Towards A Formally Verified Network-on-Chip

Tom van den Broek[1] Julien Schmaltz[12]

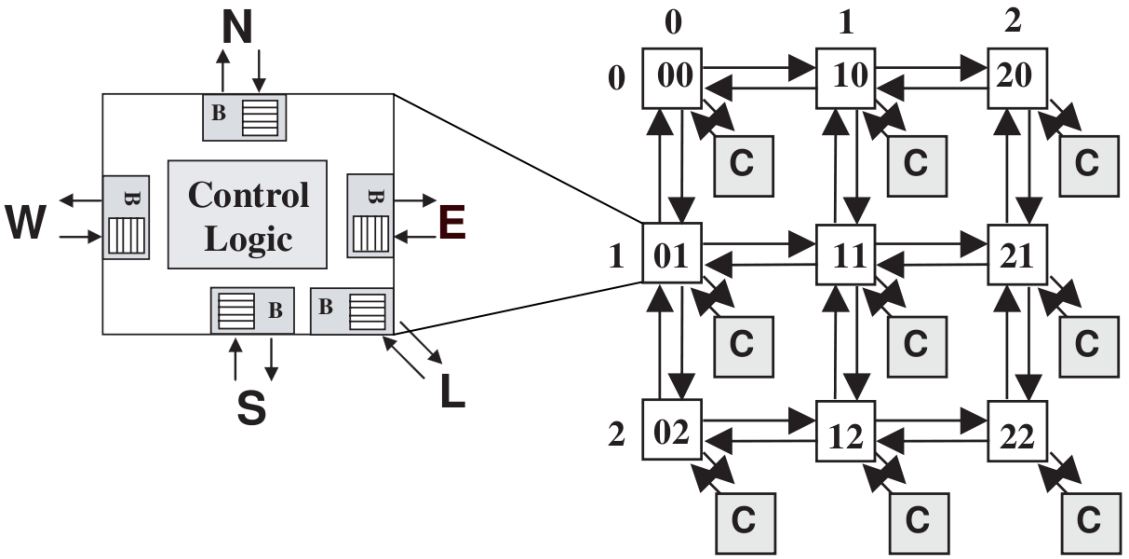
[1]Institute for Computing and Information Sciences
Radboud University Nijmegen
The Netherlands

[2]School of Computer Science
Open University
The Netherlands

t.vandenbroek@cs.ru.nl & julien.schmaltz@ou.nl

FMCAD '09

Networks-on-Chips: Hermes



- Implemented as model instance
- Characteristics:
  - XY minimal deterministic routing
  - Wormhole switching
  - Frame structure:
    - Header flit (Route Information)
    - Data flits (Payload)
    - Torn-down flit (Last flit)

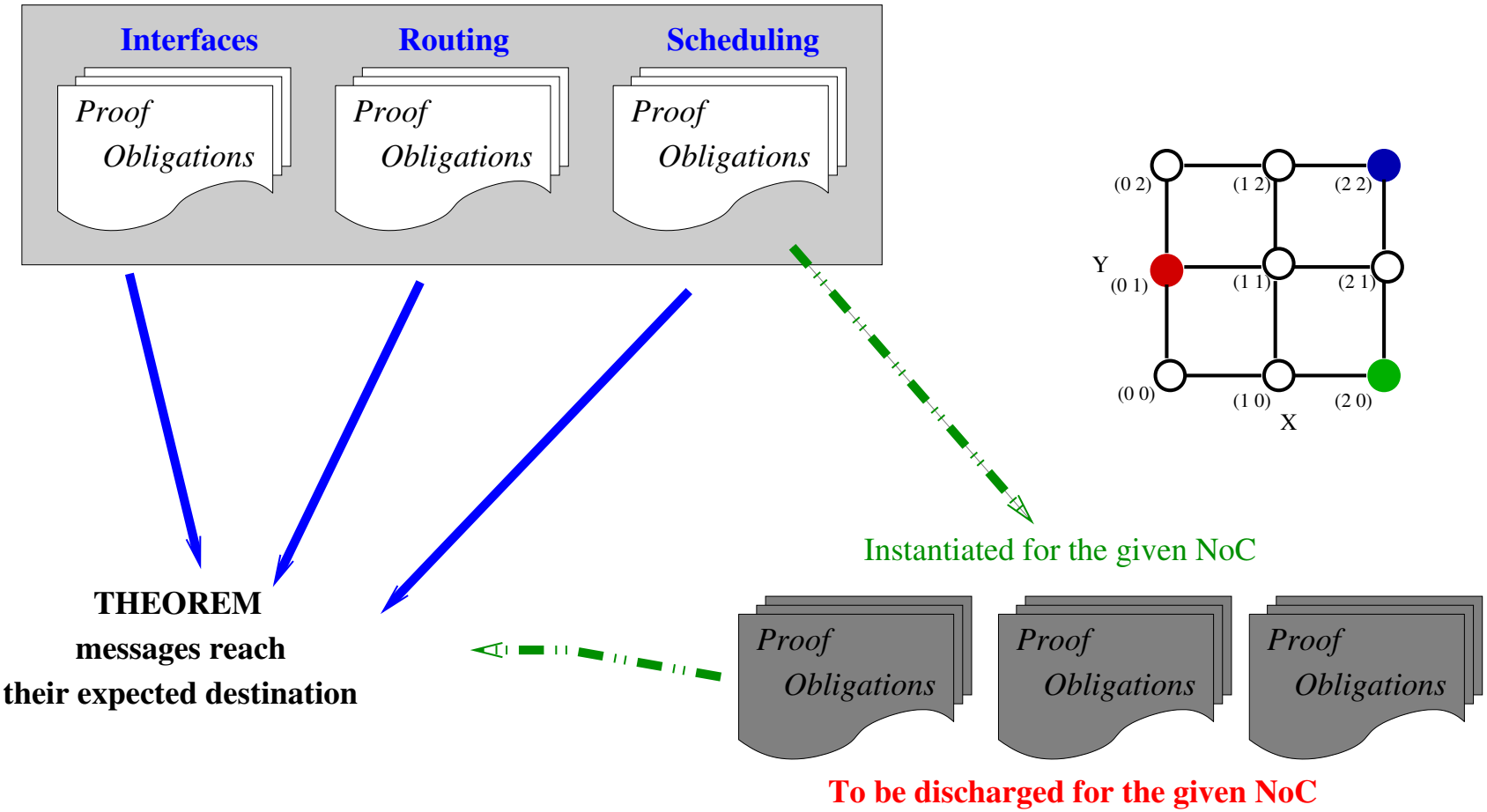
Platform-Based Design and Networks-on-Chip

- Platform-Based Design:
  - Re-use of parametric modules (*Intellectual Properties*)
  - High-level of abstraction
  - *Communication*-centric: from buses to networks
- Solves the communication issues
- The components are connected in a communication network
- Advantages
  - Scalable
  - Parallelism

Formal Methods and Networks-on-Chips

- System Verification:
  - Proof of each component
  - Proof of their interconnection
- State-of-the-Art:
  - Model checking or theorem proving of instances of systems
  - Often at hardware level (RTL)
- The GeNoC Approach:
  - A generic model for reasoning about NoCs
  - Reduces amount of the user interaction needed to prove properties on NoC instances

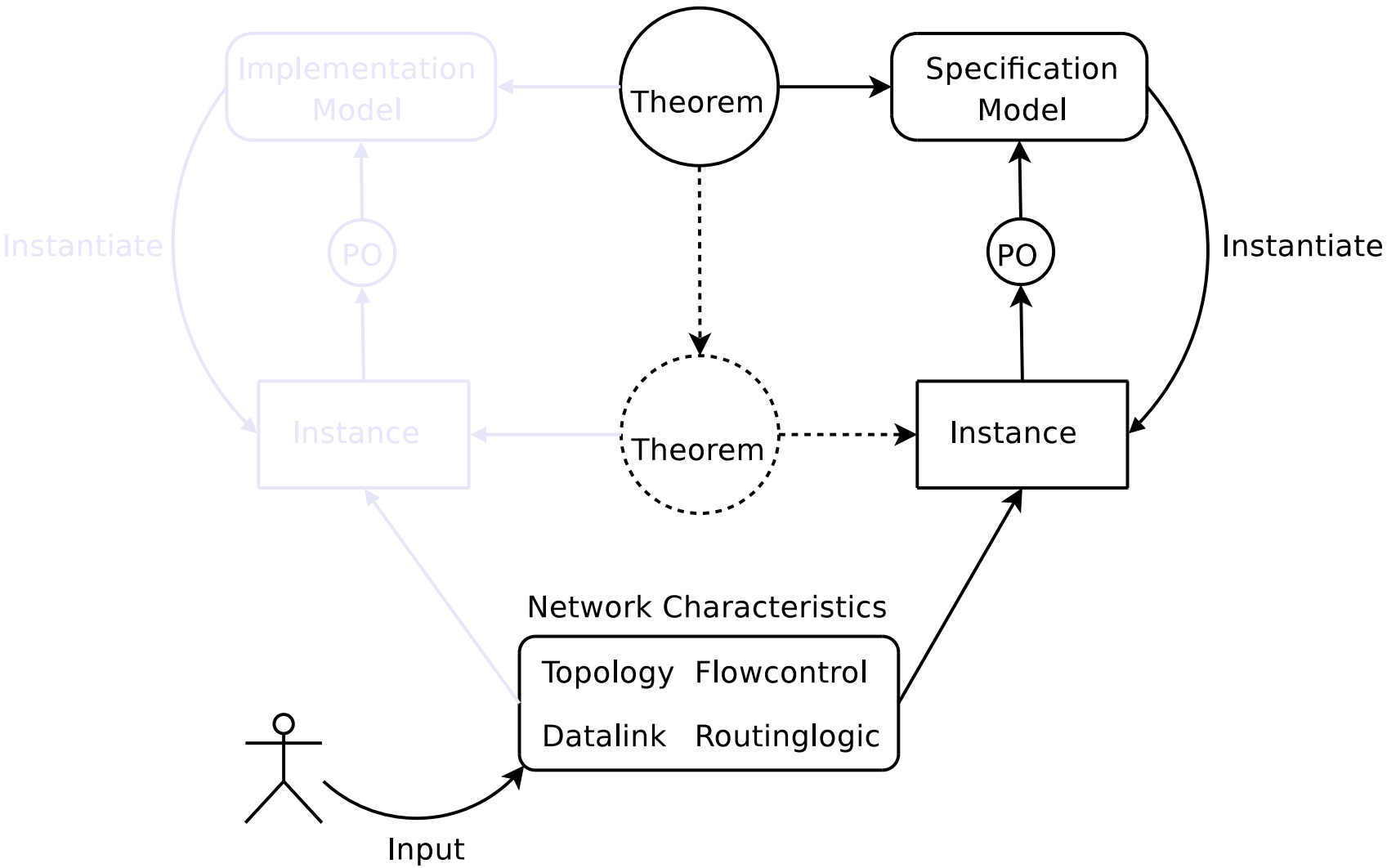
GeNoC approach



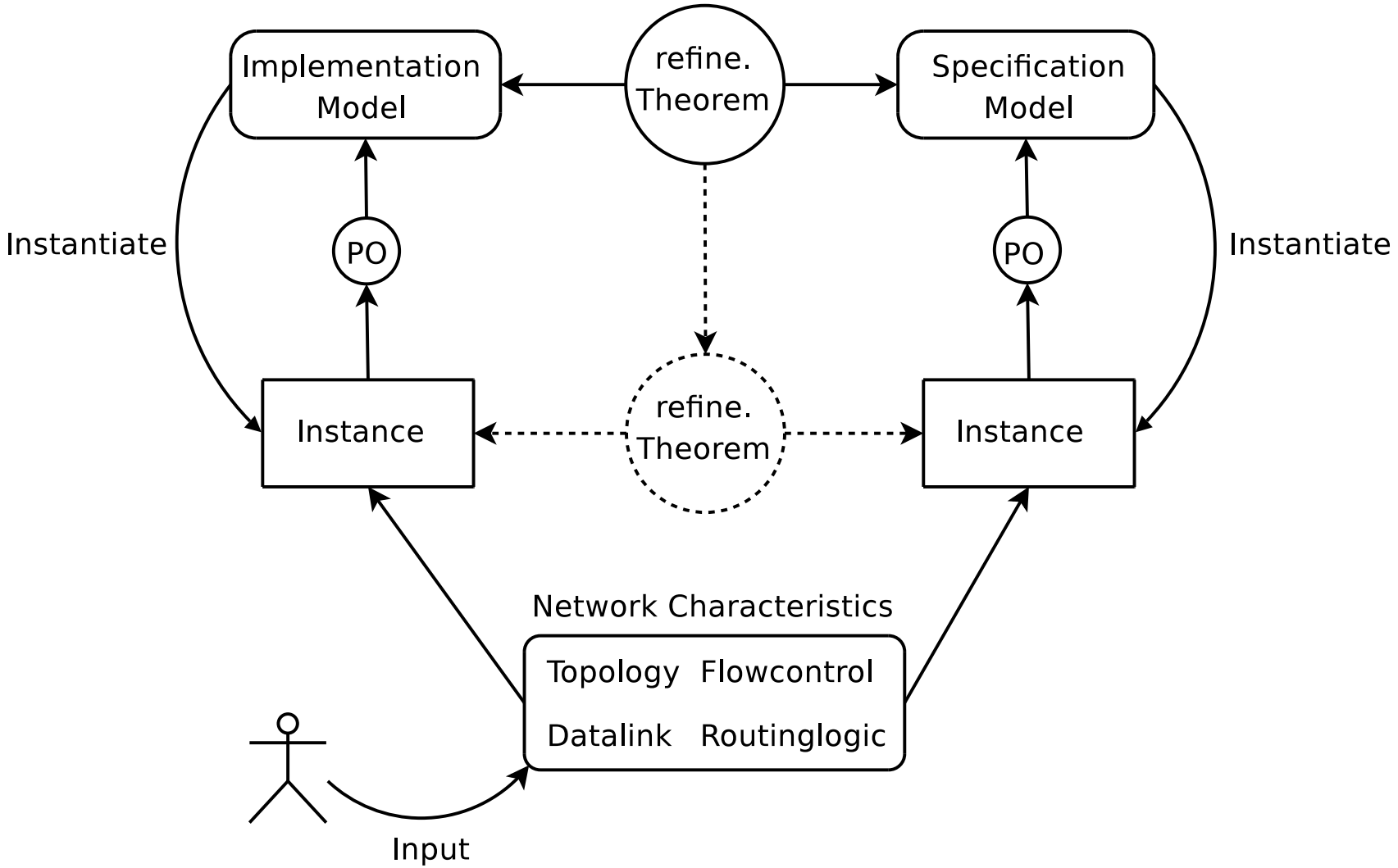
Contribution

- Original GeNoC Model
  - Highly abstract representation of the communications
  - The model has access to the complete precomputed routes the messages will traverse in the network
  - How does the specification level relates to the implementation level?
- Contribution
  - A generic implementation model
  - A (generic) specification model
  - A refinement proof between two instances of these models

Method - Specification model



Method - Contribution

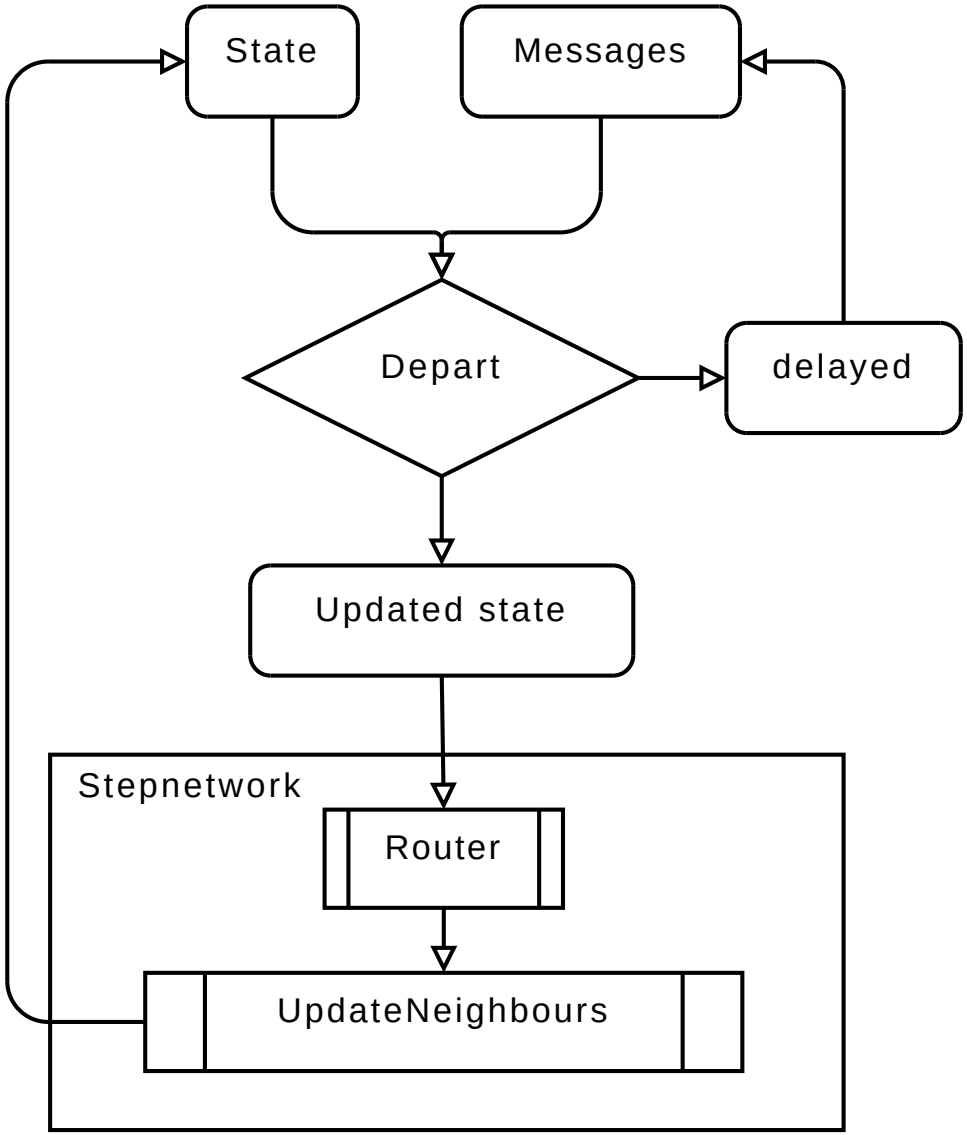


Structure of the two models

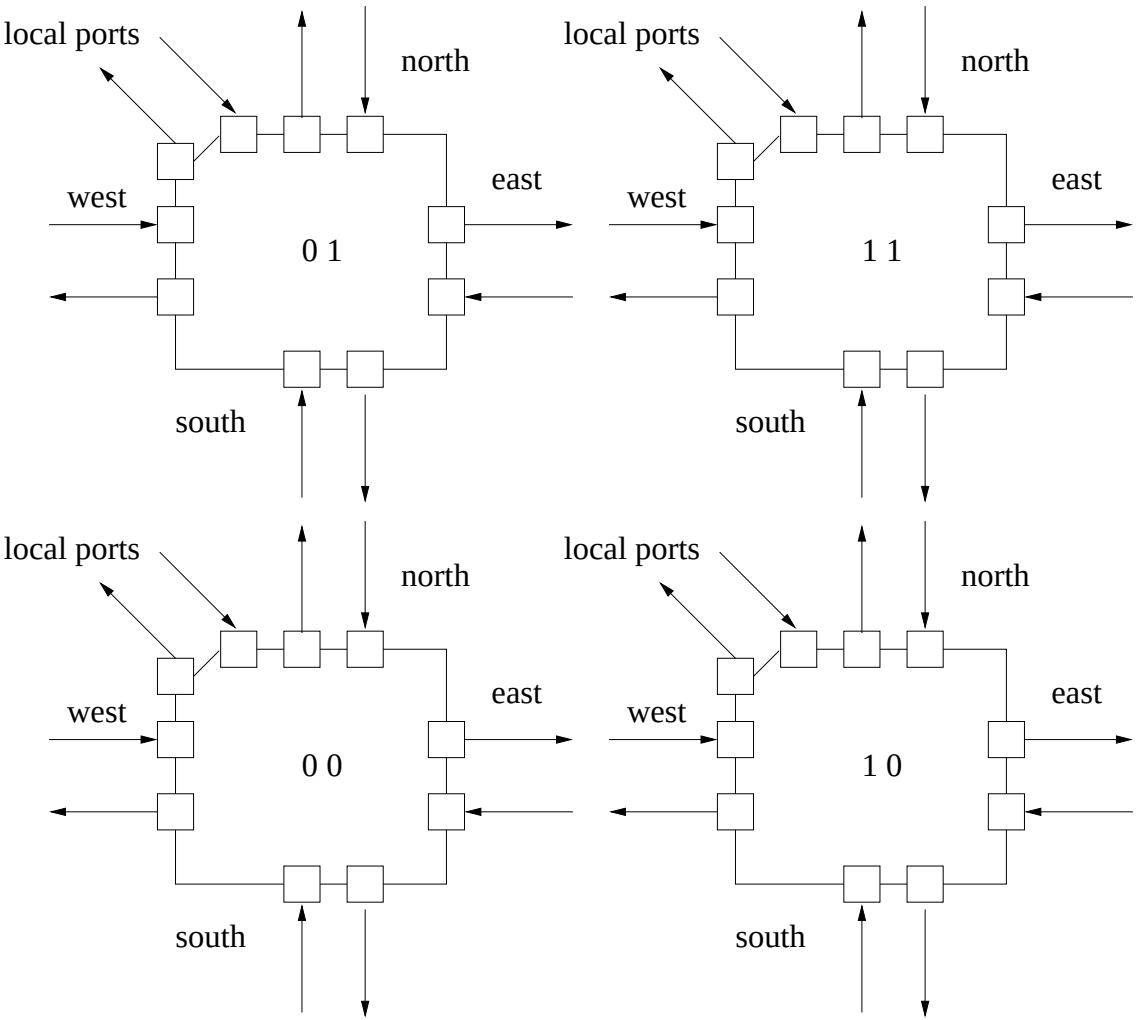
Both models consist of two main parts:

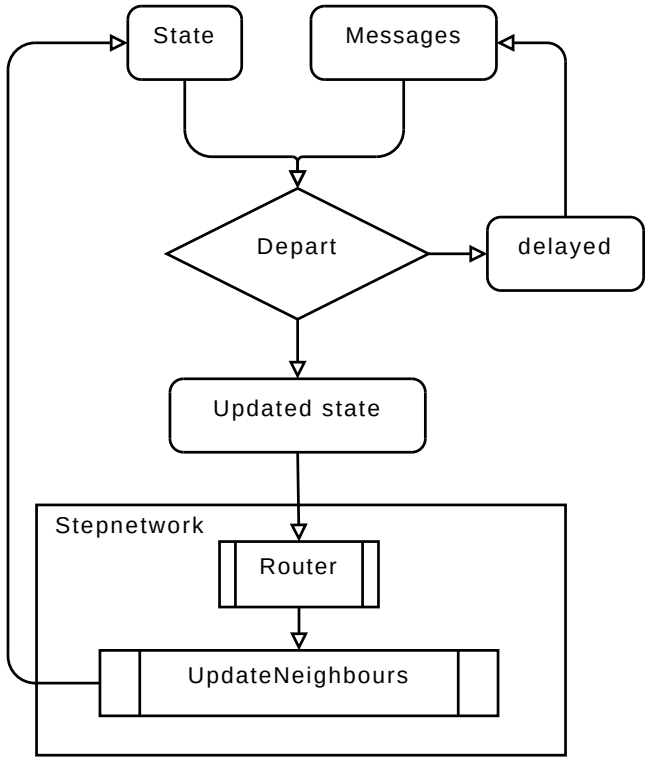
- The NoC characteristics are defined in the Network model
  - Topology
  - Router components:
    - Datalink
    - Routing
    - Scheduling
- The Network interpreter takes a network model and simulates the network
- Implemented in ACL2

The main interpreter structure

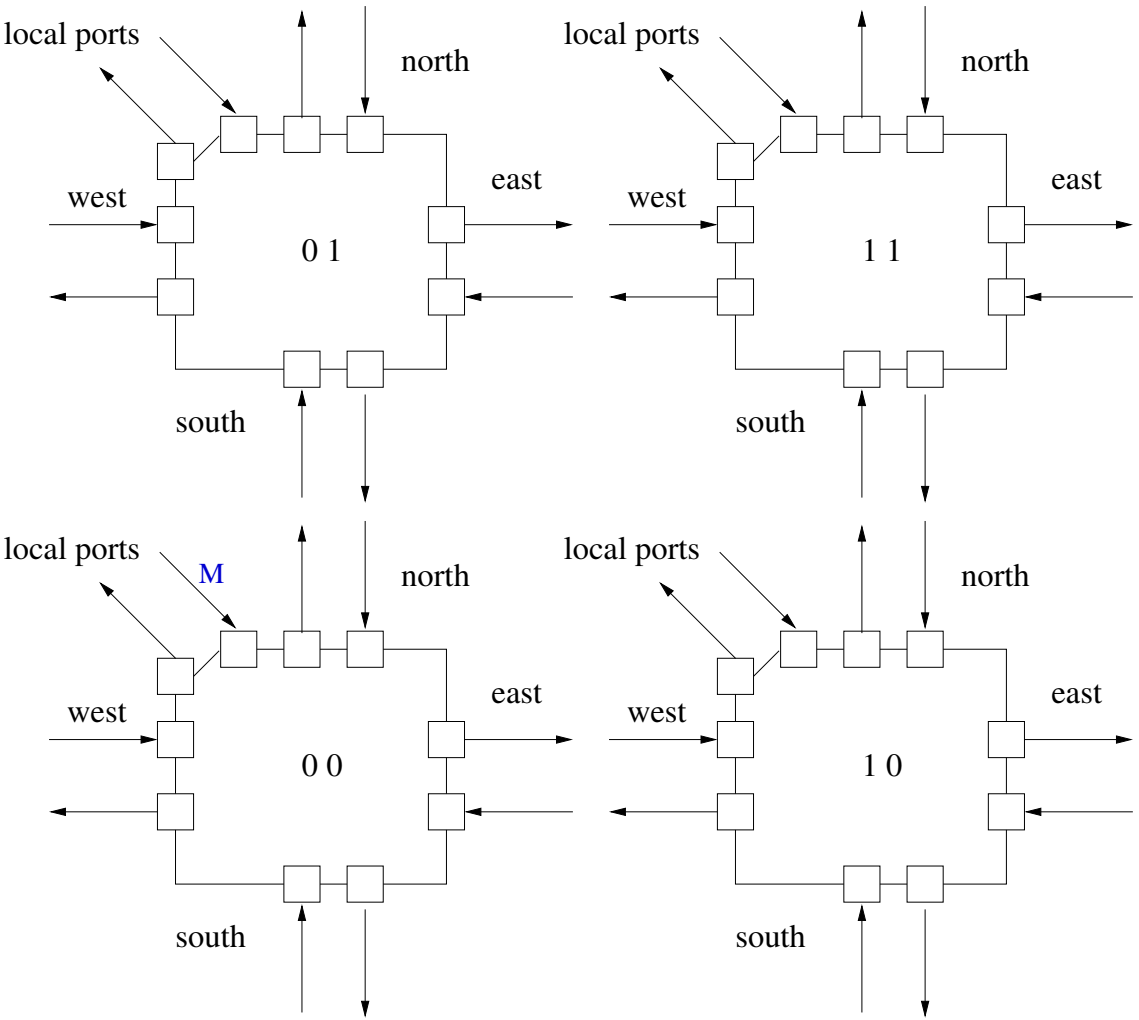


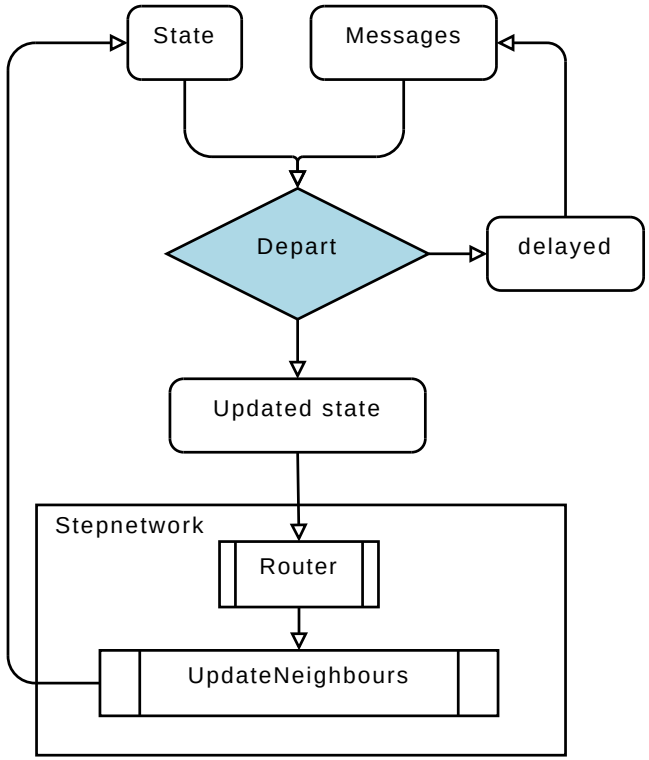
Network interpreter – Implementation Level



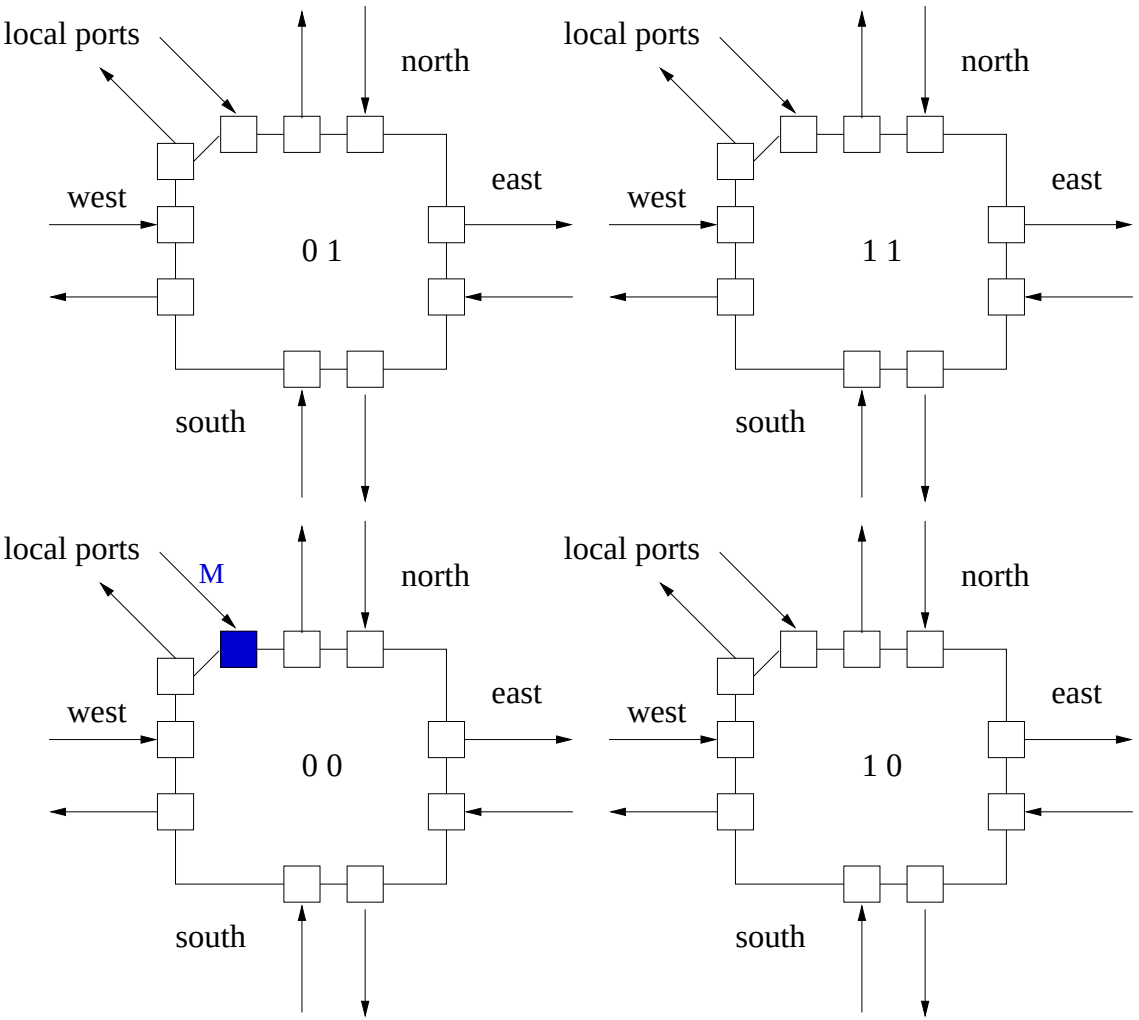


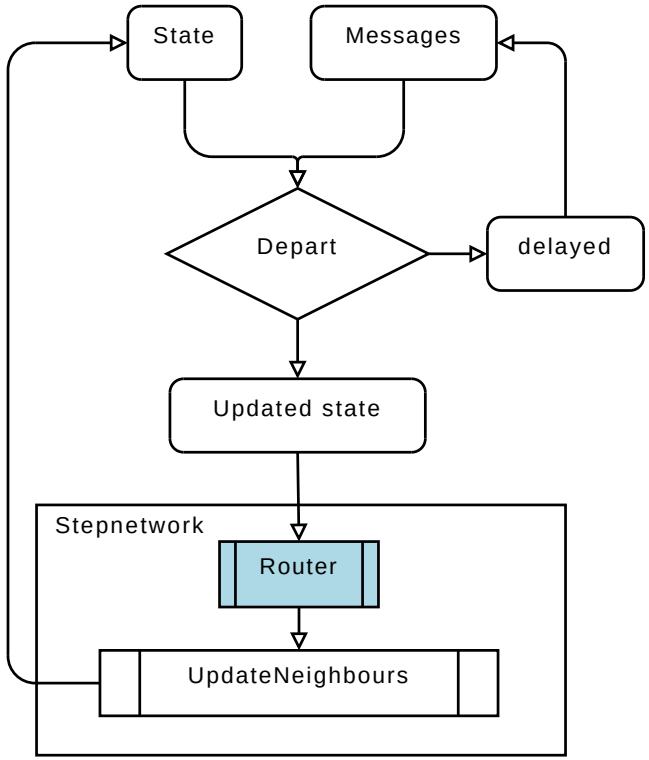
Network interpreter – Implementation Level





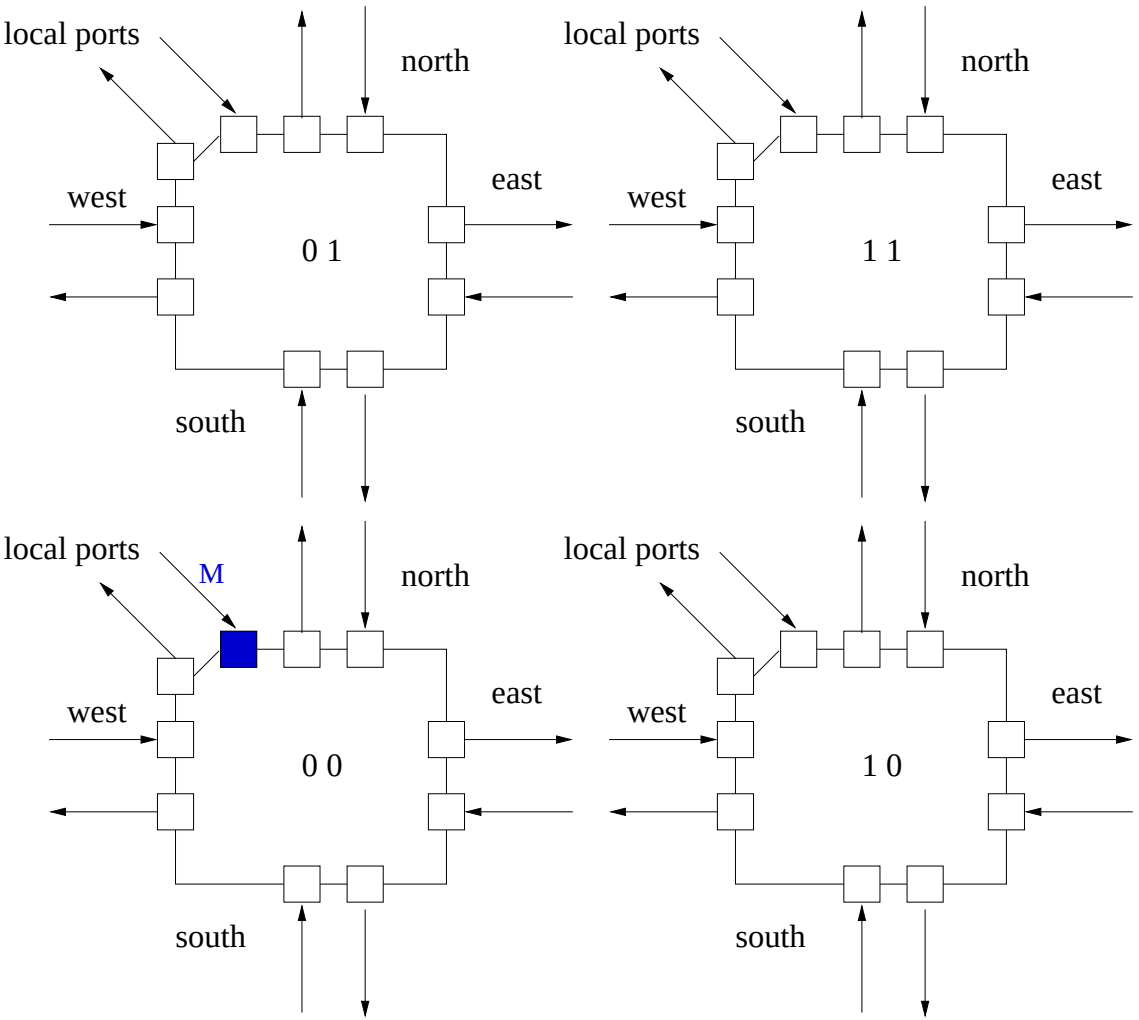
Network interpreter – Implementation Level

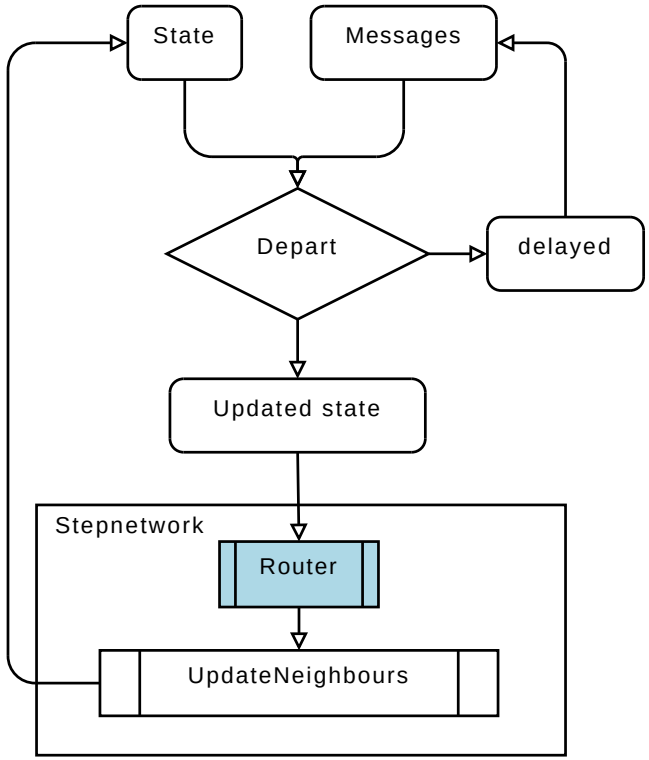




`ProcessInputs`

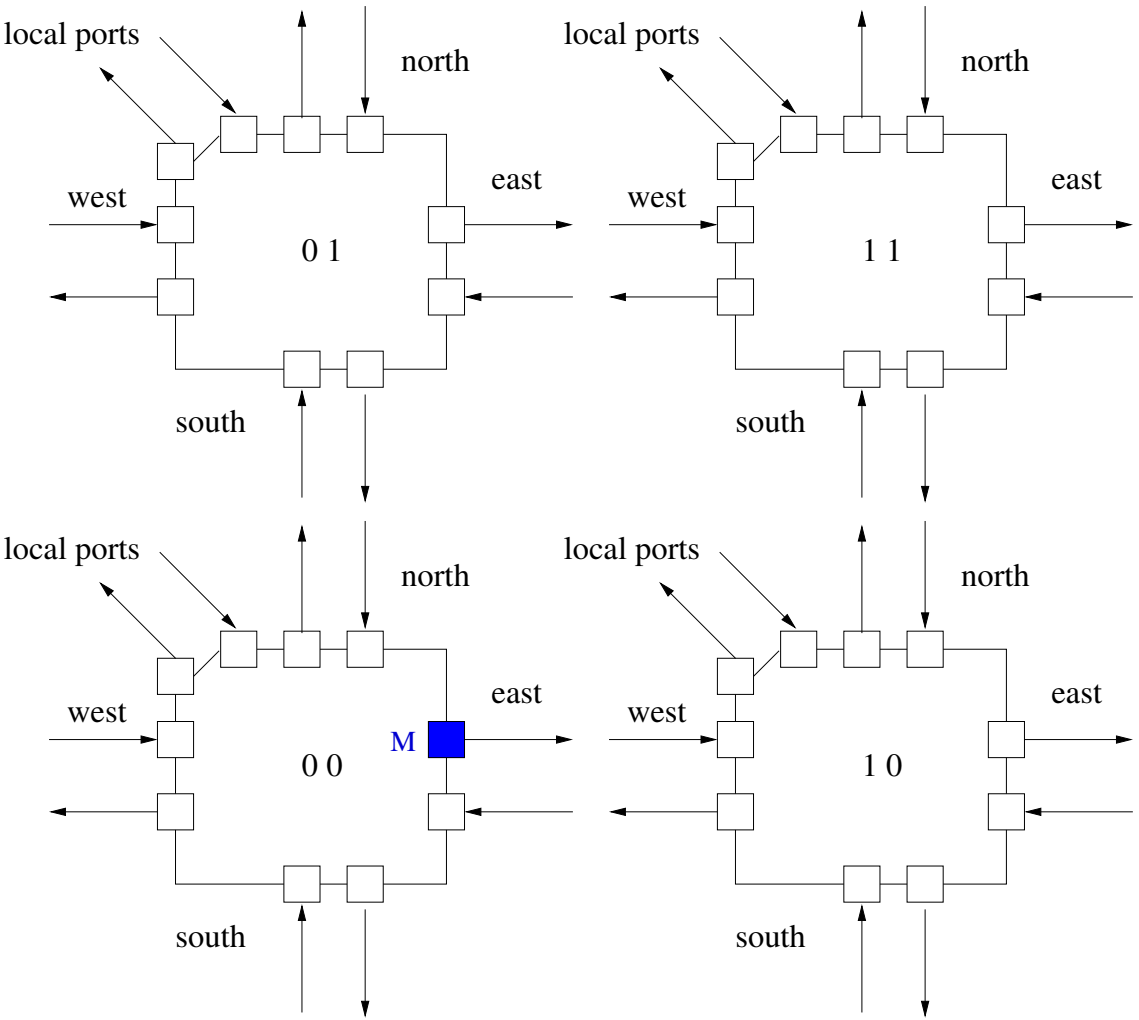
Network interpreter – Implementation Level

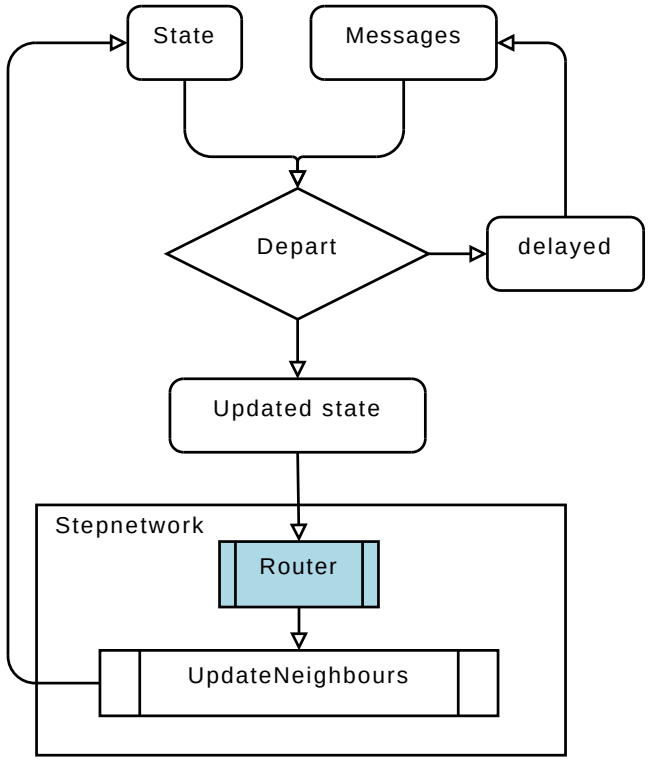




RouteControl

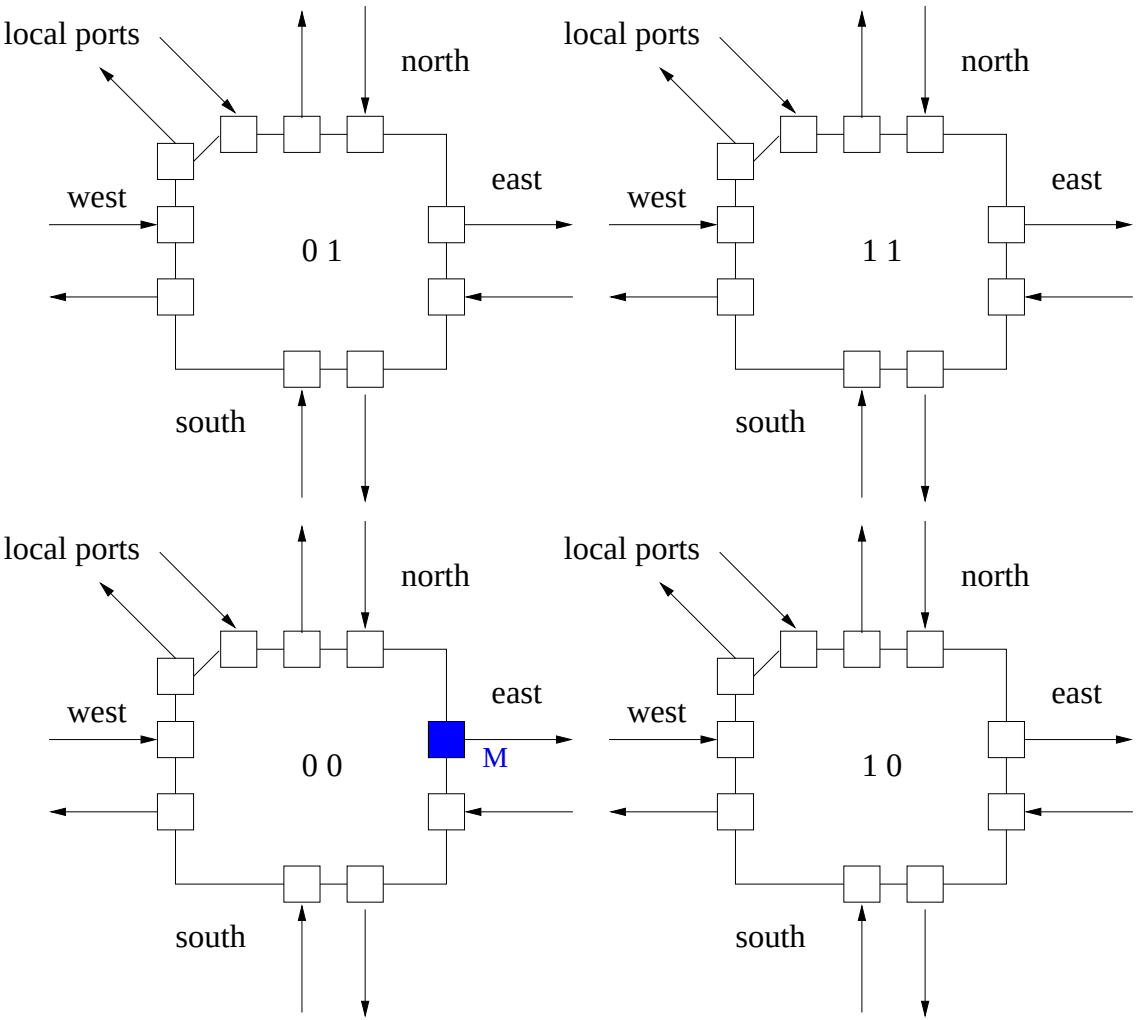
Network interpreter – Implementation Level

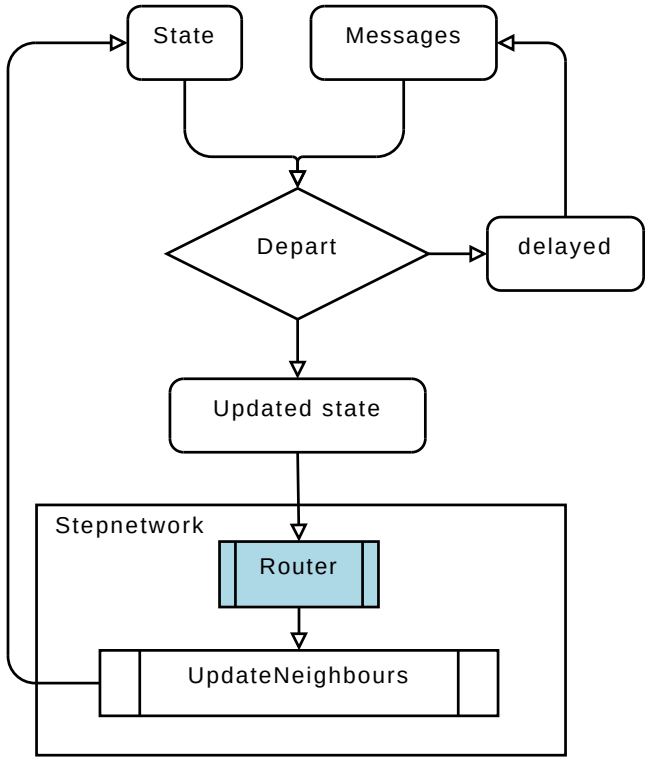




FlowControl

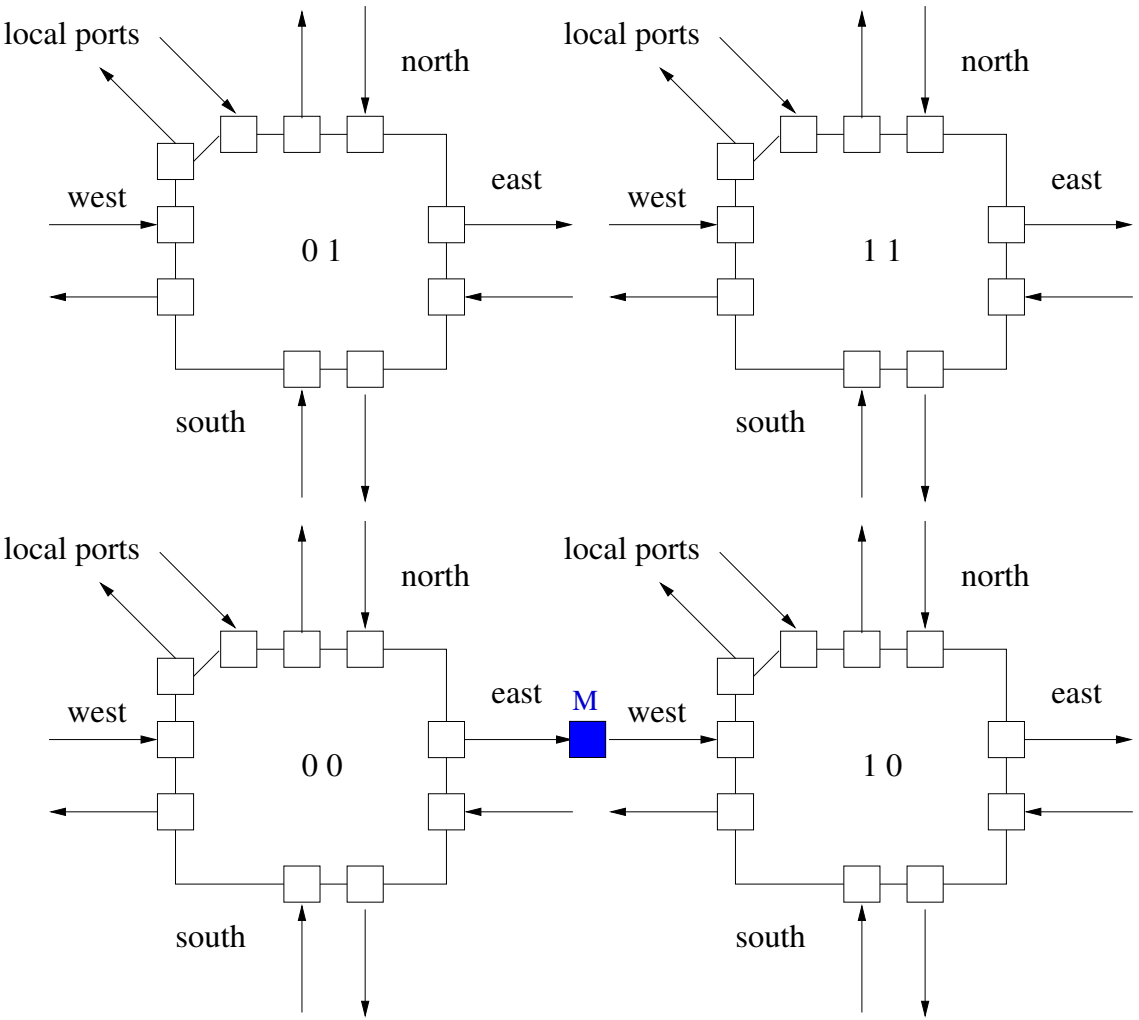
Network interpreter – Implementation Level

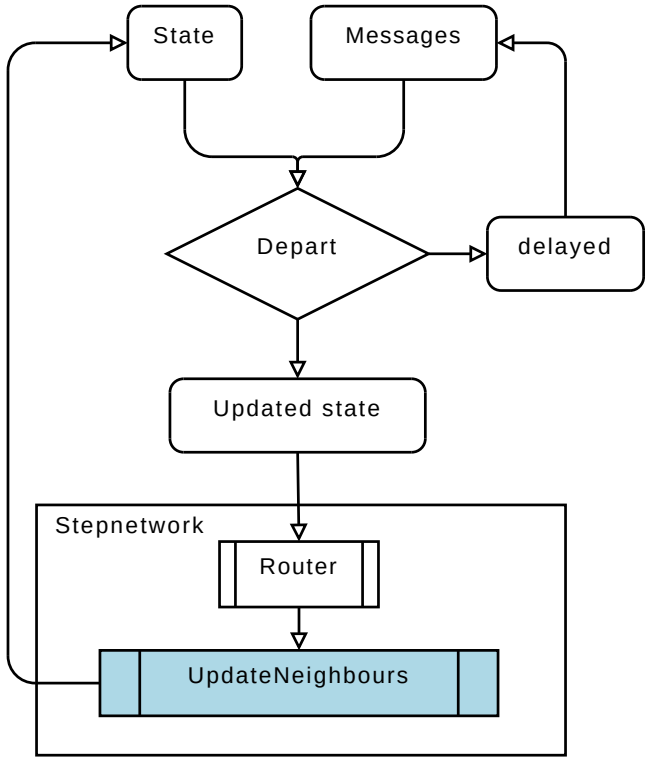




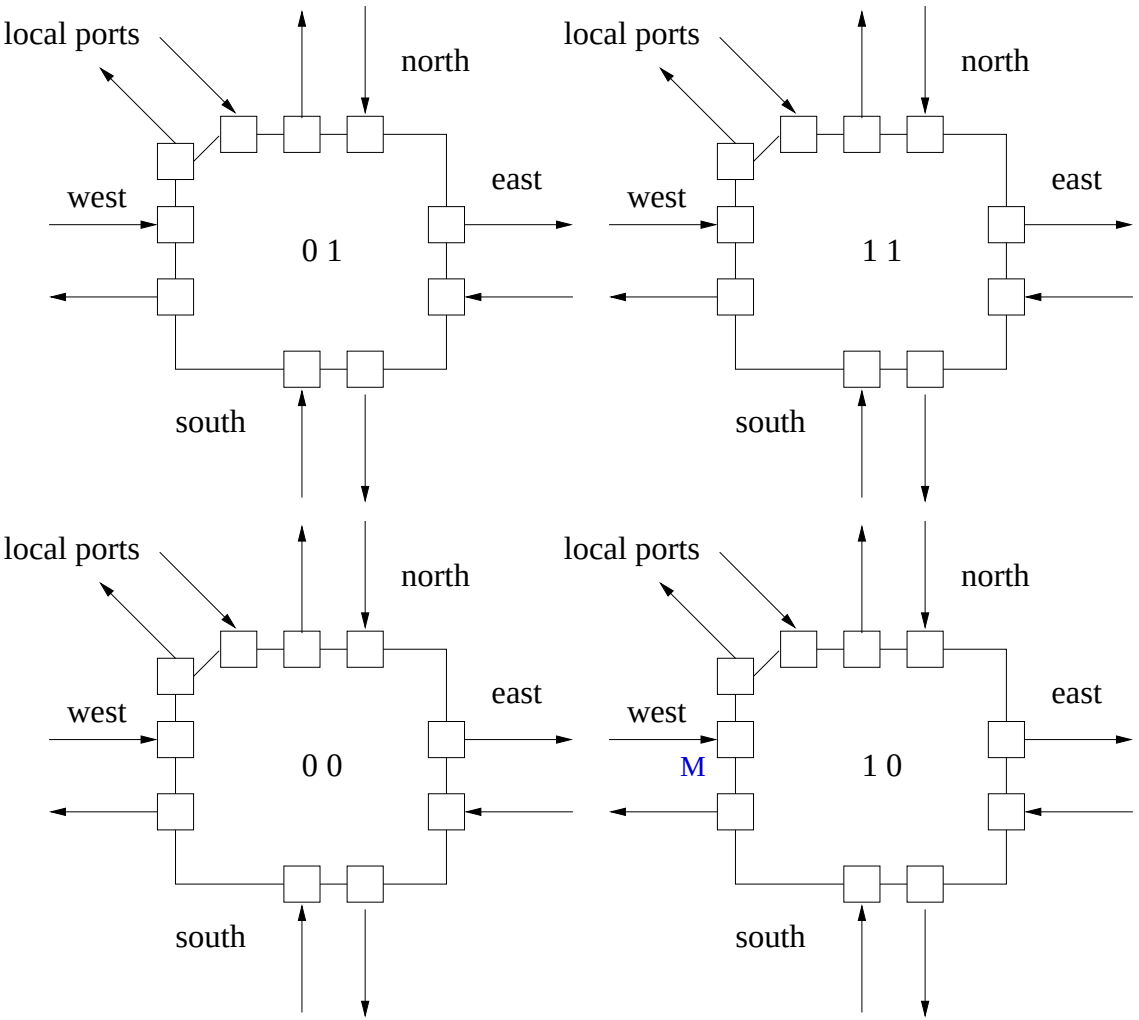
`ProcessOutputs`

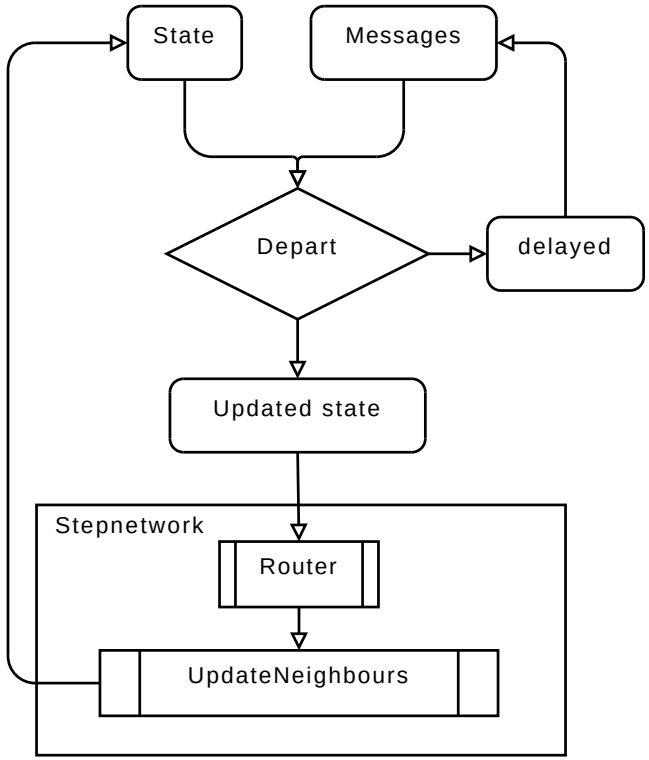
Network interpreter – Implementation Level



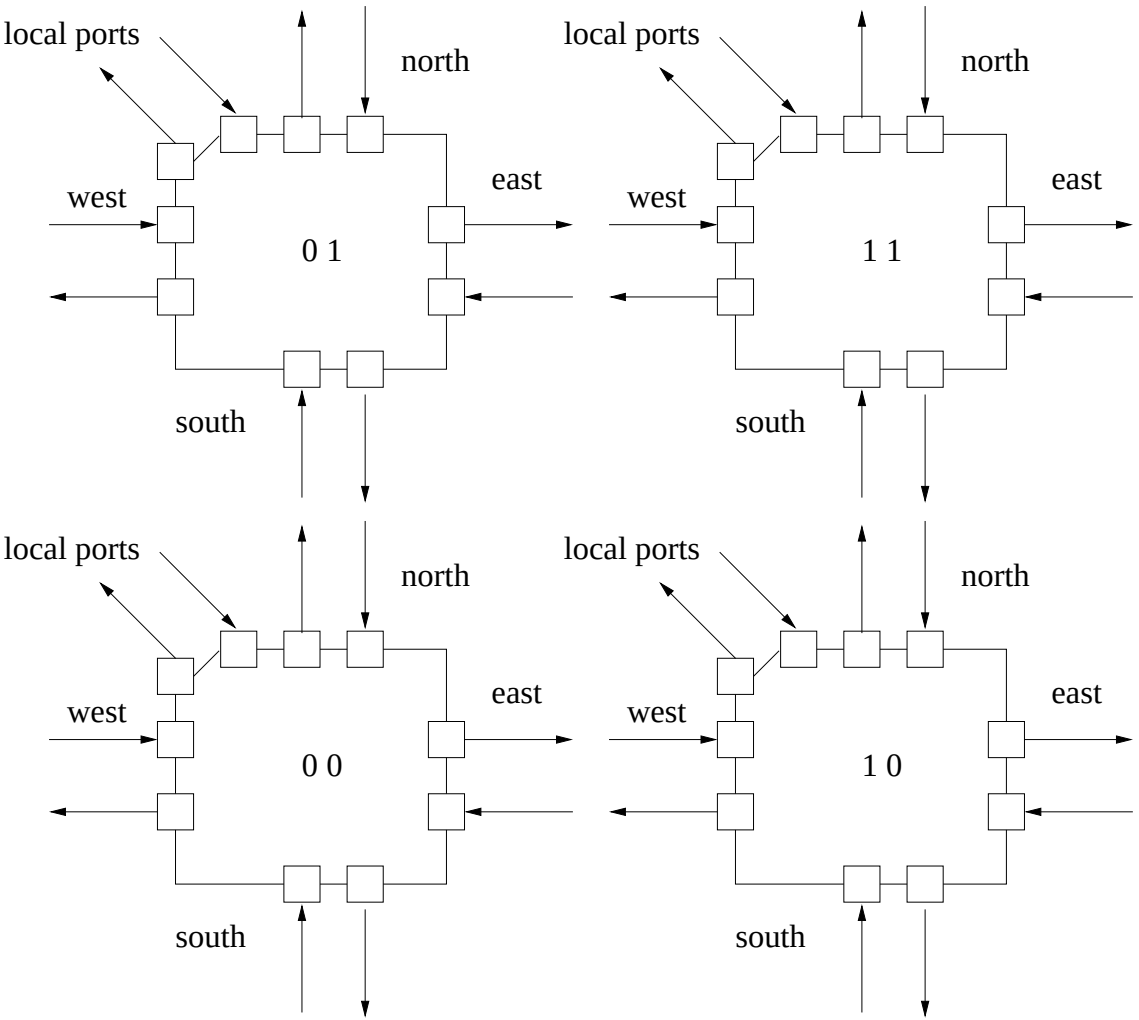


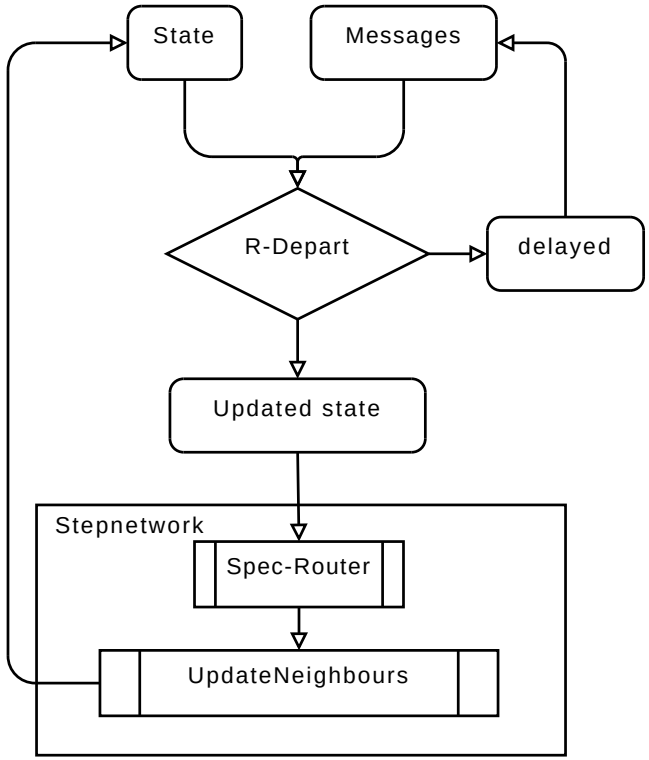
Network interpreter – Implementation Level



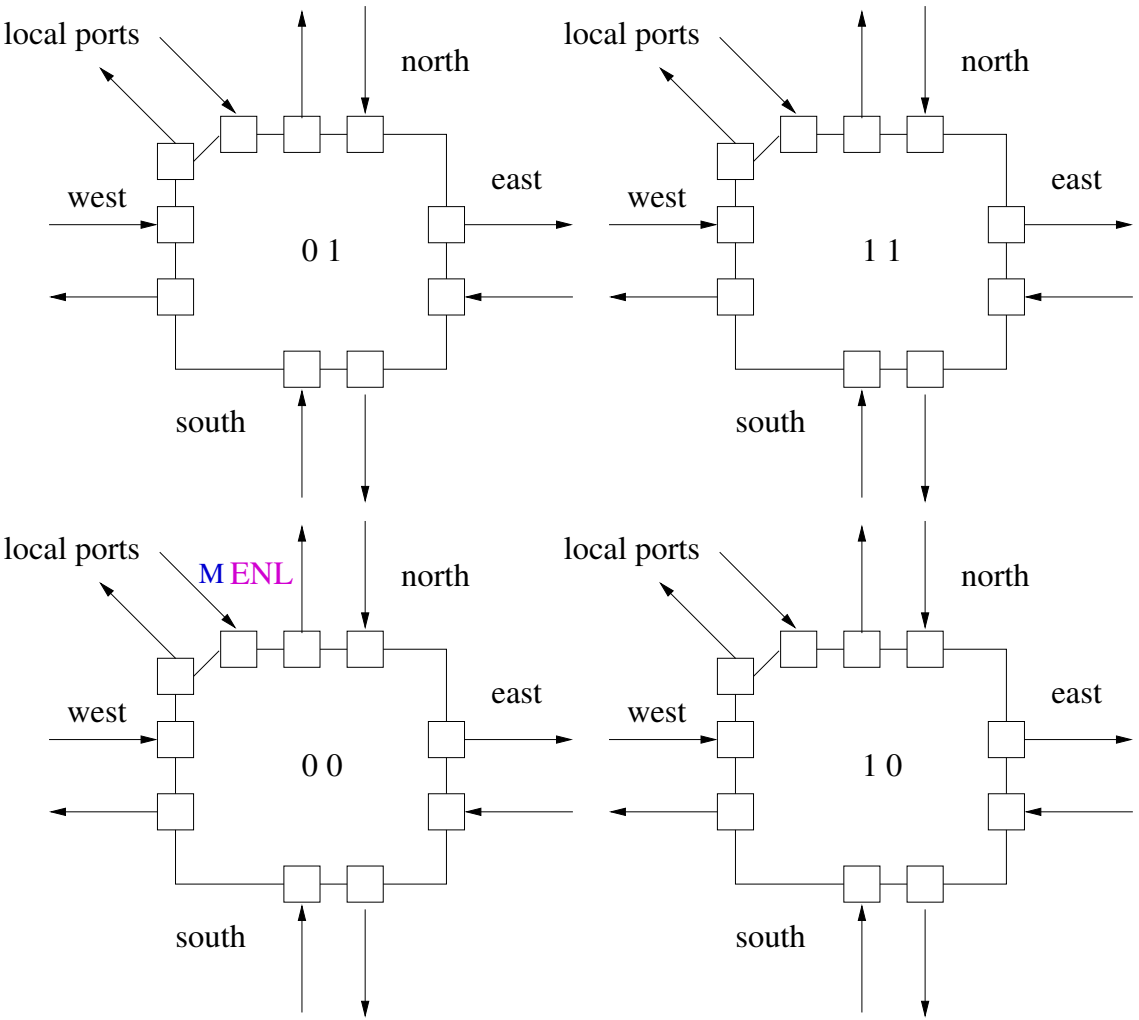


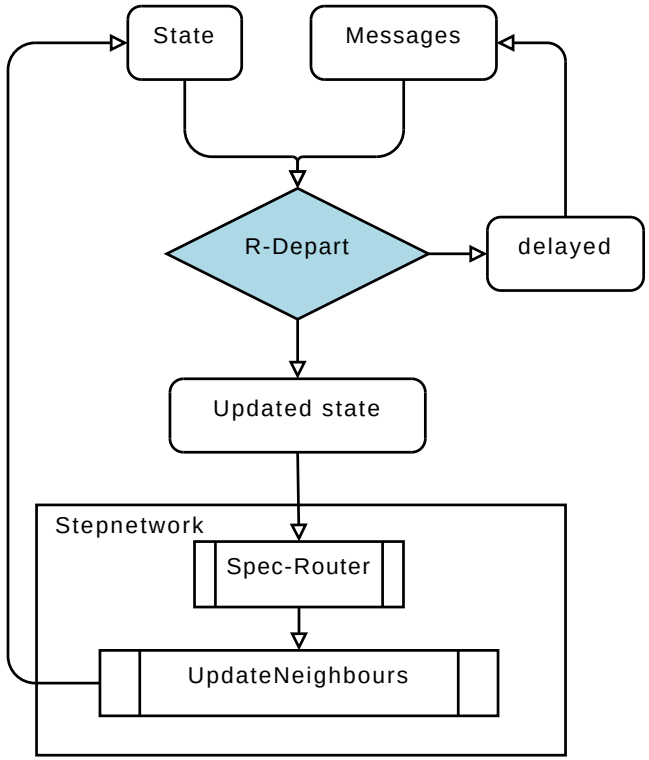
Network interpreter – Specification Level



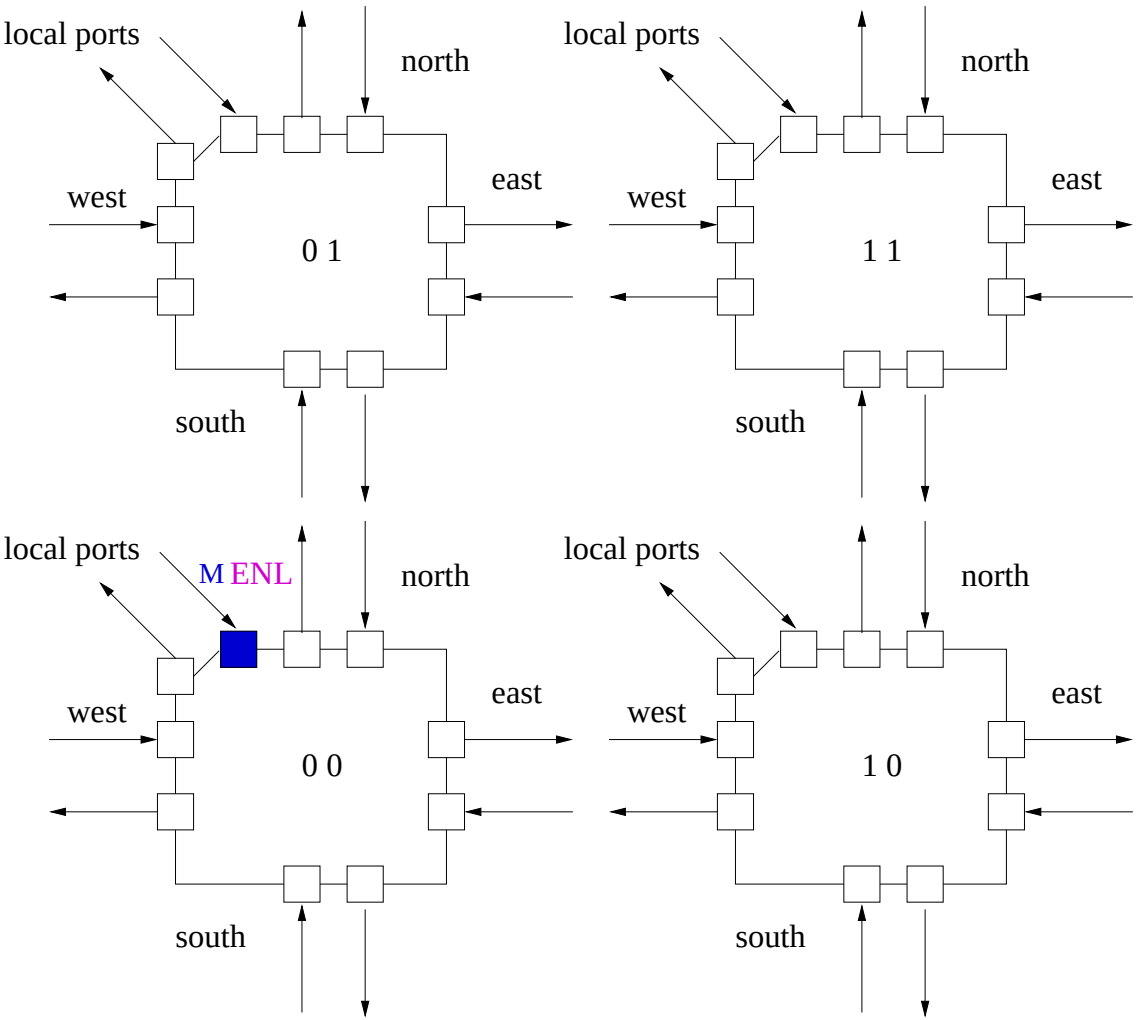


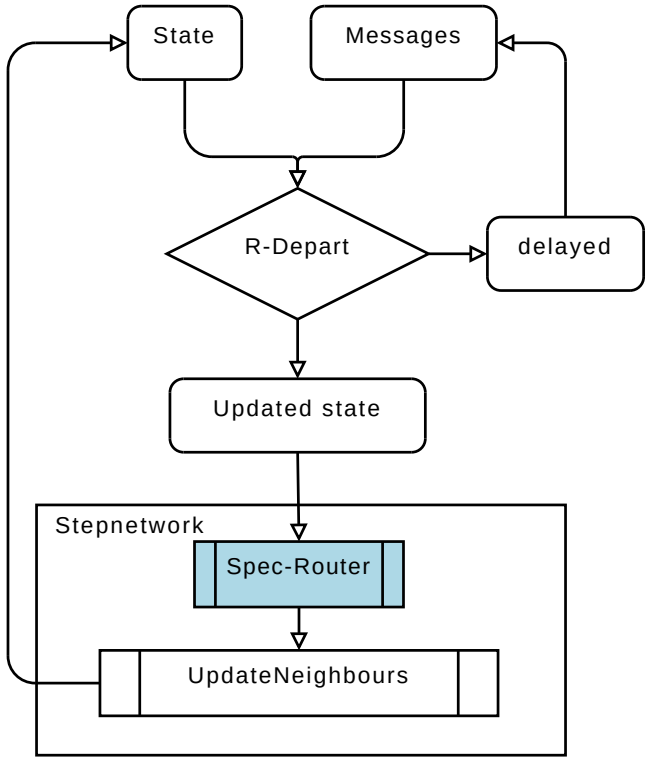
Network interpreter – Specification Level



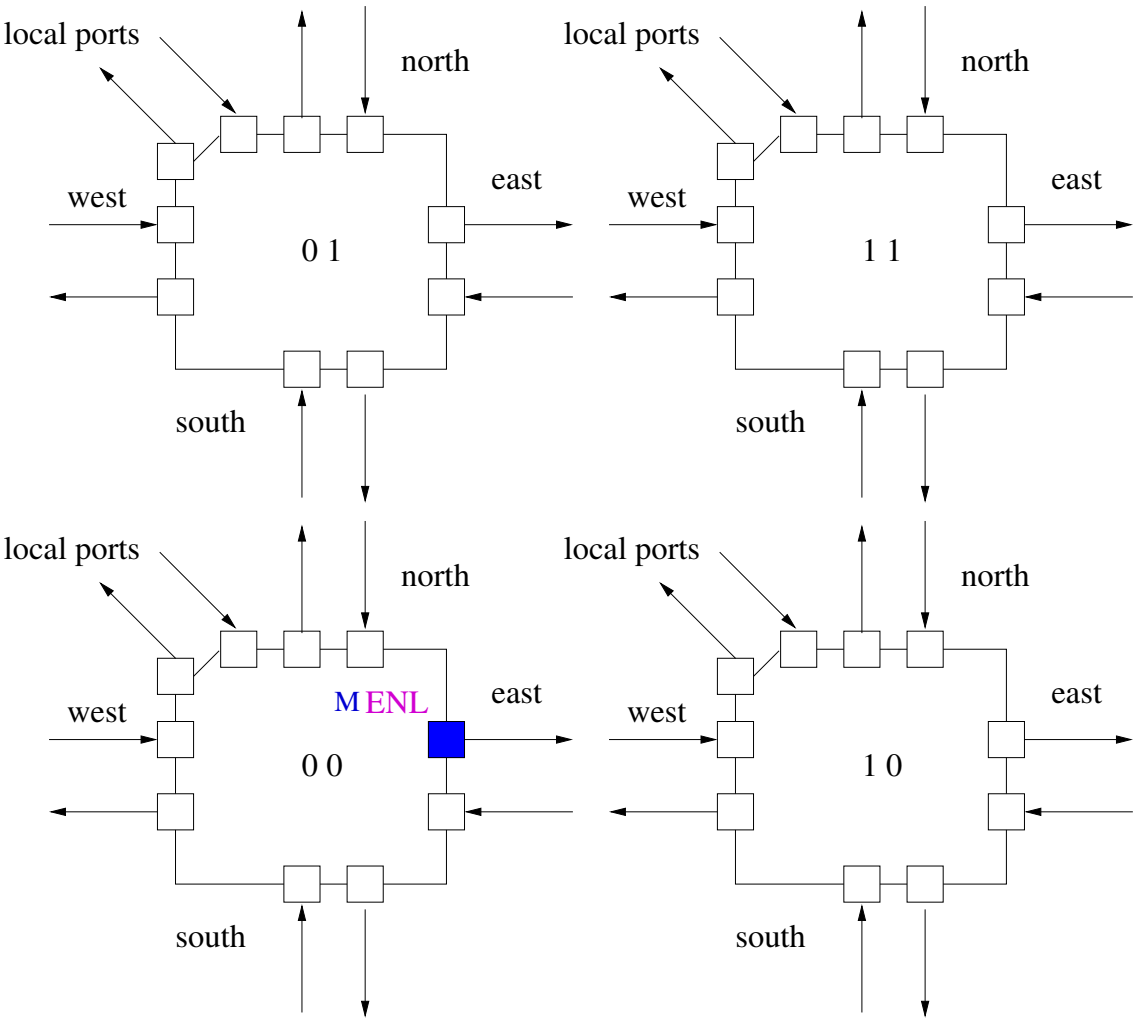


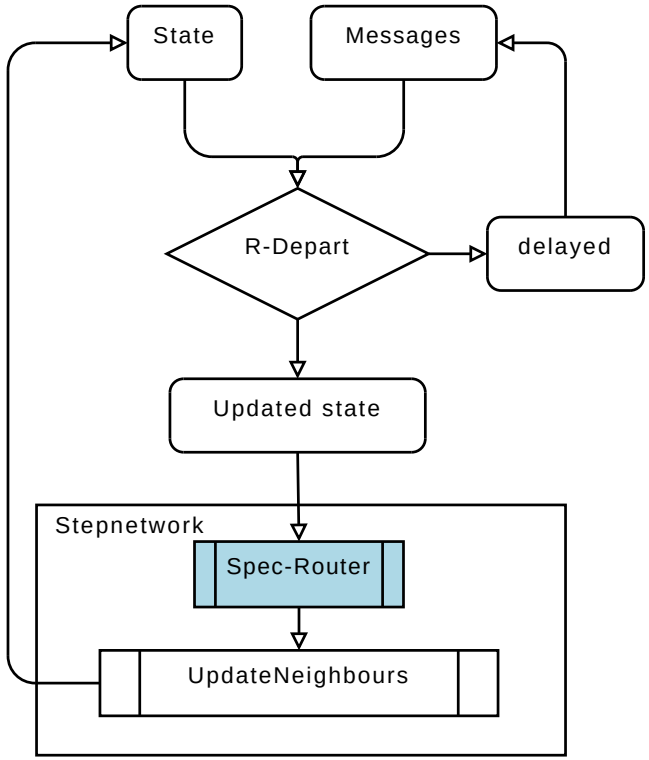
Network interpreter – Specification Level



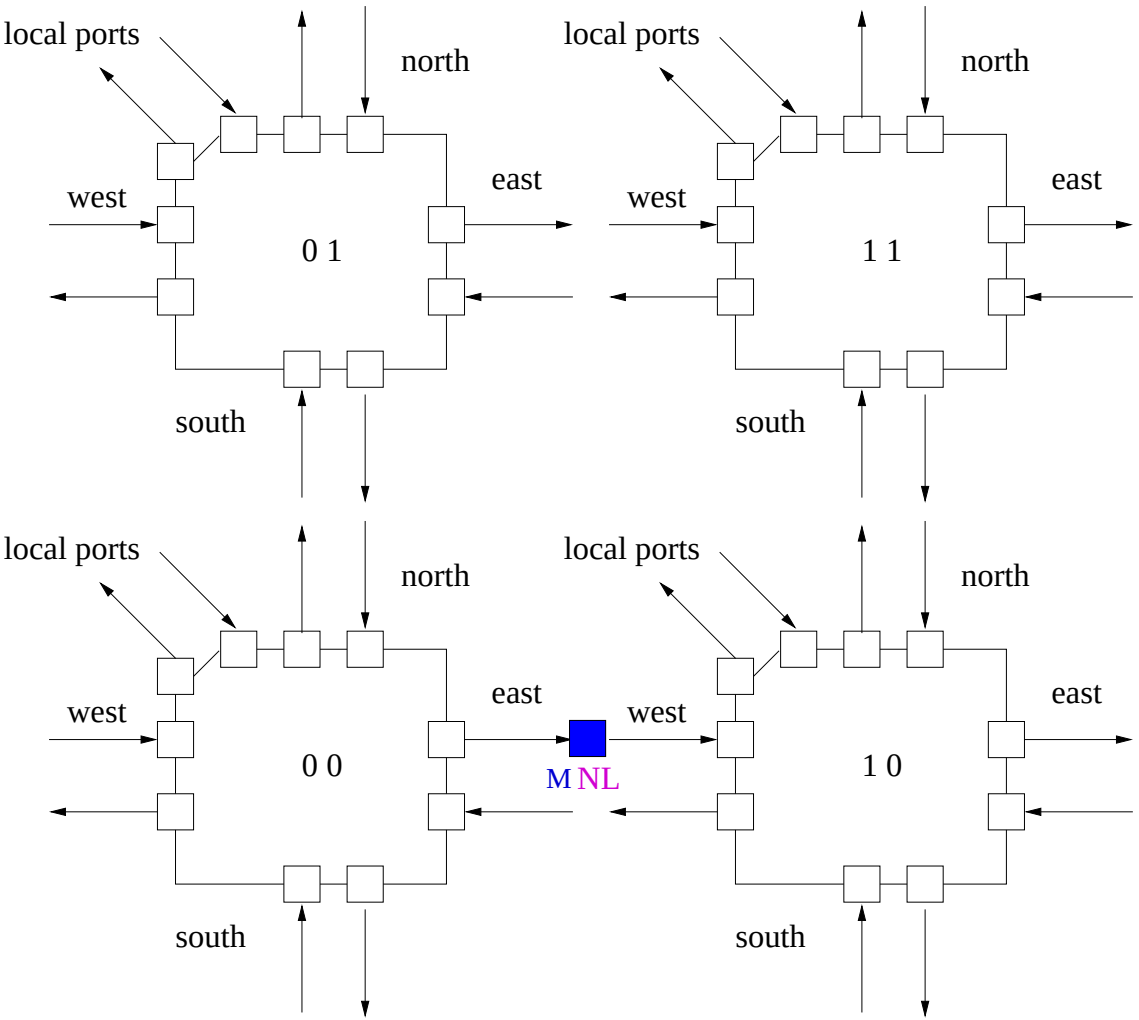


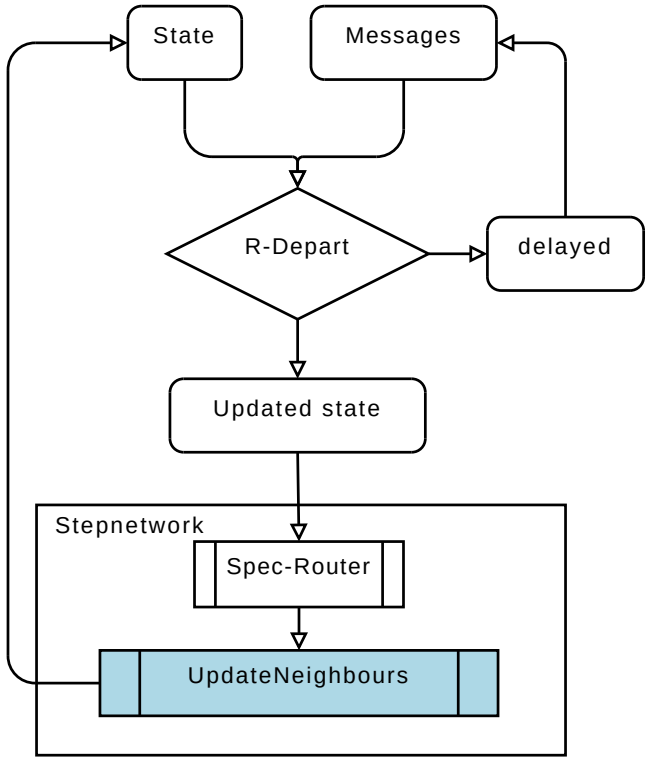
Network interpreter – Specification Level



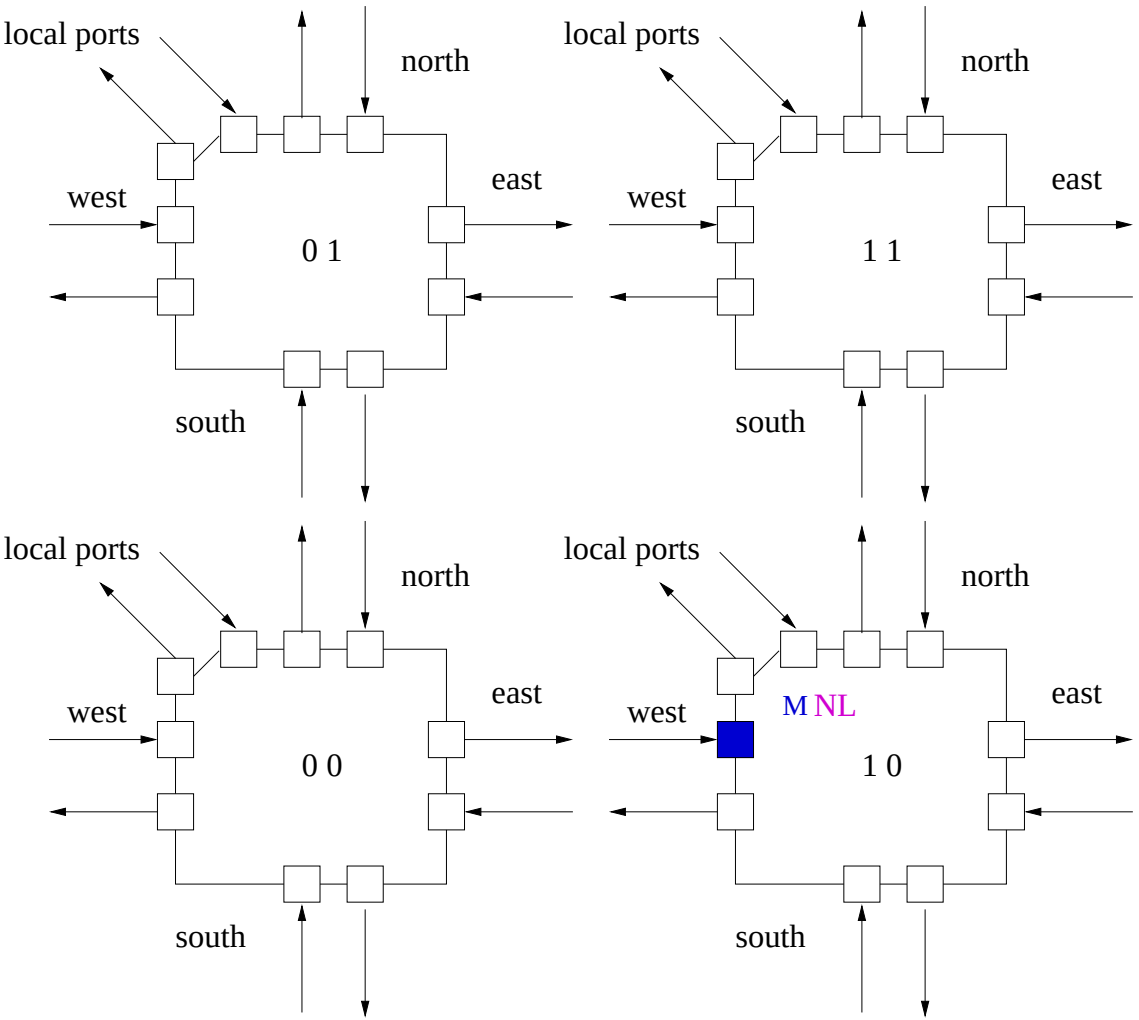


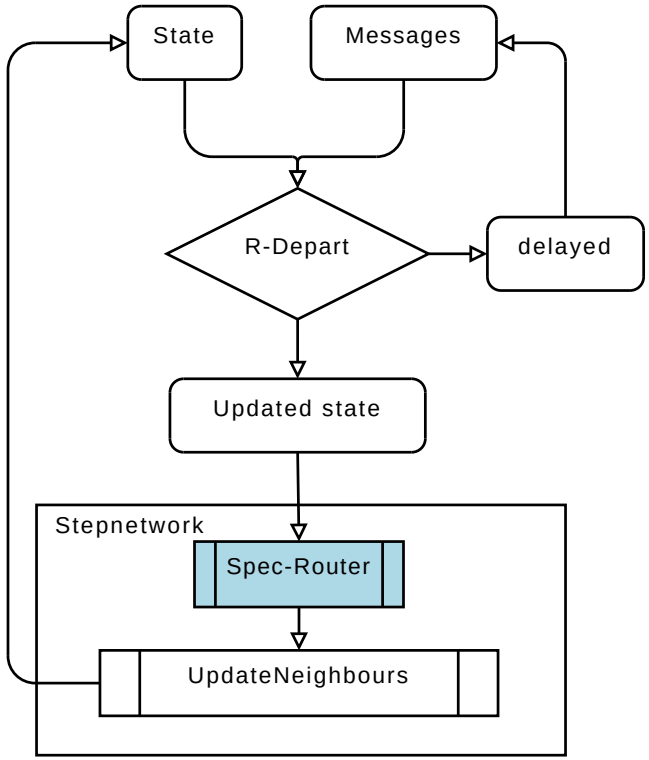
Network interpreter – Specification Level



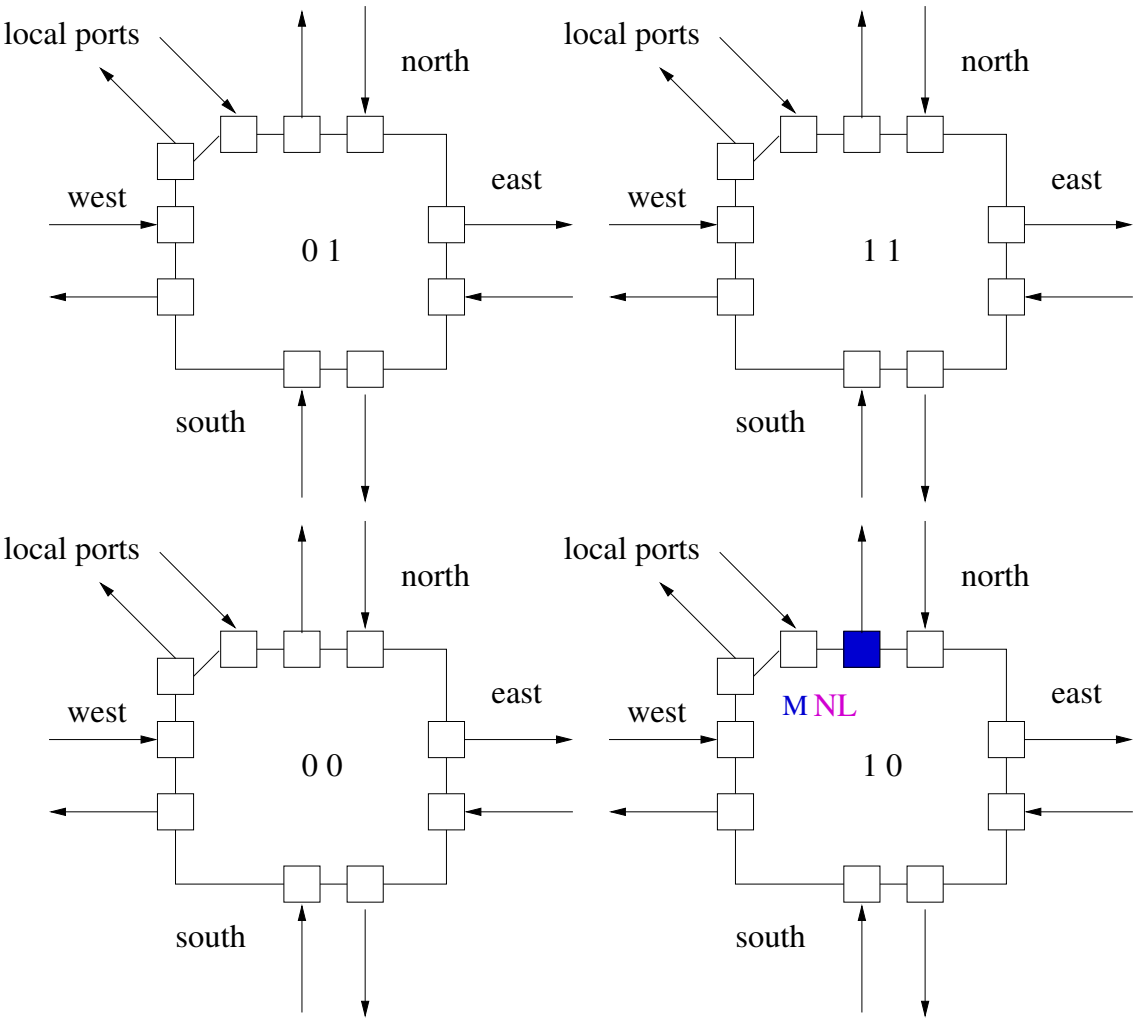


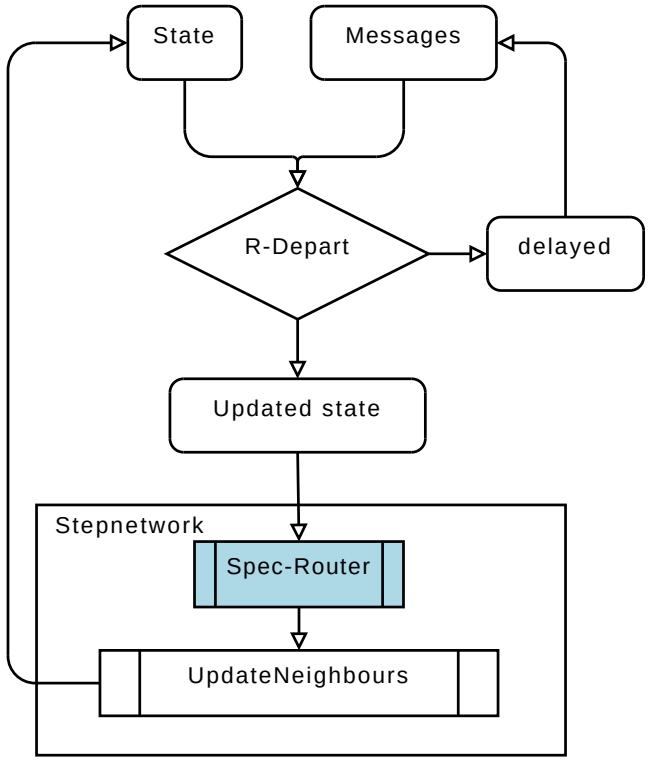
Network interpreter – Specification Level



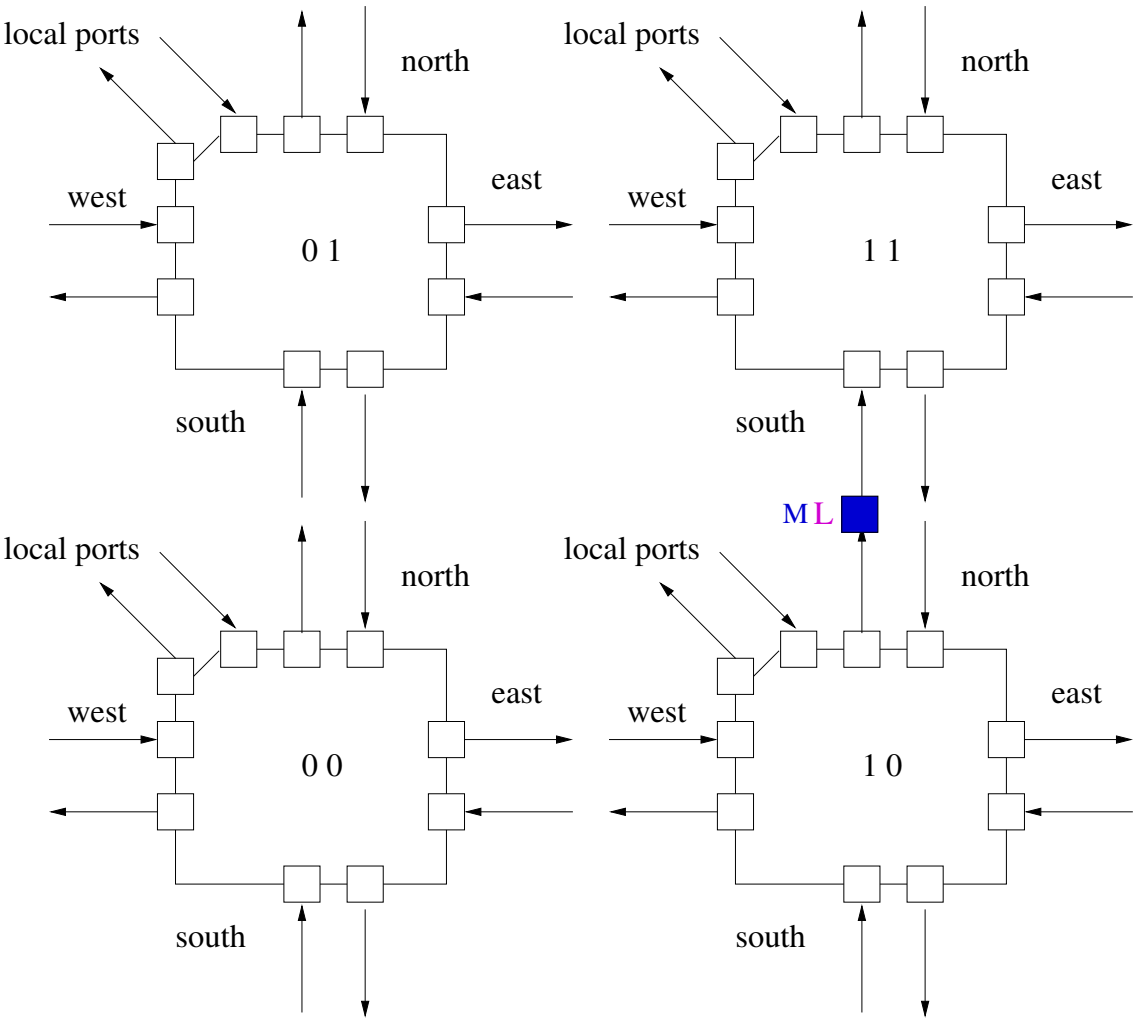


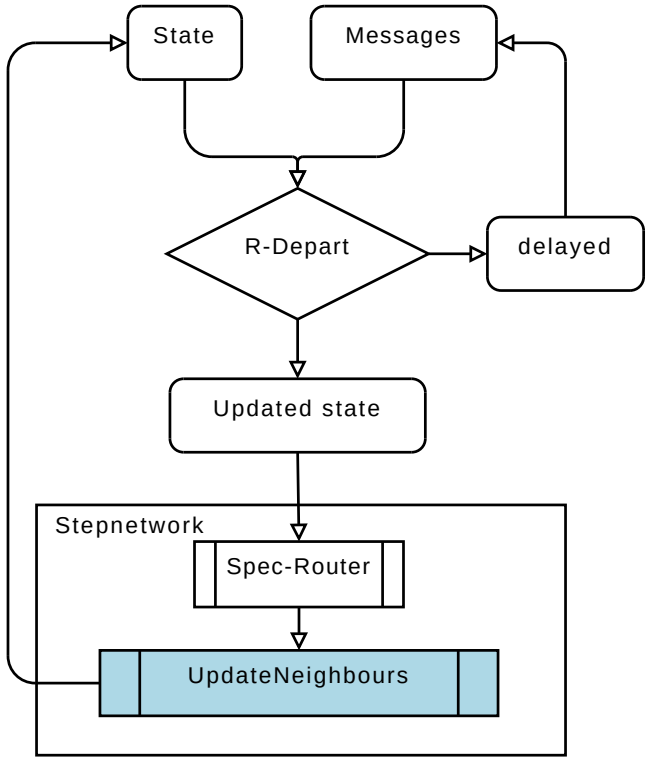
Network interpreter – Specification Level



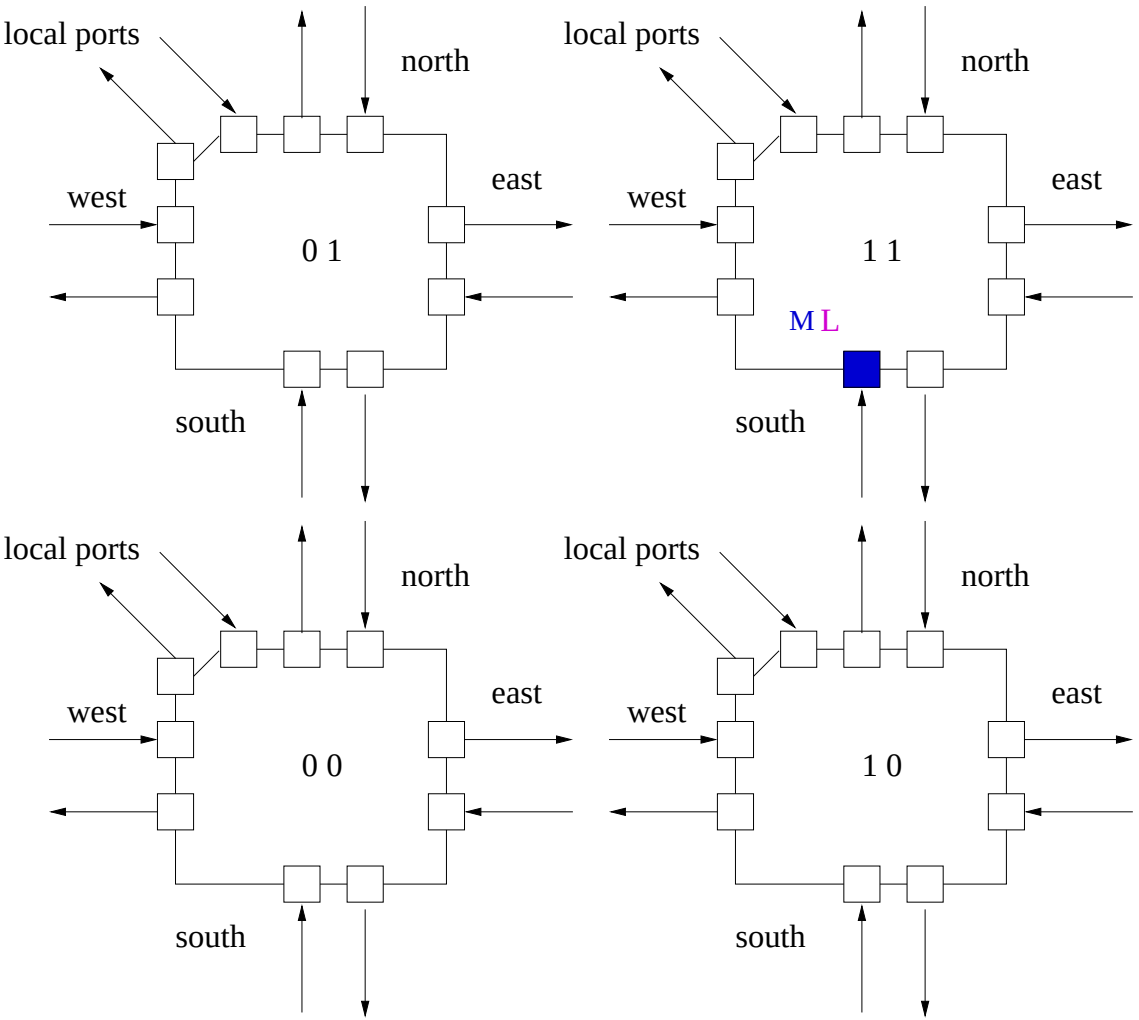


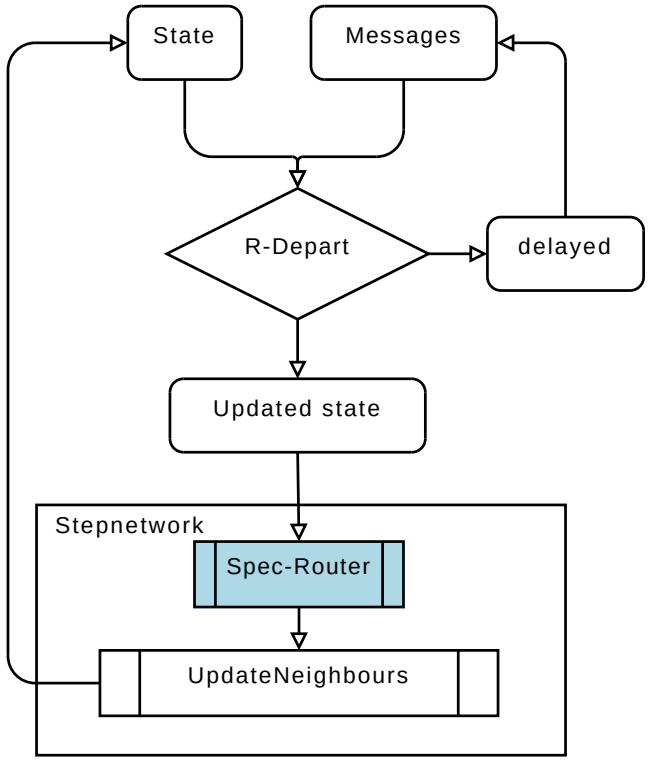
Network interpreter – Specification Level



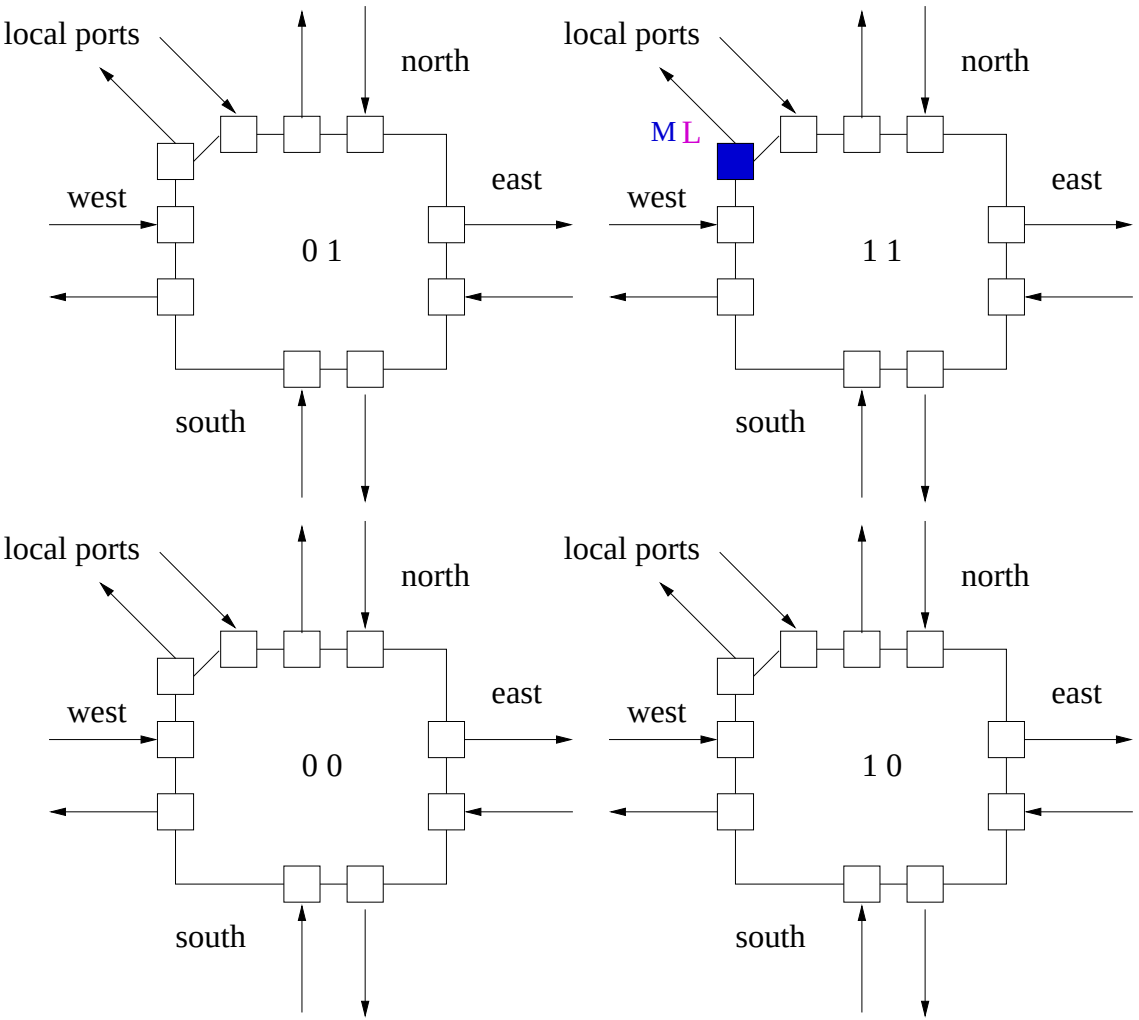


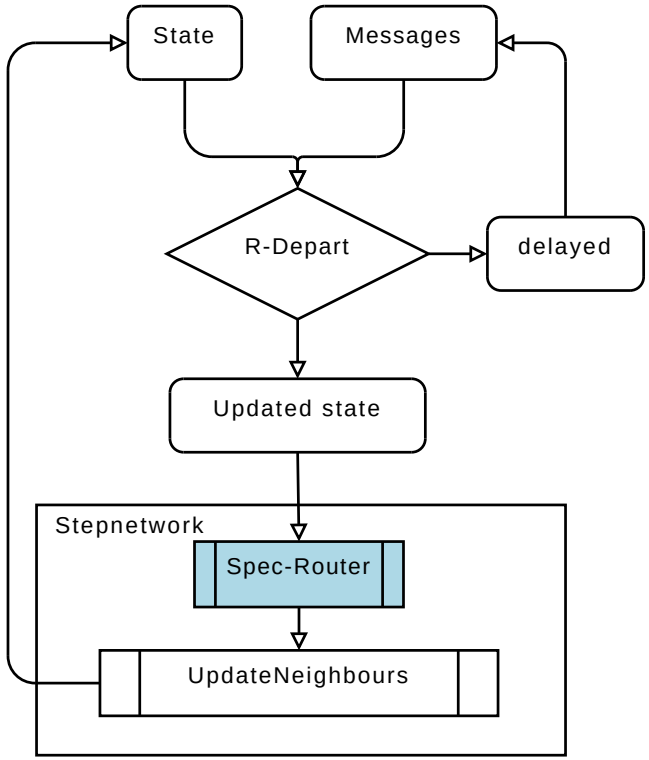
Network interpreter – Specification Level





Network interpreter – Specification Level



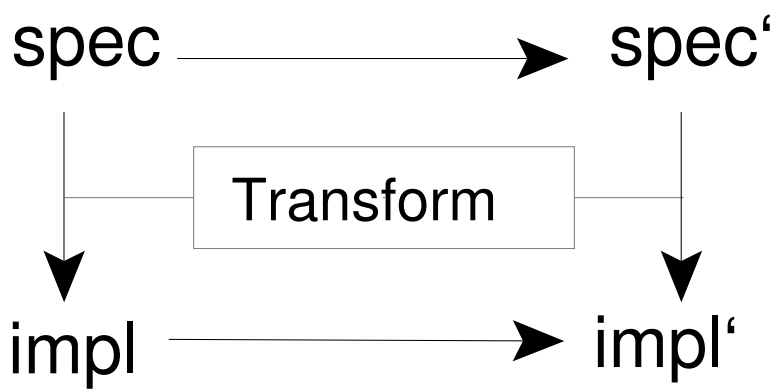


Proof concept

- The implementation model is a refinement of the specification model
  1. Given the same input the models should produce the same output
  2. The messages should traverse the same paths in the network

Proof concept

- The implementation model is a refinement of the specification model
  1. Given the same input the models should produce the same output
  2. The messages should traverse the same paths in the network



The Transform relation removes the routes from the network state

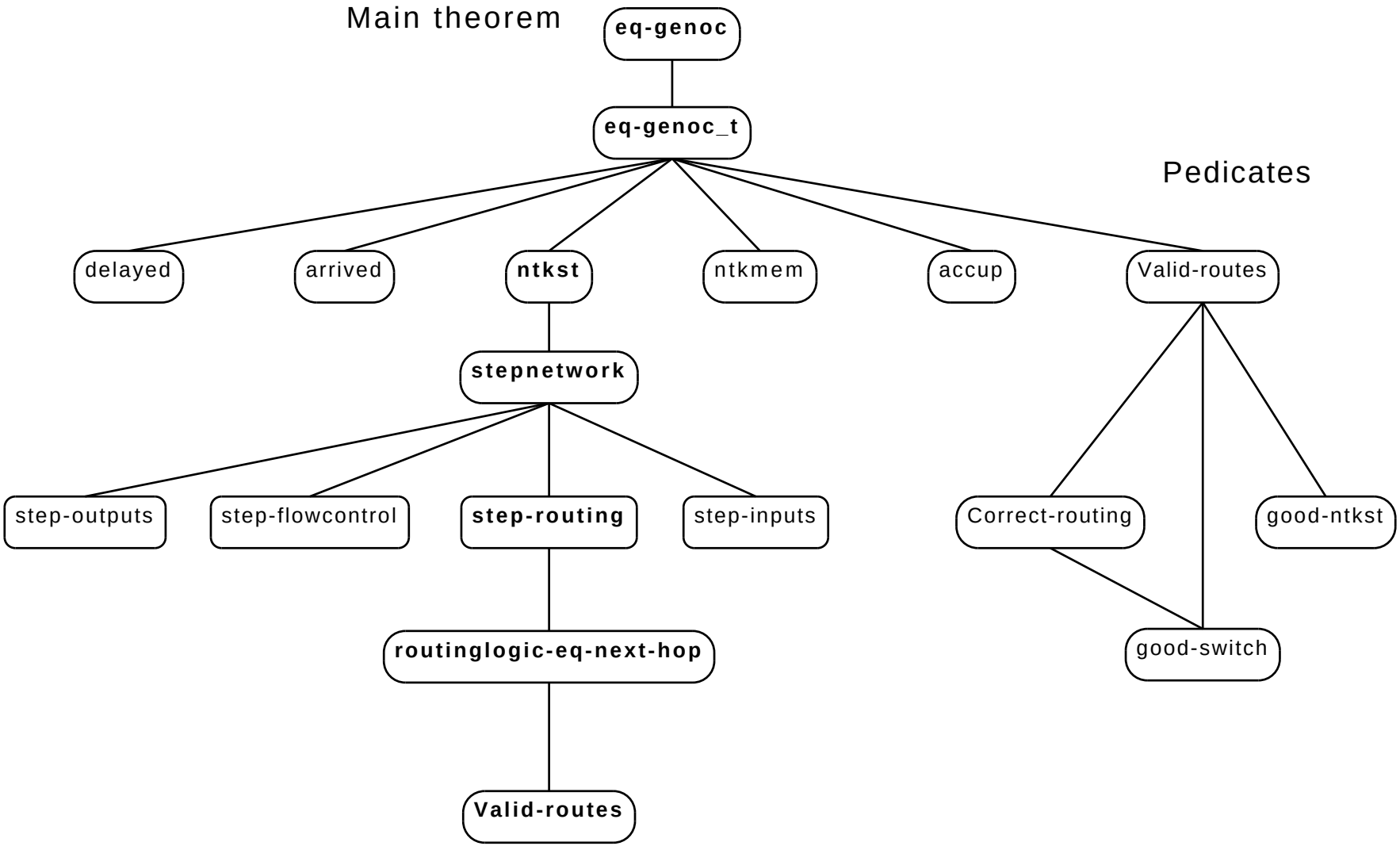
Refinement theorem (1) – Correct-GeNoC

$$\forall state, transactions :$$
$$transform(GeNoC_S(state, transactions)) = GeNoC_I(state, transactions)$$

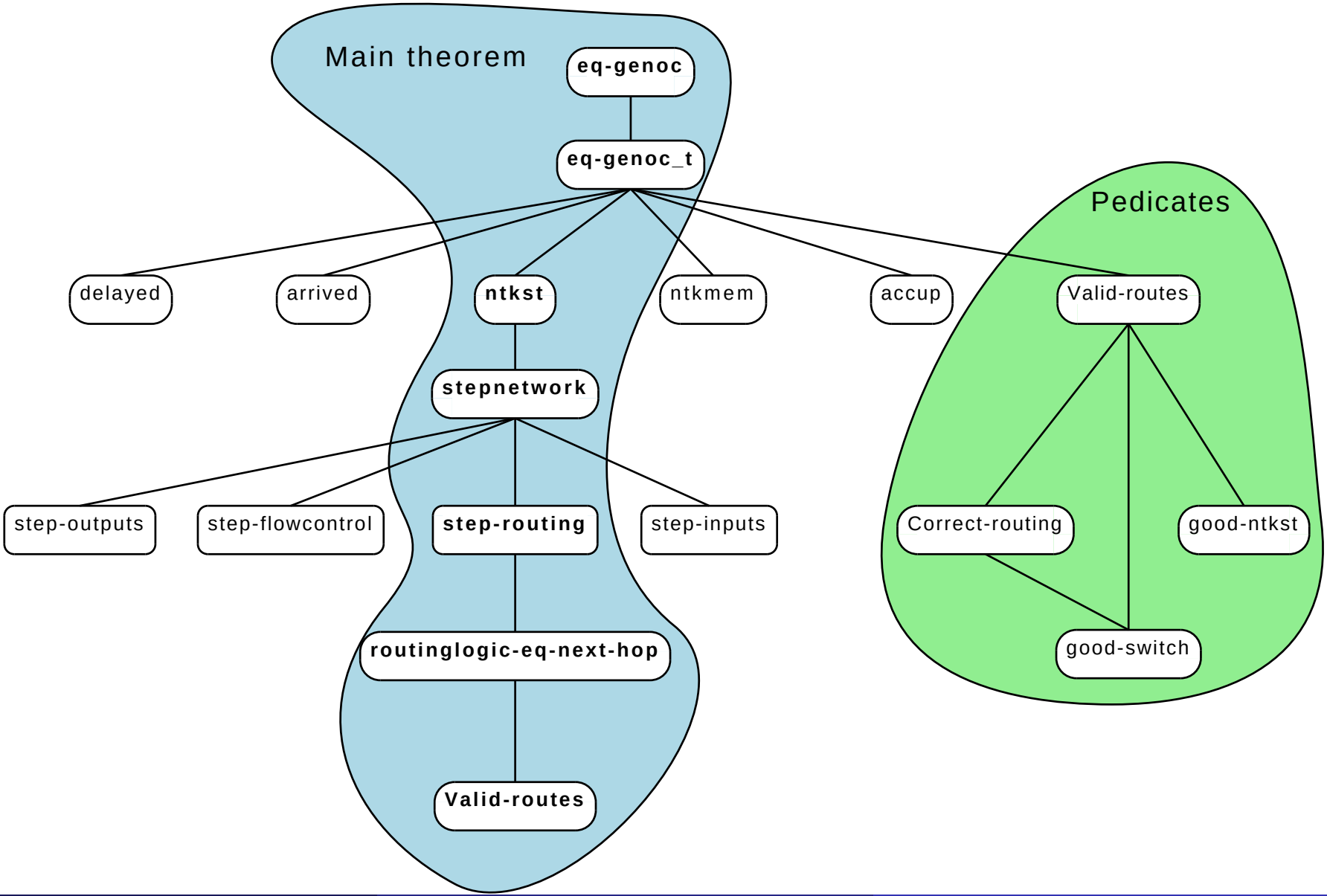
$GeNoC_I$ and $GeNoC_S$ return a tuple of $(arrived, delayed, trace)$ so this theorem can be read as:

1. The transformed arrived messages are equal
2. Delayed messages are equal
3. The transformed simulation trace is the same

Proof - Structure



Proof - Structure



Example theorem - Routinglogic-eq-next-hop

$$\forall msg : validRoute(msg) \implies computeRoute(cur(msg))(dest(msg)) = getNextHop(msg)$$

This theorem states:
A message with a valid route implies that computing the next step in the route is equal to extracting it from the precomputed route.

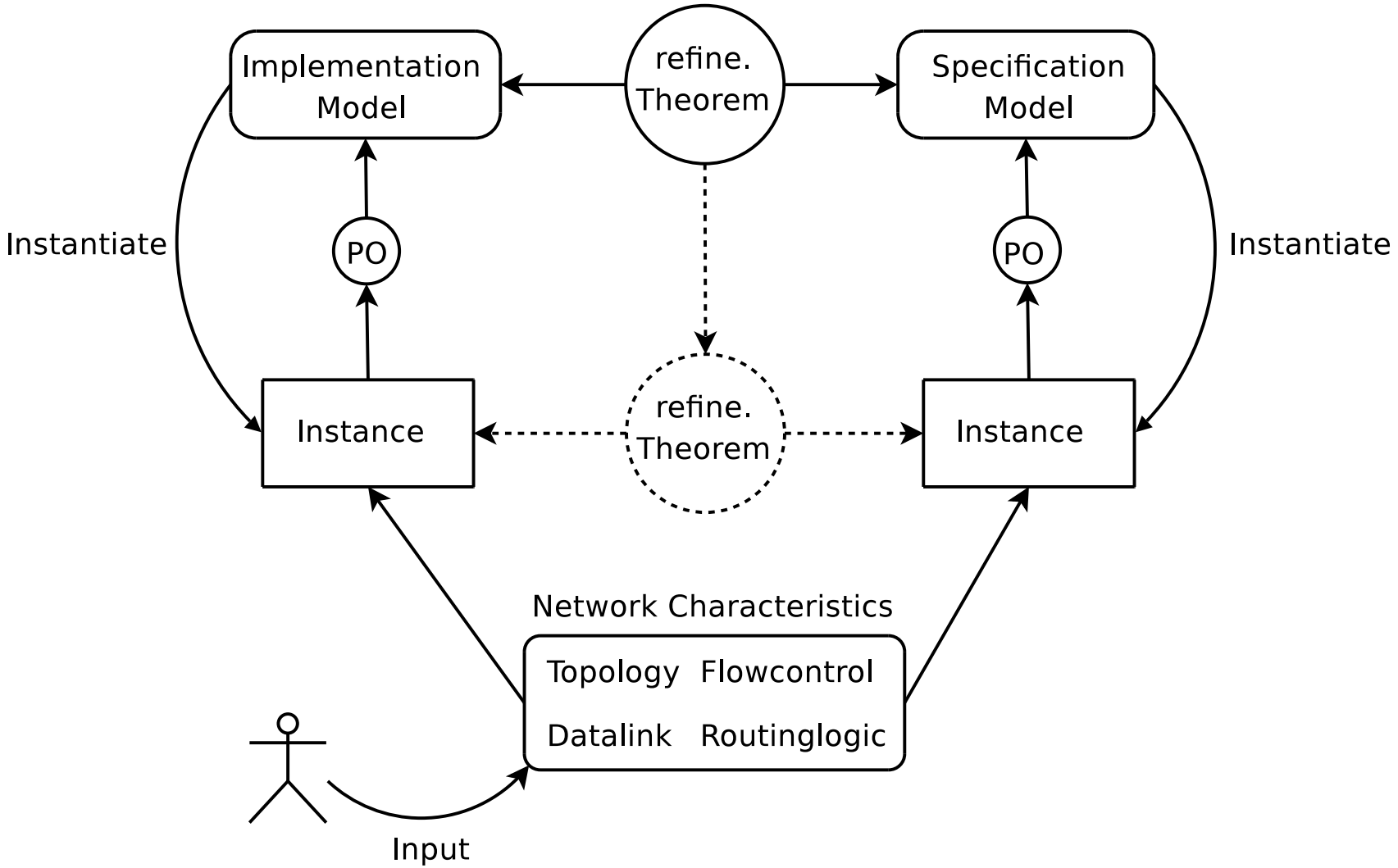
Proof - Statistics

Group	number of Theorems
Changed functions	72
Predicates	140
Not changed functions	88
Total	300

The source code of the proofs and models is available on the web [1]

[1] http://www.cs.ru.nl/~julien/Julien_at_Nijmegen/FMCAD09.html

Conclusion - overview



Conclusion - contributions

The contributions are:

- First cross-layer verification attempt of a NoC
- A realistic generic implementation model
- Multiple implementation instances of real NoCs
  - Packet, circuit, and wormhole switching
  - XY and Spidergon routing
  - Hermes NoC
  - Octagon NoC
- Instance of a NoC at the specification level
- Refinement proof between two instances

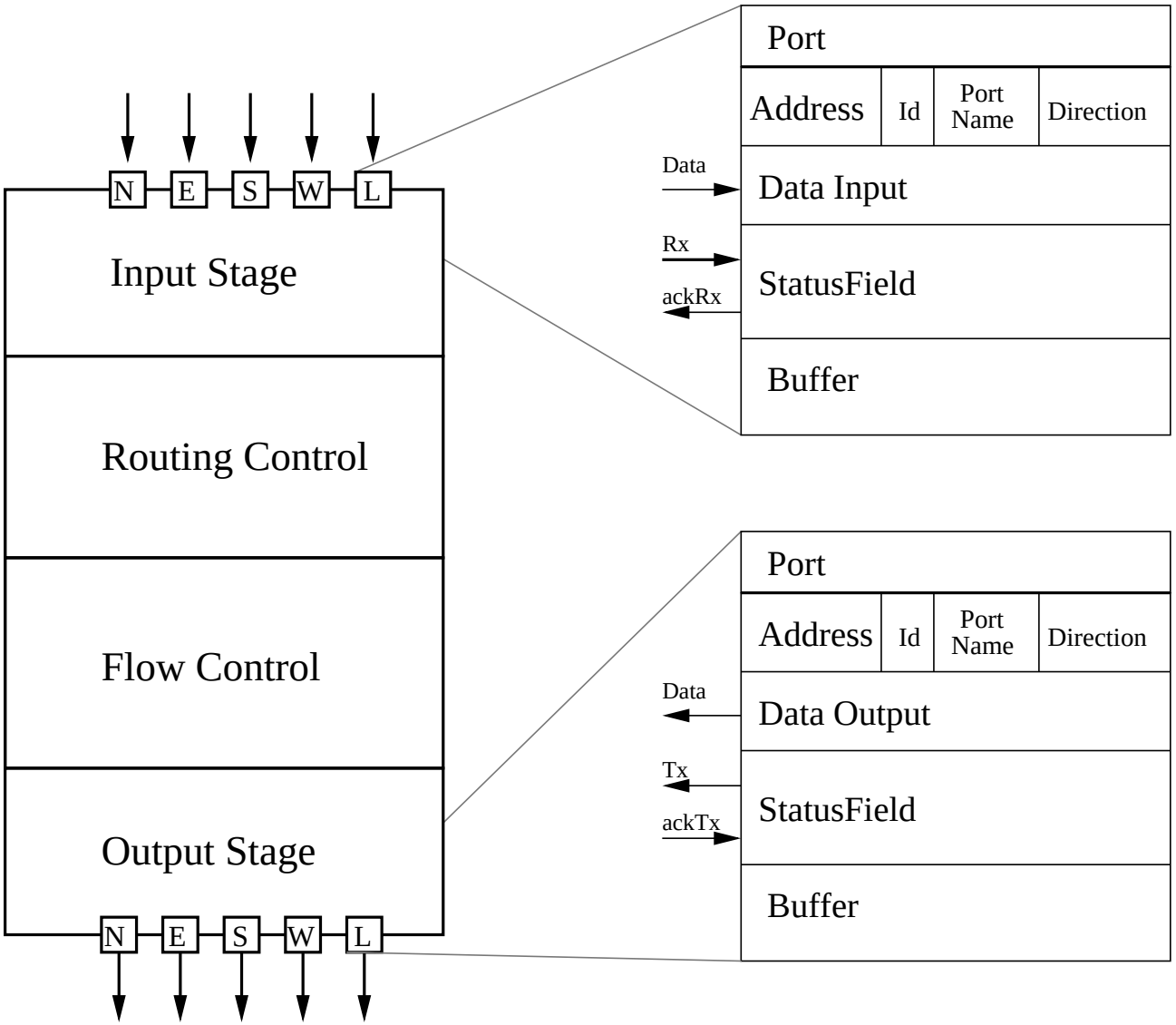
Conclusion - Future work

Current and future research directions:

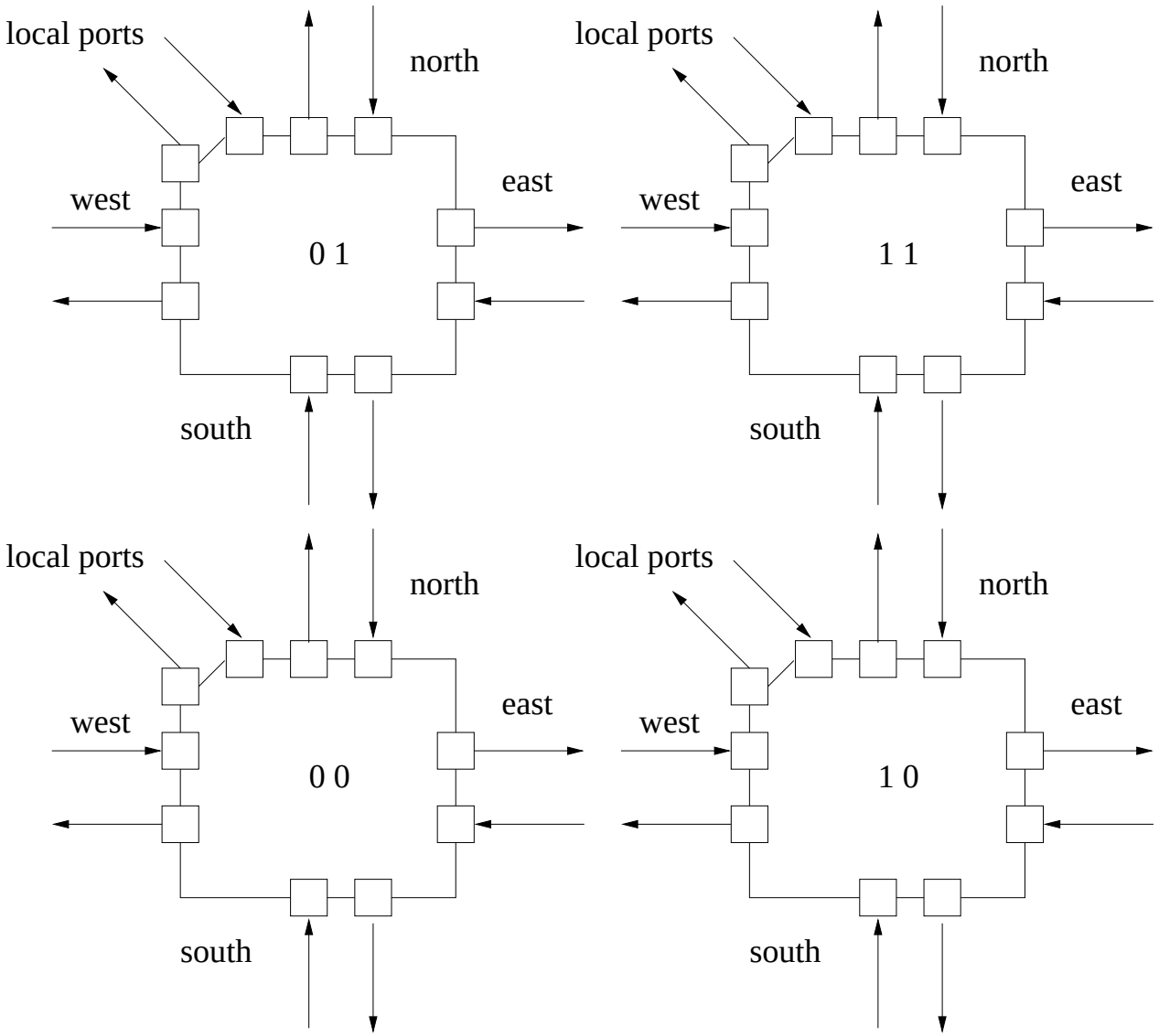
- A generic cross-layer verification method
- Proof between two generic models at two different levels
- More instances of different NoCs
- Integration of deadlock and liveness properties (Verbeek & Schmaltz ACL2 '09 and DATE '10)
- Extending the number of layers
  - Towards RTL
  - Layer with "Source" and "Distributed scheduling"

Thank you for listening!

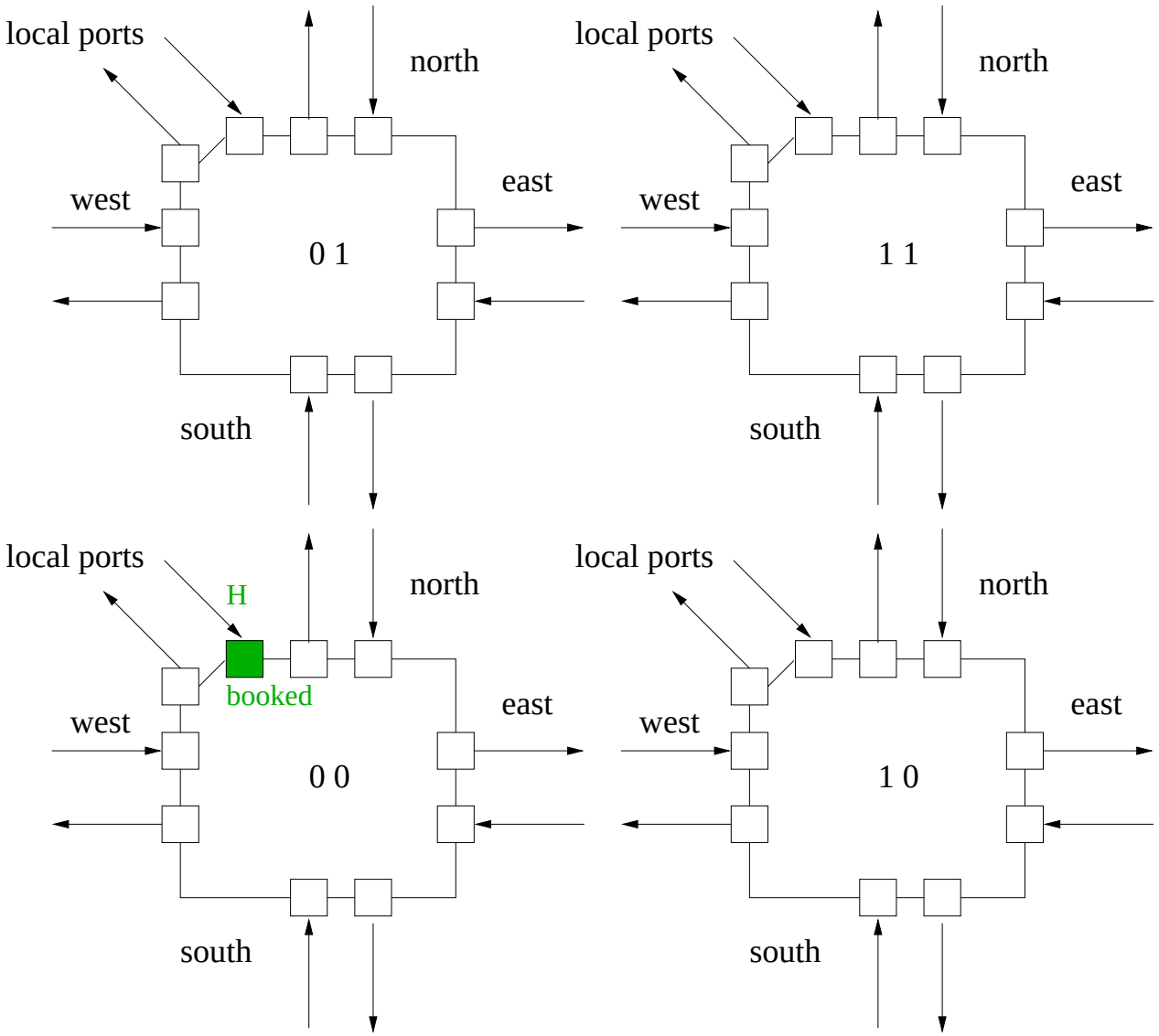
Network Model – Generic Router



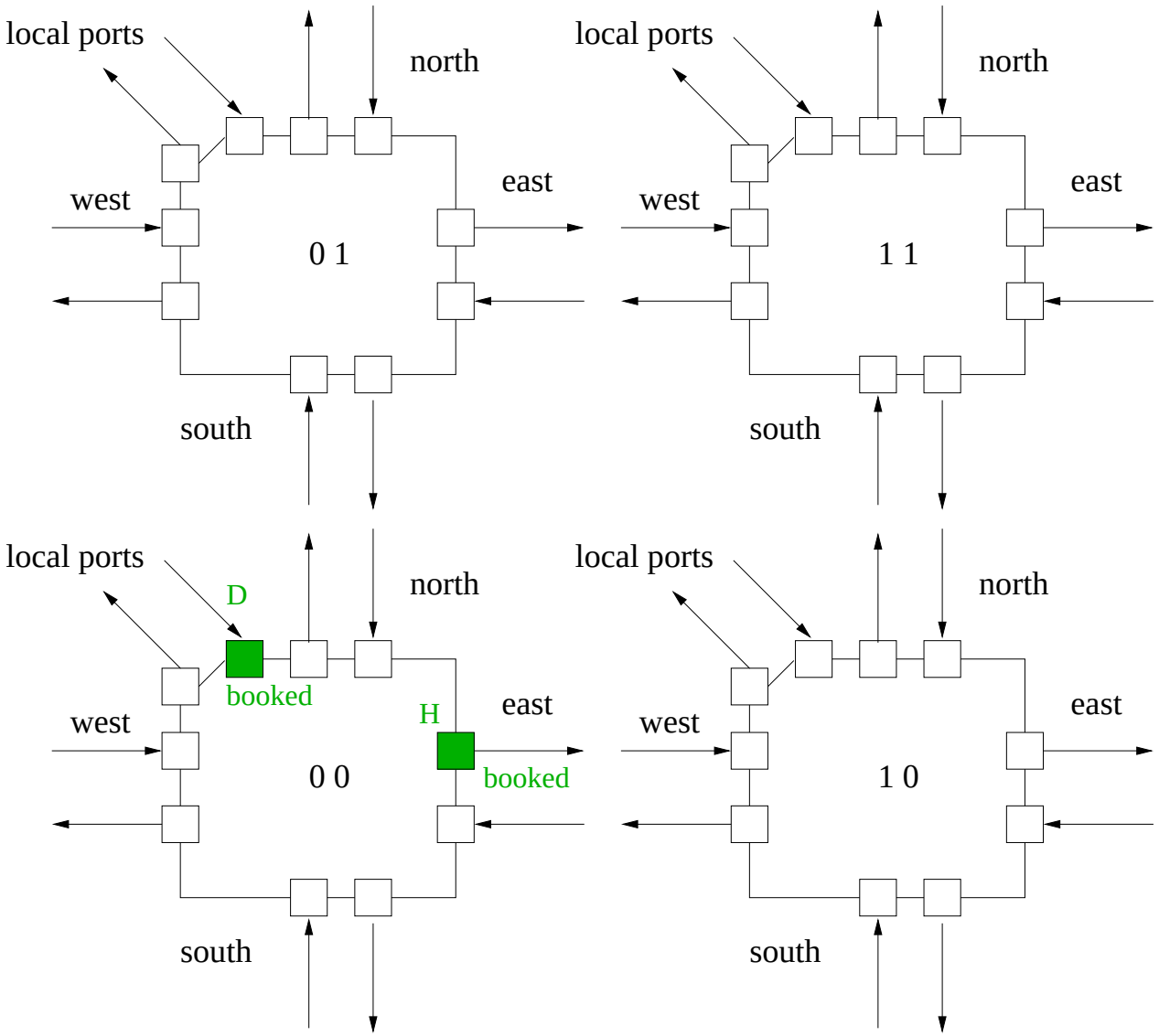
Wormhole switching and XY Routing



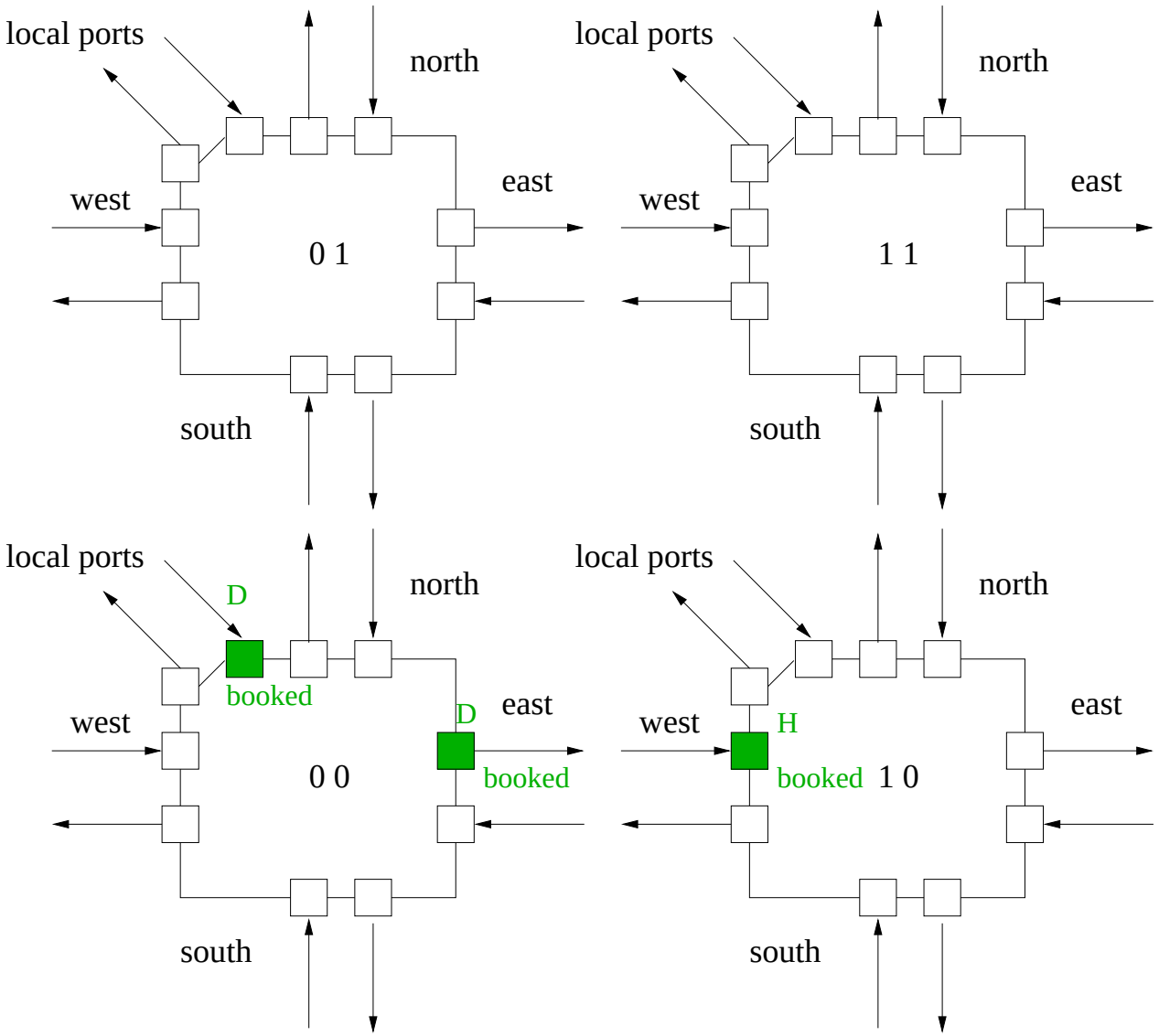
Wormhole switching and XY Routing



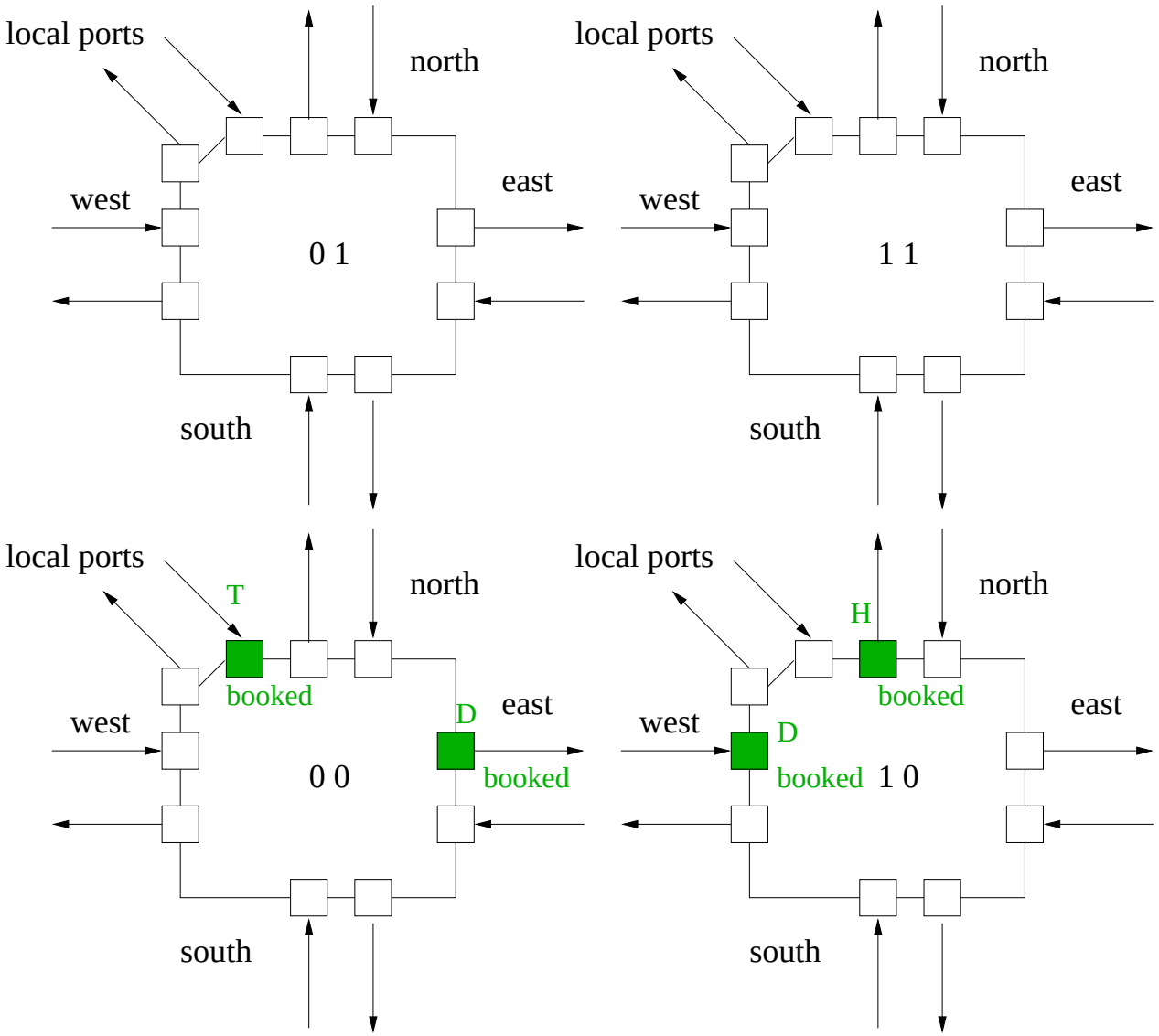
Wormhole switching and XY Routing



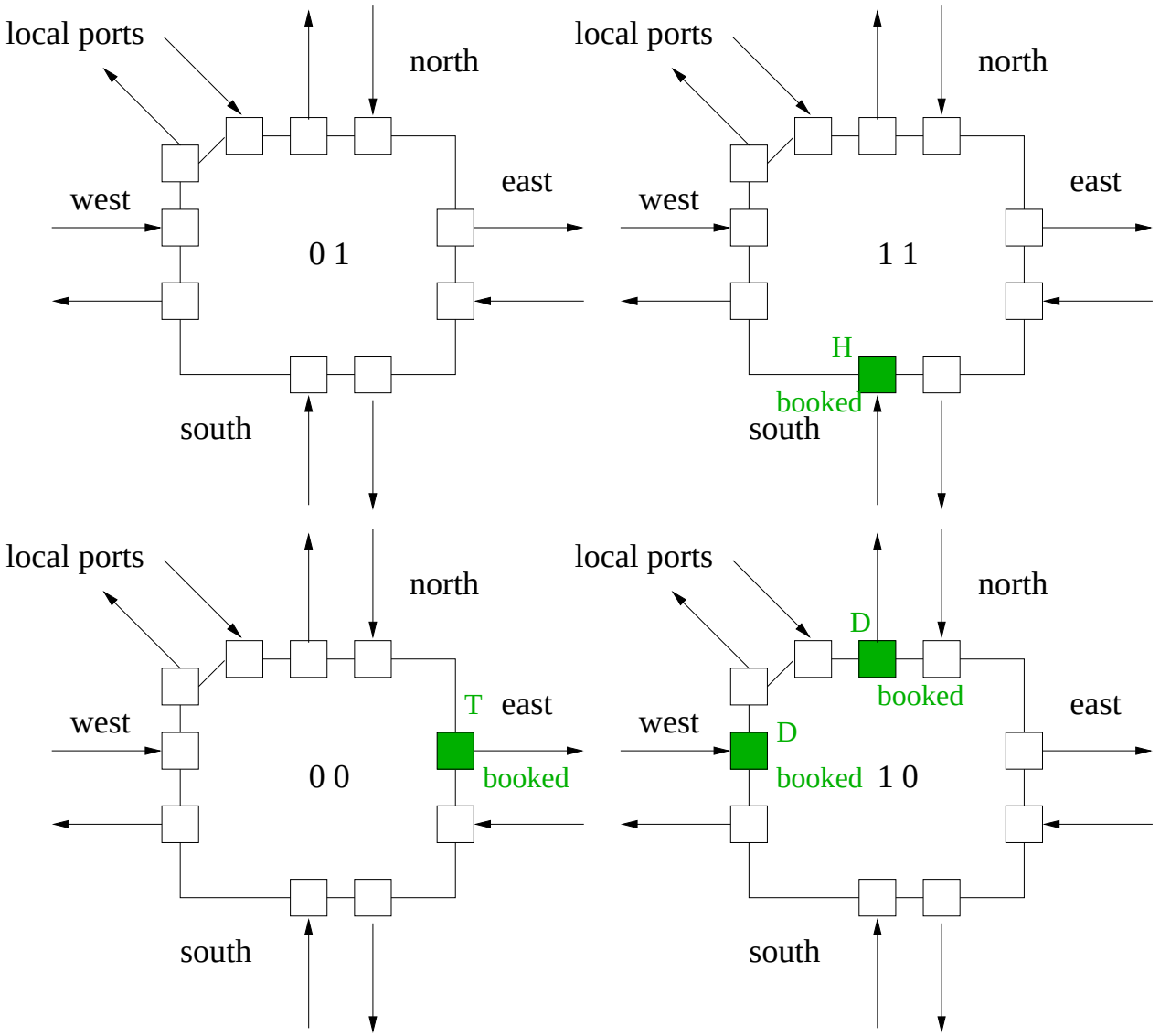
Wormhole switching and XY Routing



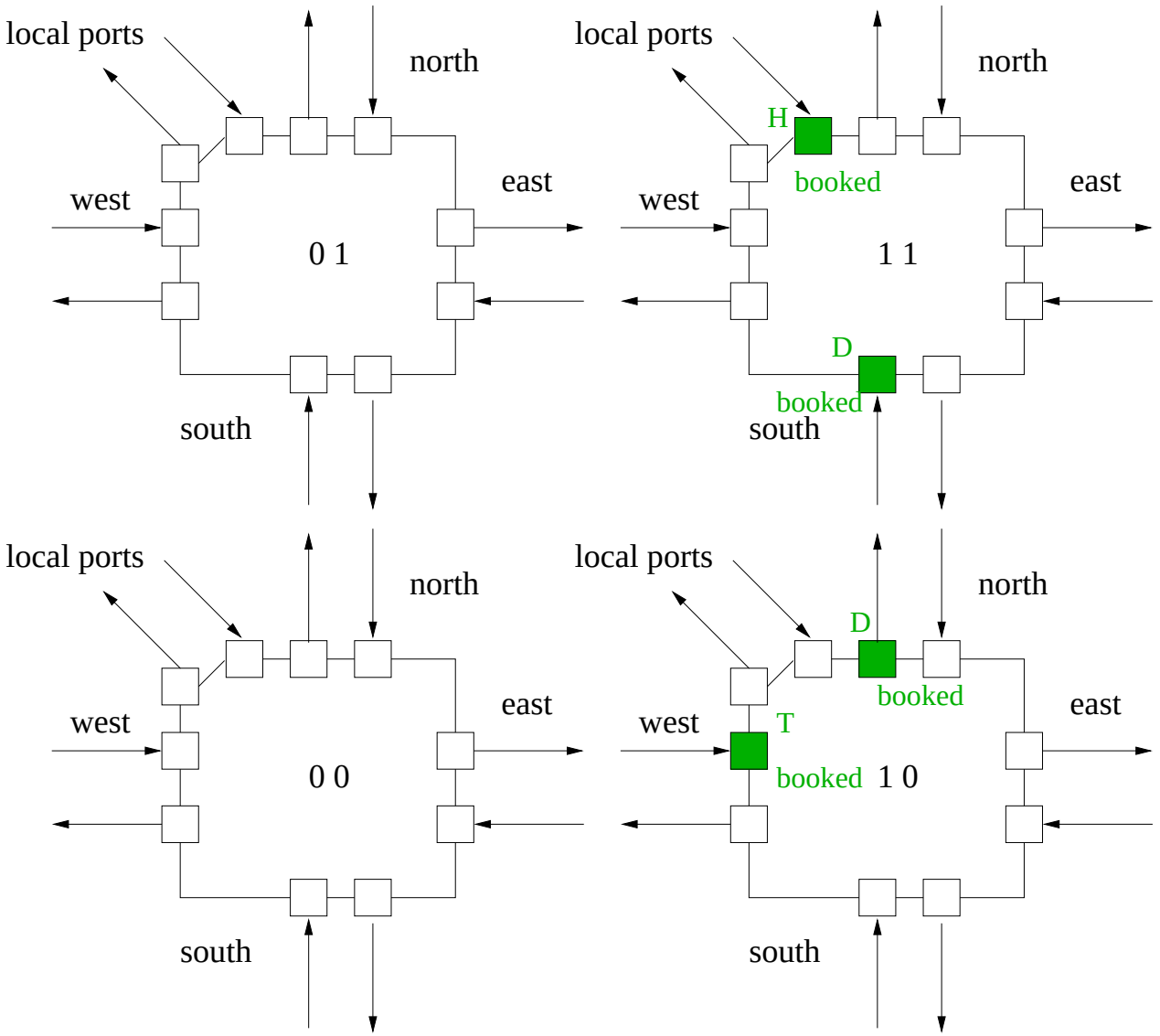
Wormhole switching and XY Routing



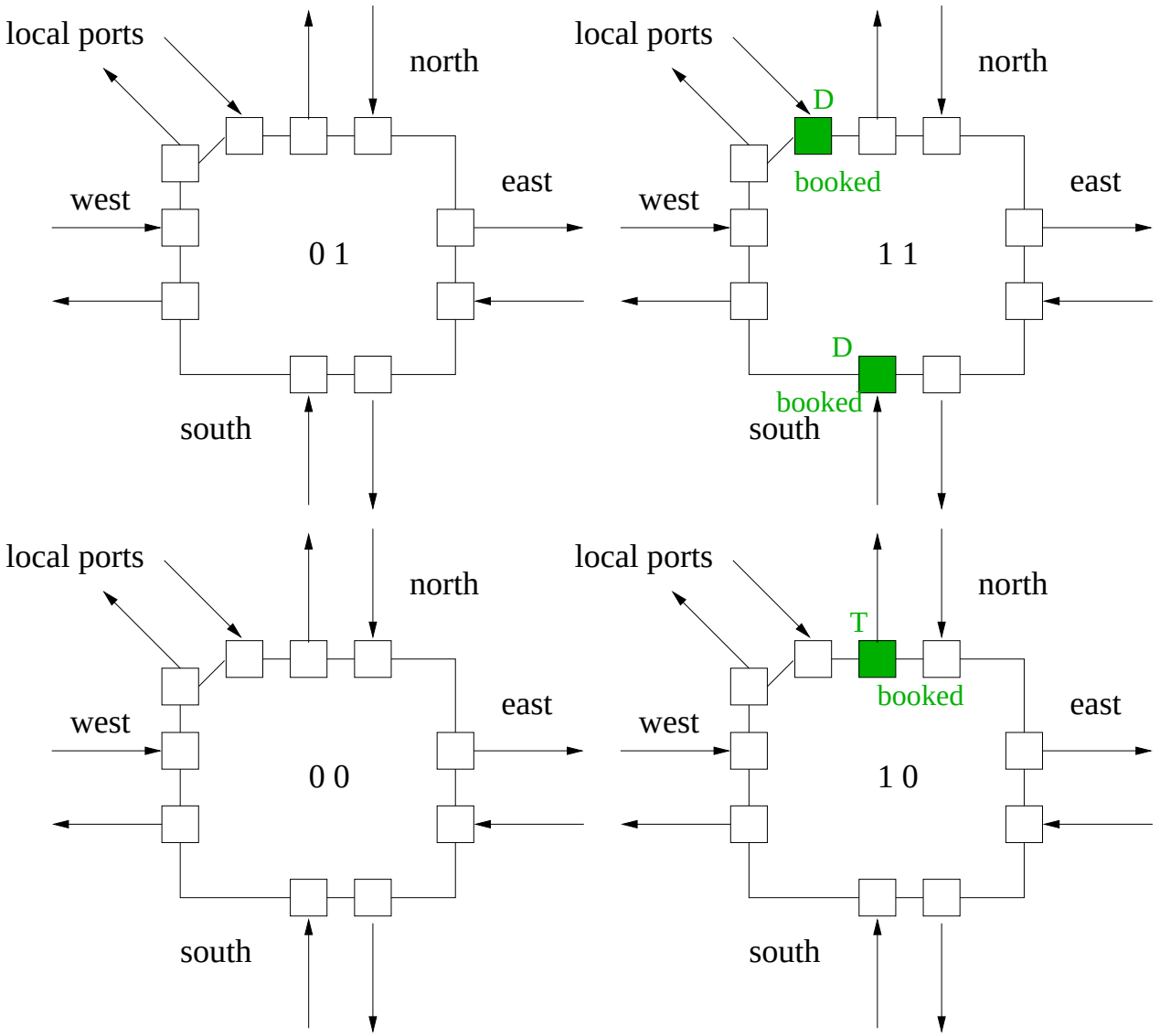
Wormhole switching and XY Routing



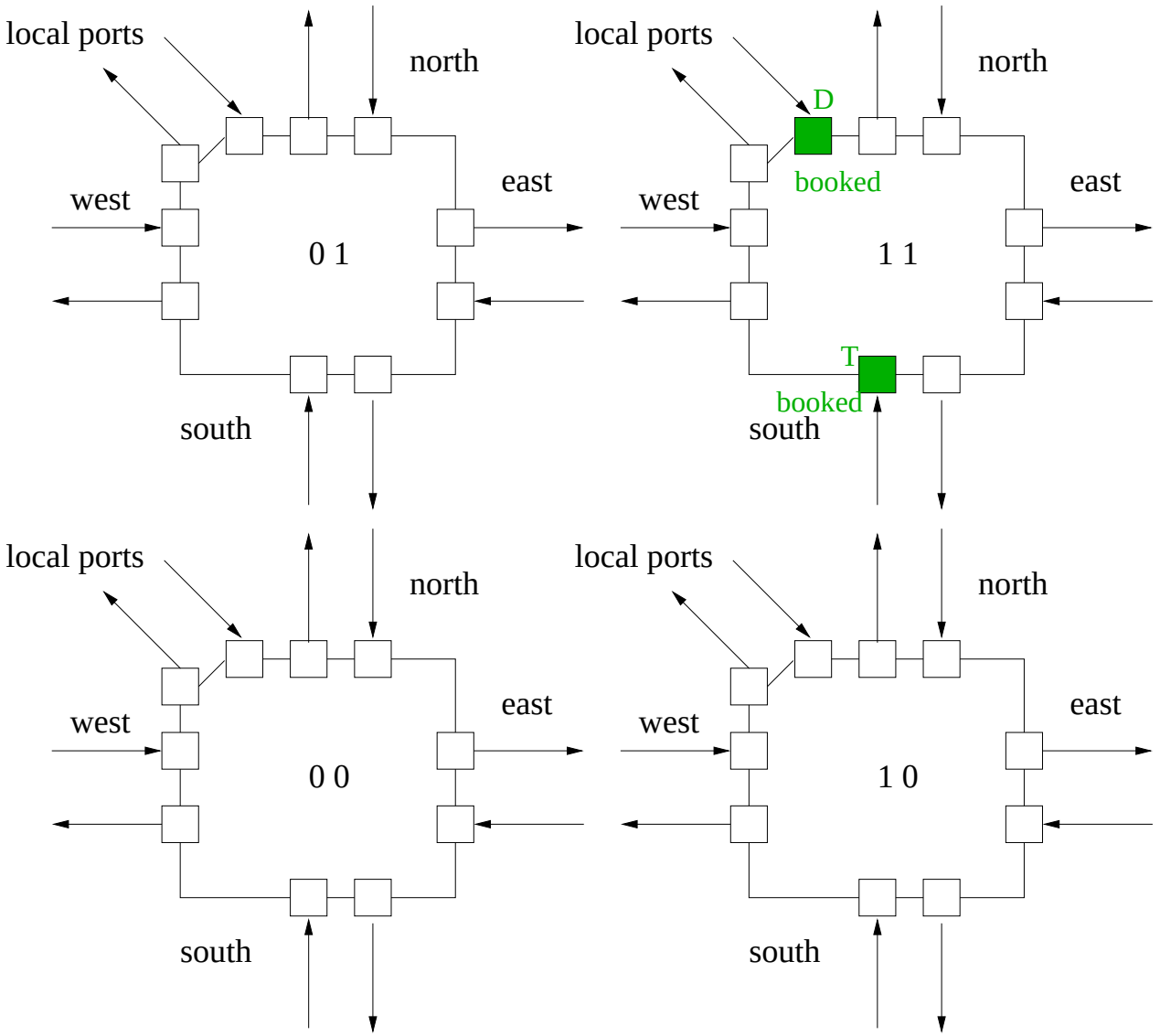
Wormhole switching and XY Routing



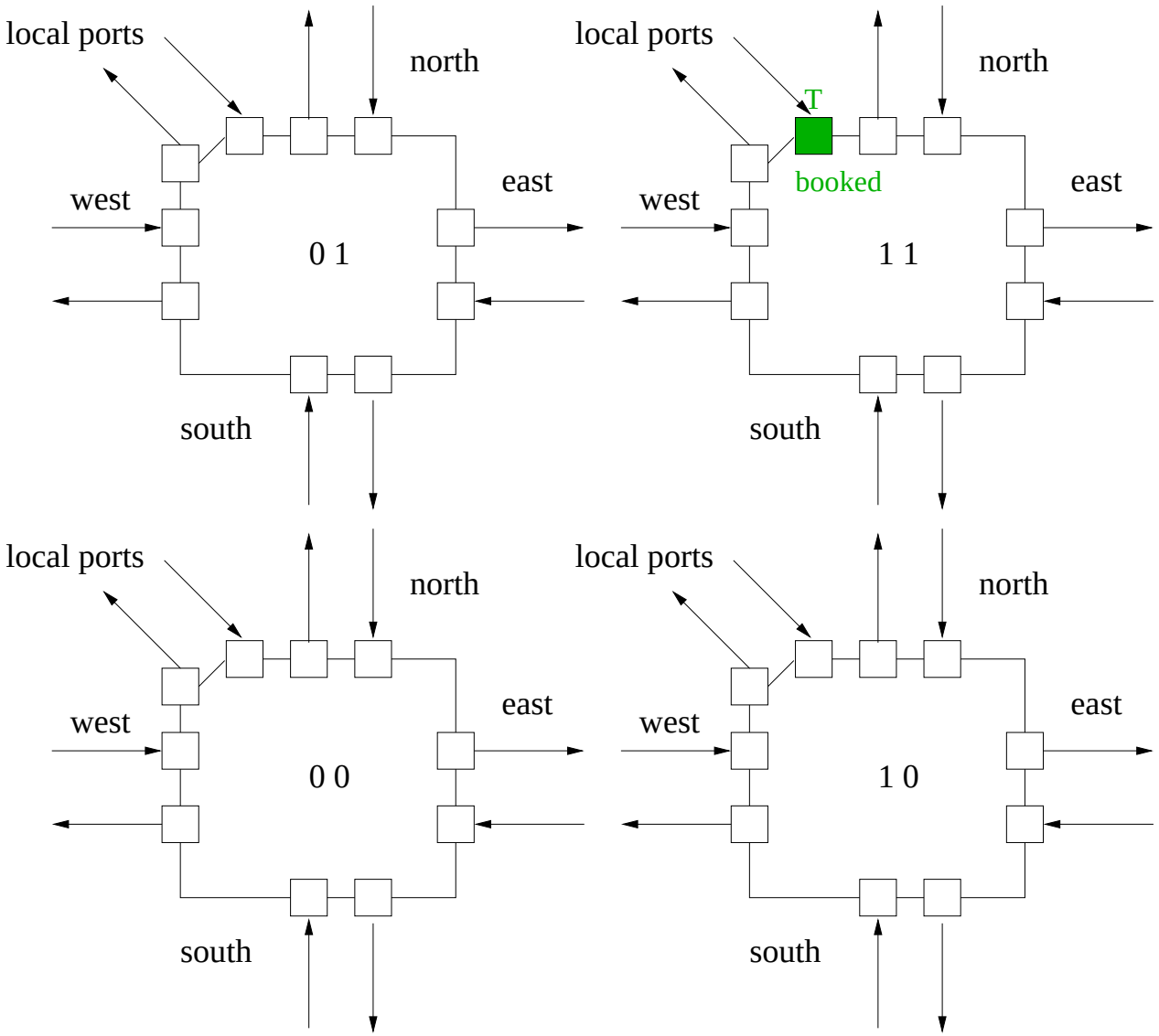
Wormhole switching and XY Routing



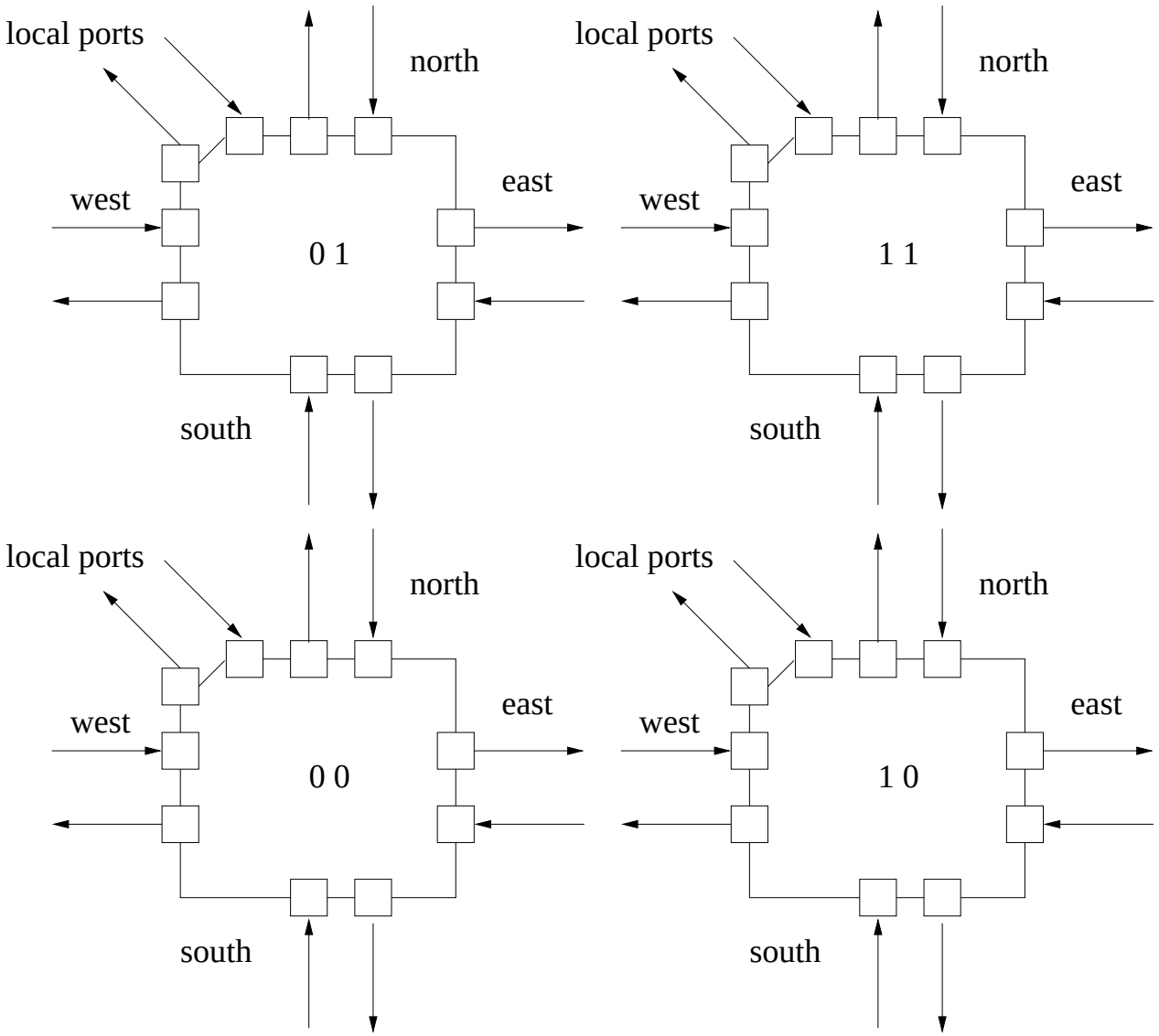
Wormhole switching and XY Routing



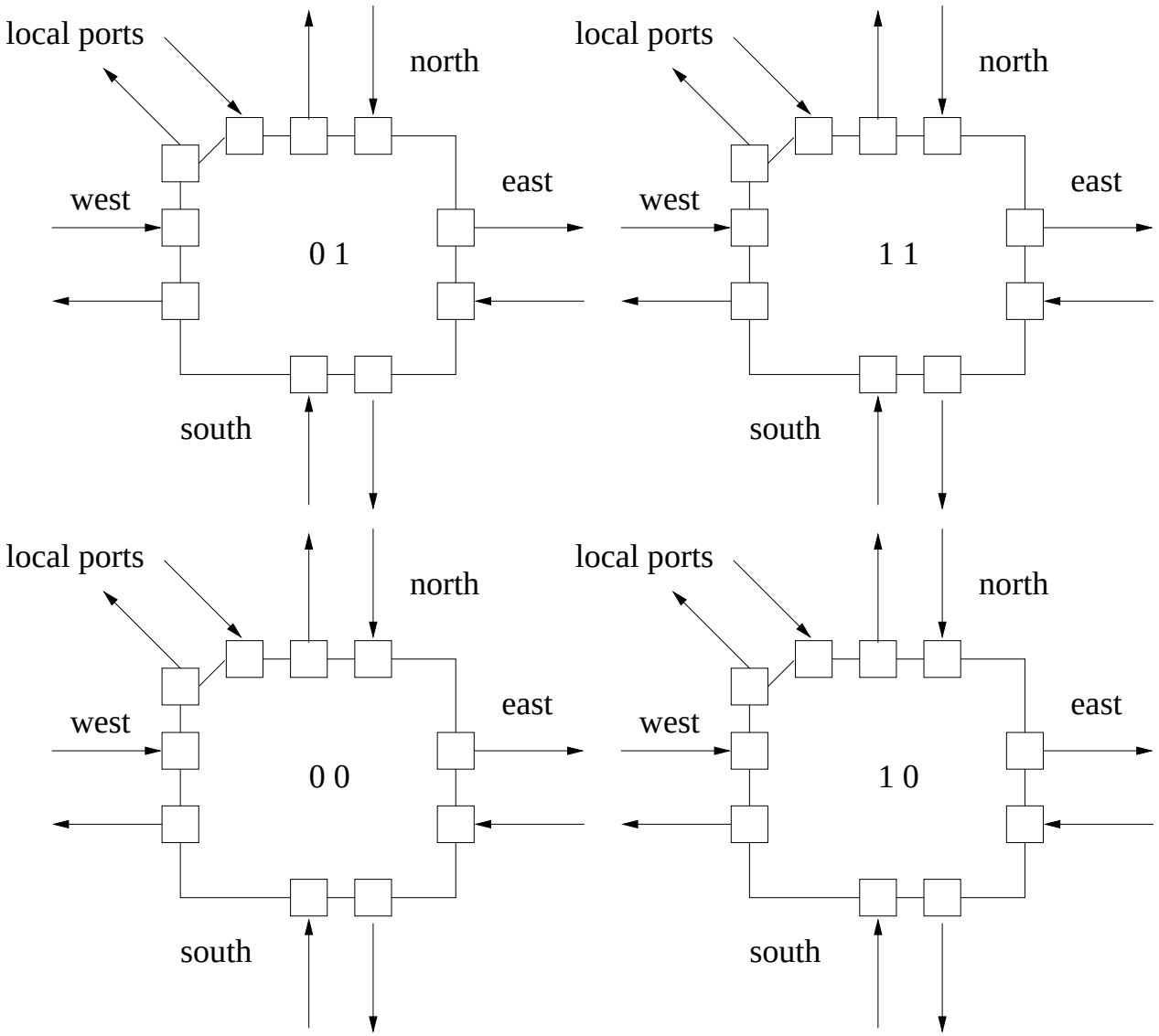
Wormhole switching and XY Routing



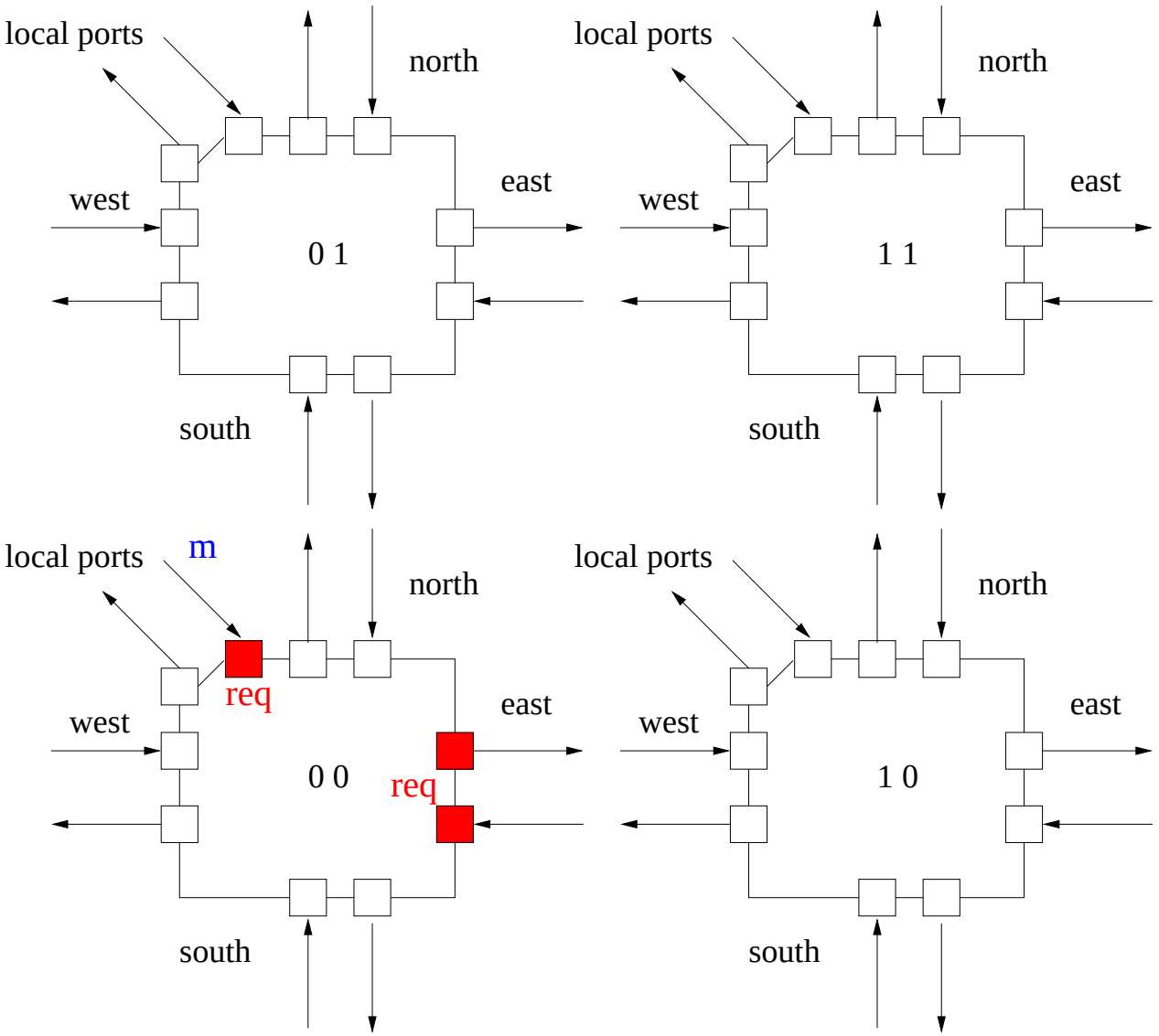
Wormhole switching and XY Routing



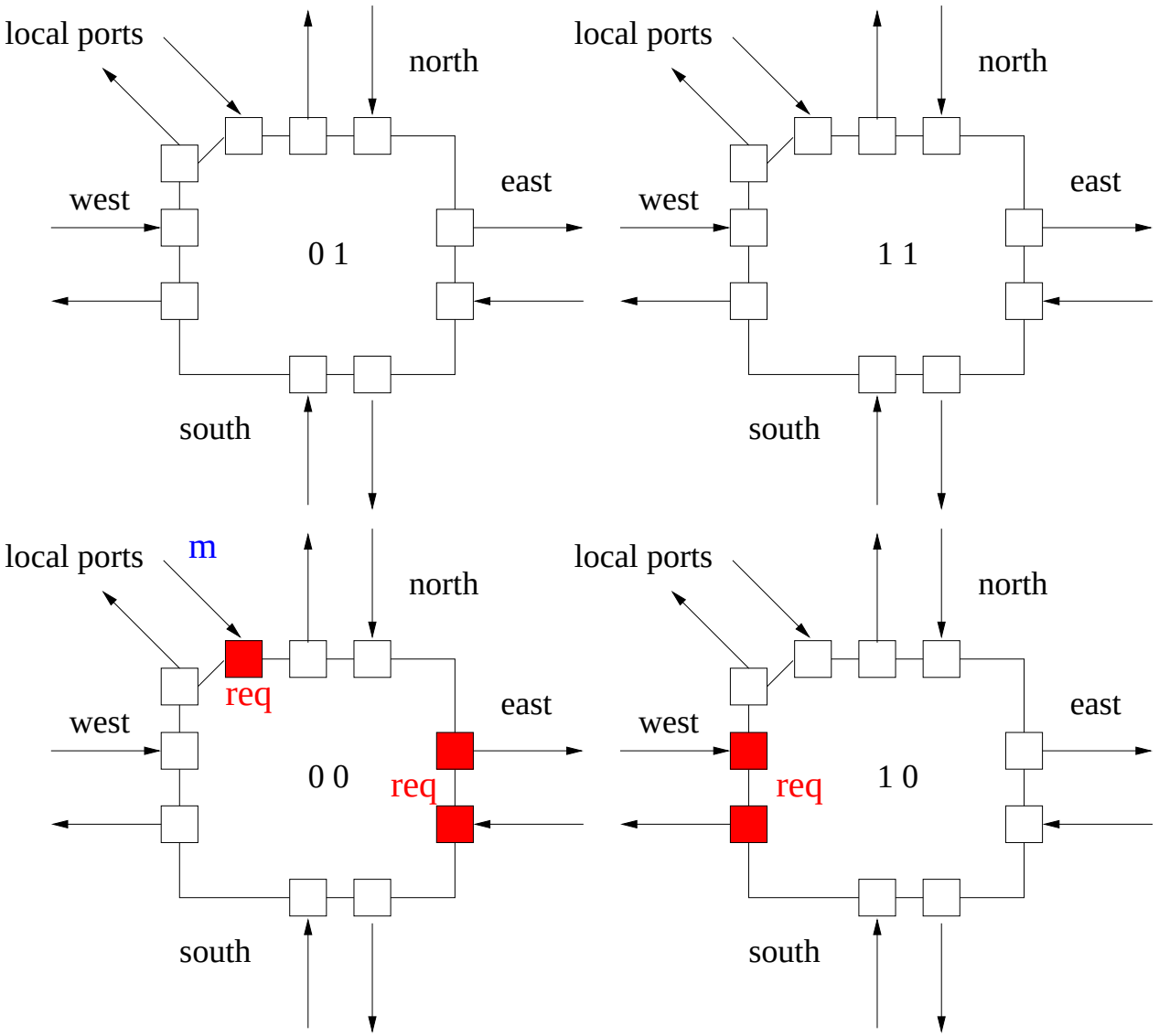
Circuit Switching



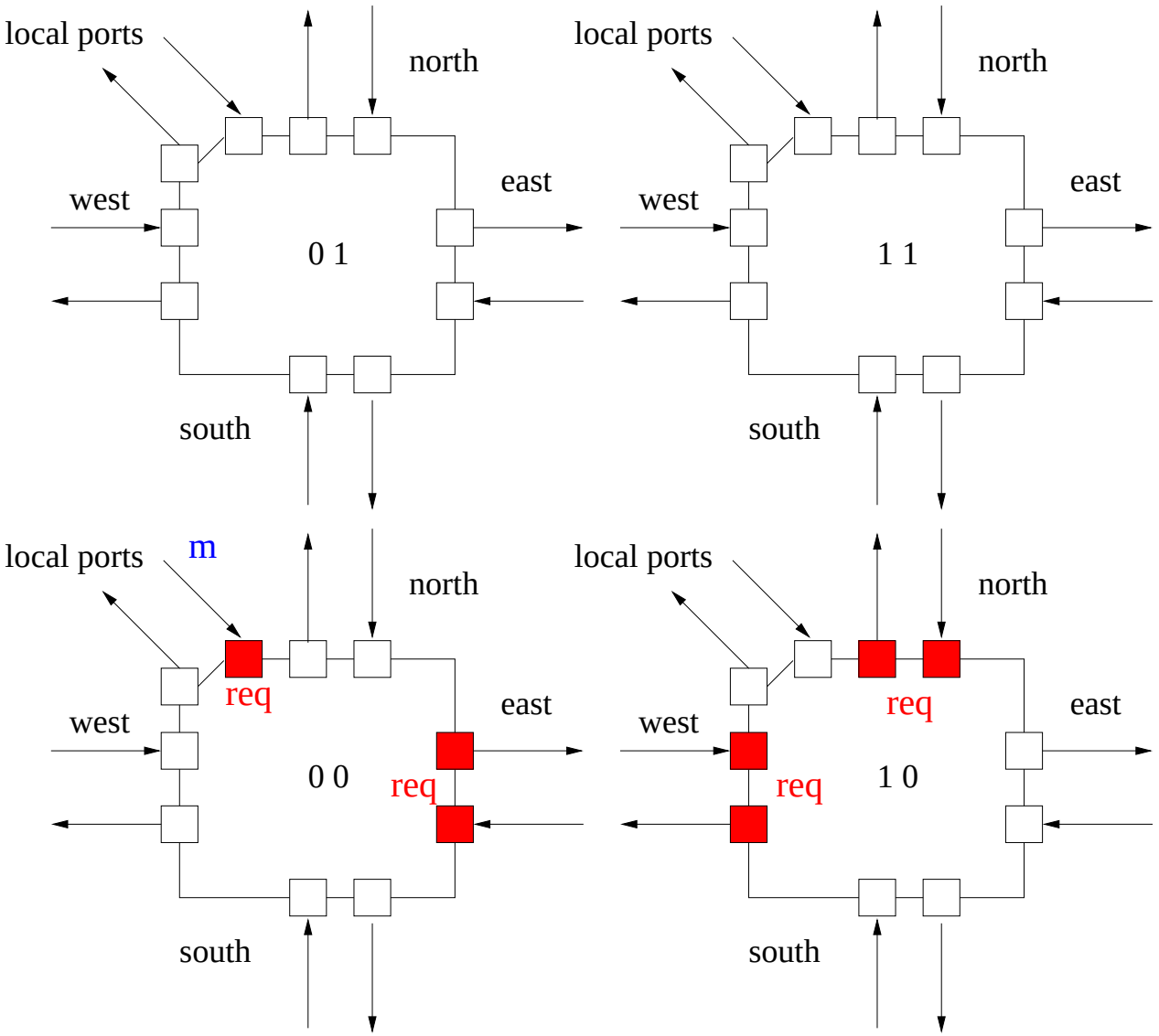
Circuit Switching



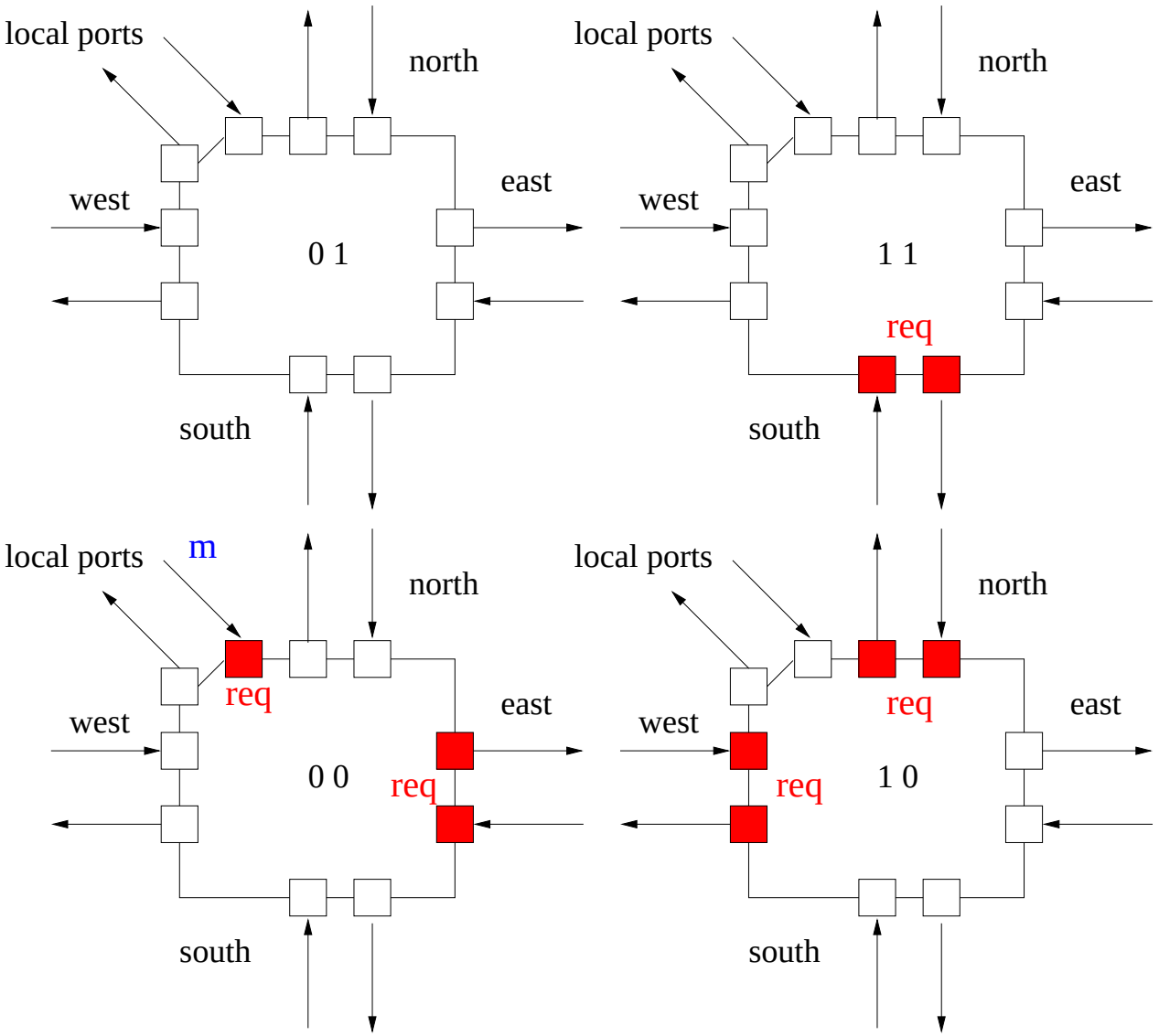
Circuit Switching



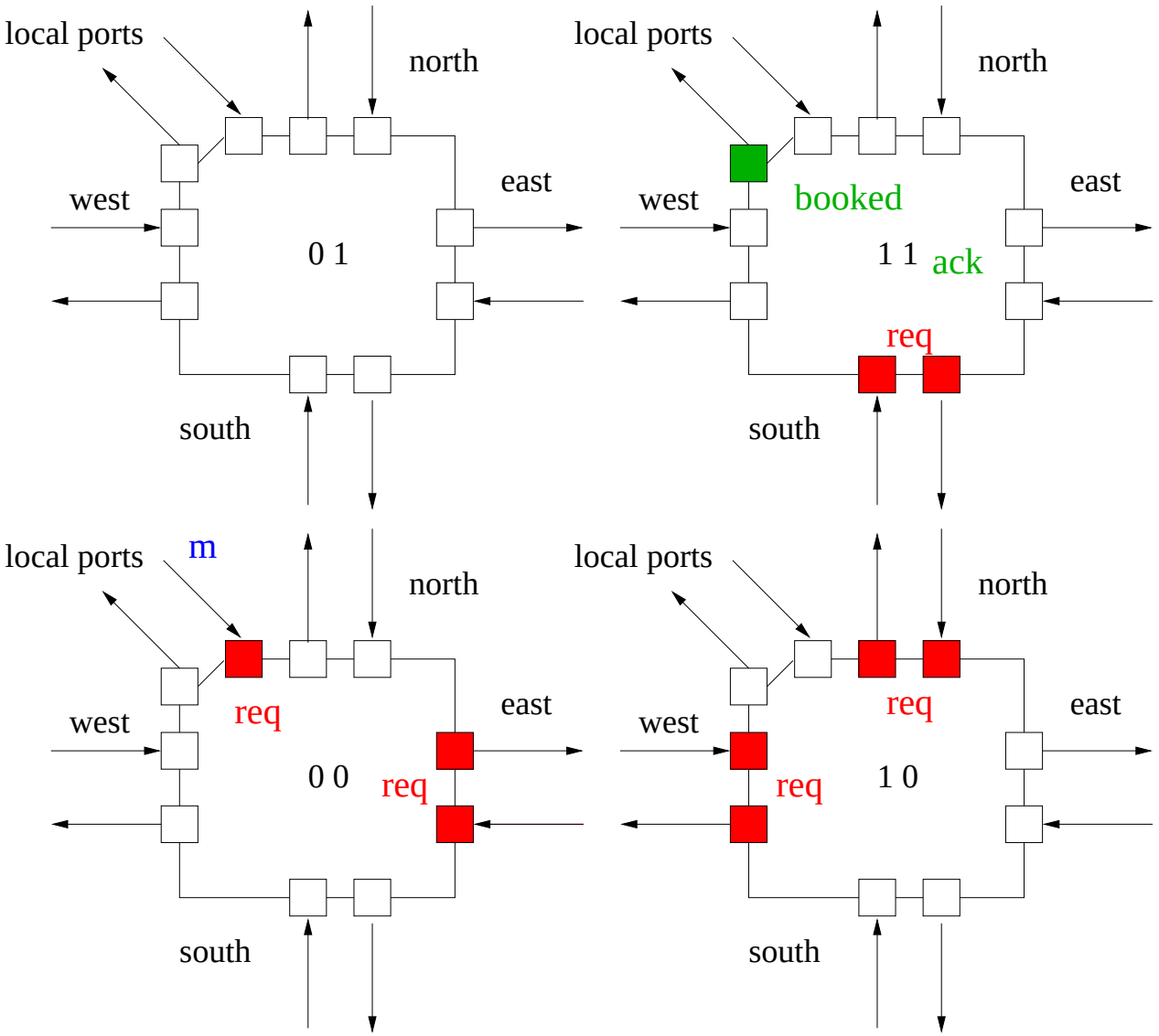
Circuit Switching



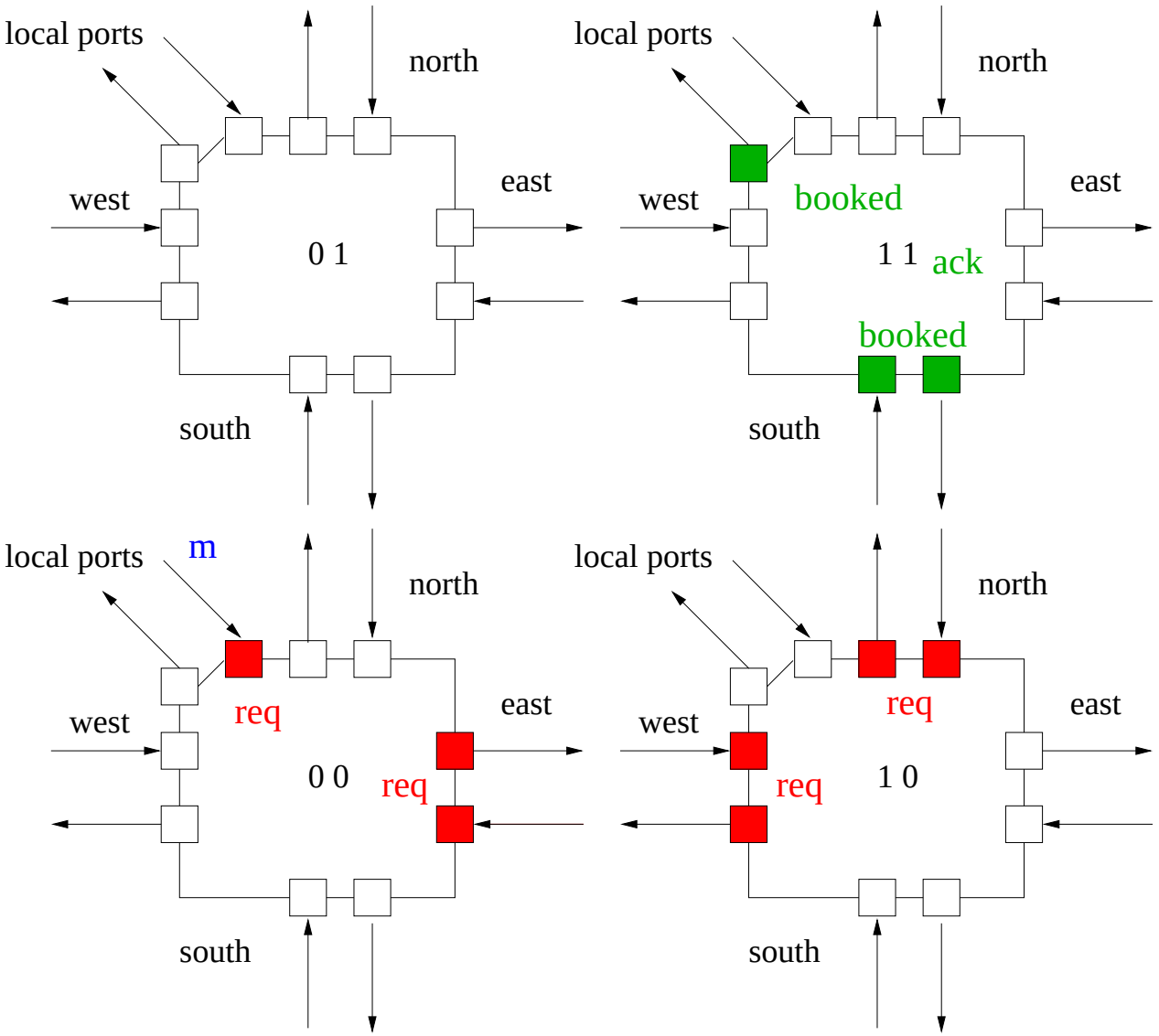
Circuit Switching



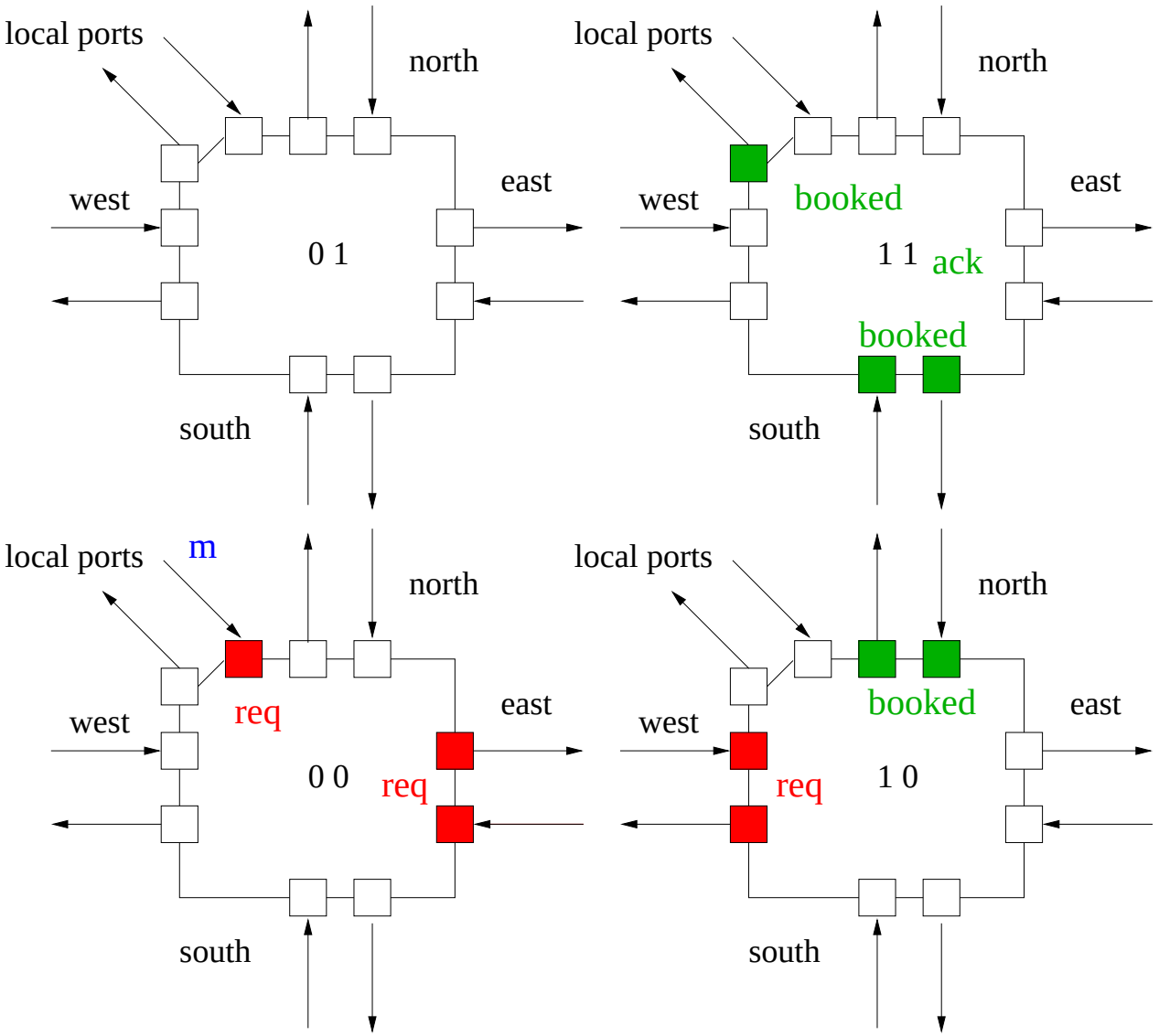
Circuit Switching



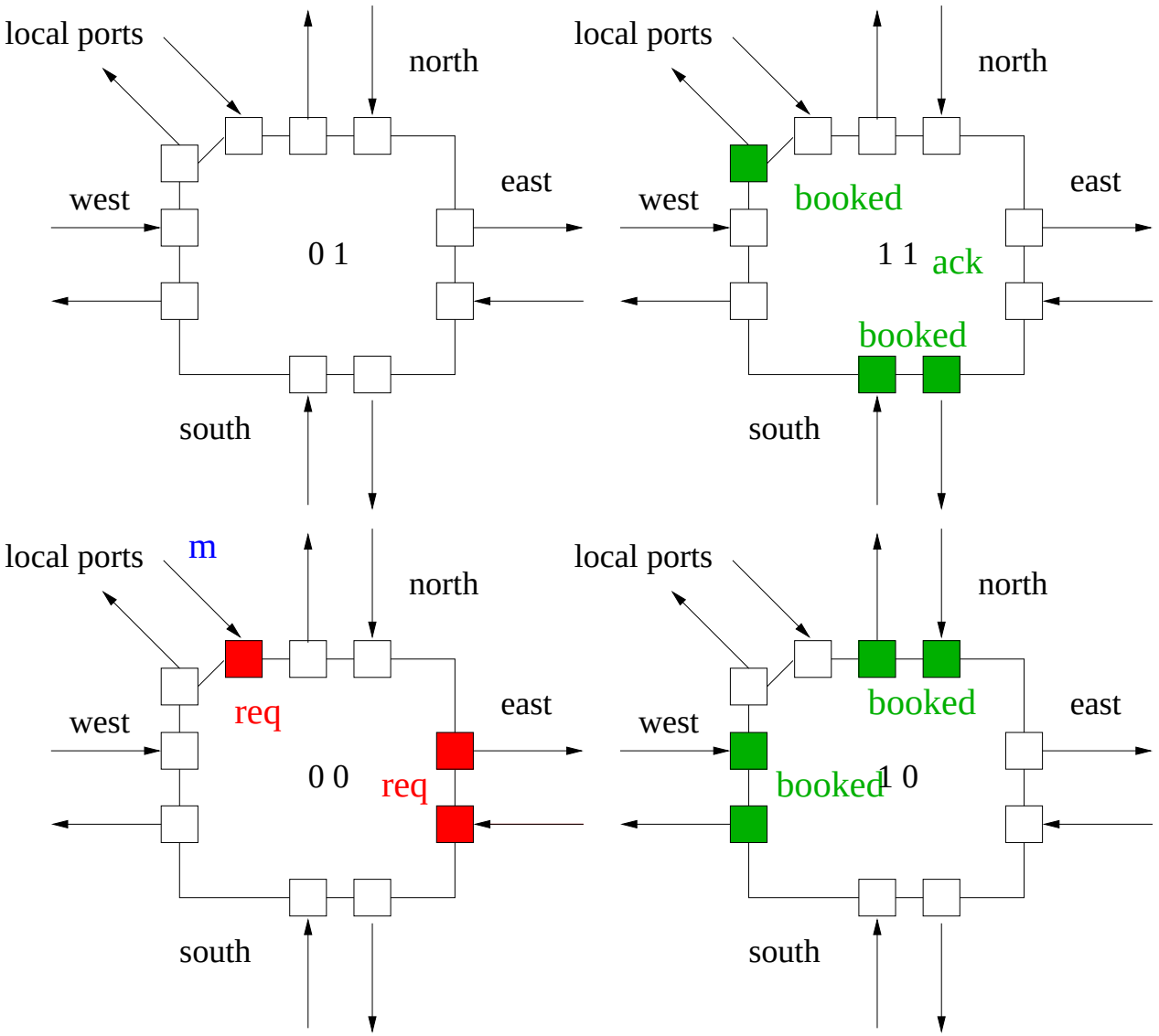
Circuit Switching



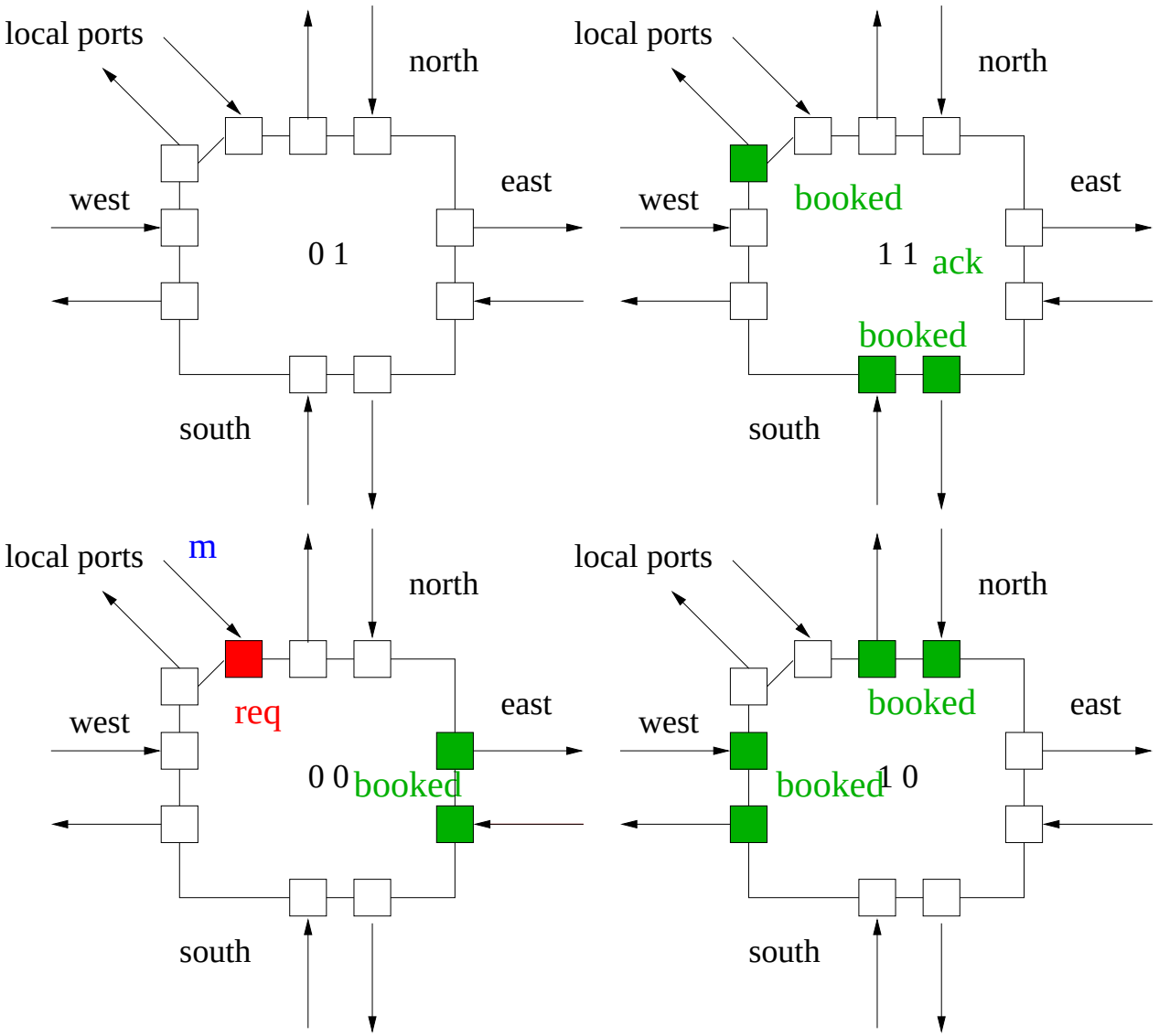
Circuit Switching



Circuit Switching



Circuit Switching



Circuit Switching