Introduction to Computer Networks and the Internet COSC 264

Network Protocols: Architectures and Basics

Dr. Andreas Willig

Dept. of Computer Science and Software Engineering
University of Canterbury, Christchurch

UoC, 2020

Outline

1. Protocol Layering
   - The Concept of Layering
   - The OSI Reference Model
   - The TCP/IP Reference Model
2. Elements of Service and Protocol Design
   - Service Primitives
   - A few Standard Protocol Mechanisms

About this Module

- We look at architectures for packet-switched networks
- Goals:
  - Understand protocol layering and two reference models
  - Understand concepts of services, protocols and their relationships
- This module is based on [6, Chap. 2], [4]
- Further references: [3], [2], [7], [1], [5]

Outline

Networking Software

- The Internet and POTS are among the most complex technical systems, they require vast amounts of software
- **Structuring principles** organize networking software to achieve:
  - Modularity and software re-use
  - Independence of network technologies (**Transparency**)
  - Separation of concerns
  - Correctness

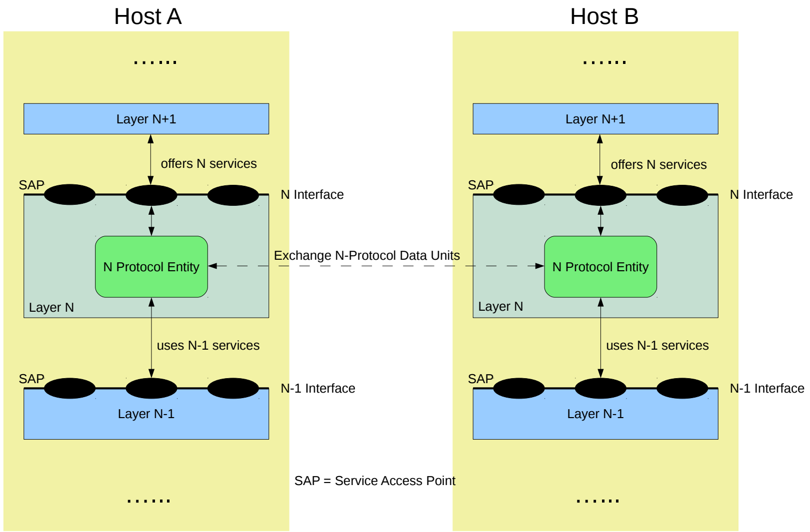
Layering

A key structuring principle for networking software is **layering**: the functionality is decomposed into a chain of layers so that layer N offers services (through an **interface**) to layer $N+1$ and itself is only allowed to use services offered by layer $N-1$.

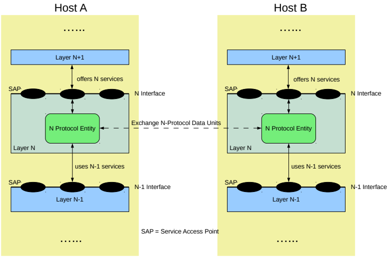
Outline

1. Protocol Layering
   - The Concept of Layering
   - The OSI Reference Model
   - The TCP/IP Reference Model
2. Elements of Service and Protocol Design

Layering Concepts

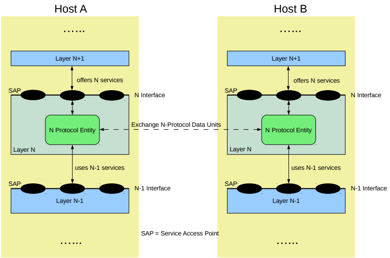


Layering Concepts (2)



- A layer N offers an N-service interface – Example: the socket API
- The next higher layer $N+1$ is only allowed to use the N-interface, but not any of the lower interfaces (e.g. the $N-1$ interface) – this applies to all layers!
- The N-interface offers services at **service access points** (SAP)
- The N-interface can offer several SAPs, this allows to multiplex between different layer $N+1$ protocols or different layer $N+1$ "connections" or "sessions"
- Example: sockets and their associated port numbers are SAP's, different applications use different port numbers

Layering Concepts (3)



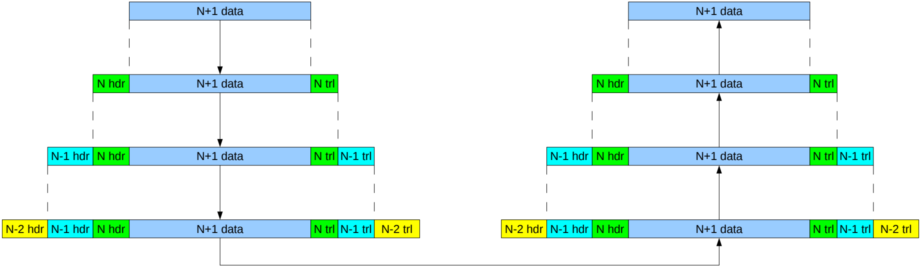
- The layer N-service is implemented through an N-protocol
- The N-protocol makes direct use of $N-1$ services
- The N-protocol makes no assumption whatsoever on what is on layer $N+1$
- It exchanges protocol data units (PDUs) with a peer N-protocol entity – it constructs these PDUs itself and hands them over to its local $N-1$-layer to deliver them to peer N-protocol entity (which in turn receives it from its local $N-1$ layer)
- “PDU” is a more fancy word for packet

General Layout of a Layer *N* PDU/Packet

N-protocol header	N+1 data = N-SDU	N trailer

- The *N*-PDU is constructed by the *N*-protocol entity
- It carries the data handed over by layer $N+1$ for transmission, also referred to as **user data**, **payload** or *N*-SDU (**service data unit**)
- The sending *N*-protocol entity adds an *N*-protocol **header** which carries control information (e.g. sequence numbers, addresses, flags) important for the *N*-protocol but not the receiving $N+1$ layer or $N-1$ layer
- It might furthermore add an *N*-protocol **trailer** (usually a checksum)
- The receiving *N*-protocol entity removes the *N* header and trailer and hands over the $N+1$ data to its local layer $N+1$ entity

Layered PDU Processing



- An N-PDU is treated as payload / user data by the $N-1$ layer
- Each layer adds own header and trailer before handing down to lower layer
- Receiving layer removes its header / trailer before handing payload to upper layer

About Interfaces

- Interfaces specify a **service** that a certain layer offers
- Example:
  - The socket interface on a stream socket offers reliable, in-sequence and byte-oriented data transfer through an interface resembling a file system interface
  - The TCP protocol implements this service (and in turn makes use of the “best effort” service provided by the IP protocol)
  - Applications just use the socket interface and are not concerned with the operation of the TCP protocol

Important Point

Standardized interfaces allow higher layers to ignore the operation and properties of lower layers

Outline

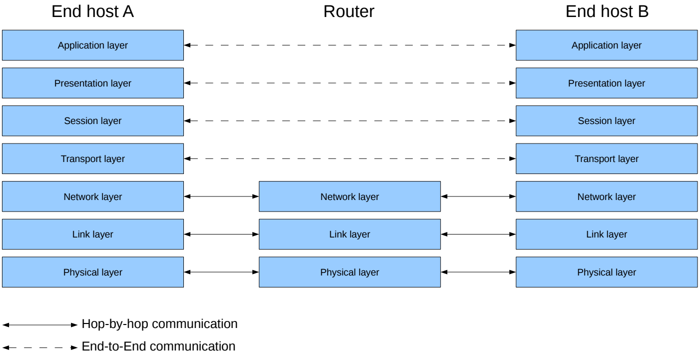
1. Protocol Layering
   - The Concept of Layering
   - The OSI Reference Model
   - The TCP/IP Reference Model
2. Elements of Service and Protocol Design

The OSI Seven Layer Model

- Layer 7: Application layer
- Layer 6: Presentation layer
- Layer 5: Session layer
- Layer 4: Transport layer
- Layer 3: Network layer
- Layer 2: Link layer
- Layer 1: Physical layer

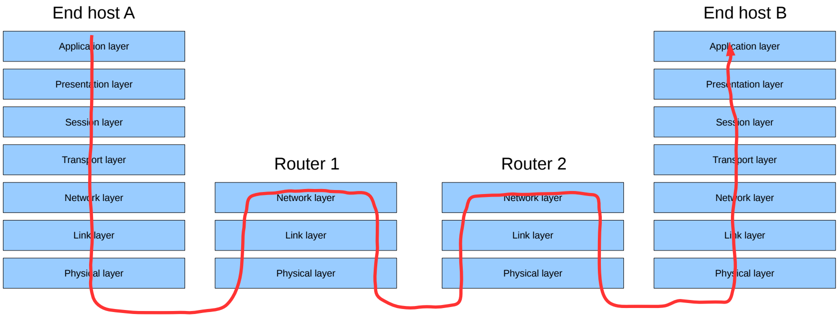
- OSI = Open Systems Interconnection
- Set of standards and protocols created by ISO
- See [7]
- The model was not commercially successful, but helped greatly to clarify networking architectures and concepts, and in this sense is foundational to networking

The OSI Seven Layer Model – A Second View



- Lowest two layers have strictly “single-hop scope” and exchange PDUs only between physically connected hosts
- Network layer uses hop-by-hop communication to achieve end-to-end communication
- Upper four layers exchange PDUs between end hosts (perhaps over several intermediate nodes, called **routers**), they have strictly “end-to-end scope”
- This hints at a network architecture where end nodes are interconnected through routers
- Routers only work on the lowest three layers

The OSI Seven Layer Model – A Third View



- This shows the order of processing that a packet experiences along its path through a multi-hop network

OSI RM – Physical Layer

- Often referred to as “PHY”
- Concerned with transmission of digital data (e.g. bits, bytes) over a physical medium, using modulated waveforms / signals
- Often involves specification of:
  - Cable types (wired) or frequencies / bandwidth (wireless)
  - Connectors
  - Electrical specifications
  - Modulation / demodulation and signal specification
  - Carrier- or bit synchronization methods

UC UNIVERSITY OF CANTERBURY Te Whare Wānanga o Waitaha CHRISTCHURCH NEW ZEALAND

OSI RM – Link Layer

- Task: (reliable) transfer of messages over one physical link
- Link layer messages are often called **frames**
- Often involves specification of:
  - Framing:
    - delineation of frame start and end
    - choice of frame size
    - frame format
  - Error control (e.g. coding- or retransmission-based)
    - Error-correction coding is also often regarded as a PHY functionality
  - Medium access control
    - distributes right to send on shared channel to several participants
    - often considered as a separate "sub-layer" of link layer
  - Flow control
    - Avoid overwhelming a slow receiver with too much data

OSI RM – Network Layer

- Concerned with:
  - Providing a link technology-independent abstraction of entire network to higher layers
  - Addressing and routing
  - End-to-end delivery of messages
- Network- and higher-layer messages are called **packets**
- Often involves specification of:
  - Addressing formats
  - Exchange of routing information and route computation
  - Depending on technology: establishment, maintenance and teardown of connections

OSI RM – Transport Layer

- Concerned with:
  - (reliable, in-sequence, transparent) end-to-end data transfer
  - programming abstractions (interface) to higher layers
- Often involves specification of:
  - Error-control procedures (**Question**: why again?)
  - Flow control procedures
  - Congestion control procedures
    - Protect network against overloading
    - Can also be considered a network-layer issue

OSI RM – Session and Representation Layer

- Session layer:
  - Concerned with establishing communication sessions between applications
  - A session can involve several transport layer connections in parallel or sequentially
  - A session might control the way in which two partners interact, for example enforce that partners speak alternatingly
- Representation layer:
  - Translates between different representations of data types used on different end hosts
  - Example: host A uses low-endian integers, host B big-endian

OSI RM – Application Layer

- Application support functions useful for many applications
- Examples:
  - File transfer services
  - Directory services
  - Transaction processing support (e.g. two-phase commit)

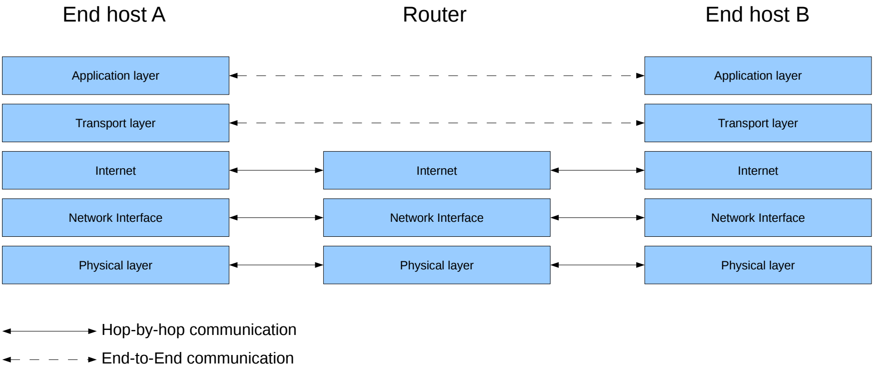
Outline

The TCP/IP Reference Model

Layer 5: Application
Layer 4: Transport layer
Layer 3: Internet
Layer 2: Network Interface
Layer 1: Physical layer

- This model is used in the Internet
- This is broadly equivalent to the OSI RM with the session and presentation layer being removed
- The Internet follows the so-called **end-to-end principle**: Layers 3 and below are kept simple, most complexity resides in transport layer
- Or in other words: keep routers simple!

The TCP/IP Reference Model – A Second View



- This reference model also uses a network architecture where end nodes (called **hosts**) are interconnected through routers!

The Application Layer

- Consists of applications using services of transport layer
- Accesses transport layer through **socket interface**
- There are well-known application-layer protocols, e.g.:
  - SMTP (email)
  - HTTP (web)
  - FTP (file transfer)
  - RTP (real-time video and audio)

The Transport Layer

- Provides end-to-end communications to applications
- Offers its services through **socket interface**
- Standard transport layer protocols:
    - TCP: reliable, in-sequence byte-stream transfer
    - UDP: unreliable, un-ordered message transfer

  but other protocols can be used as well (e.g. SCTP)
- SAPs are called **ports**, used for **application multiplexing**
    - Several applications / processes can use transport service
    - A port is bound to one application
    - Ports are identified by numbers
    - The PDUs generated by TCP / UDP are called **segments**
    - TCP / UDP segments include the port number
    - TCP / UDP receiver delivers incoming segment to the application denoted by the port number (through an associated socket)

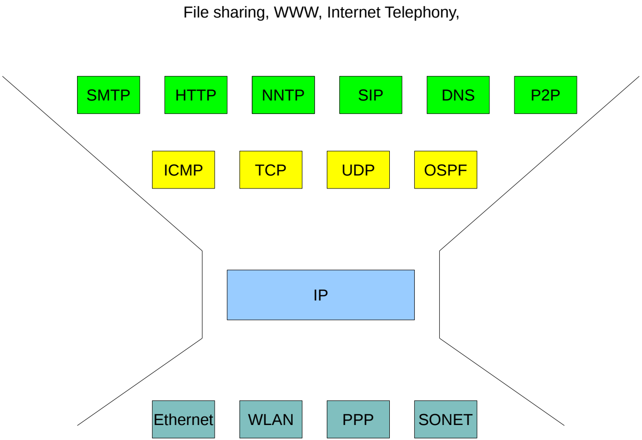
The Transport Layer (2)

- TCP has mechanisms for:
  - Error control (retransmission-based) and in-order delivery
  - Flow control
  - Congestion control
- UDP has none of these features
- For transmission, TCP and UDP hand over segments to the Internet layer
- For reception, TCP and UDP get incoming segments from the Internet layer

The Internet Layer

- This is a key part of the TCP/IP reference model
- Uses IP (Internet Protocol), its PDUs are called **datagrams**
- All higher-layer segments are encapsulated in datagrams
- The IP protocol:
  - specifies an addressing scheme (IP addresses)
  - provides end-to-end delivery of datagrams (forwarding)
  - does **not** specify how routing is done, left to dedicated protocols
  - has no mechanisms for error-, flow- and congestion control
  - can send IP datagrams over any network interface

The Internet Layer (2)



- “Everything over IP, IP over everything”

The Physical and Network Interface Layer

- The physical layer is similar to the PHY in the OSI RM
- The Network Interface Layer:
    - Similar to the link layer in the OSI RM
    - Accepts IP datagrams and delivers them over physical link
    - Receives IP datagrams and delivers them to local IP layer
    - Includes medium access control, framing, address resolution
    - Might also include link-layer error- and flow control

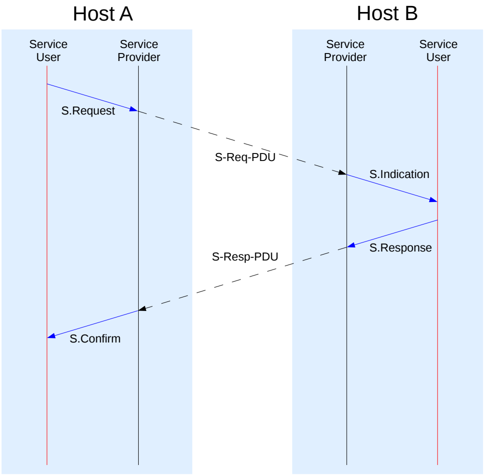
Outline

Outline

Service Providers and Service Users

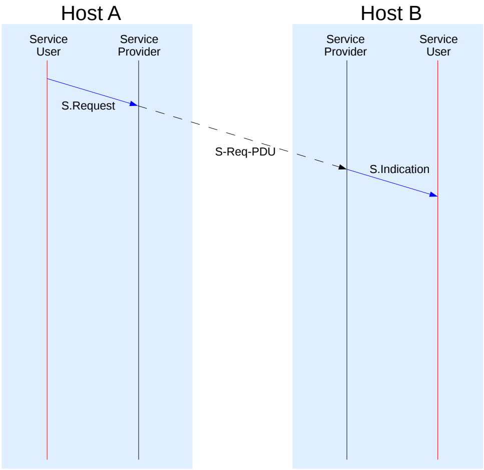
- An N-protocol implements an N-service
- Stated differently: the N-protocol is the N-service provider!
- An N+1-protocol (or the application) is the N-service user
- **Guiding question**: How do service provider and user interact?
- Service provider and user:
  - talk to each other through **service primitives**
  - have to obey rules in the usage of services
    - Example: before a telephone can use any “send voice data” service, it must have used “connection setup” service before
    - Example: before you can read from a file, you have to open it
- Standard service primitives for a service `S`:
  - `S.request`
  - `S.indication`
  - `S.response`
  - `S.confirmation`

Confirmed Service



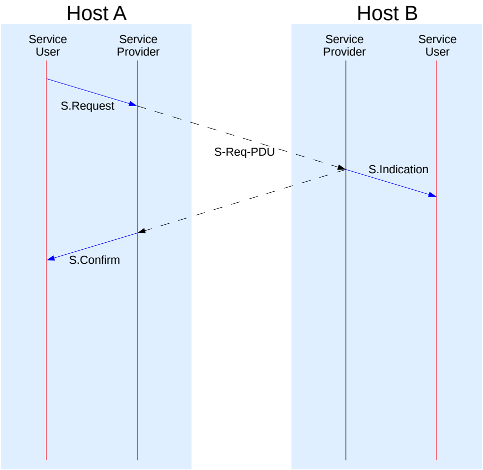
- Service user at A issues an S.request service primitive, possibly carrying user data
- The service provider for S (a protocol) generates one or more PDUs and sends them to host B
- Service user at B is informed about A's service request through an S.indication primitive
- Service user at B prepares response (possibly with data), gives it to local service provider through S.response
- B's response is made known to A's service user through S.confirm primitive
- Key point: response comes from B's service user!
- Do you know an example?

Unconfirmed Service



- Service user at A issues an S.request primitive
- Service provider for S generates one or more PDUs and sends them to host B
- Service user at B is informed through an S.indication primitive
- Service user at A has no clue whether service request reached B
- Do you know an example?

Confirmed Delivery Service



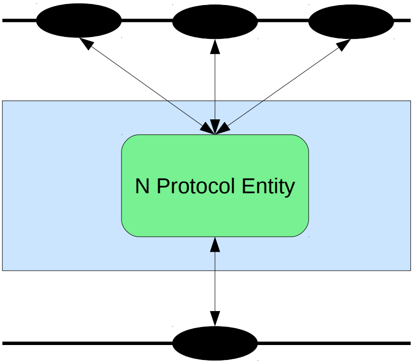
- Roughly similar to confirmed service
- Key difference: it is B's service provider generating a response, not B's service user!
- Thus, A's service user has no information about the behaviour of B's service user
- Do you know an example?

Outline

1. Protocol Layering

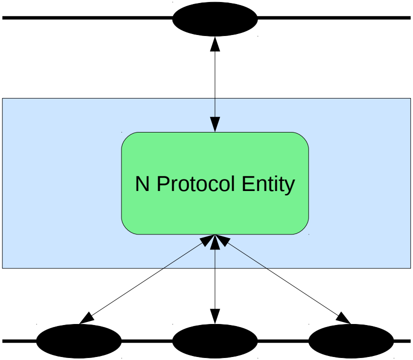
2. Elements of Service and Protocol Design
   - Service Primitives
   - A few Standard Protocol Mechanisms

Multiplexing



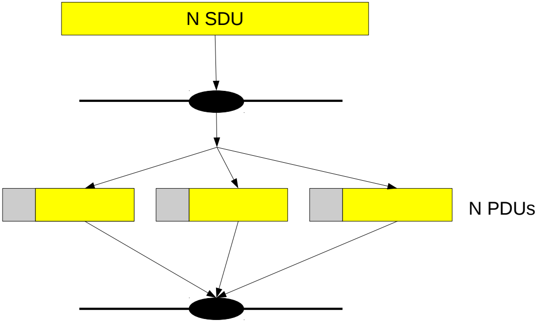
- Multiplexing allows to transmit data from several *N* SAPs over a single $N-1$ SAP
- When several *N* SAPs are used in parallel, the *N* protocol entity needs to make scheduling decisions to decide which *N* SAP to serve next
- Sending *N* entity needs to include an SAP identifier into the *N* PDU to allow receiver entity to deliver an incoming *N*-PDU to the right SAP
- Example: TCP supports several SAPs through port numbers, port numbers are part of TCP header

Splitting



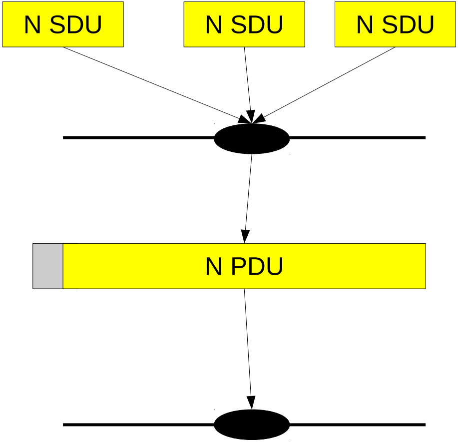
- An N-entity can transmit data received from higher layers via N-SAP over several $N-1$ SAPs
- Allows transmission of data over several channels to increase throughput and / or reliability through parallel transmission
- N-entity needs to make scheduling decisions on which $N-1$ SAP(s) to use for a given PDU
- Additional mechanisms for sequencing might become necessary

Fragmentation and Reassembly



- PDUs often have a limited size – on the lower layers this is usually for physical reasons
- To make PDU sizes transparent to higher layers, an *N*-layer can accept large *N*-SDUs and partition the data into several *N*-PDUs (**fragments**), each having own header, and transmit them separately
- Fragments must be numbered to allow receiver re-assembly
- **Question**: How should the receiver deal with losses of fragments?
- Disadvantage: higher overhead

Blocking and Deblocking



- Sometimes higher layers produce very small *N*-SDUs
- Instead of putting each *N*-SDU into separate *N*-PDU, transmitter waits until several *N*-SDUs are present (**blocking**) and puts them into one *N*-PDU to save overhead
- Receiver entity decomposes received *N*-PDU (**deblocking**) and delivers several *N*-SDUs to higher layers, this requires markers in the *N*-PDU separating the *N*-SDUs
- **Question**: when should sender stop collecting *N*-SDUs and send an *N*-PDU?

Sequence Numbers

- An *N*-entity can maintain a sequence number
- For each newly constructed PDU the sequence number is written into the *N*-PDU header, afterwards the sequence number is incremented
- Sequence numbers allow the receiver to:
  - Detect duplicate PDUs (and drop them)
  - Detect lost PDUs (possibly requesting retransmission from sender)
  - Put *N*-PDUs back in the right order when network reordered them
- Implementation issues:
  - Sequence number space is finite, wrapovers need to be handled
  - Choice of initial sequence number

[1] Mung Chiang, Steven H. Low, A. Robert Calderbank, and John C. Doyle.
Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures.
Proceedings of the IEEE, 95(1):255–312, January 2007.

[2] Douglas E. Comer.
Internetworking with TCP/IP – Principles, Protocols and Architecture, volume 1.
Prentice Hall, Upper Saddle River, New Jersey, fifth edition, 2006.

[3] John Day.
Patterns in Network Architecture – A Return to Fundamentals.
Prentice Hall, Upper Saddle River, New Jersey, 2008.

[4] Gerard J. Holzmann.
Design and Validation of Computer Protocols.
Prentice Hall, Englewood Cliffs, 1992.

[5] Vikas Kawadia and P. R. Kumar.
A Cautionary Perspective on Cross-Layer Design.
IEEE Wireless Communications, 12(1):3–11, February 2005.

[6] William Stallings.
Data and Computer Communications.
Prentice Hall, Englewood Cliffs, New Jersey, fourth edition, 2006.

[7] Hubert Zimmermann.
OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection.
IEEE Transactions on Communications, 28(4):425–432, April 1980.

UC UNIVERSITY OF CANTERBURY Te Whare Wānanga o Waitaha CHRISTCHURCH NEW ZEALAND