Protocol Design and Analysis

ITS 413 – Internet Technologies and Applications

Contents

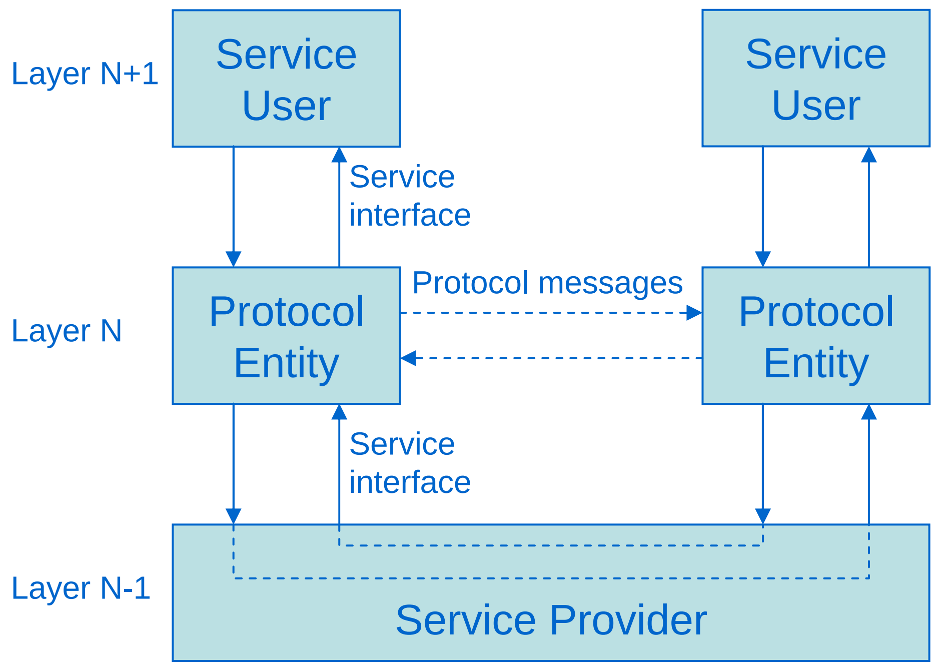
Motivation

- Many protocols are used in the Internet to communicate between computers:
  - Application and transport: end-to-end communication
  - Network: IPv4 and IPv6
  - Link layer and Physical: for link communications
- Important that protocols have good designs
  - Operate as we want them too; don't cause our computers to crash
  - Perform well
  - Secure; don't allow unnecessary security attacks
- This lecture gives some examples of:
  - How are protocols specified?
  - How are protocols analysed?
- Applies to all protocols and networks (not just Internet)

5 Elements of a Protocol

1. Service
   - What service does the protocol provide to other users?
   - Example: IEEE 802.11 MAC provides reliable delivery of frames between stations
2. Assumptions
   - What do we assume about the environment we are operating in?
   - Example: IEEE 802.11 MAC assumes physical layer can transmit bits in certain order to receiver
3. Vocabulary
   - What vocabulary of messages is used in the protocol?
4. Encoding
   - What is the format of each message?
5. Procedure (or Protocol) Rules
   - What rules do we follow to have consistent exchange of messages?
   - This is the hardest and most complex part!

Protocols and Layers

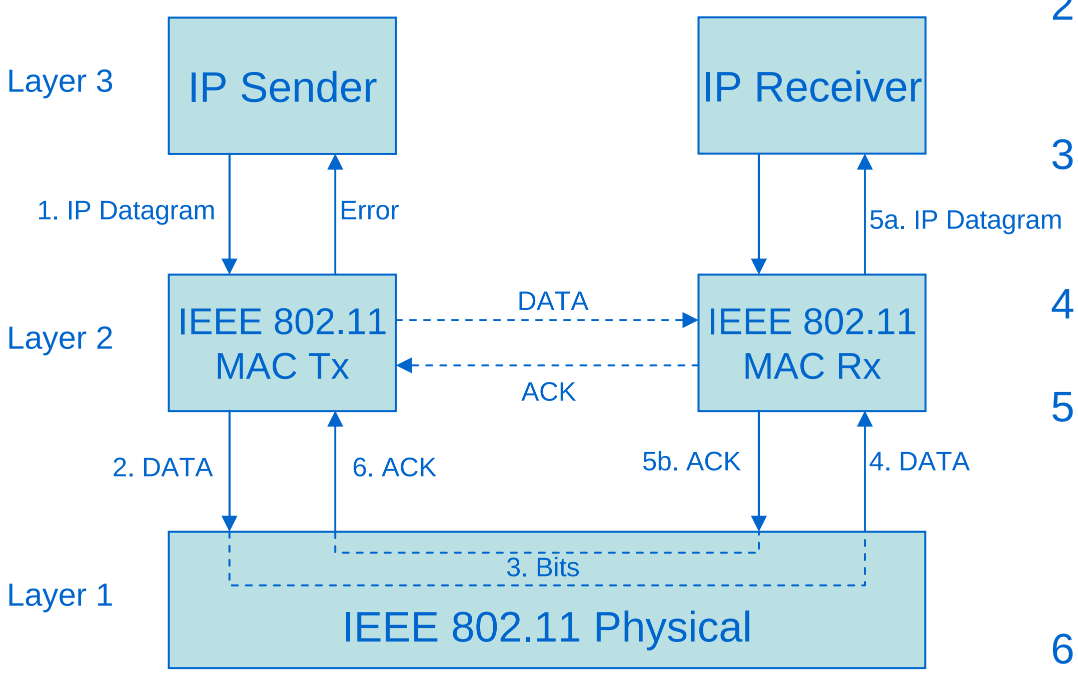


- Protocol is defined by the two (or more) protocol entities that exchange messages
- The protocol provides a service to upper layer users
- The protocol uses the service of lower layer

- Example 1:
  - Layer 5 users: FTP Client and FTP Server
  - Layer 4 protocol: TCP Protocol Entities at sender and receiver
  - Layer 3 service provider: IP
- Example 2:
  - Layer 4 users: TCP sender and receiver
  - Layer 3 protocol: IP
  - Layer 2 service provider: IEEE 802.11 MAC protocol

IEEE 802.11 Example

- Assume 802.11 DCF basic access mode



1. IP Sender generates an IP datagram and sends to MAC
2. MAC Tx protocol entity follows DCF rules and transmits DATA frame
3. Physical layer sends DATA frame as bits over wireless to receiver
4. Receiver passes DATA frame to MAC Rx protocol entity
5. MAC Rx protocol entity follows DCF rules:
   a. Sends IP datagram to IP receiver
   b. Sends ACK frame
6. ACK frame goes back to MAX Tx protocol entity via physical layer

Service Specification and Assumptions

- Specify the interface between protocol and upper layer users
- Usually use abstract messages (with parameters). Examples:
  - TCP sockets: open(), send(), recv()
  - OSI style: Invoke.request (ID=1, Flag=True); Invoke.confirm (ID=1)
- Interface should be independent of protocol specification
  - Higher layer users don’t care about protocol details, only concerned with service interface
- Assumptions about Environment:
  - What do you assume about users and lower layers?
    - Reliable delivery, ordering of messages, …

Vocabulary and Encoding

- Define the protocol message types and formats
- Example - IEEE 802.11 MAC:
  - Message types: RTS, CTS, DATA, ACK
  - Message formats, e.g. DATA frame format

Frame Control	Duration	A1	A2	A3	Sequence Control	A4	Frame Body	FCS

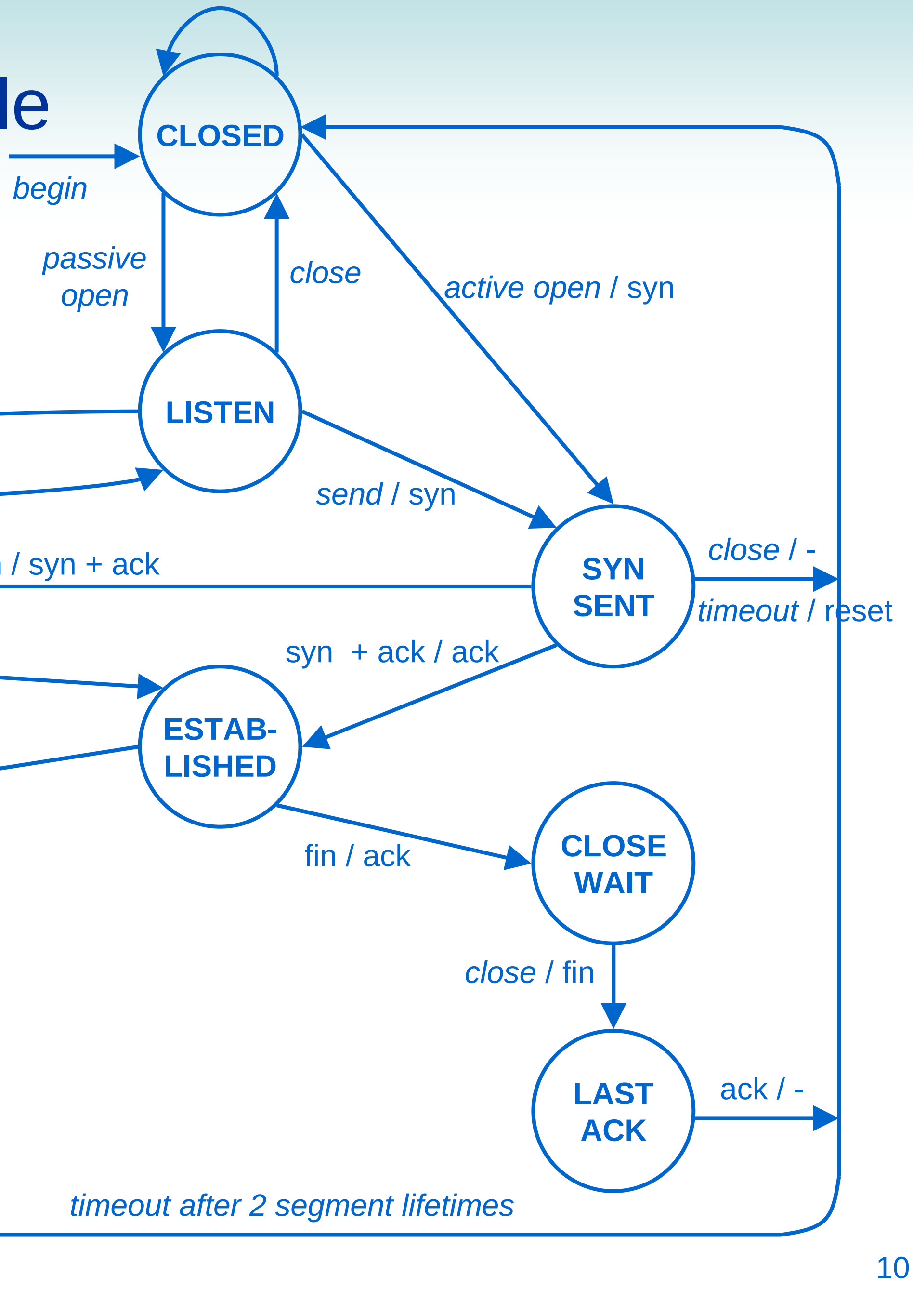
- Usually user header/frame format diagrams to illustrate each possible message

Protocol Rules

- What rules does each protocol entity follow?
  - IEEE 802.11 MAC DCF rules for how and when to transmit a DATA and ACK frame between stations
- Often described by a state machine:
  - Protocol entity can be in 1 of a set of states
    - E.g. IDLE, WAIT_FOR_DIFS, WAIT_FOR_ACK, …
  - In each state, a set of events can occur
    - Events are: receive data from higher layer; receive message from protocol entity; or timeout
    - E.g. ReceiveIPDatagram, ReceiveACK, BackoffComplete
  - Events lead to actions and possible change of state
    - Actions can be: send message, start timer, change counter
    - E.g. SendACK, StartBackoff, IncrementRetryCounter

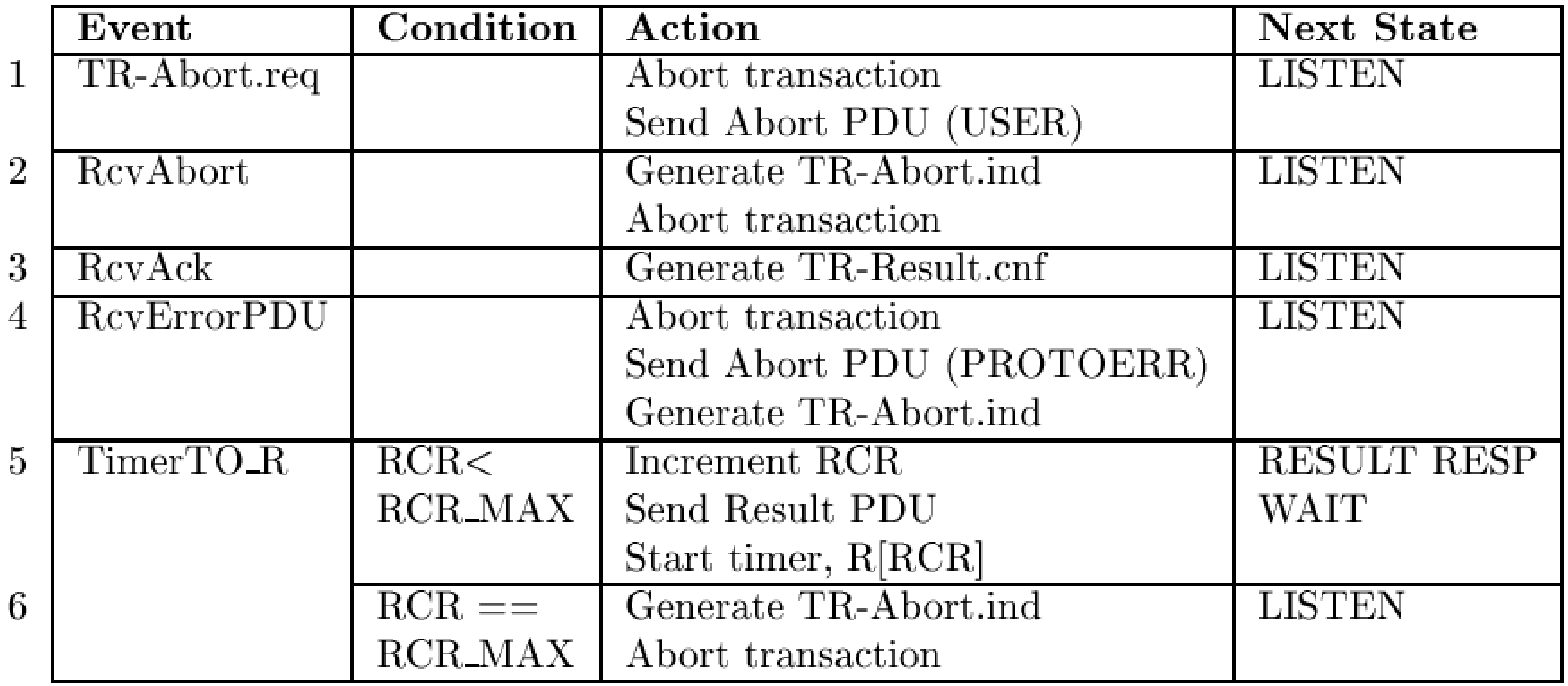
Protocol Rules Example

TCP State Machine



Protocol Rules Example

- WAP Transaction Protocol RESULT RESP WAIT table
  - There are 10 other similar tables for other states



	Event	Condition	Action	Next State
1	TR-Abort.req		Abort transaction Send Abort PDU (USER)	LISTEN
2	RcvAbort		Generate TR-Abort.ind Abort transaction	LISTEN
3	RcvAck		Generate TR-Result.cnf	LISTEN
4	RcvErrorPDU		Abort transaction Send Abort PDU (PROTOERR) Generate TR-Abort.ind	LISTEN
5	TimerTO_R	RCR< RCR_MAX	Increment RCR Send Result PDU Start timer, R[RCR]	RESULT RESP WAIT
6		RCR == RCR_MAX	Generate TR-Abort.ind Abort transaction	LISTEN

Protocol Specification Examples

- IETF RFCs
  - Text based, some text-based graphics and tables
  - Usually informal descriptions of protocols
- IEEE standards
  - Usually quite detailed standards with formal descriptions (e.g. state machines)
- OSI standards
  - Usually more detailed standards than RFCS and include some formal descriptions
- Many ETSI and mobile communication standards are closer to OSI style
  - WAP, GSM, 3G, …

Analysing Networks and Protocols

- Make sure the system operates correctly and efficiently
- Types of Analysis
  - Functional Analysis
    - Does the system do what it is supposed to do? Does it operate as we expect it to?
  - Performance Analysis
    - Does the system meet our performance goals? What are the performance limits and bottlenecks?
  - (Security Analysis)
    - Is the system secure?
- When is the analysis performed?
  - Design stage: ideally, in the design of a new protocol, all analysis stages should be completed before you implement and deploy
  - Network Management: during network operation we may want to fix bottlenecks and see what will happen if we change the network or protocols

Functional Analysis Techniques

- Informal
  - Use logic and experience to reason about the design
  - Problem: very hard to consider all possible cases
    - Like software testing – you can cover many cases, but cannot prove absence of bugs or certain properties
- Formal
  - Use mathematical techniques to model the protocol/network and then prove properties
    - Does the protocol have any deadlocks (can't get out of the state)?
    - Does the protocol get stuck in any loops?
    - …
  - Examples: state machines, Petri nets, automata, process algebra, …
  - How do you prove properties:
    - State space exploration: explore all possible states of system (also called model checking)
    - Infer properties using theorems
  - Problem: often many states (millions, billions and more), therefore hard to generate and analyse all of them
  - Need good software tools to support modelling and analysis

Performance Analysis Techniques

- Informal
  - Based on experience with other protocols and components of networks, reason about (estimate) the performance
  - Useful for high level results, e.g.
    - “this protocol will perform better than another protocol”
    - “this network will have acceptable performance”
- Mathematical
  - Create a mathematical model (e.g. set of equations) of the protocol or network and obtain performance results
  - Examples: queuing theory for networks; information theory for physical transmission of bits
  - Can obtain results for any set of parameters
  - Problem: Very hard to accurately model modest to large networks/protocols; need to make a lot of assumptions

Performance Analysis Techniques

- Simulation
  - Create a discrete event model of the protocol/network in software then execute for many different cases
  - Example simulation tools: OPNET, NS2, GlomoSim, Matlab, …
  - Problem: Does not cover all cases (unlike mathematical models); need assumptions or simplified models
- Experimentation
  - Deploy the protocols/network and experiment with small test cases
  - Problem: costly and complex to setup; does not cover all cases
- Hybrids
  - Many simulation software tools now support integration with mathematical models and experiment results or networks
    - E.g. connect simulation software with real network

Modelling Protocols

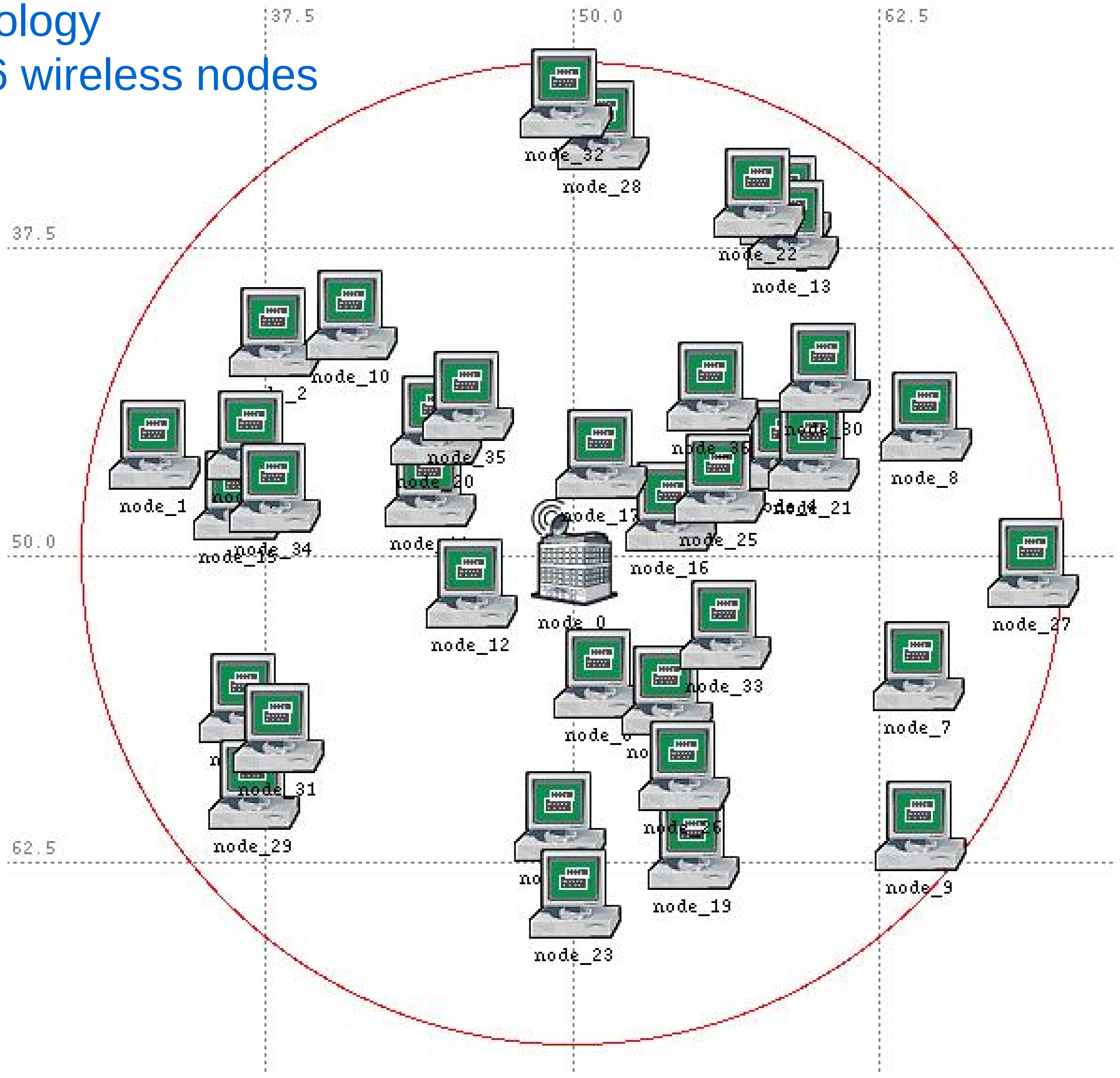
- Why create models?
  - Analysis techniques use models:
    - Functional Analysis: generate all possible states from model
    - Performance Analysis: simulate execution of models
    - Security Analysis: based on state models and theorem proving
  - Help with understanding of system
    - Can see how it is going to work
  - Cheaper than implementing and deploying in a real network
- What are the models?
  - Often state-based models (same as protocol rules)
    - In some state, if an event occurs, perform some action and change to a state
  - Assumptions to simplify implementation
    - Model source code may be simpler than the real protocol source code
  - GUI-based models make development faster and more convenient

OPNET Modelling Software

- OPNET IT Guru
  - Network simulation software from OPNET for performance analysis
  - User selects network components (routers, hosts, servers, links, …) and creates network topology
  - User can configure parameters for network components
  - Network components are generated from node and state models for protocols (written in C)
  - User selects traffic that is generated and statistics to be collected then runs simulations
  - Results can be viewed and analysed in OPNET
- OPNET Modeller
  - Includes features of OPNET IT Guru, plus access to node and state models (e.g. source code for protocol models so you can modify or develop your own protocols)

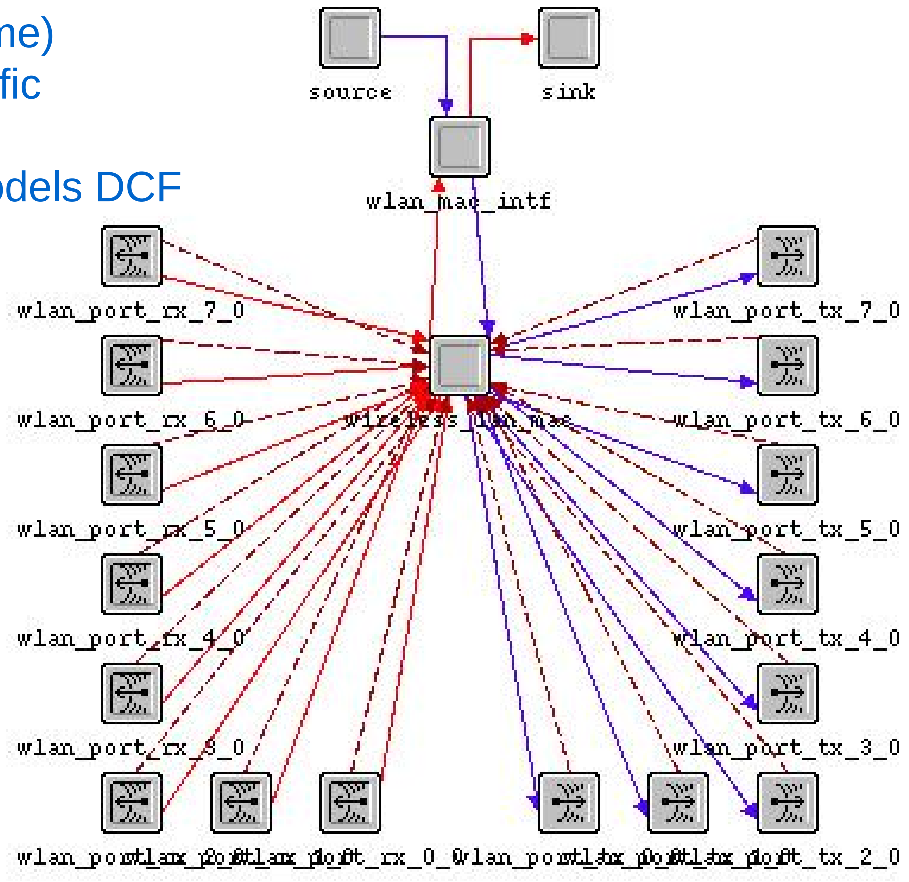
OPNET Model Example

Network Topology
- contains 36 wireless nodes

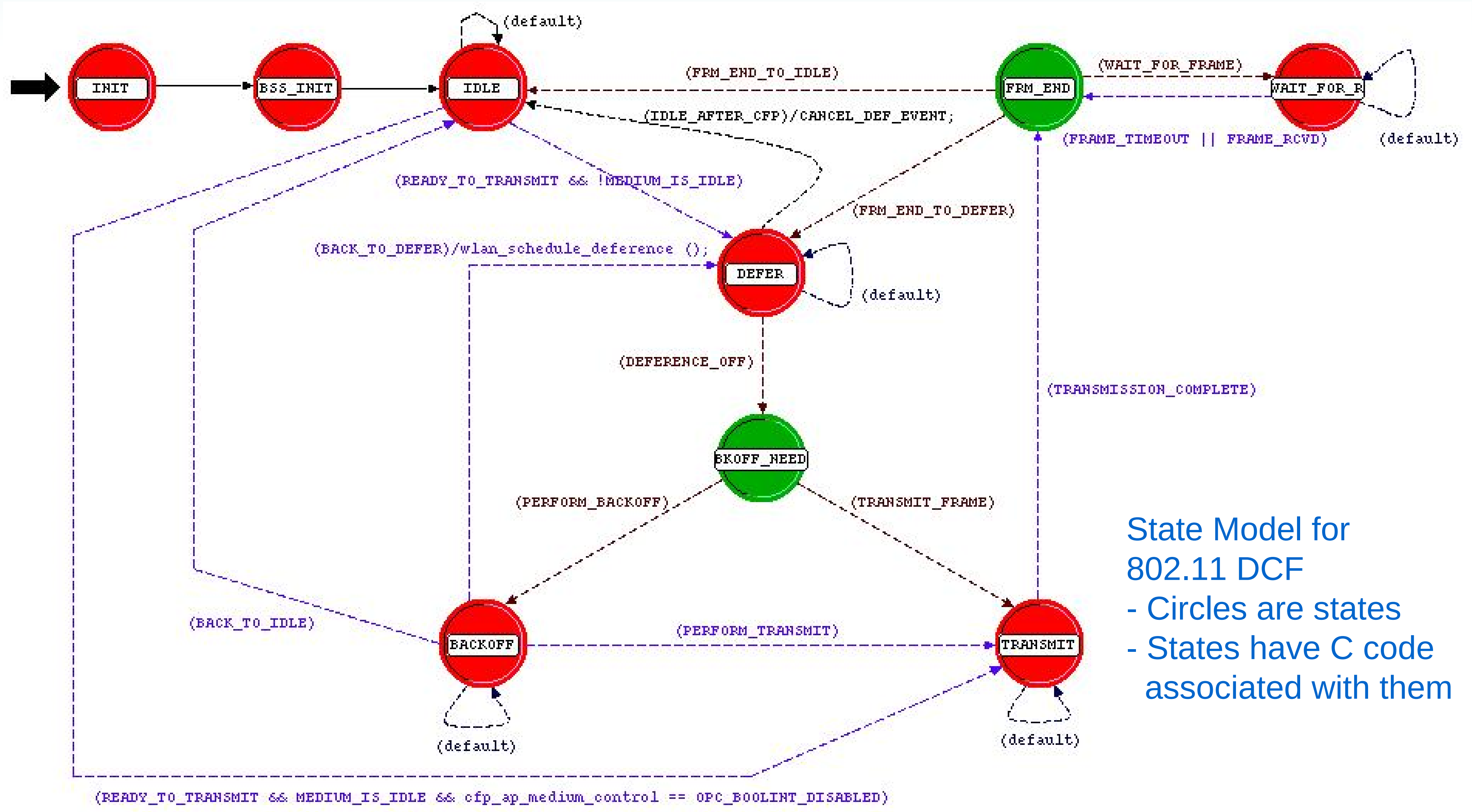


OPNET Model Example

Node Model
 - model of 1 wireless node
(all other nodes are same)
 - source generates traffic
 - sink receives traffic
 - wireless_lan_mac models DCF



OPNET Model Example



State Model for 802.11 DCF
- Circles are states
- States have C code associated with them