reMOTEable

Embeded Software Engineering Course

Winter 2005/06

Prof. Kirsch

Alois Hofstätter

Bernhard Kast

Horst Stadler

26.01.2006

Overview

- Introduction
- Design
- Implementation
- Demo
- Outlook

Introduction – Project

- Project outline
  - Agent system
  - Independent embedded devices
  - Radio communication
  - Sensor the environment

Introduction - Motes

- Telos mote IV Revision A (mote = A very small particle)
- Wireless Ad hoc Sensor Networks (e.g. habitat monitoring)
- Advantages:
  - small
  - durable
  - low power usage
  - communication devices
  - sensors (optional)

Introduction – Existing Solutions

- Agilla
  - existing running agent system
  - strong mobility (code + state)
  - too high hardware requirements (41.6k code and 3.59k data)
- Maté
  - not an agent system
  - a mote reprogramming system
  - hence not suitable for our project
- Sensorware
  - high hardware requirements & weak mobility

Introduction – Conclusion

- Own requirements
  - run on Telos mote IV Rev A (2k RAM, 60k ROM)
  - strong mobility
  - multiple agents in one network
  - mote2mote (IEEE 802.15.4) and mote2pc (USB) communication
    - inject new agents
    - get collected data from motes

=> implement own virtual machine on top of TinyOS

Design - Overview

- Virtual Machine
- Protocols
- Agents
- Example Agent

Design – VM (1)

- Architecture:
  - 4 Registers (addressing 2 bit, value size 8 bit)
  - half byte alignment (addresses, commands)
  - Code Segment and Data Segment (CS + DS < 418 half-bytes)
  - Flags (cmp, remainder, overflow)

Design – VM (2)

- Basic Assembler commands
  - logical operations
  - arithmetic operations
  - branches (cmp, jnz)
  - memory operations (load, store)
- Enhanced high level calls (use of Capability model)
  - capcall package_number, procedure_number (4 bit each)
  - e.g. send_agent(), get_neighbours(),...

Design – Protocols

	Agents	Application Layer
unreliable Protocol	reliable Protocol	Transport Layer
TinyOS		Network Layer
		Data Link Layer
IEEE802.15.4		Physical Layer

Design – Discovery Protocol

- Header
  - source address (4 bit)
  - destination address (4 bit)
- no payload
- Discovery
  - Request (broadcast)
  - Answer (unicast – cast type used to distinguish packets)
- Distinguish Discovery Protocol from Transport Protocol (length field of TinyOS Packet)

Design – Transport Protocol (1)

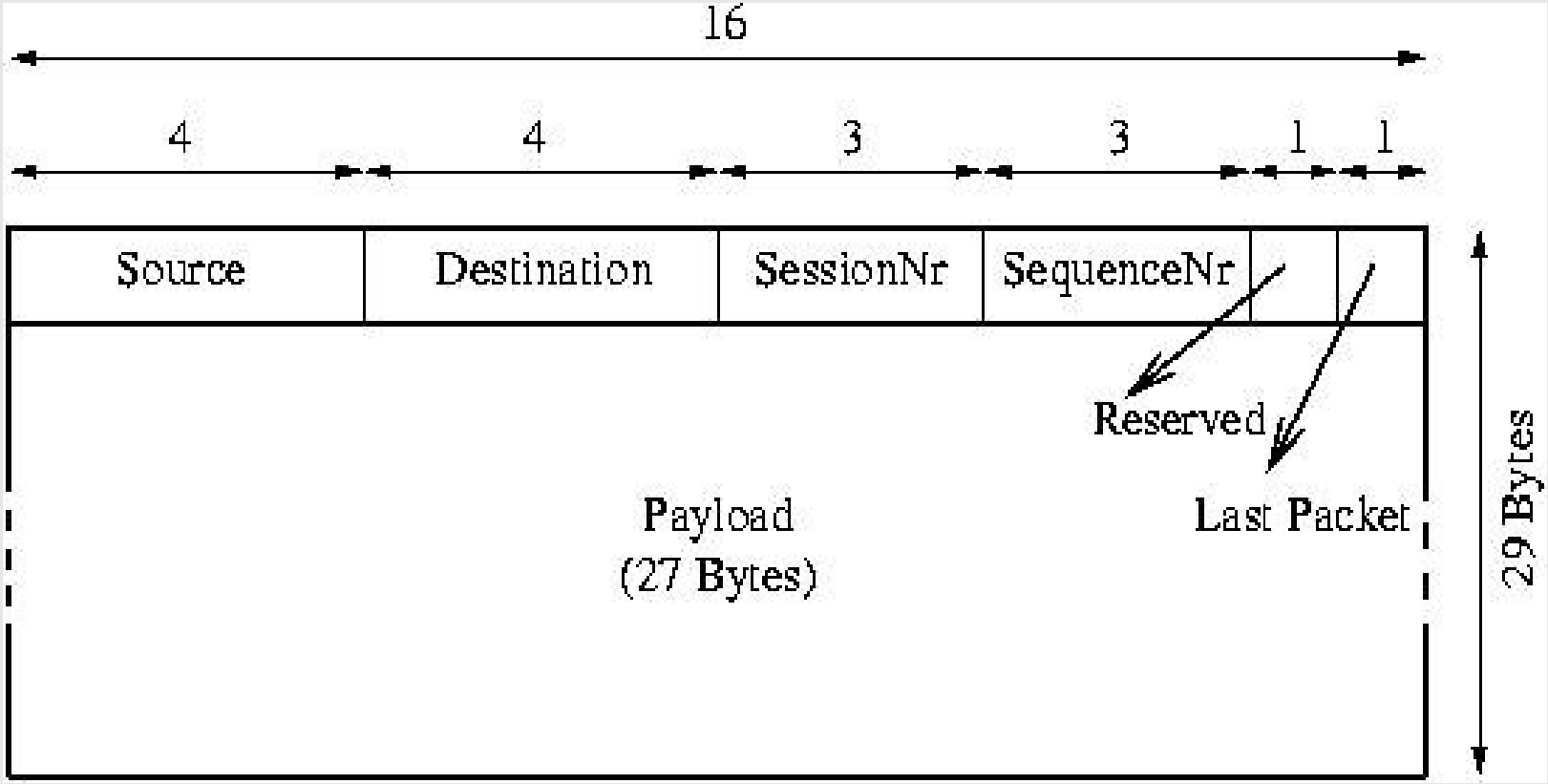
- Header (2 byte)
  - source address (4 bit)
  - destination address (4 bit)
  - session number (3 bit)
  - sequence number (3 bit)
  - reserved (1 bit)
  - last packet bit (1 bit)

Design – Transport Protocol (2)

- 27 byte payload
- reliable
- used to transport agent fragments
- no route information (mote to mote – one hop)

  => Agent contains route information if required

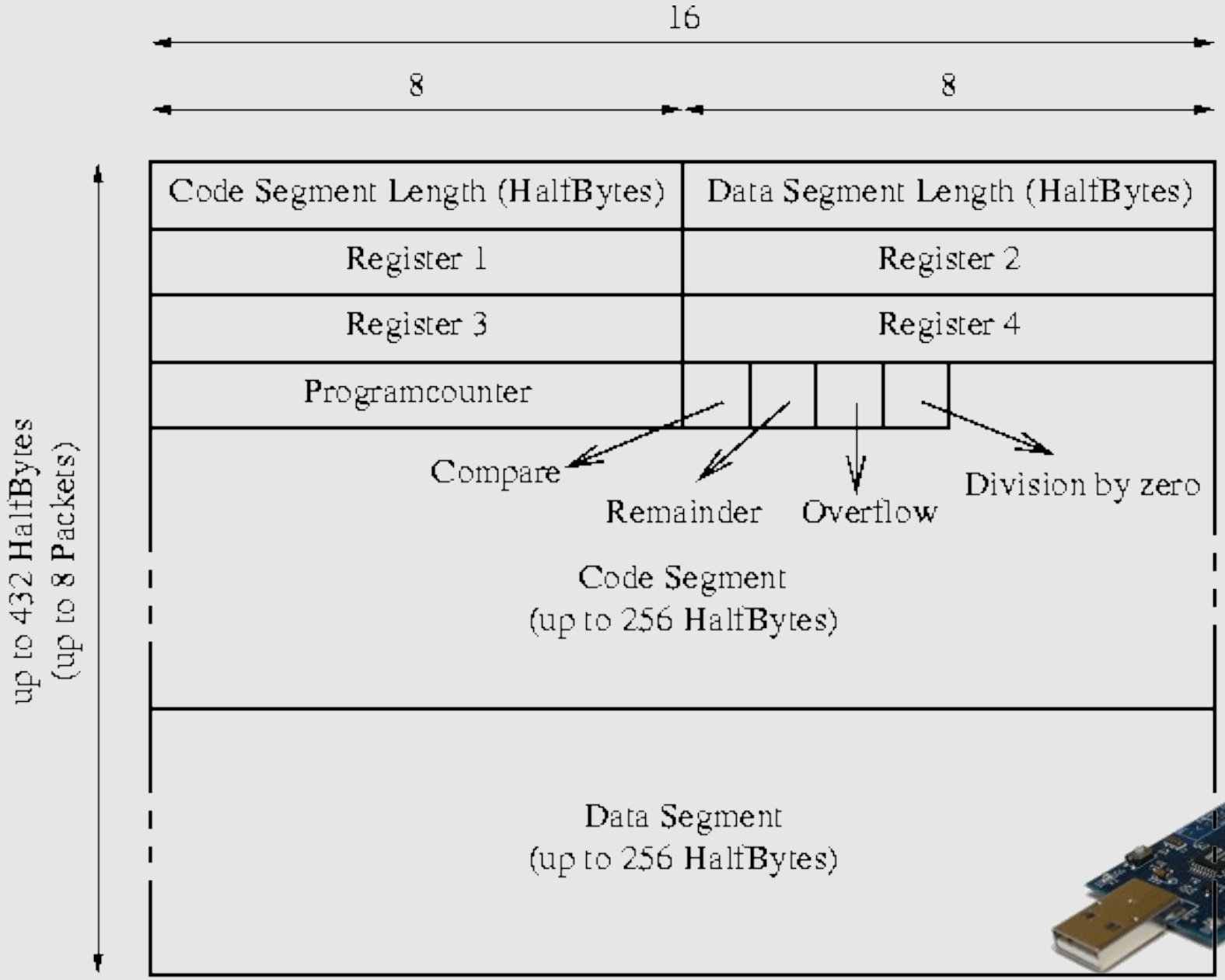
Design – Transport Protocol (3)

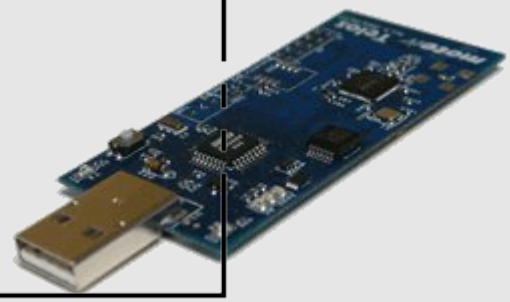


Design – Agent (1)

- Strong mobility
  - Highly Stateful (PC, registers, CS, DS, flags)
  - Mote provides
    - point-to-point communication
      - Transport Protocol (reliable)
      - Discovery Protocol (unreliable)
    - neighbour mote cache
  - whole program logic contained in the agent
- Data
  - variable size of CS and DS

Design – Agent (2)





Design – Agents

- Route Acquiring Agent
  - Basic agent that is hard coded onto every mote
  - Replicates itself around the network to get a route to the target mote
- Data Collection Agent
- ...

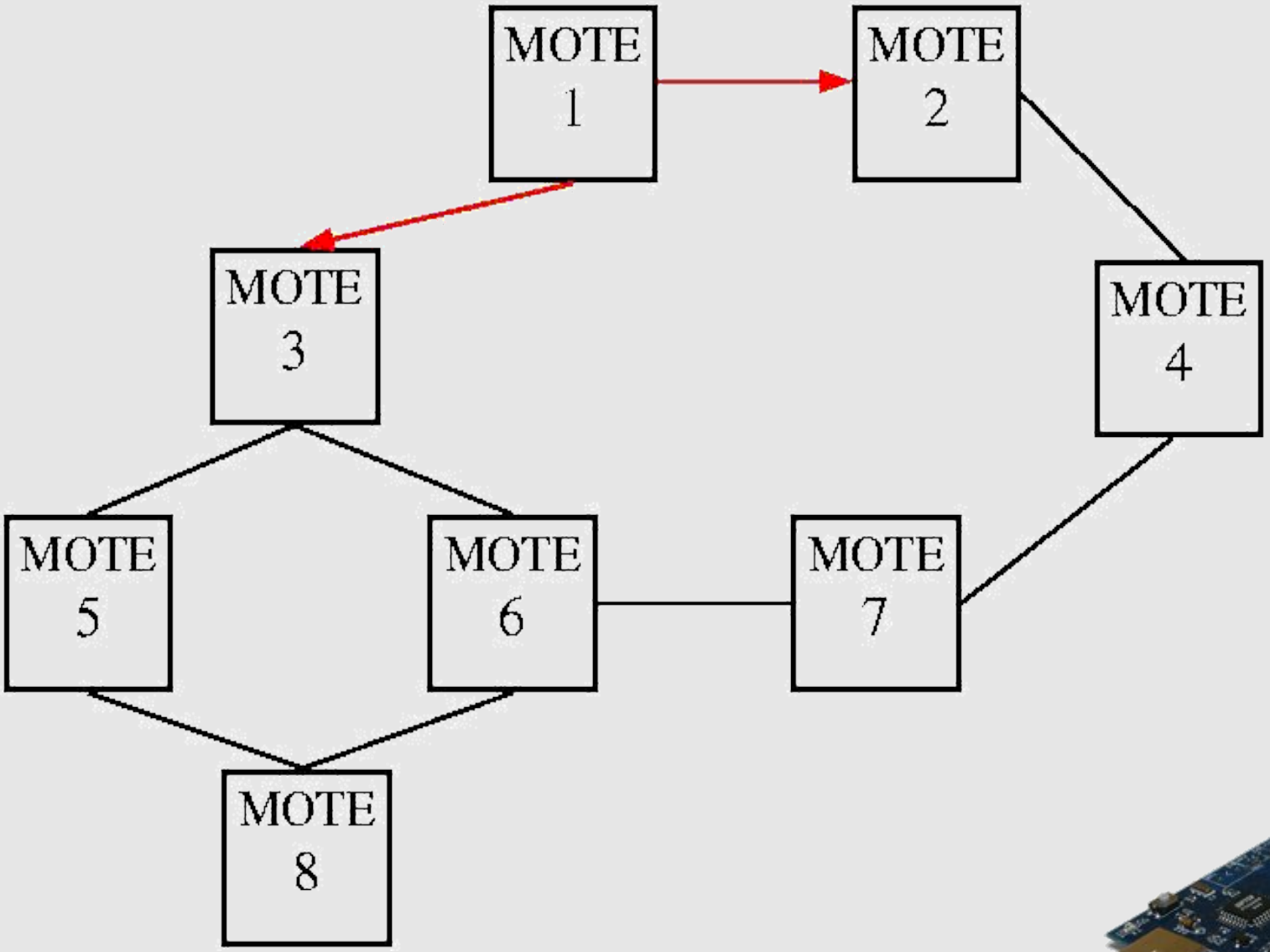
Design – RAA Rules

- save current mote address (route information)
- new neighbours -> replicate agent to each neighbour
- only known neighbours (circle!) -> kill agent
- target found -> go back to source using route information
- dead end -> kill agent
- back home -> transfer route information
  to calling agent

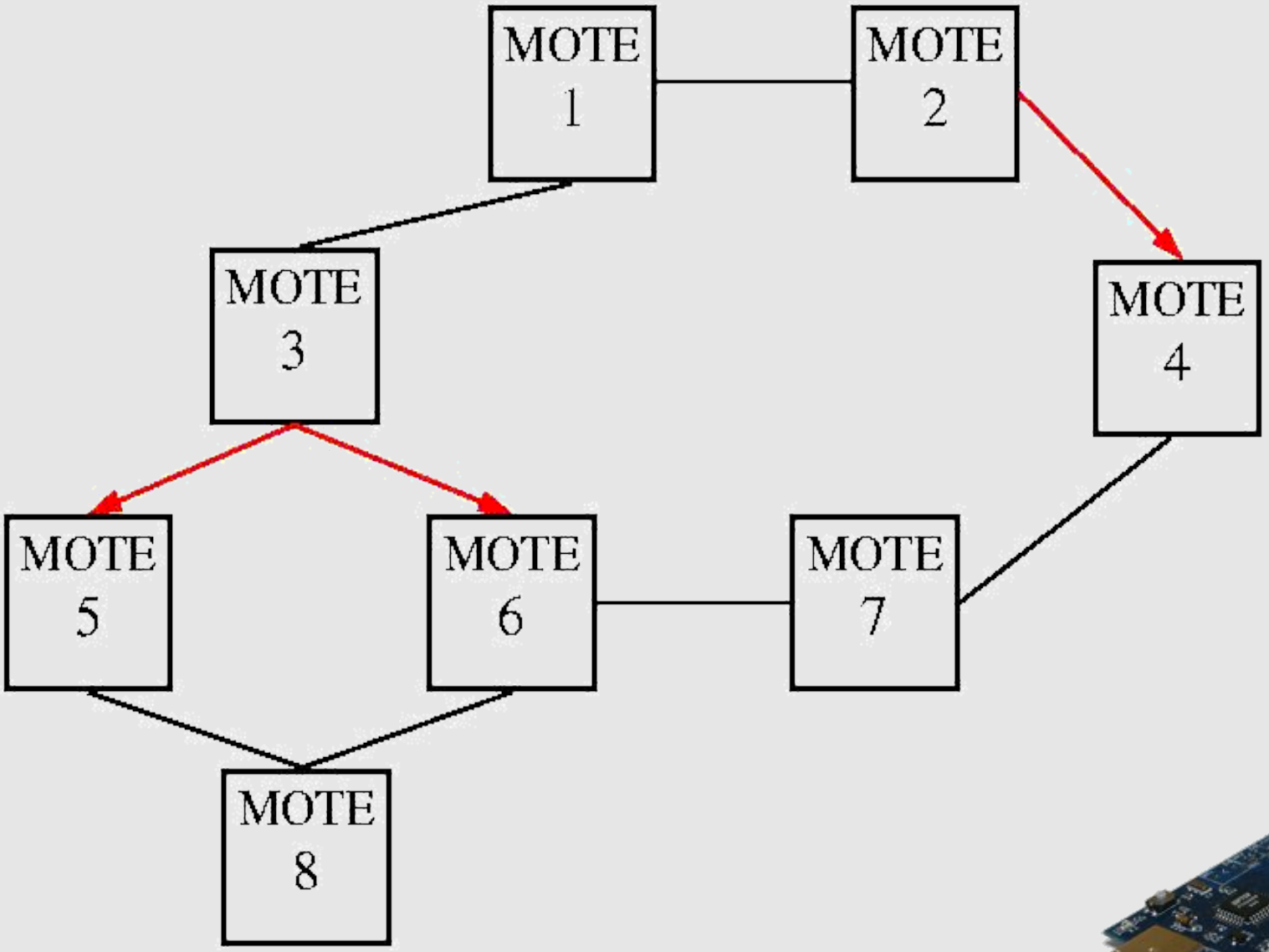
Design - RAA

- Type 1:
  RAA - redundant for reliability
  Explained in the following slides
- Type 2:
  RAA - optimized for less traffic
  Little modifications to Type 1

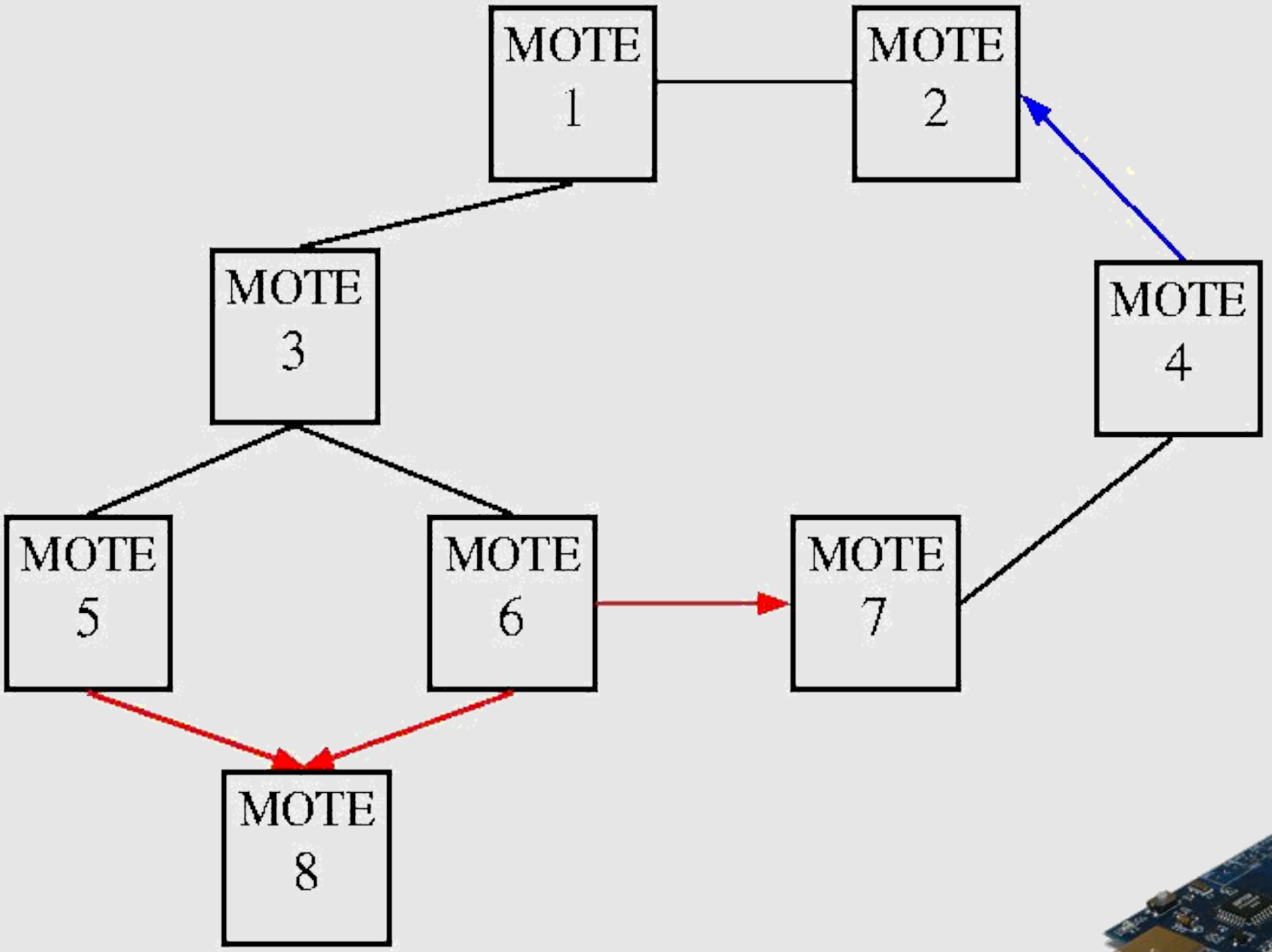
Design – RAA – Step 01



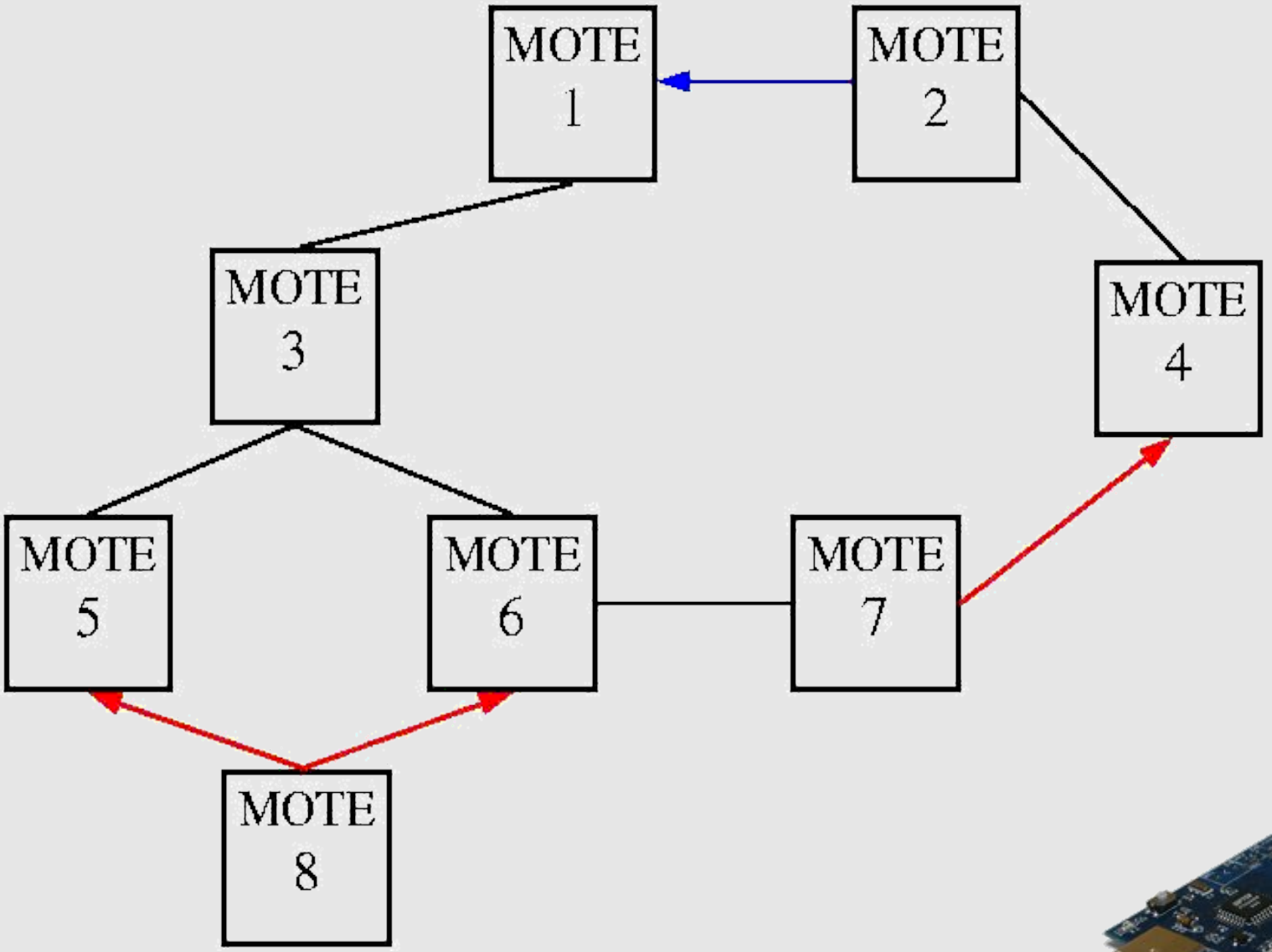
Design – RAA – Step 02



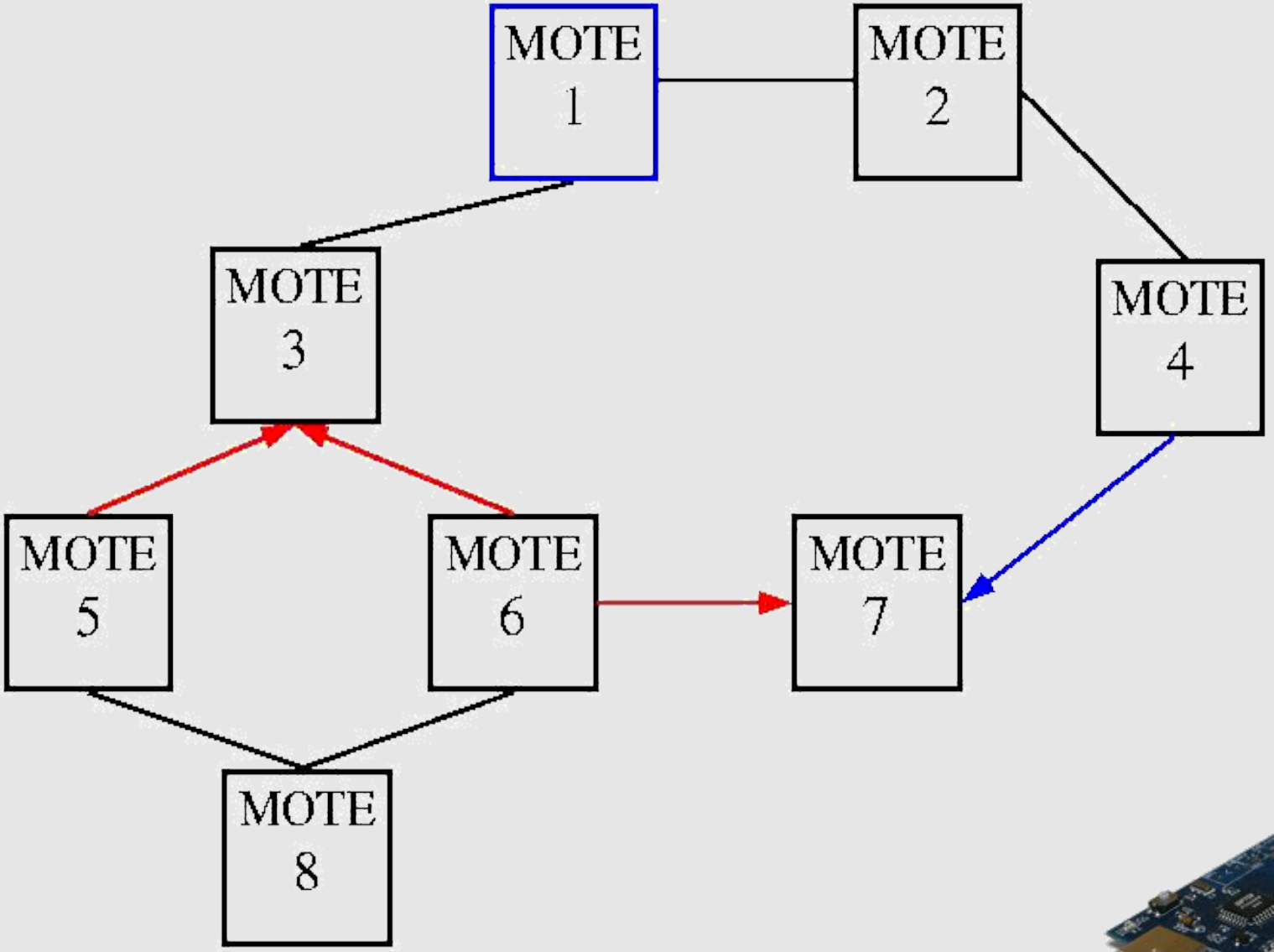
Design – RAA – Step 03



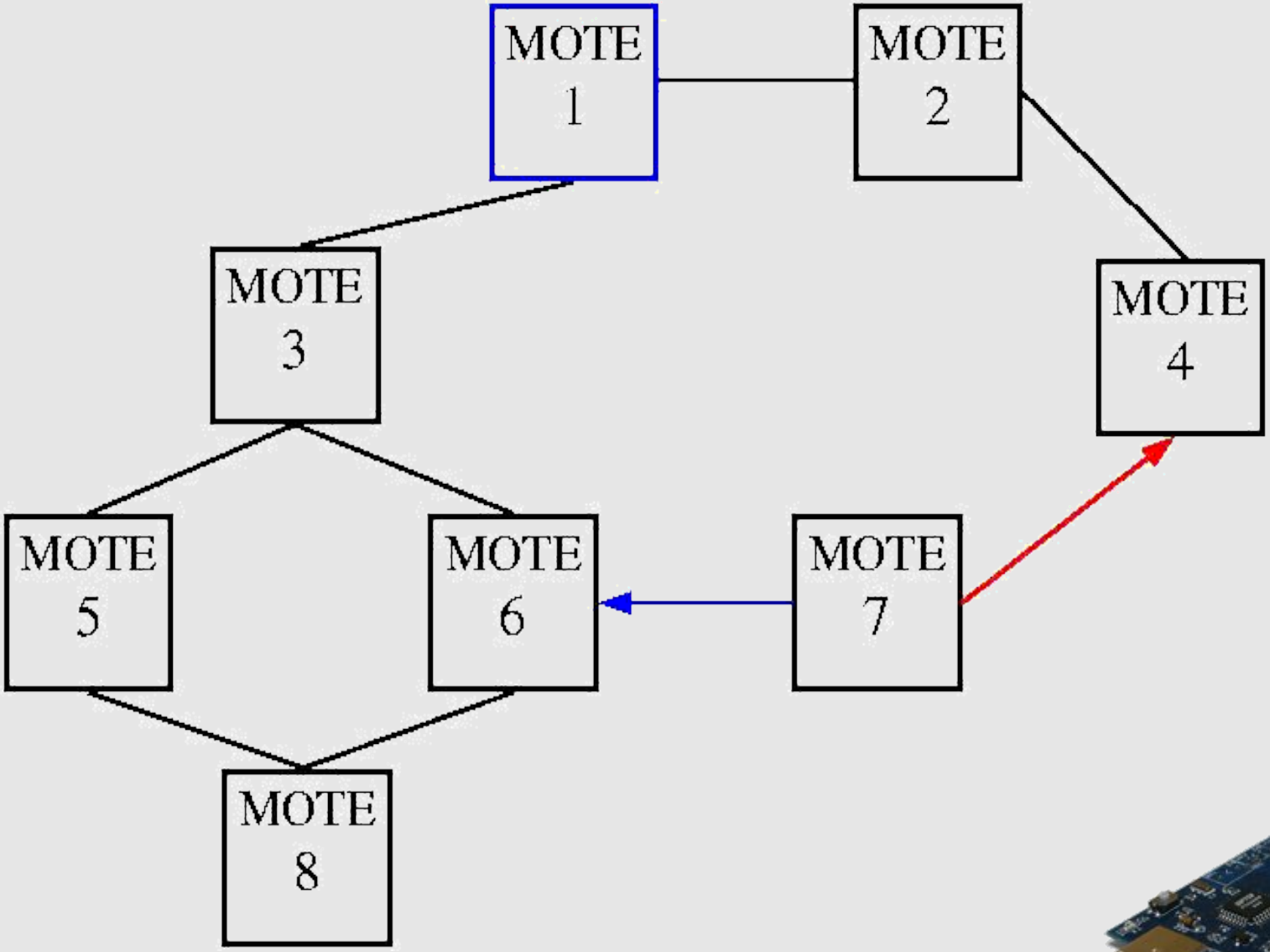
Design – RAA – Step 04



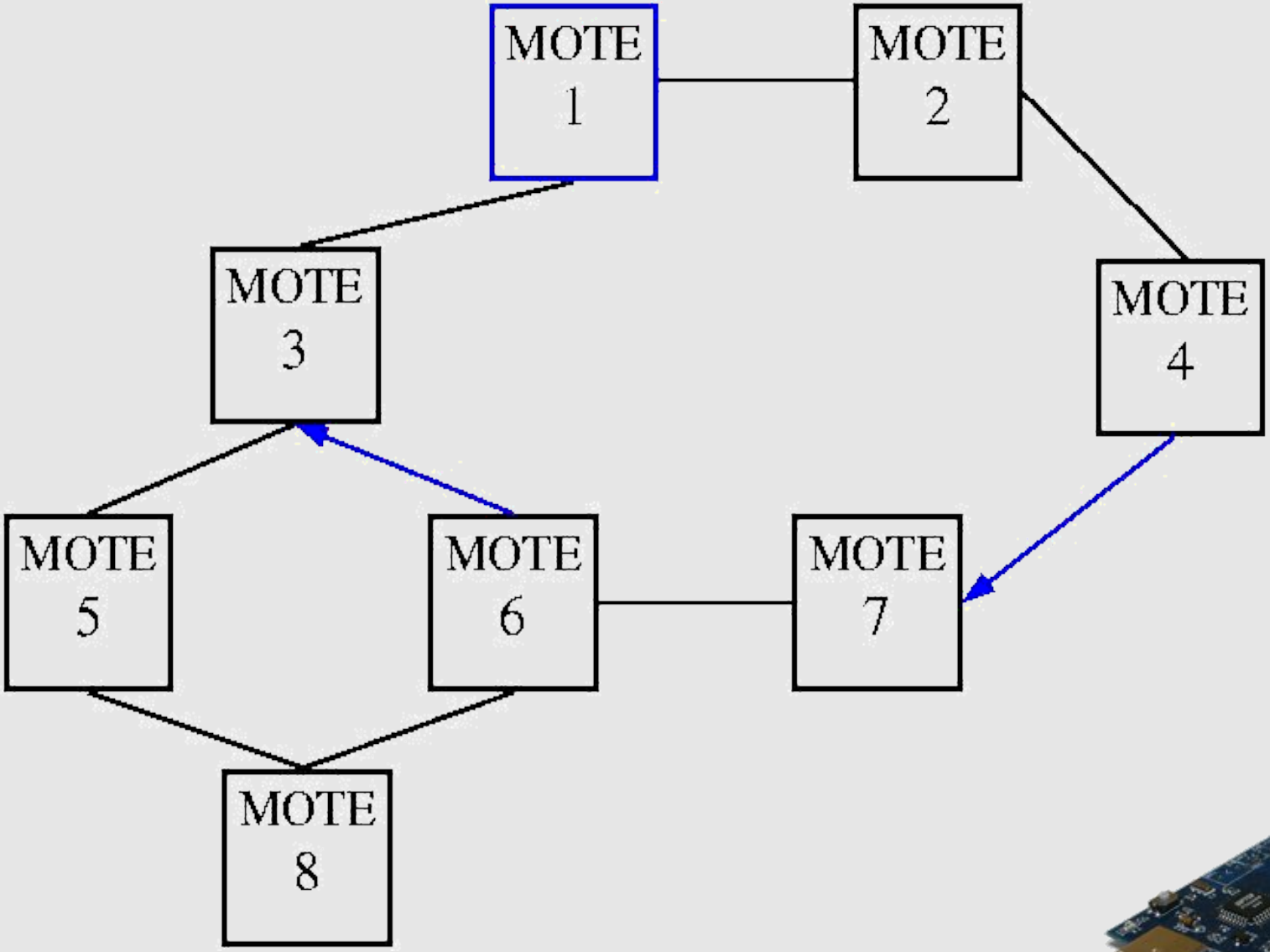
Design – RAA – Step 05



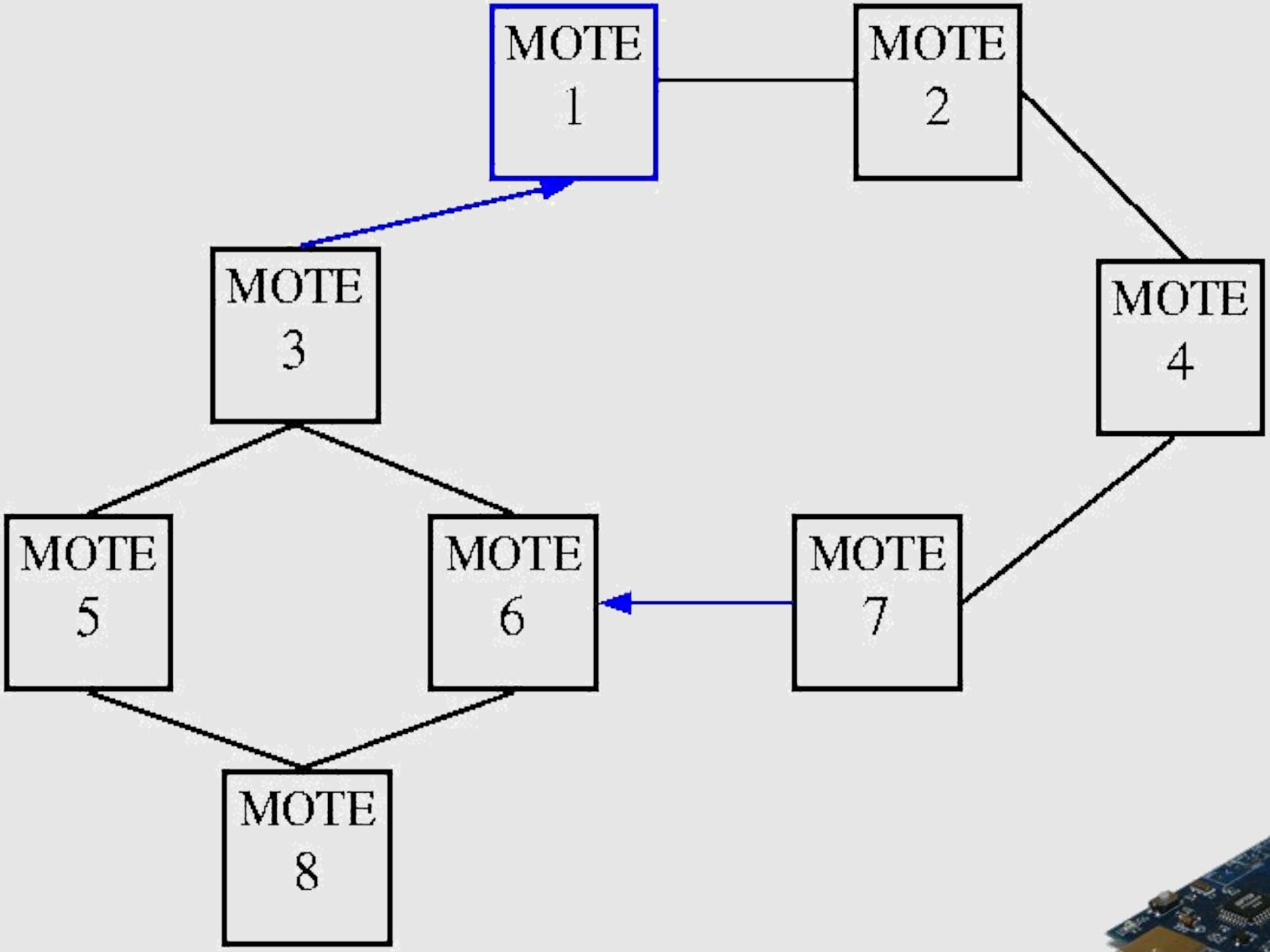
Design – RAA – Step 06



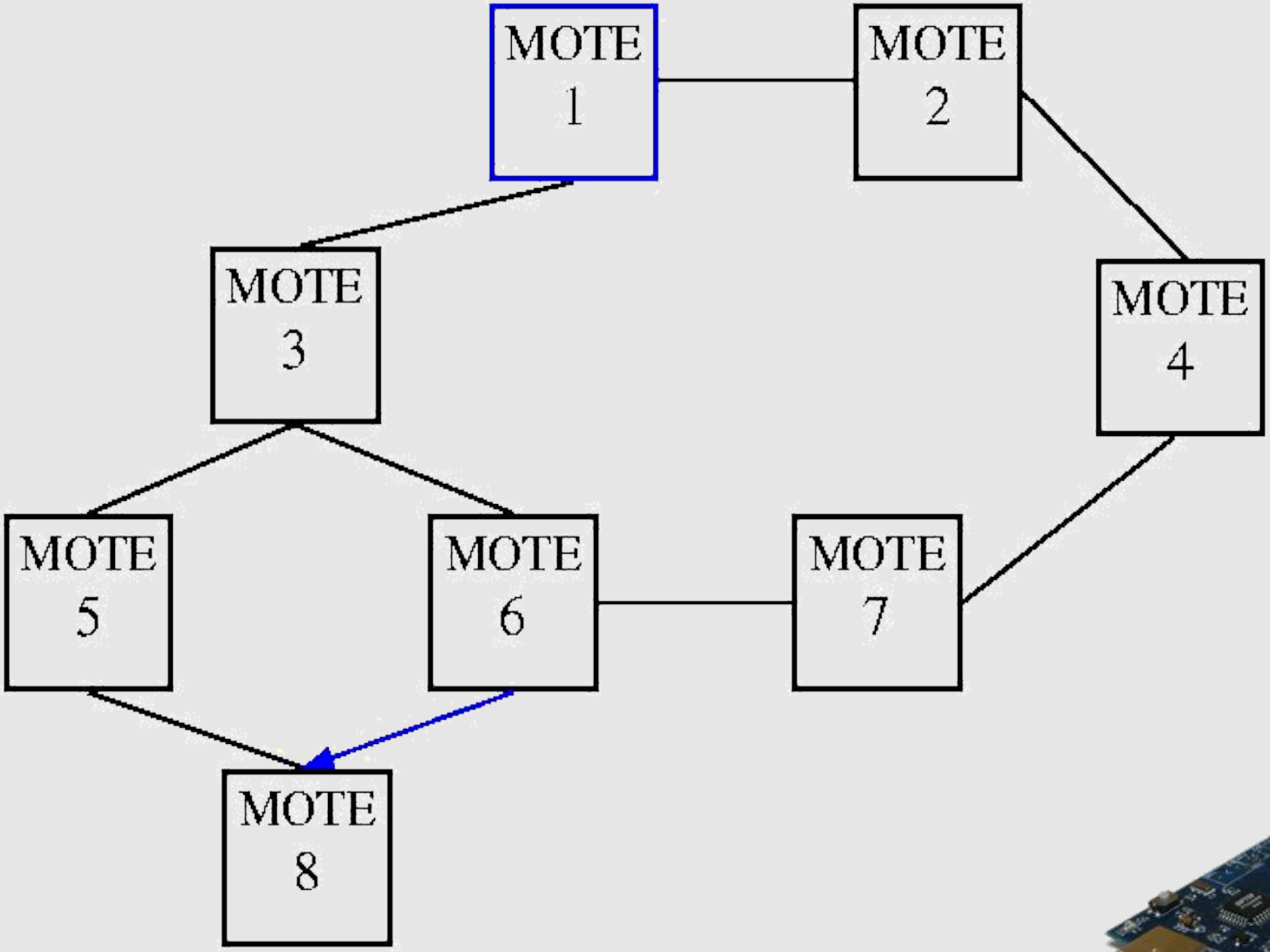
Design – RAA – Step 07



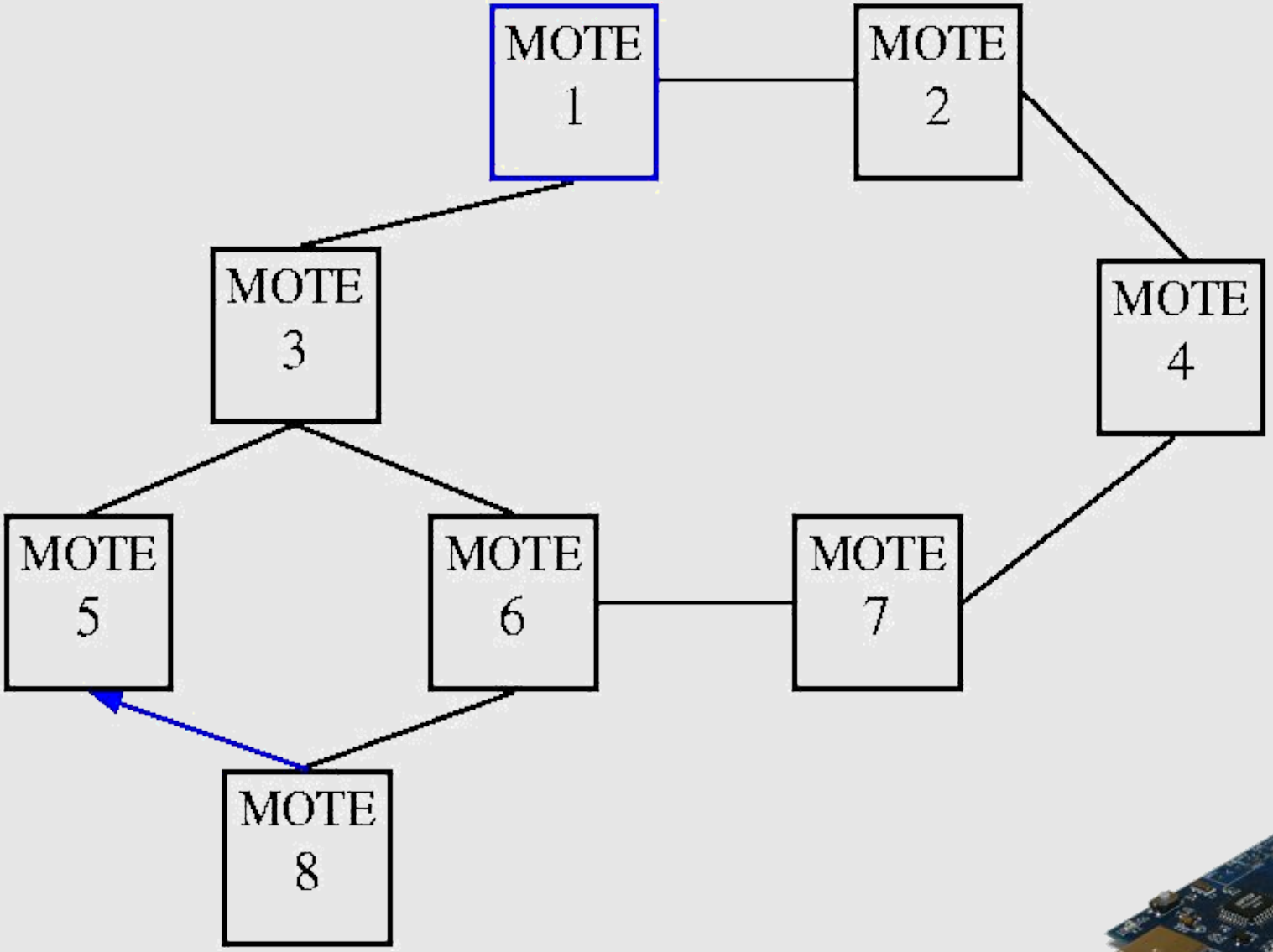
Design – RAA – Step 08



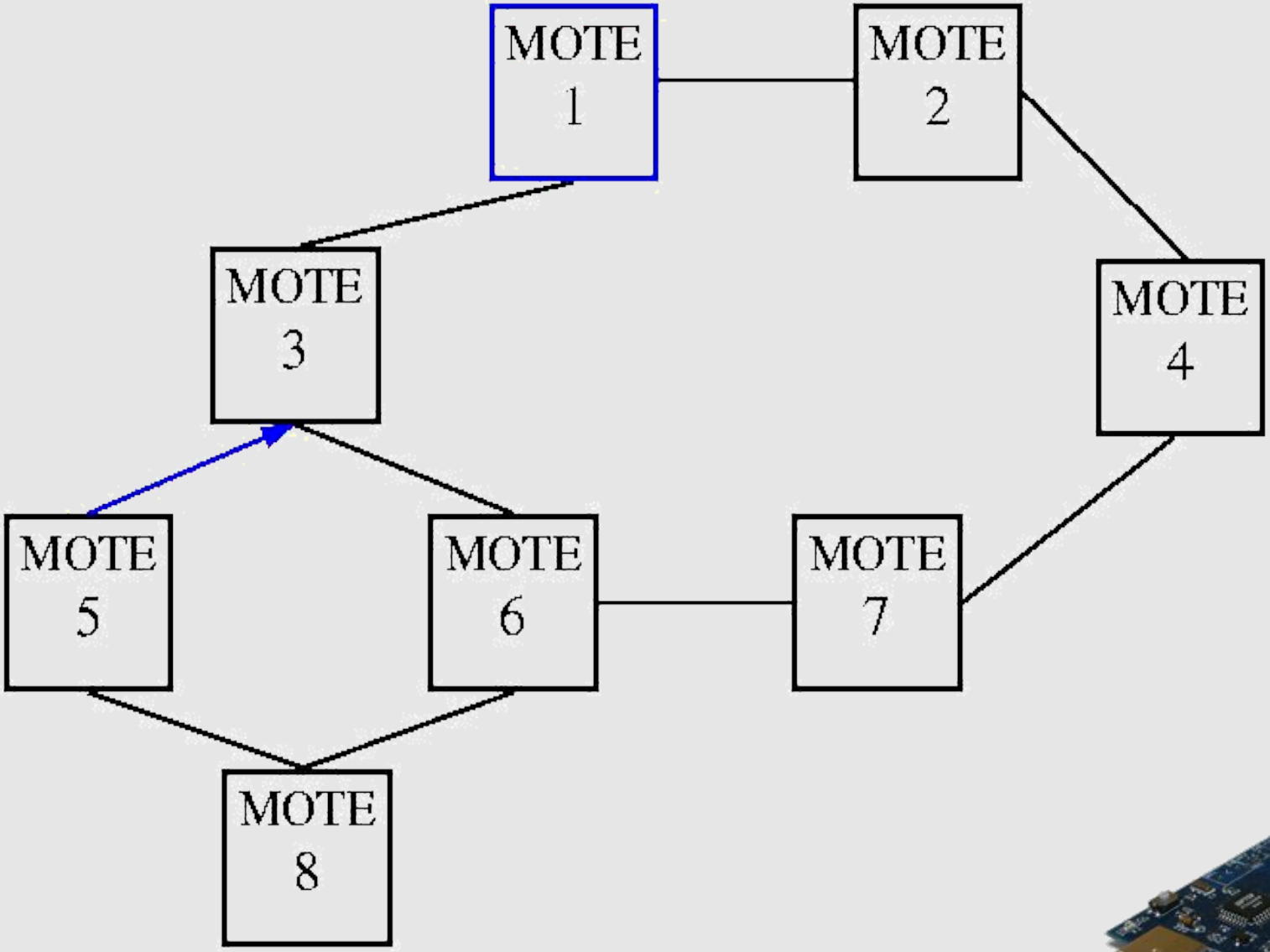
Design – RAA – Step 09



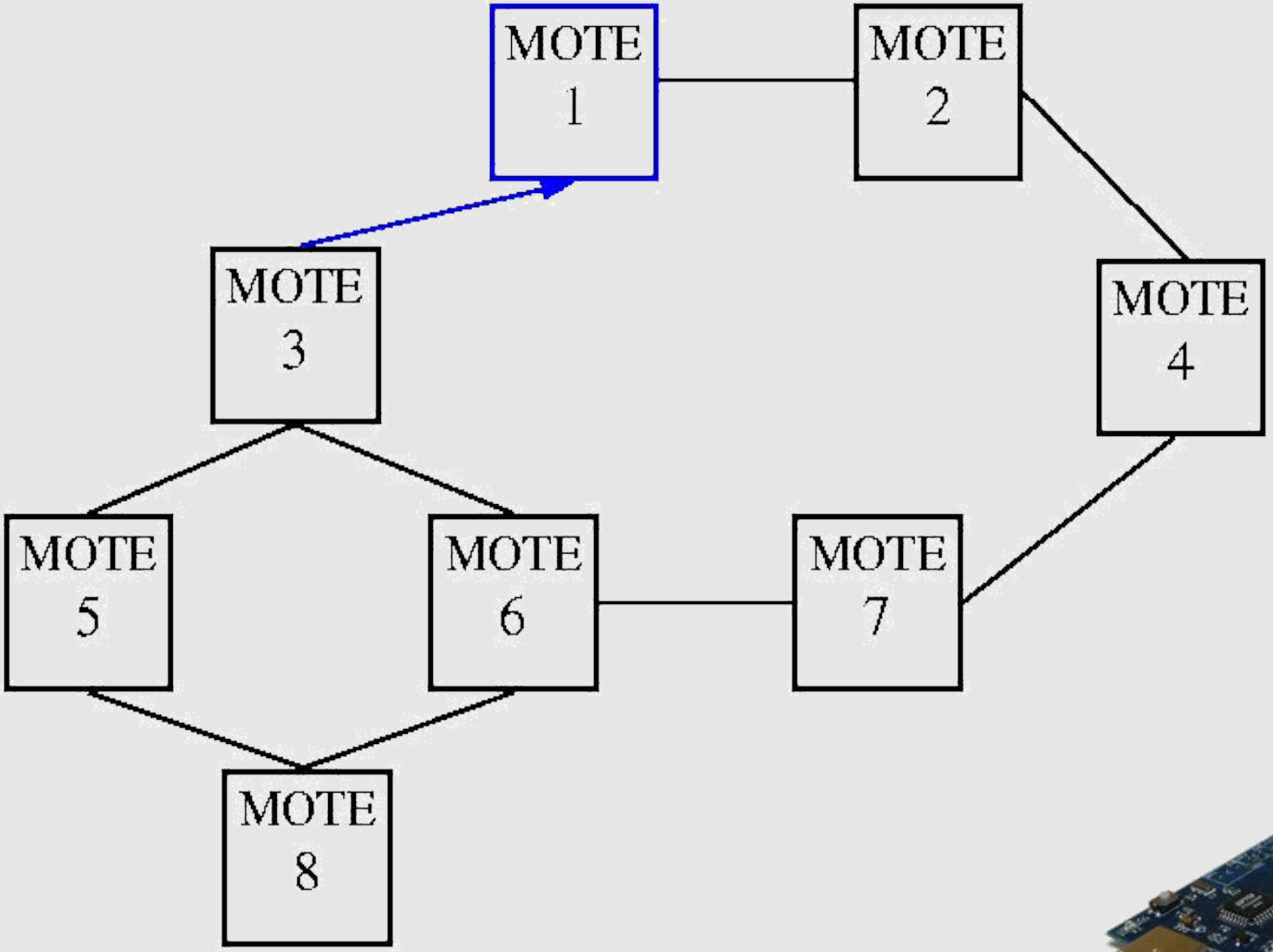
Design – RAA – Step 10



Design – RAA – Step 11



Design – RAA – Step 12



Implementation - Overview

- OS: TinyOS
- Programming language: nesC
- Virtual Machine also in C for testing/debugging purpose
- mote2pc communication via javax.comm

Implementation - Status

- Virtual Machine
  - all assembler commands working and tested
  - few capability calls working and tested
  - support four agents at the same time (simple scheduling policy)
- Discovery Protocol
  - Request
  - Answer
  - Neighbour caching

Demo - Test of Discovery Protocol

- Simple Agent which loops forever and calls request_neighbours()
- five motes with the same implementation
- Leds
  - blue led toggles if request sent
  - red led toggles if answer received
  - green led toggles if request received

Outlook

- Transport Protocol
- Application for injecting new agents into and retrieving data from the system
  - Translator for agent code to virtual machine code
  - UI for reading data
- Some basic Agents

The End

Questions &
Discussion