Design Examples and Case Studies of Program Modeling and Programming with RTOS-1:

Lesson-3

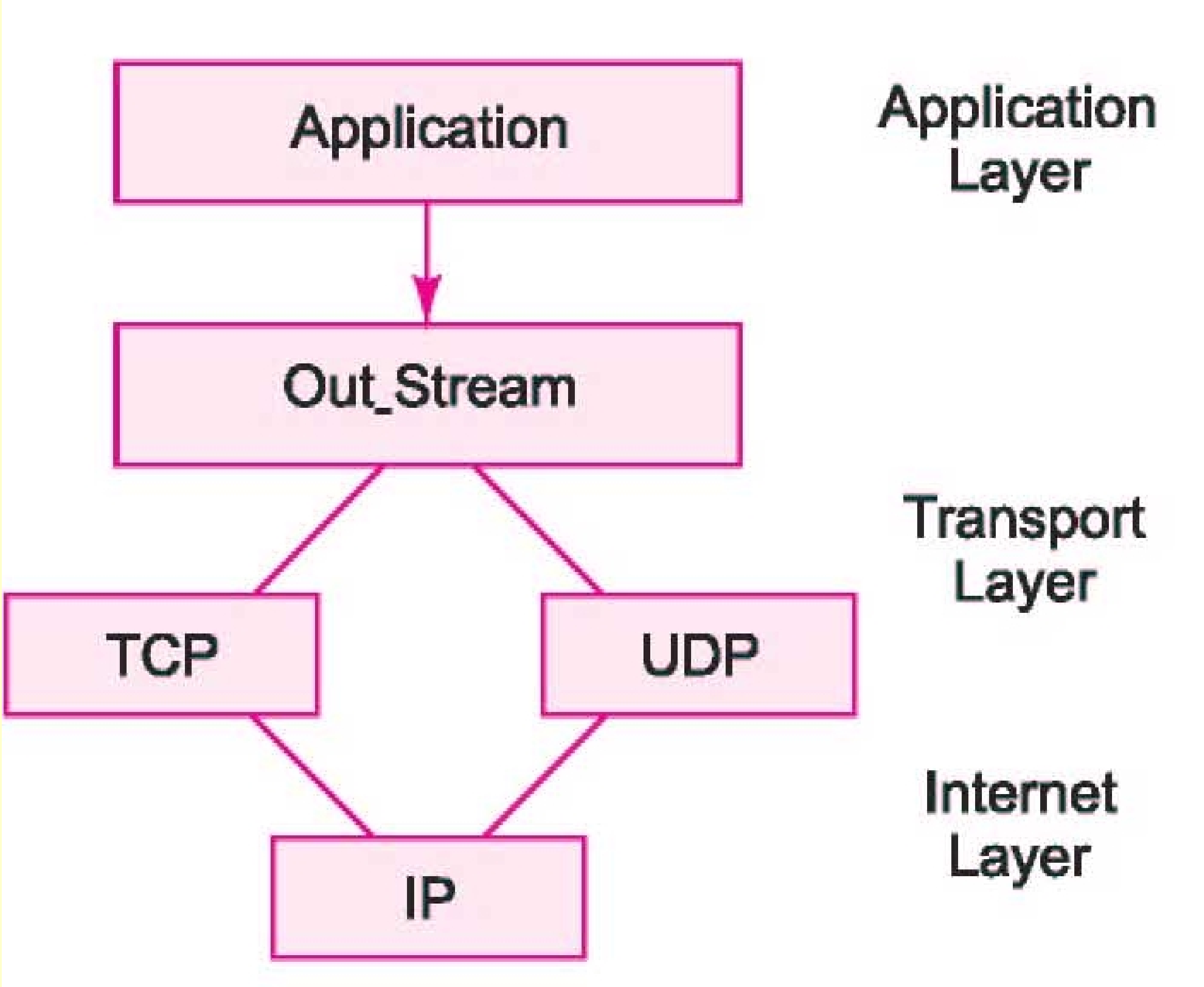
CASE STUDY OF CODING FOR SENDING APPLICATION LAYER BYTE STREAMS ON A TCP/IP NETWORK USING RTOS VxWorks

1. Specifications

TCP/IP Stack

- In TCP/IP suite of protocols, application layer transfers the data with appropriate header words to transport layer.
- At transport layer, either UDP or TCP protocol is used and additional header words placed.

TCP/IP Stack transmitting subsystem



TCP/IP Stack

- UDP is connection-less protocol for datagram transfer of total size including headers of less than 2^{16} Byte.
- TCP is connection oriented protocol, in which data of unlimited size can be transferred in multiple sequences to the internet layer
- At internet layer, the IP protocol is used and IP packets are formed, each packet with an IP header transfers to the network through network driver subsystem.

TCP/IP stack

- A TCP/IP stack network driver is available in the RTOSes
- It has a library, *sockLib* for socket programming.
- The reader may refer to VxWorks Programmer's Guide when using these APIs and *sockLib*.

TCP/IP stack

- Since the objective of case study is to learn the use of IPCs in multitasking RTOS through a case study.
- However, the present case study, without resorting to the network socket driver APIs in VxWorks, the required software to be embedded in the system is described to understand applications of IPC functions in VxWorks.

2. Requirements

Purpose

- To generate the byte stream for sending on the network using TCP or UDP protocol at transport layer and IP protocol at network layer

Inputs

- Bytes from application layer
- Notification SemTCPFlag or SemUDPFlag in case of TCP sequences or UDP datagram, respectively

Signals, Events and Notifications

- After forming the packets at IP layer, SemPktFlag for network driver task

Outputs

- TCP or UDP Byte stream to destination socket

Functions of the system

- An HTTP application data is sent after encoding headers at transport and network layers.
- Tasks are scheduled in five sequences (i) Task_StrCheck, (ii)Task_OutStream,

  (iii) Task_TCPSegment or Task_UDPDatagram,

  (iv) Task_IPPktStream

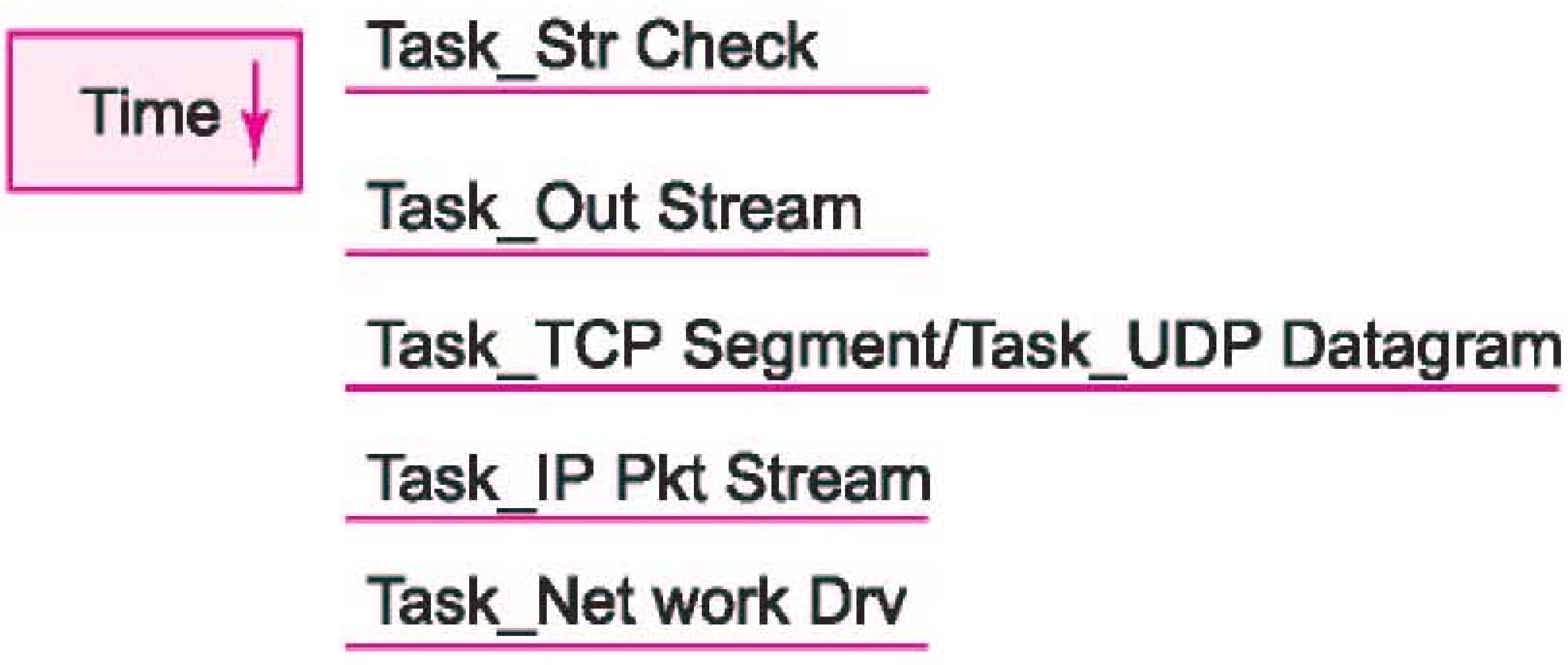
  (v) Task_NetworkDrv

Test and validation conditions

- A loop back from destination socket should enable retrieval of application data stream as originally sent
- Buffer Memory over flow tests

3. Classes and Objects

Tasks and their scheduling sequence in TCP/IP Stack transmitting subsystem



Time ↓

Task_Str Check

Task_Out Stream

Task_TCP Segment/Task_UDP Datagram

Task_IP Pkt Stream

Task_Net work Drv

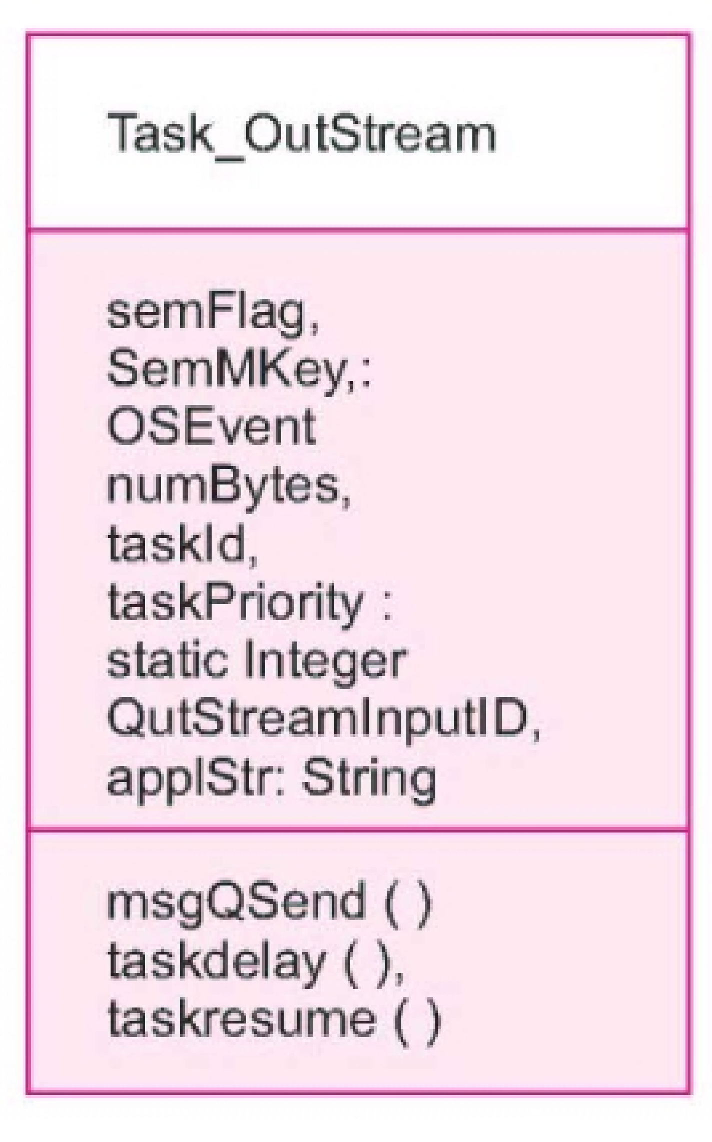
Classes for TCP or UDP byte tream on TCP/IP network

- Task_TCP is an abstract class
- Extended class(es) is derived to create objects TCP or UDP packet streams to a network.

Classes for TCP or UDP byte tream on TCP/IP network

- Task_StrCheck is to check and get the string.
- Task_OutStream extends from the two classes Task_StrCheck and Task_TCP.
- Task_TCPSegment creates a stream from TCP segment for IP layer. When datagram is to be sent then Task_UDPDatagram creates a stream using from Task_OutStream output bytes.

Class Task_OutStream



Task_OutStream

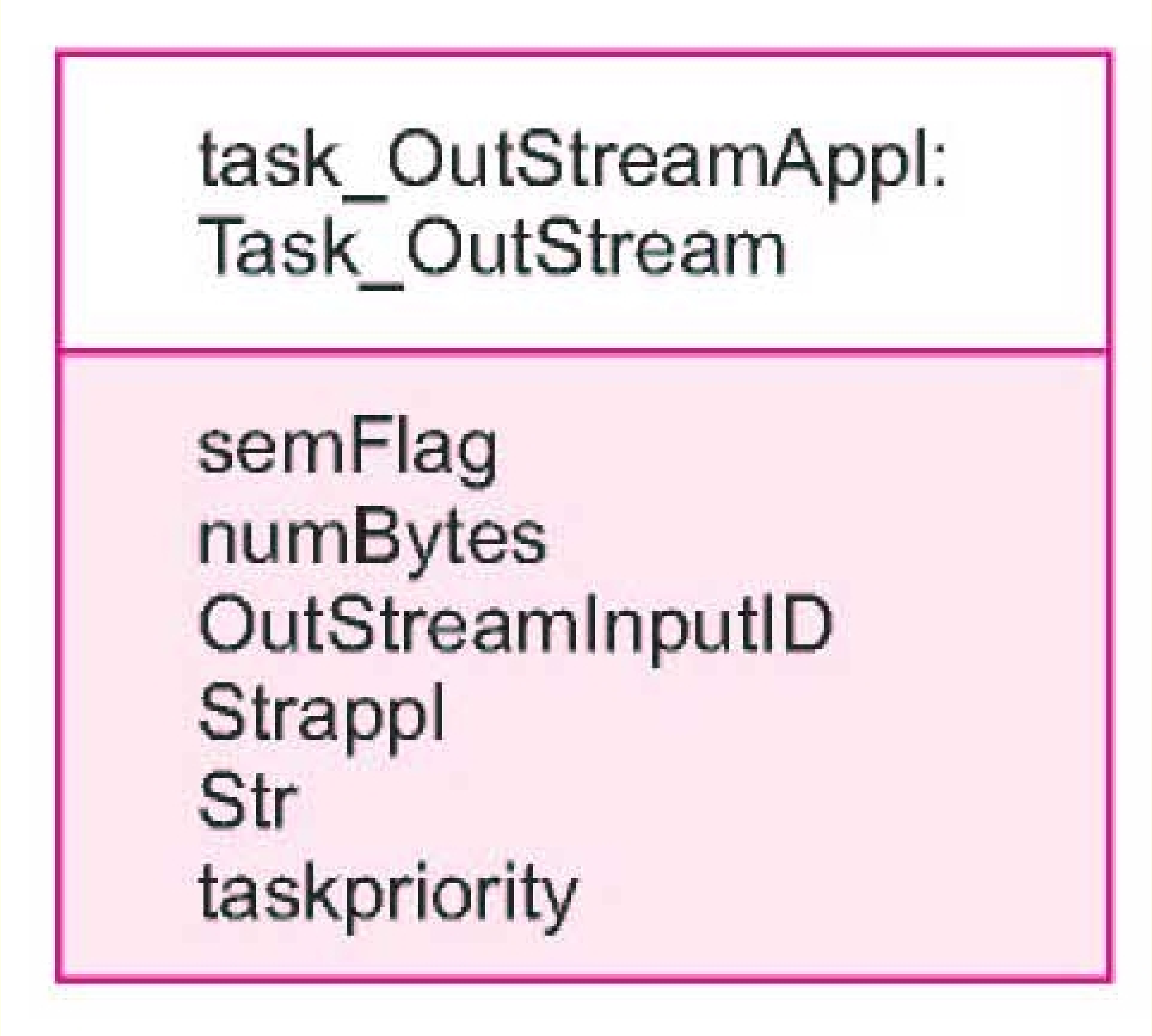
semFlag,
SemMKey,:
OSEvent
numBytes,
taskId,
taskPriority :
static Integer
QutStreamInputID,
applStr: String

msgQSend ( )
taskdelay ( ),
taskresume ( )

task objects

- The task objects are instances of the classes (i) Task_StrCheck, (ii)Task_OutStream, (iii) Task_TCPSegment or Task_UDPDatagram, (iv) Task_IPPktStream (v) Task_NetworkDrv.

Object task_OutStreamAppl

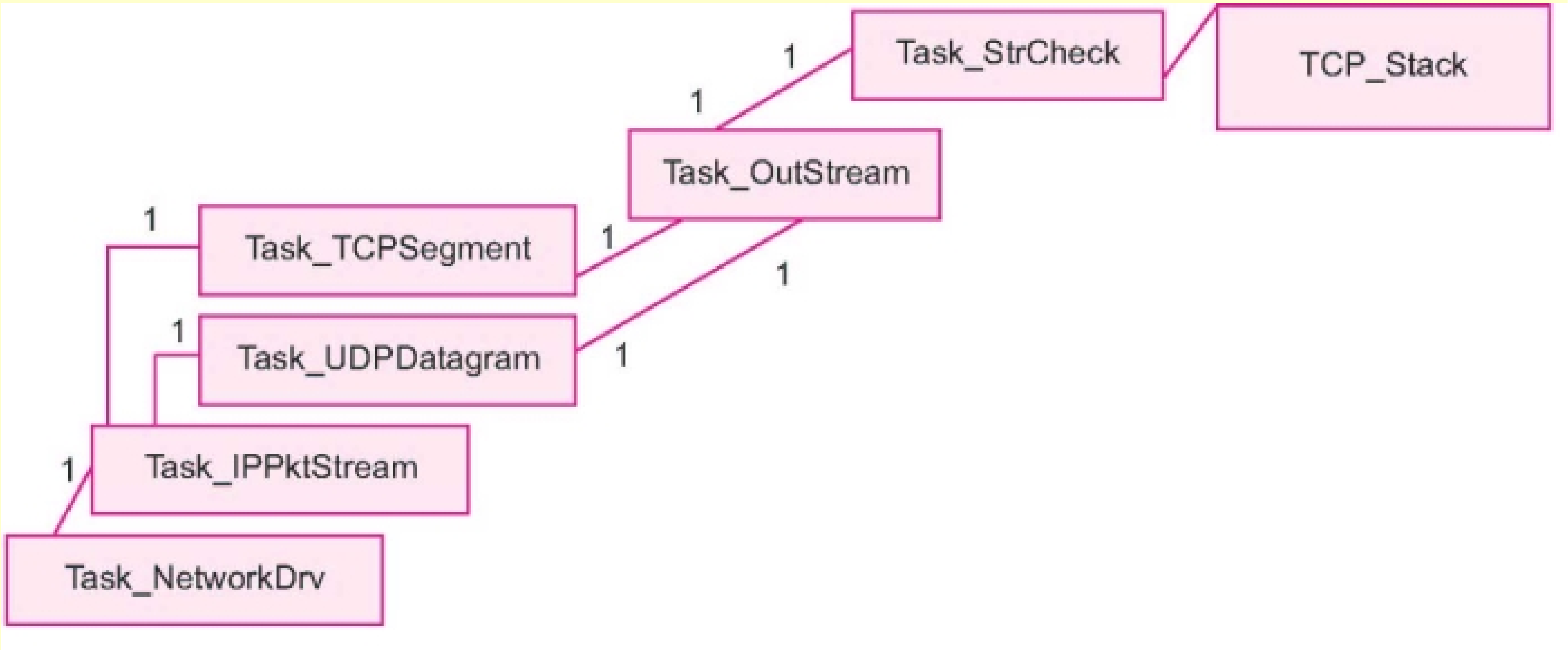


task_OutStreamAppl:
Task_OutStream

semFlag
numBytes
OutStreamInputID
Strappl
Str
taskpriority

4. Class diagrams

Class diagram for sending TCP/IP stack



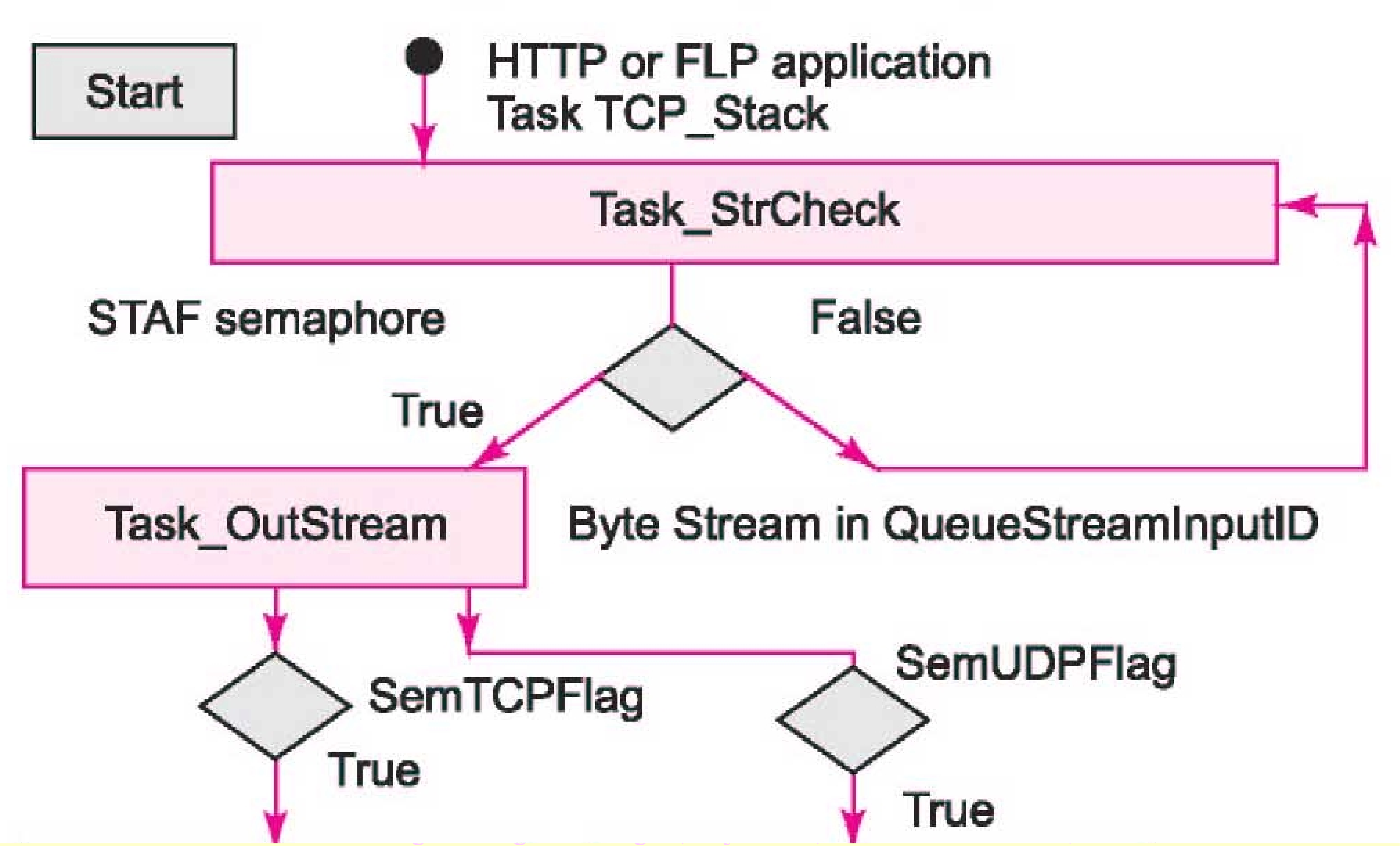
5. Hardware architecture

Hardware

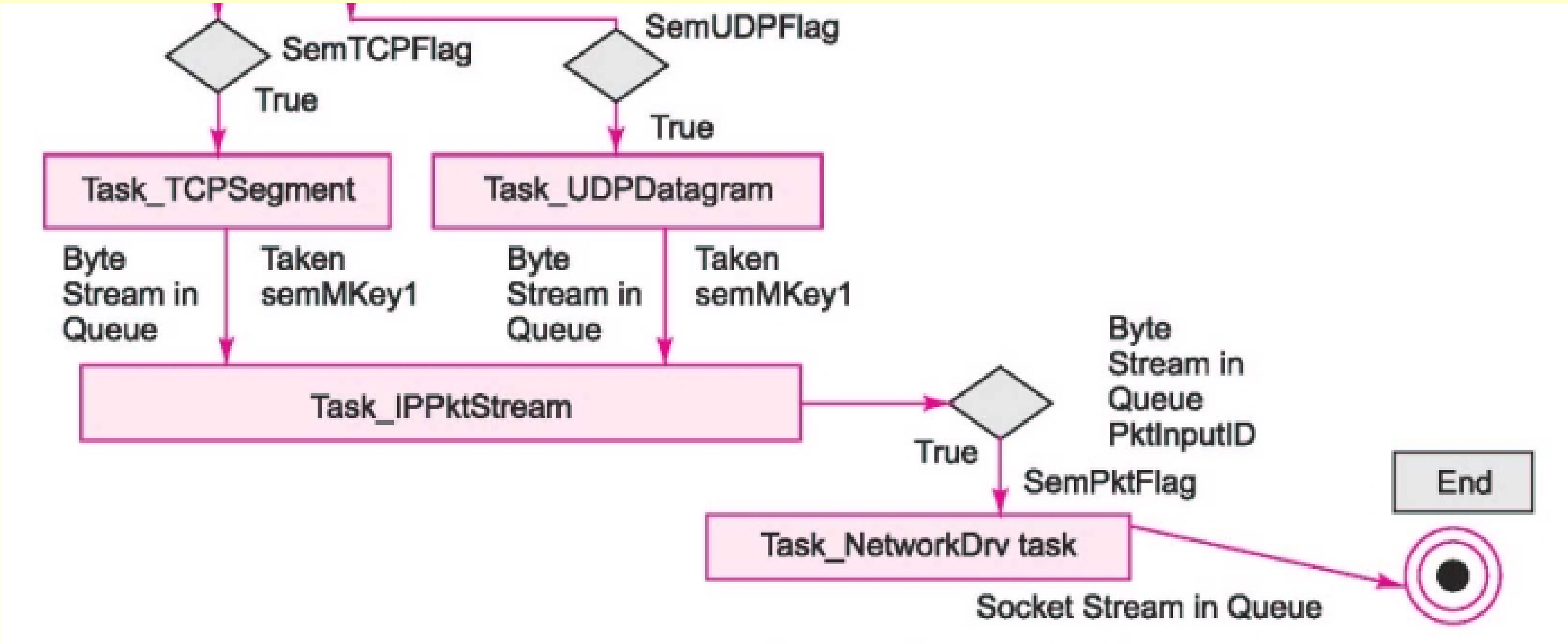
- TCP stack will run on same hardware as for the system which is networked with other system.
- Only additional hardware needed is memory for codes, data and queue for data streams, packets and sockets.
- A single TCP packet or UDP datagram is of maximum 2^{16} bytes.
- 2 MB (512 × 216 bytes) RAM can be taken as additional memory requirement

6. Modeling of State diagram for of TCP/IP stack tasks

State diagram for synchronization of TCP/IP stack tasks- Part 1

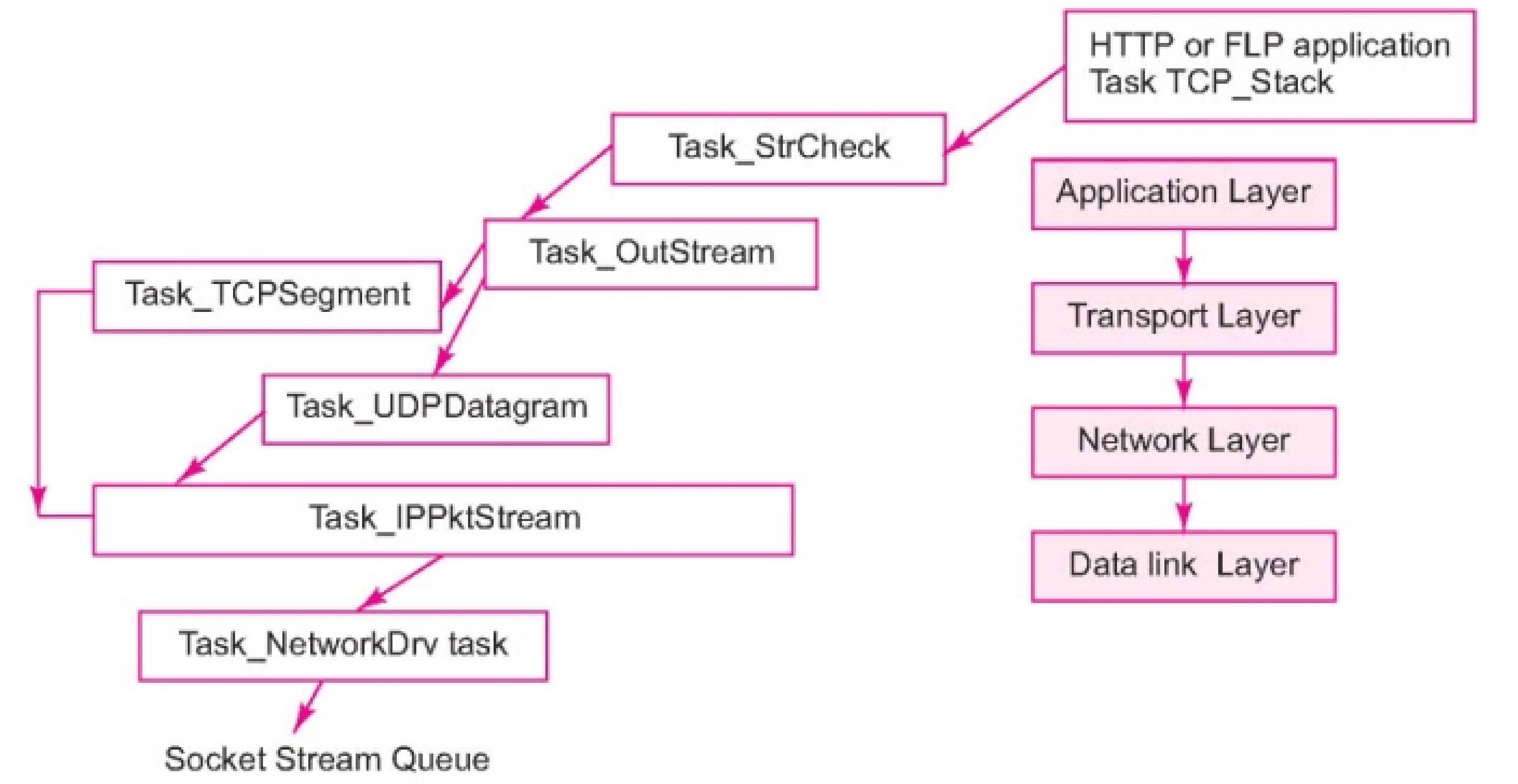


State diagram for synchronization of TCP/IP stack tasks- Part 2



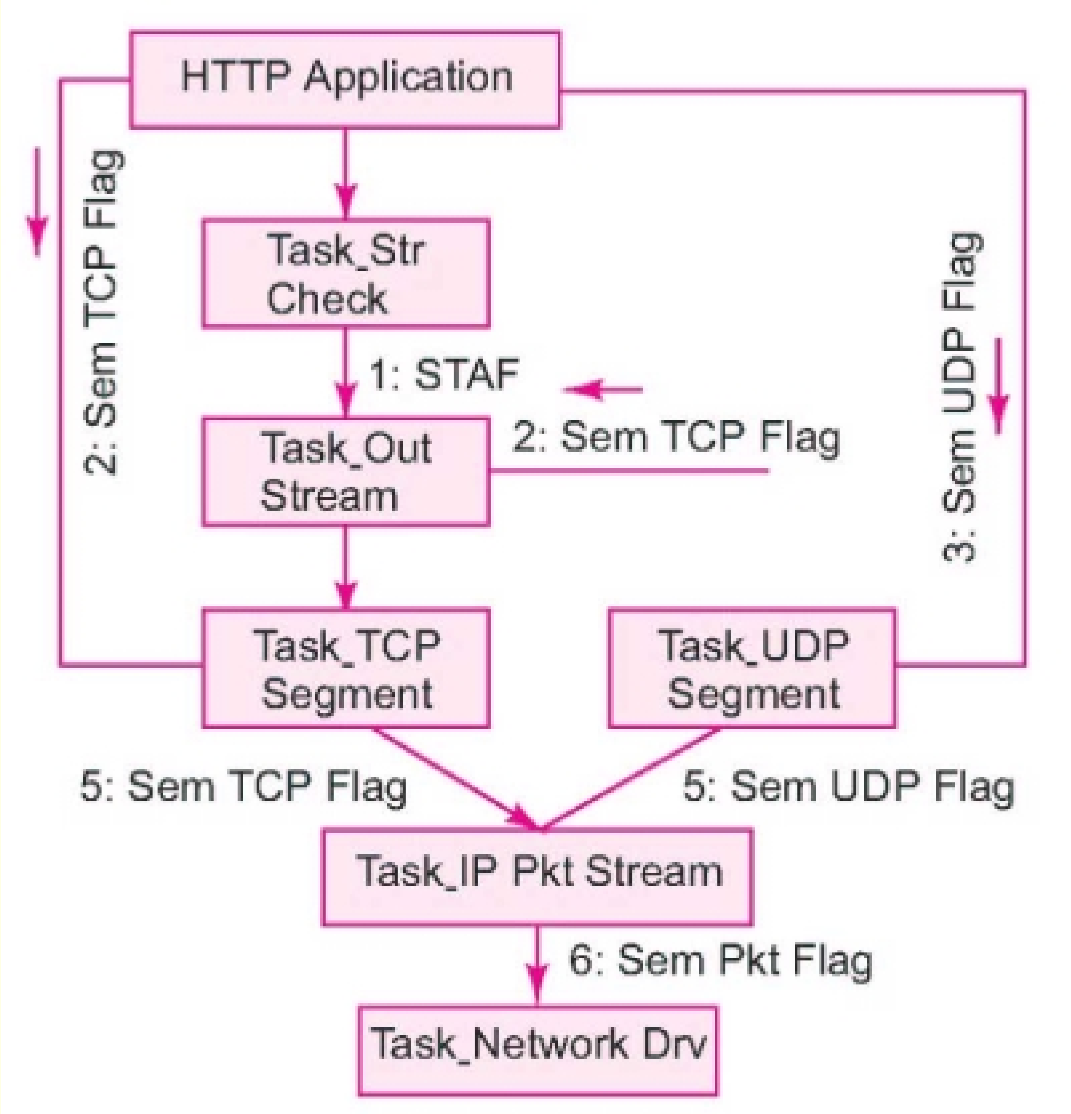
7. Software architecture

Software architecture TCP/IP Stack



8. Multiple tasks and their synchronization model

Multiple tasks and their synchronization model using semaphores and mailbox messages



9. Tasks and their priority, action and IPCs

Application Layer Task_StrCheck

- Priority─ 120
- Action─ Get a string from the application
- IPC pending: ─
- IPC posted: SemFlag1

Application Layer Task_OutStream

- Priority ─ 121
- Action─ Read string and put bytes into an output stream
- IPC pending: SemFlag1, SemFlag2
- IPC posted: *QStreamInputID*

Transmission Control (Transport) Task_TCPSegment

- Priority ─ 122
- Action─ Insert TCP header to the stream
- IPC pending: SemTCPFlag, SemMKey1, QStreamInputID
- IPC posted: QStreamInputID and SemMKey1

Transmission Control (Transport) Task_UDPDatagram

- Priority ─ 122
- Action─ Insert UDP header to the stream sent as a datagram
- IPC pending: SemUDPFlag, SemMKey1, QStreamInputID
- IPC posted: QStreamInputID, SemM-Key1

internet (network) layer Task_IPPktStream

- Priority ─ 124
- Action─ Form the packets of maximum 2^{16} bytes
- IPC pending: SemMKey1, QStreamInputID
- IPC posted: SemMKey1, SemPktFlag, QPktInputID

Task_NetworkDrv

- Priority — 124
- Action— Send the packets as the frames
- IPC pending: SemPktFlag, QPktInputID, SemMKey1
- IPC posted: SemFinishFlag, SemFlag2

10. Coding using VxWorks RTOS IPC functions

Coding using VxWorks RTOS

- Refer Example 11.2 in Section 11.3.4
- At each step the explanation for the set of statements given there.

Summary

We learnt

- TCP stack generation is modeled by class Task_TCP, which is an abstract class from which the extended class (es) is derived to create TCP or UDP packet to a socket.
- The task objects are instances of the classes (i) Task_StrCheck, (ii)Task_OutStream, (iii) Task_TCPSegment or Task_UDPDatagram, (iv) Task_IPPktStream (v) Task_NetworkDrv

We learnt

- VxWorks RTOS used for embedded system codes for driving a network card after generating the TCP/IP stack.
- Exemplary codes show a method of code designing for sending the byte streams on a network.

End of Lesson-3 of chapter 11 on

CASE STUDY OF CODING FOR SENDING APPLICATION LAYER BYTE STREAMS ON A TCP/IP NETWORK USING RTOS VxWorks