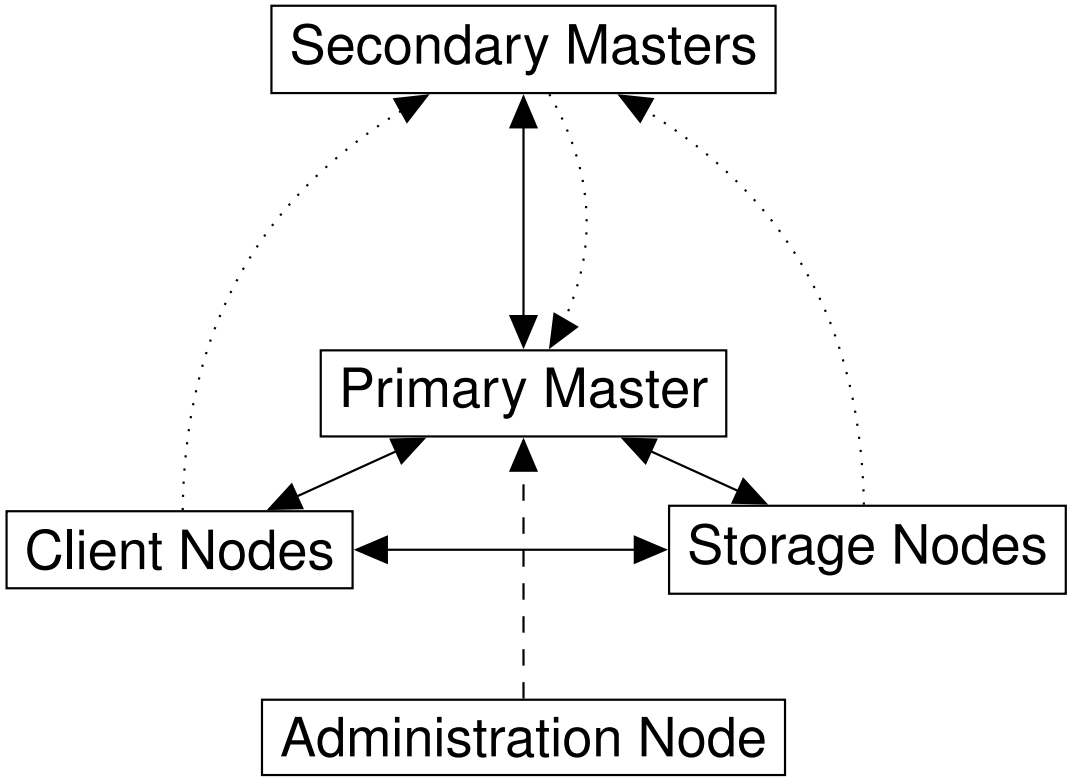


Figure: The NEO protocol topology

1. Understand Structure

Protocol Lifeline

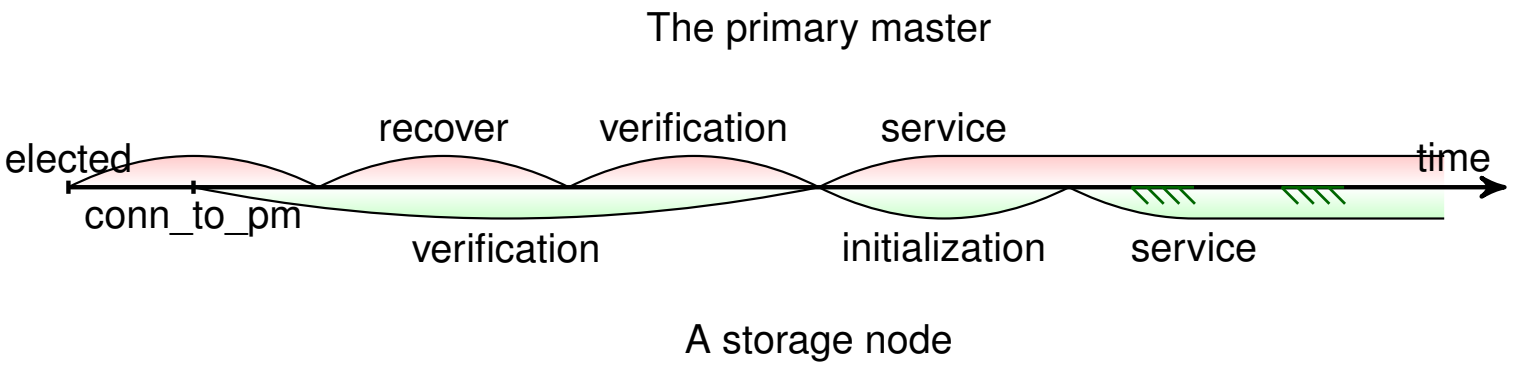


Figure: Primary and storage phases in time

- For each phase there is a special message handler that treats all incoming messages.
- Each handler treats certain message types.

2. Find Key elements

Properties to check

1. In the end of election phase there is exaclty one primary master.
2. PM eventually reaches the service state.
3. All SNs arrive finally to the service state.
4. In the end of the bootstrap phase there is no more unfinished transaction.
5. There is no untreated message in the network.
6. All lost (or containing lost objects) transactions are deleted.

2. Find Key elements

Types of nodes

- **Master**
- **Storage**
- Client
- Admin

Every node has a **run** method that is called in the end of class constructor. It contains an infinite loop that is executed until shut down is requested.

Stages

- Election of the primary master
- Bootstrap, contain different **phases**:
  - For the primary master node: Recovery, Verification
  - For storage node: Verification, Initialisation
- Service

3-4. Interactions and Secondary elements

Interactions

- Network
  network
  MESS
- Partition table
- Global variables
  has_pt_ni_lid
  SNxPTxNIxLID

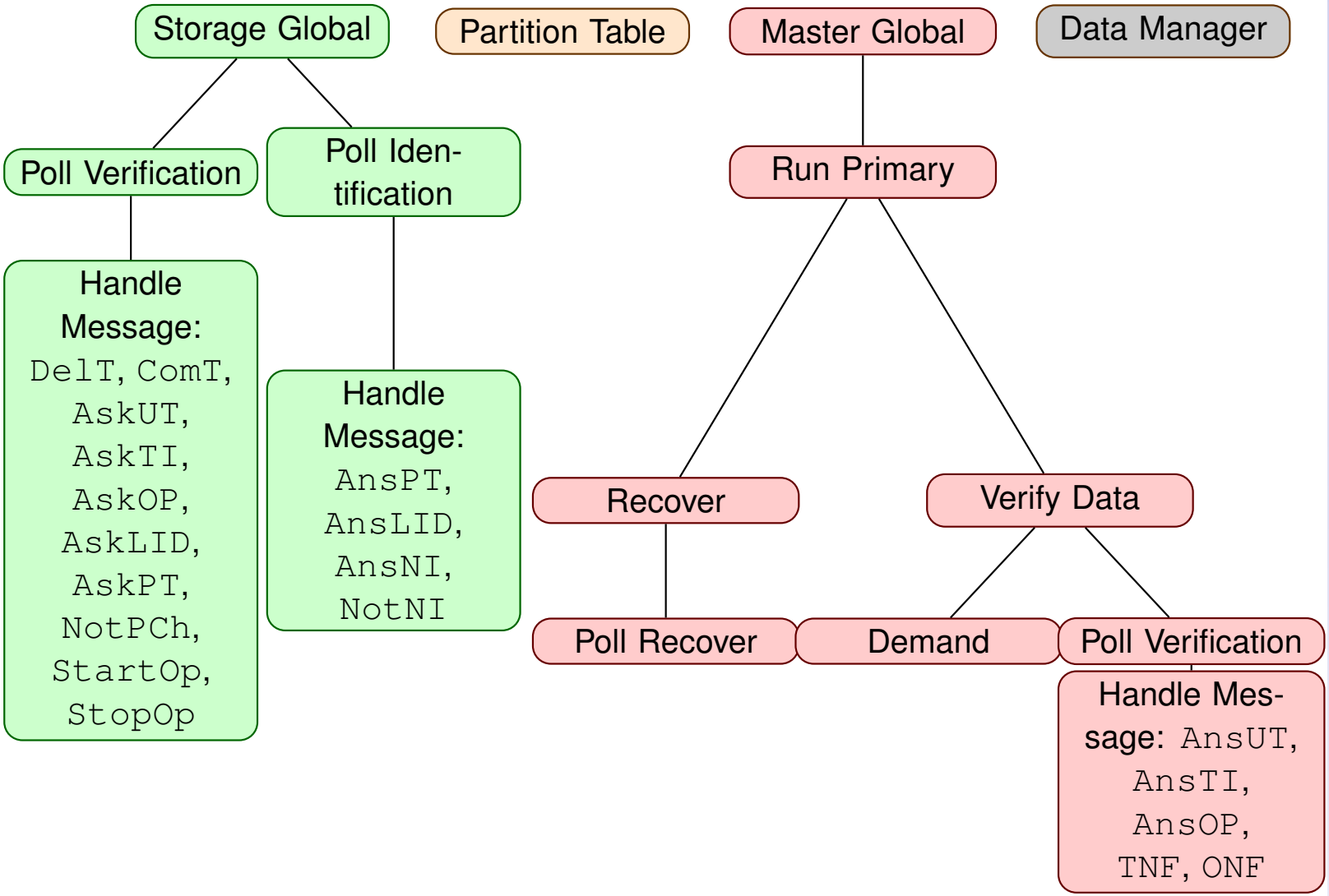
	sn 1	sn 2	sn 3	sn 4
p 0	+	+		
p 1			+	+
p 2	+	+		
p 3			+	+
p 4	+	+		
p 5			+	+
p 6	+	+		
p 7			+	+
p 8	+	+		
p 9			+	+

Secondary elements

- poll method
- Data manager class

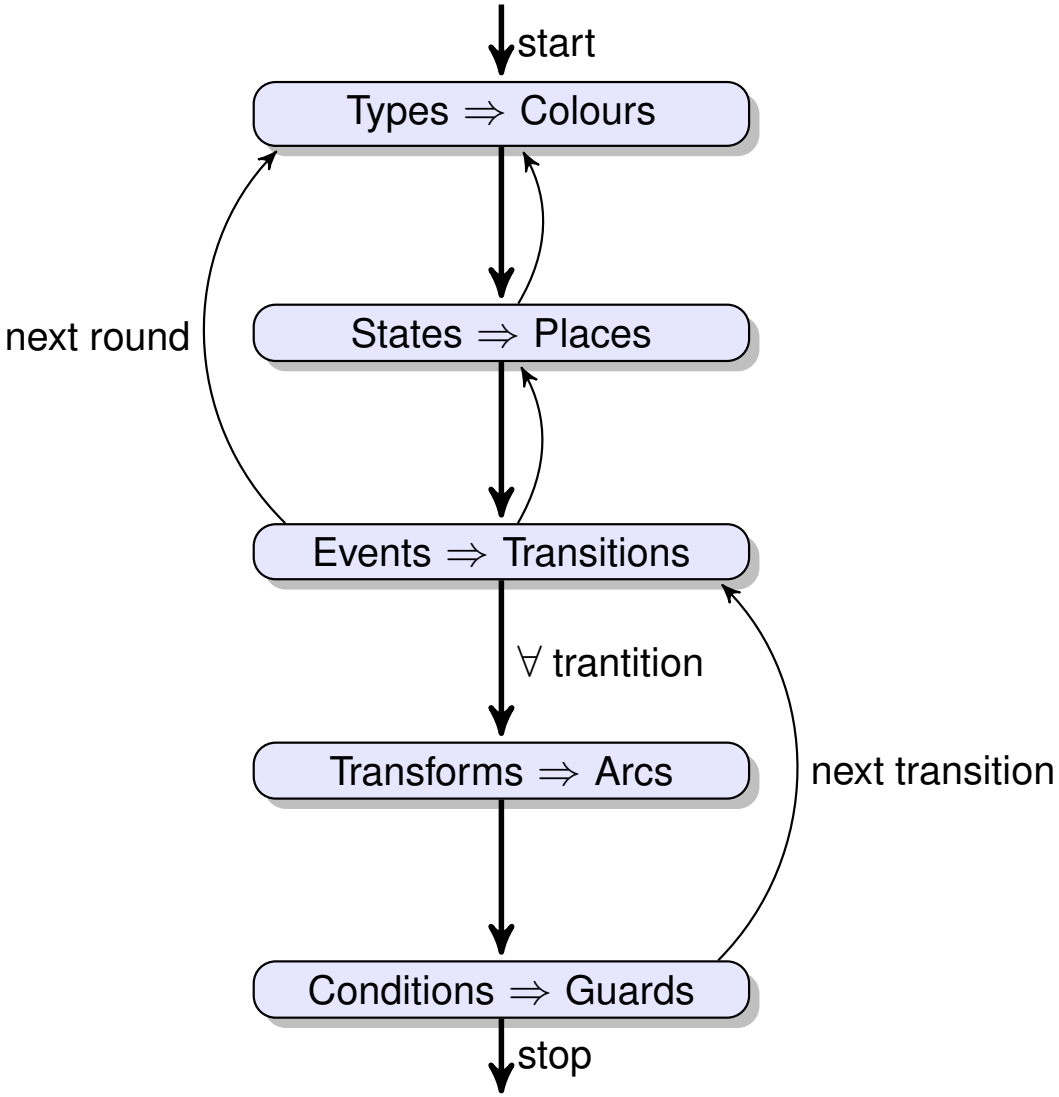
5. Divide into Modules

NEO Model Hierarchy



Outline

Code Analysis



1. Data structures

- Storage node: `colset SN = index sn with 0..N;`
- Master node: `colset MN = index mn with 0..M;`
- Node: `colset NODE = union s1:SN + m1:MN;`
- Message type:

```
colset MTYPE = with StopOp | StartOp |
AnsUT |AnsNI | AnsPT | AnsLID | AnsTI | AnsOP |
NotNI | NotPCh | DelT | ComT | ONF | TNF |
AskUT | AskPT | AskNI | AskLID | AskTI | AskOP;
```

- Message:

```
colset MESS = product MTYPE*NODE*NODE*INT;
```

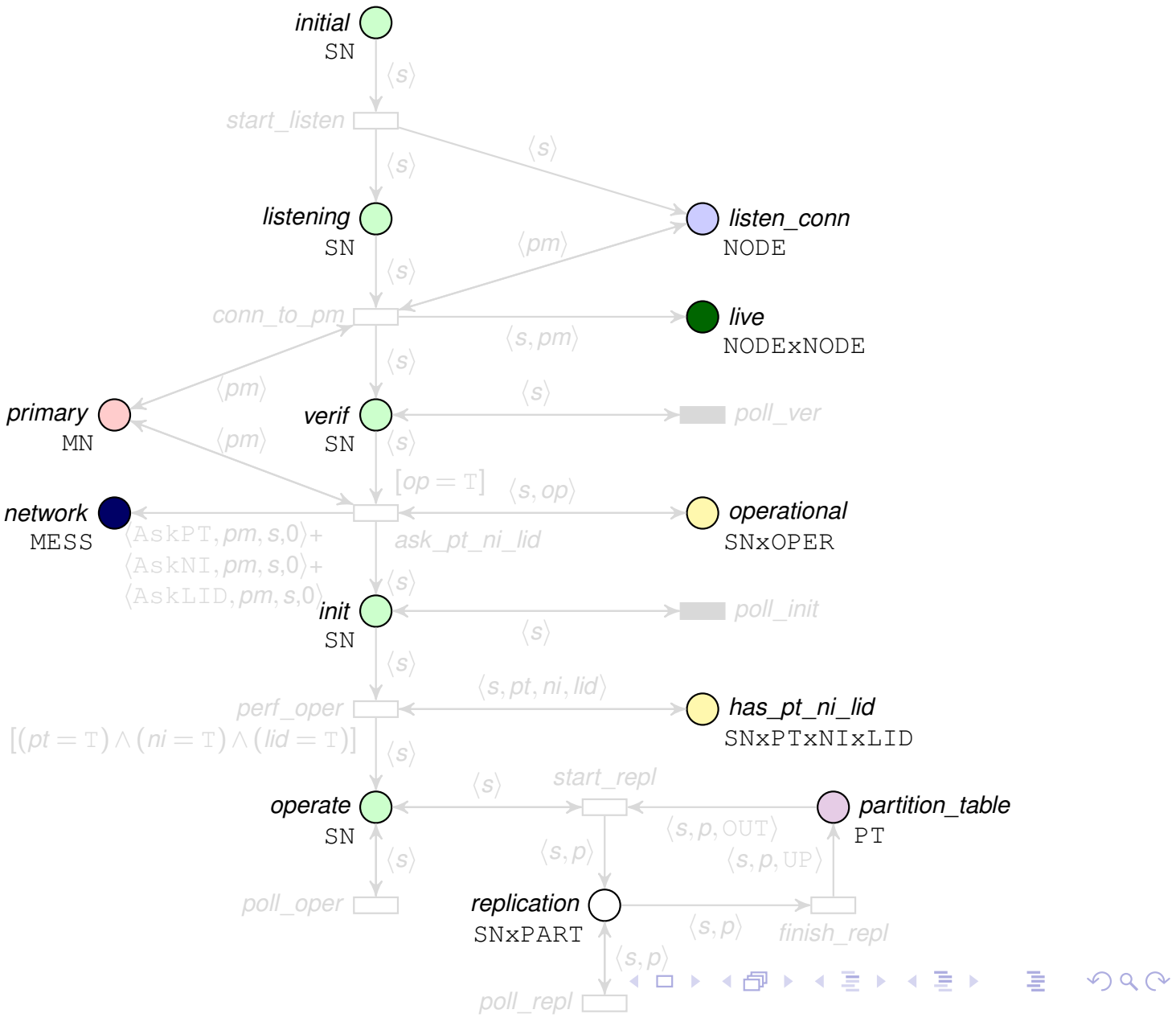
2-3. Places and transitions

Source Code of Storage Node

```
1  def _run(self):
2          ...
3          self.listening_conn = ListeningConnection(self.em, handler,
4              addr=self.server, connector=self.connector_handler())
5
6          while True:
7              self.ready = False
8              self.operational = False
9              if self.master_node is None:  self.connectToPrimary()
10             ...
11             try:
12                 self.verifyData()
13                 self.initialize()
14                 self.doOperation()
15                 raise RuntimeError, 'should not reach here'
16             except OperationFailure, msg:
17                 logging.error('operation stopped: %s', msg)
18             except PrimaryFailure, msg:
19                 logging.error('primary master is down: %s', msg)
20                 self.master_node = None
21
22 def doOperation(self):
23         handler = master.MasterOperationHandler(self)
24         self.master_conn.setHandler(handler)
25         ...
26         while True:
27             self.em.poll()
28             if self.replicator.pending():
29                 self.replicator.act()
```

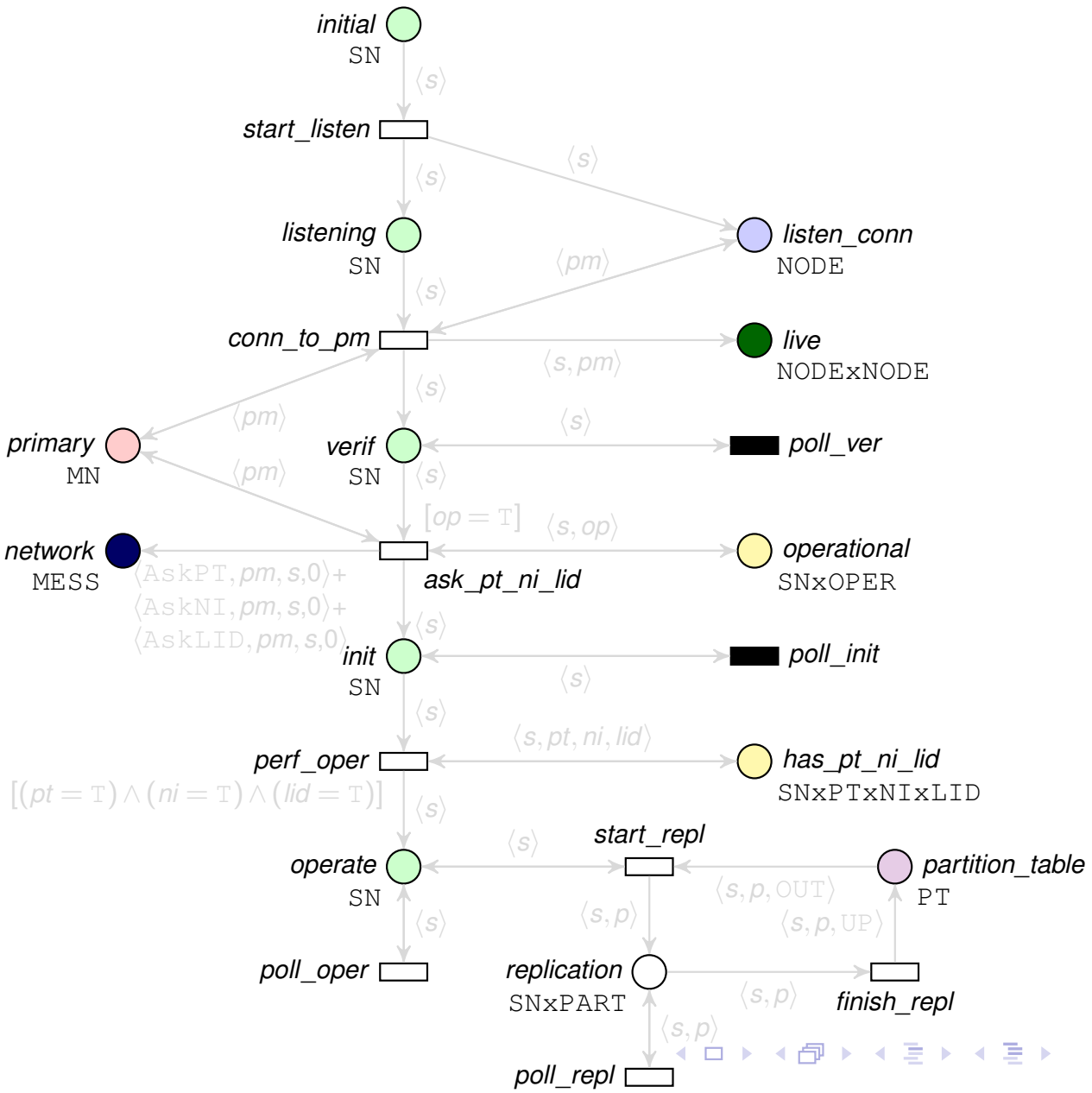
2-3. Places and transitions

Model of Storage Node - Global Level



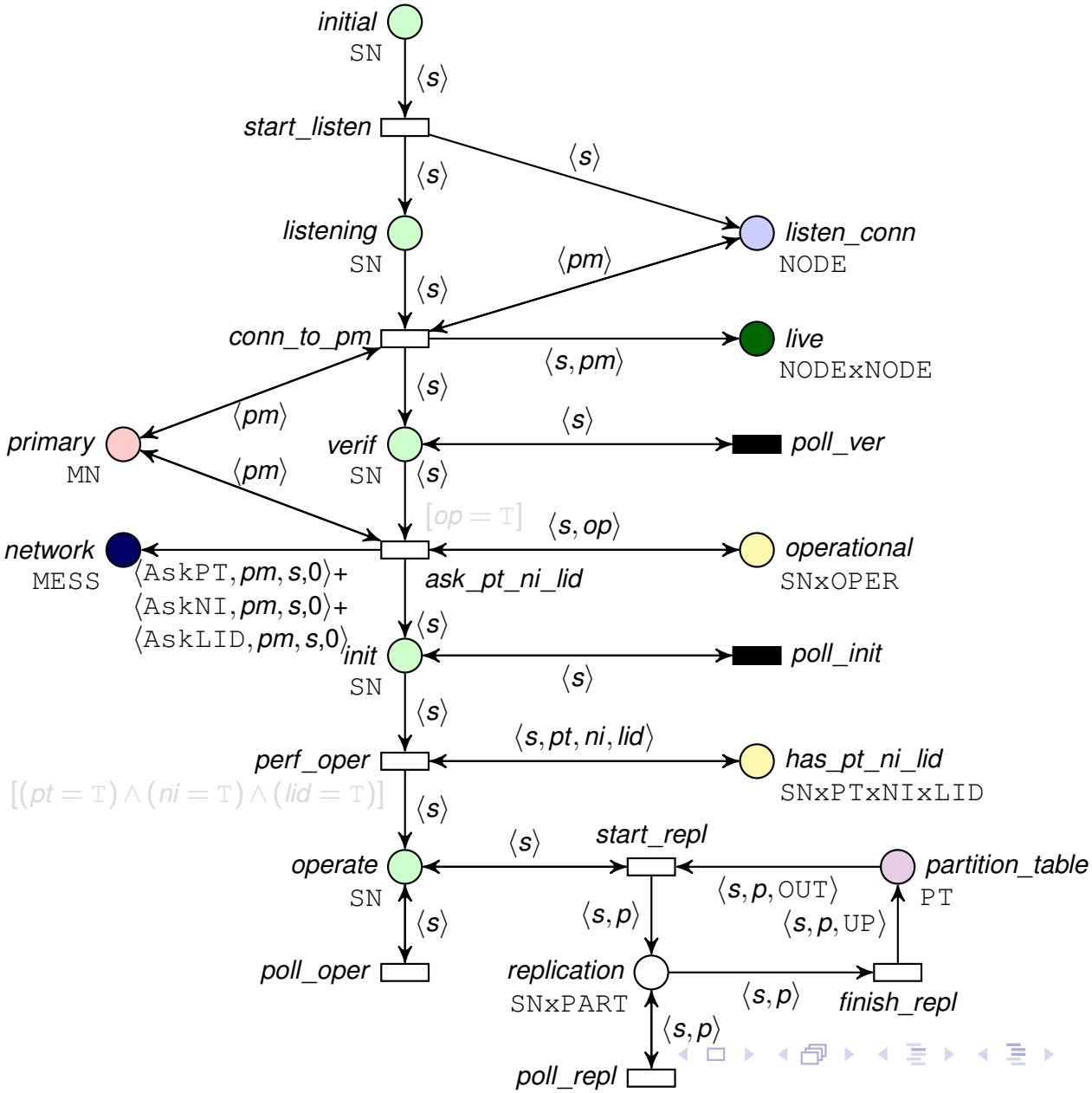
2-3. Places and transitions

Model of Storage Node - Global Level



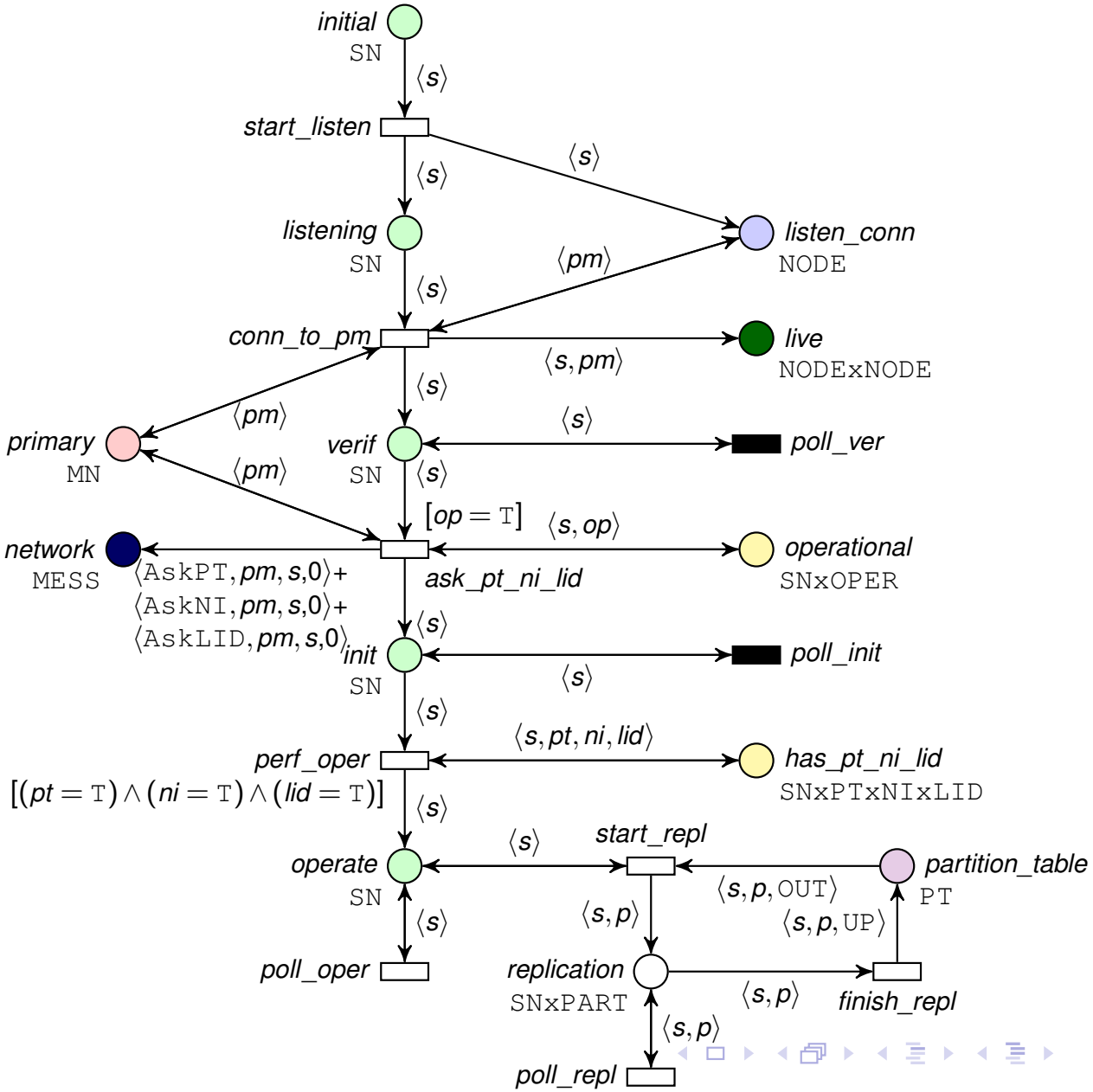
2-3. Places and transitions

Model of Storage Node - Global Level



2-3. Places and transitions

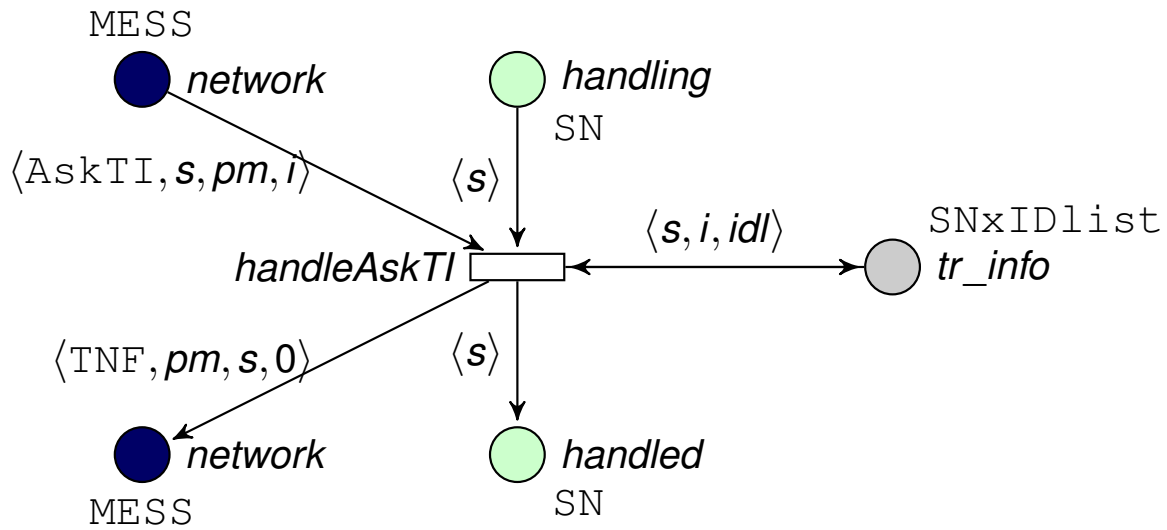
Model of Storage Node - Global Level



4. Arcs

Verification phase handlers : Ask Transaction Information (Transaction Not Found)

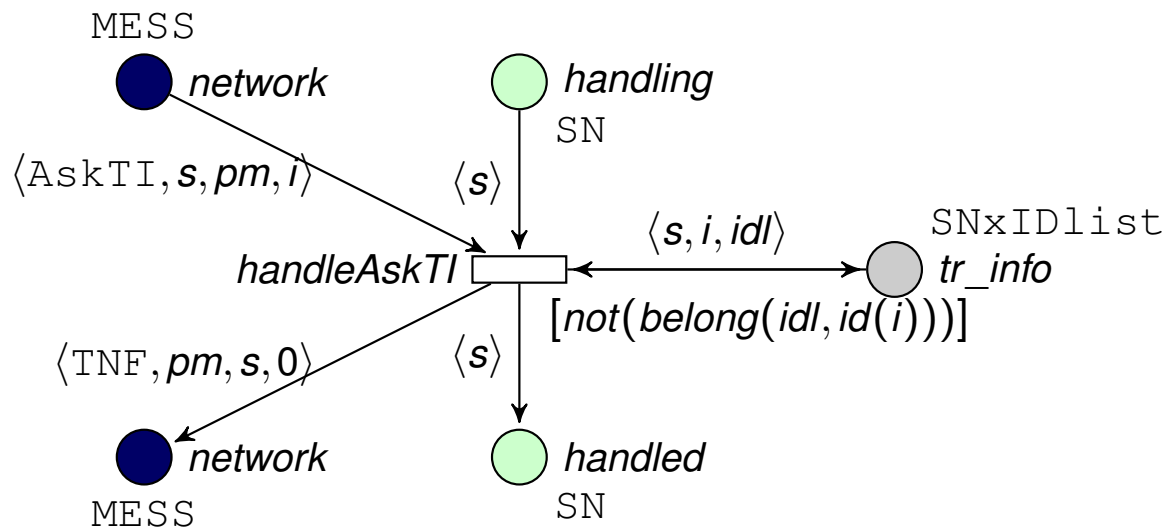
```
1   def askTransactionInformation(self, conn, tid):
2       app = self.app
3       t = app.dm.getTransaction(tid, all=True)
4       if t is None:
5           p = Errors.TidNotFound('%s does not exist' % dump(tid))
6       else:
7           p = Packets.AnswerTransactionInformation(tid, t[1], t[2], t[3],
8                   t[4], t[0])
9       conn.answer(p)
```



5. Guards

Verification phase handlers : Ask Transaction Information (Transaction Not Found)

```
1    def askTransactionInformation(self, conn, tid):
2        app = self.app
3        t = app.dm.getTransaction(tid, all=True)
4        if t is None:
5            p = Errors.TidNotFound('%s does not exist' % dump(tid))
6        else:
7            p = Packets.AnswerTransactionInformation(tid, t[1], t[2], t[3],
8                    t[4], t[0])
9        conn.answer(p)
```




```
fun belong ([], _) = false | belong (item ::  l, item') = if item = item'
then true else belong (l, item')
```

Outline

Summary

Grounding

1. Structure
2. Key elements
3. Interactions
4. Auxiliary elements
5. Modules

Analisys of code

- Types ⇒ Colours
- States ⇒ Places
- Events ⇒ Transitions
- Transforms ⇒ Arcs
- Conditions ⇒ Guards