Session 8A

RPC, RTC and RTCP

Mouli Sankaran

Session 8A: Focus

- Remote Procedure Call (RPC)
  - Mechanisms
  - Protocols
  - Use Cases
  - Quiz 1 to 2
- Multimedia Streams
  - Real-Time Protocol (RTP)
  - Real-time Transport Control Protocol(RTCP)
  - Quiz 1 to 3

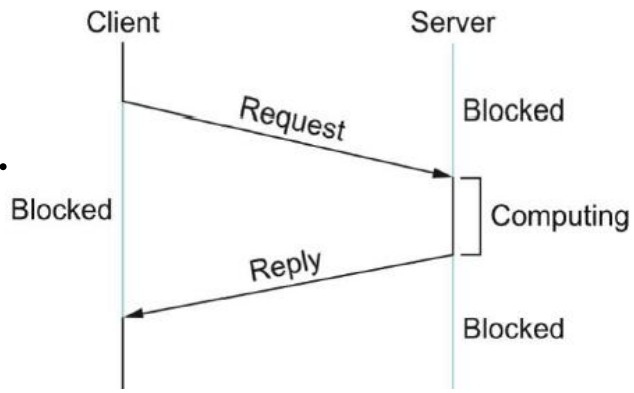
Course page where the course materials will be posted as the course progresses:

Remote Procedure Call (RPC)

RPC: Remote Procedure Call

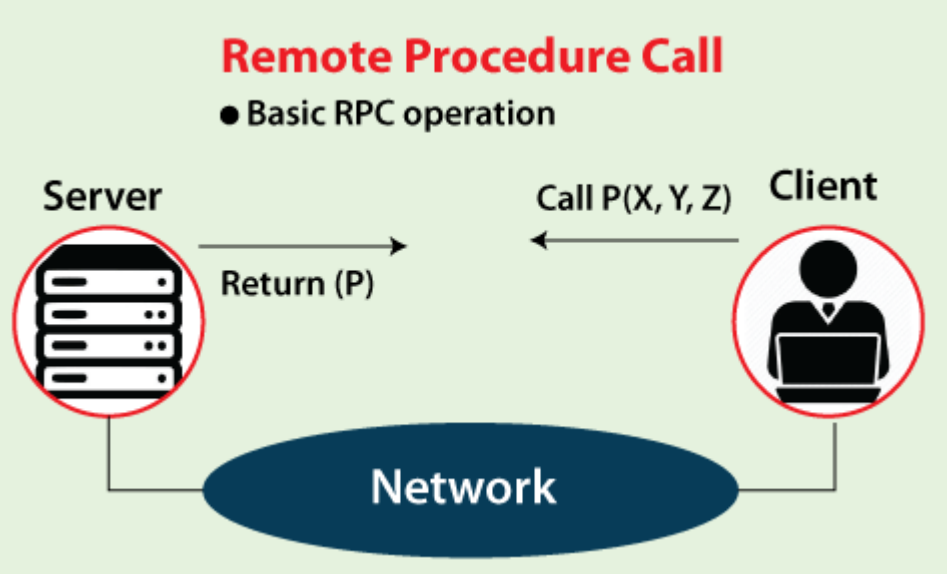
- **RPC** follows **request/reply message transaction**.

1. A client sends a request message to a server.
2. The server responds with a reply message.
3. The client blocking (suspending execution) to wait for the reply.



- **RPC** is not a protocol; it is a mechanism for structuring **distributed systems**.
- Here, an application program makes a call into a procedure without regard for whether it is local or remote and blocks until the call returns.
- When the procedures being called are actually, methods of remote objects in an object-oriented language, RPC is known as **Remote Method Invocation (RMI).**

RPC: Problems



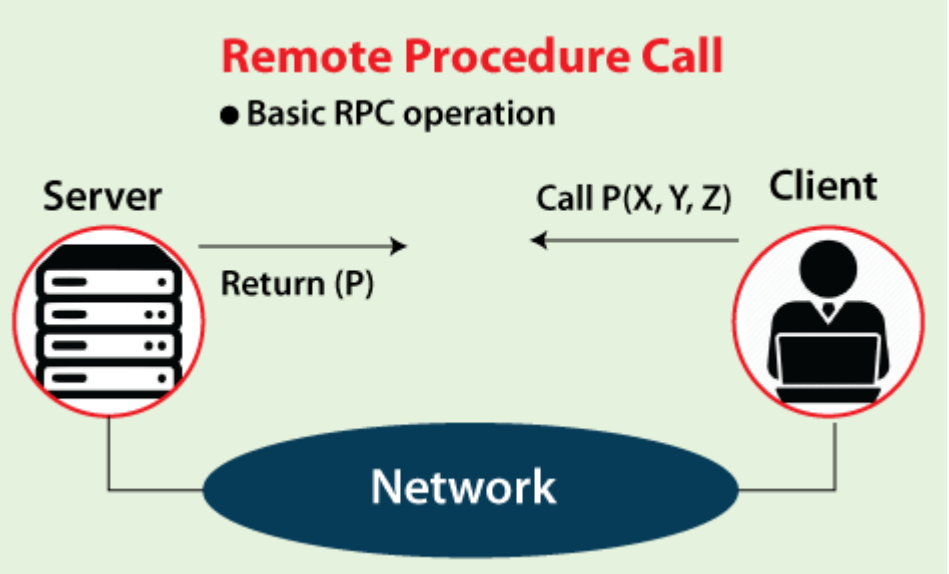
- Calling a procedure to be executed over a network is lot more different from calling it within a computer.
- The **two main problems** are:

1. Network can limit message sizes and also it has a tendency to lose or reorder messages sent over it.
2. The computers on which the calling and called processes run may have significantly different architectures and data representation formats, etc.

RPC: Major Components

- A complete **RPC mechanism** involves **two major components**

1. A protocol to manage the messages sent between the client and the server to deal with the potentially undesirable properties of the underlying network.

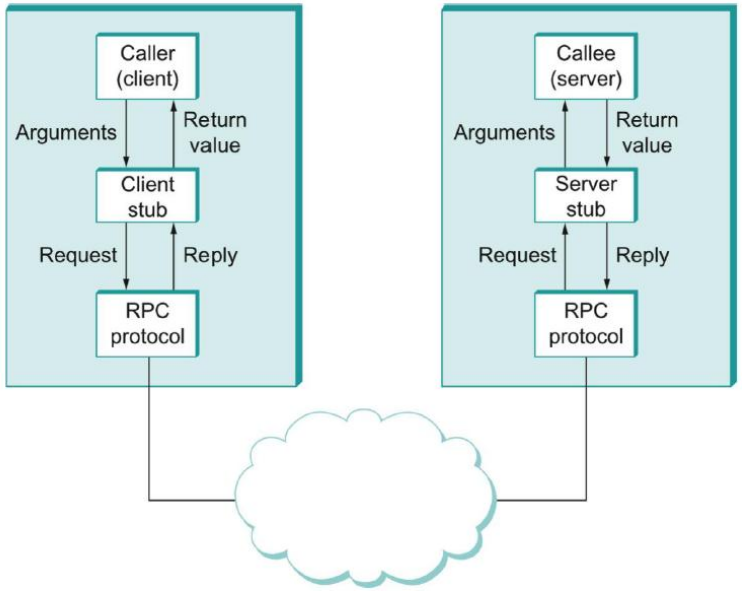


2. Programming language and compiler that support packaging **arguments** to the method into a request message on the client machine and then translates the message back into the arguments on the server machine before running.
3. Likewise, it handles the **return value** from the method on the client side (this piece of the RPC mechanism is usually called a **stub compiler**).

RPC Mechanism

RPC Mechanism

- The client calls a local stub for the procedure, passing it the arguments required by the procedure.
- This stub hides the fact that the procedure is remote by translating the arguments into a request message and then invoking an RPC protocol to send the request message to the server machine.

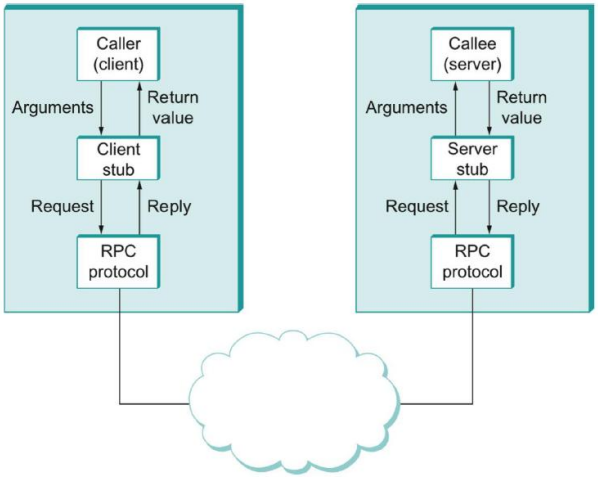


- At the server, the RPC protocol delivers the request message to the server stub, which translates it into the arguments to the procedure and then calls the local procedure.
- After the server procedure completes; it returns in a reply message that it hands off to the RPC protocol for transmission back to the client.
- The RPC protocol on the client passes this message up to the client stub, which translates it into a return value that it returns to the client program.

RPC Protocol

RPC Protocol

- RPC protocol, sometimes referred to as a **request/reply protocol**, that transmits messages between client and server.
- The term RPC refers to a type of protocol rather than a specific standard like TCP, so specific RPC protocols vary in the functions they perform.



- **Two functions** that must be performed by any **RPC protocol** are:
  - Provide a **name space** for uniquely identifying the procedure to be called
  - **Match** each reply message to the corresponding request message
- Especially in systems with multiple outstanding requests, concurrent clients, asynchronous messaging.
- It is essential for correct sequencing of responses, retransmissions or retries and avoiding mismatches in multi-threaded or multi-client environments

RPC: Key Features

- **Transparency**: The remote function appears local to the caller.
- **Stubs**: Automatically generated code that handles network communication.
- **Protocol-agnostic**: Can run over TCP, UDP, HTTP, or other transports.
- **Synchronous or Asynchronous**: RPC can block until the result is received, or it can work non-blocking in async mode.
- **Stateless or Stateful**: Depending on the implementation.

RPC: Use cases - Info

1. Microservices Communication

- Frameworks like **gRPC** (Google RPC) use Protocol Buffers and HTTP/2 to enable fast, efficient service-to-service communication in cloud-native systems.

2. Distributed Systems

- RPC is the foundation for communication between nodes in distributed file systems, clustered databases, and cloud platforms.

3. Client-Server Applications

- Traditional client-server apps (especially in enterprise) use RPC for calling functions on backend systems from a UI or client.

4. OS-level RPC (ONC RPC)

- Used in **NFS (Network File System)**, **NIS**, and other UNIX services.

5. Web & Mobile Apps

- RPC-style APIs (like **JSON-RPC**, **XML-RPC**) offer simple ways to call backend procedures over HTTP.

6. Game Networking

- Multiplayer games use RPC-like calls to sync actions between clients and servers in real time.

RPC: Quiz 1 to 2

Quiz 1: RPC

- Which of the following is a potential drawback of using RPC in a networked environment?

A. Increased complexity in error handling due to network failures
B. Limited support for multiple programming languages
C. Inability to scale beyond a single network segment
D. Requirement for specialized hardware to handle remote calls

ANS: A

Quiz 2: RPC

- In the context of RPC, what is the role of a 'stub'?

A. It serves as a placeholder for unimplemented functions
B. It translates requests and responses between client and server
C. It manages memory allocation for remote procedures
D. It establishes the physical connection between client and server

ANS: B

Multimedia

Quiz 1: Examples of Multimedia

1. Streaming stored audio/video
2. Streaming live audio/video
3. Real-time interactive audio/video

Quiz: Identify the **type of examples** given below

Multimedia Steams over the Network

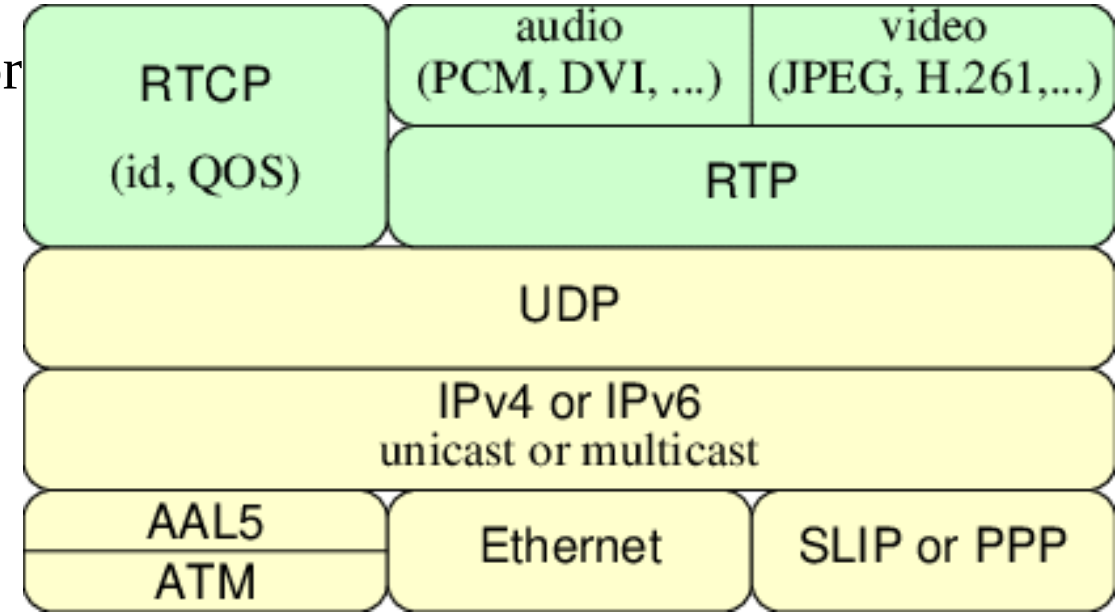
- **Multimedia streams** — like audio, video, and real-time media, have several unique characteristics
- **Timing is critical** — packets must arrive in time to play the media smoothly
- Delays (latency), jitter (variation in delay), or loss can directly affect playback quality.
- Unlike file downloads, late packets are often useless for real-time media
- It involves a constant stream of packets, not just a single request-response.
- Some packet loss is acceptable — better to skip a frame than to wait for retransmission
- Real-time streams prioritize timeliness over reliability.
- Protocols like RTP run over UDP (not TCP) to avoid retransmission delays.

Real-Time Transport Protocol (RTP)

RTP: Introduction

- **RTP** is a protocol used for delivering real-time data,
  - Audio, Video, Sensor data, Interactive media (VoIP, video conferencing)
- It was standardized by the IETF in RFC 3550 and is often used alongside **RTCP** (Real-time Control Protocol)
- **RTP** itself **does not provide reliability**. It's intentionally lightweight to support **low-latency delivery**.

- It works only over **UDP** for a low-latency delivery
- RTP is designed primarily for speed, not reliability.
- RTP uses the demultiplexing function support of UDP using its port numbers.



RTP: Explained

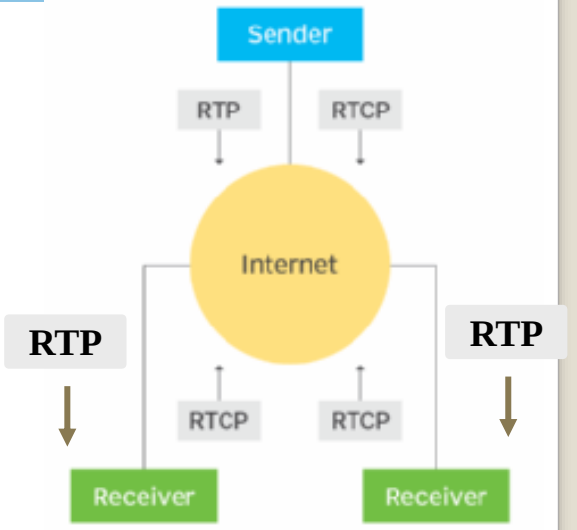
- Designed to handle time-sensitive data like audio, video, and sensor streams.
- It provides continuous, time-aligned delivery.
- RTP includes a sequence number field to detect packet loss and to maintain correct order of playback
  - Receivers buffer the data before playing back to avoid **jitter**
- Each packet includes a **timestamp** to synchronize the media streams (both audio and video). Timestamps are based on **sampling clock**.
- RTP includes a **payload type** field to notify the receiver the codec used (e.g., G.711, Opus, H.264) for the media stream
- RTP supports both **unicast** and **multicast** delivery.
  - Multicast for conferences or broadcasts and unicast for one-to-one communication like VoIP

Jitter: Variation in the time taken by the data packets to travel across a network

Real-time Transport Control Protocol (RTCP)

RTCP: Explained

- **RTCP** is the control protocol that works hand-in-hand with RTP to monitor transmission quality and help synchronize media streams
- **RTP** handles sending the actual **media data** over the network
- **RTCP** handles **reporting, feedback, and synchronization**.



- The following RTCP packets are send periodically (every few seconds)

1. **Sender reports (SR)** from RTP senders used for stream synchronization and throughput monitoring
2. **Receiver reports (RR)** sent by the RTP receivers that includes packet loss, jitter, round-trip time, to help the sender to adapt sending rates
3. **Source description (SDES)** shares sender identity useful in the multi-party conference calls.
4. **Bye** indicates a source is leaving (e.g., user disconnects)

RTP and RTCP: Quiz 1 to 3

Quiz 1: RTP

- Which transport layer protocol is typically used by RTP?

ANS: B

A. TCP
B. UDP
C. IP
D. ICMP

Note:
RTP prefers UDP due to its low-latency, connectionless nature — essential for real-time communication.

Quiz 2: RTCP

- What does RTCP provide in an RTP-based session?

A. Encryption for media
B. Routing of RTP packets
C. Feedback on media delivery quality
D. Compression of RTP data

ANS: C

Quiz 3: RTCP types

- Which of the following RTCP packet types is used to report packet loss, jitter, and delay?

A. SR

B. RR

C. SDR

D. Bye

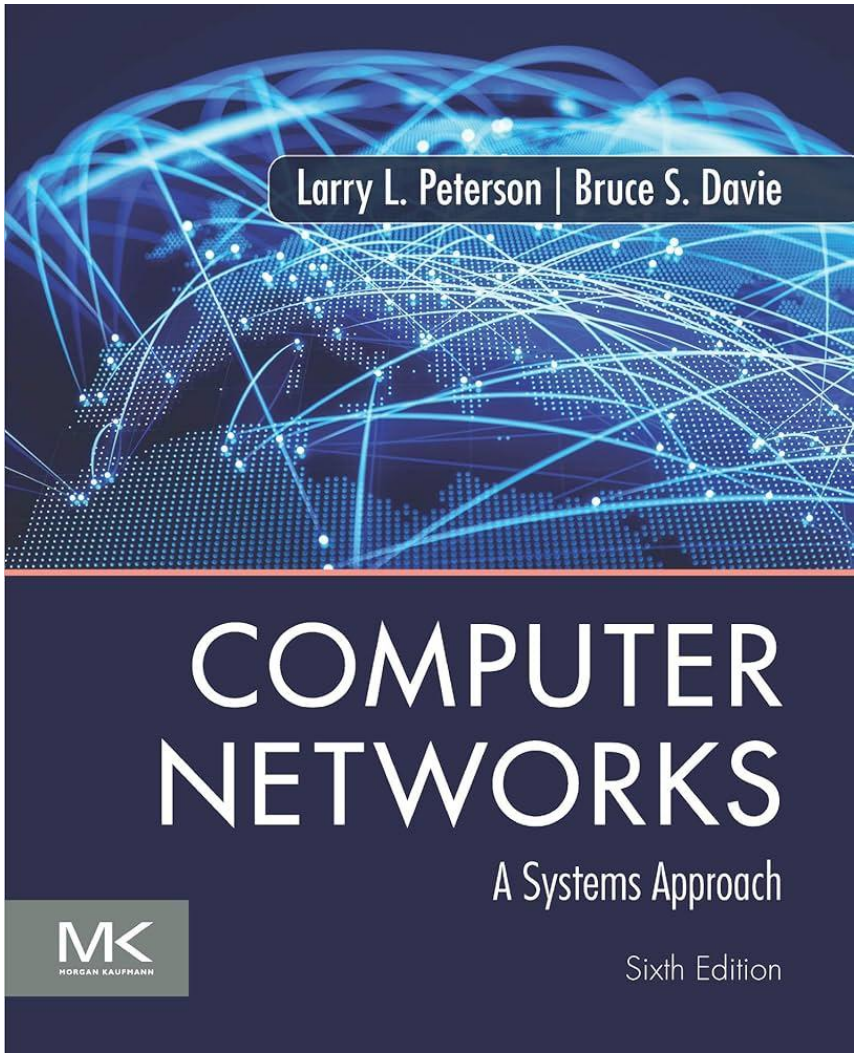
ANS: B

Session 8A: Summary

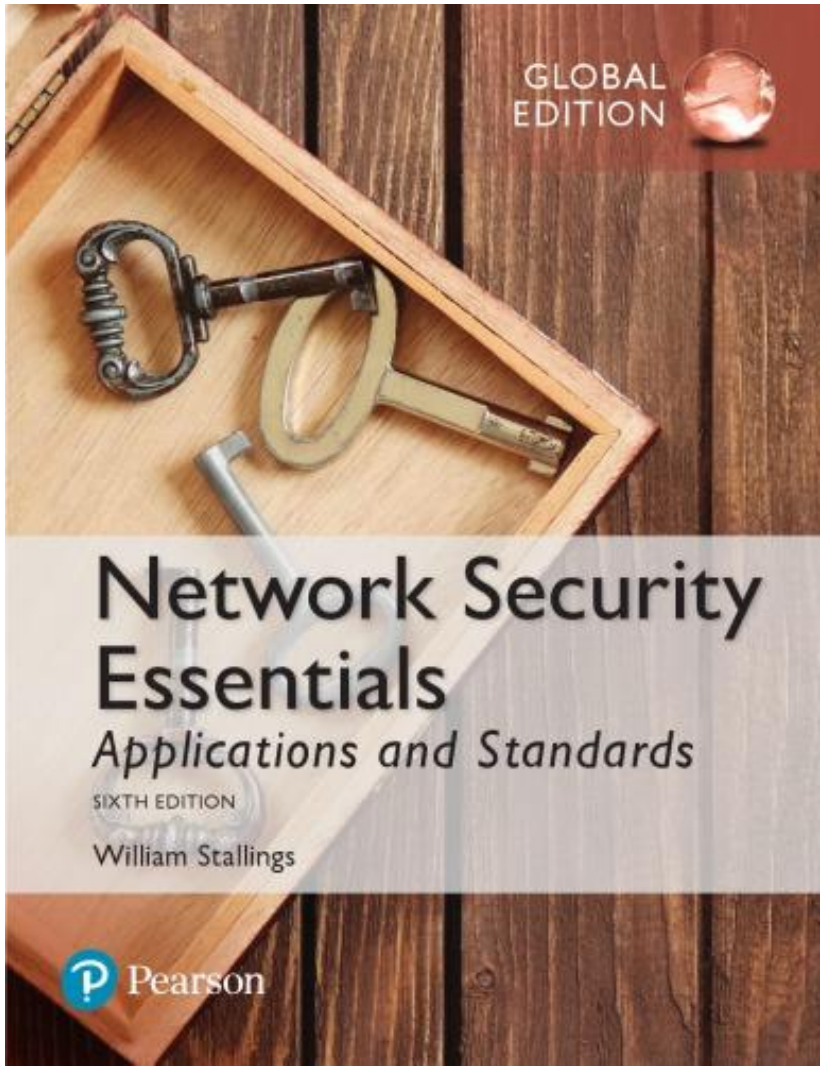
- Remote Procedure Call (RPC)
  - Mechanisms
  - Protocols
  - Use Cases
- Multimedia Streams
  - Real-Time Protocol (RTP)
  - Real-time Transport Control Protocol(RTCP)
  - Quiz 1 to 3

Textbooks

Textbook 1

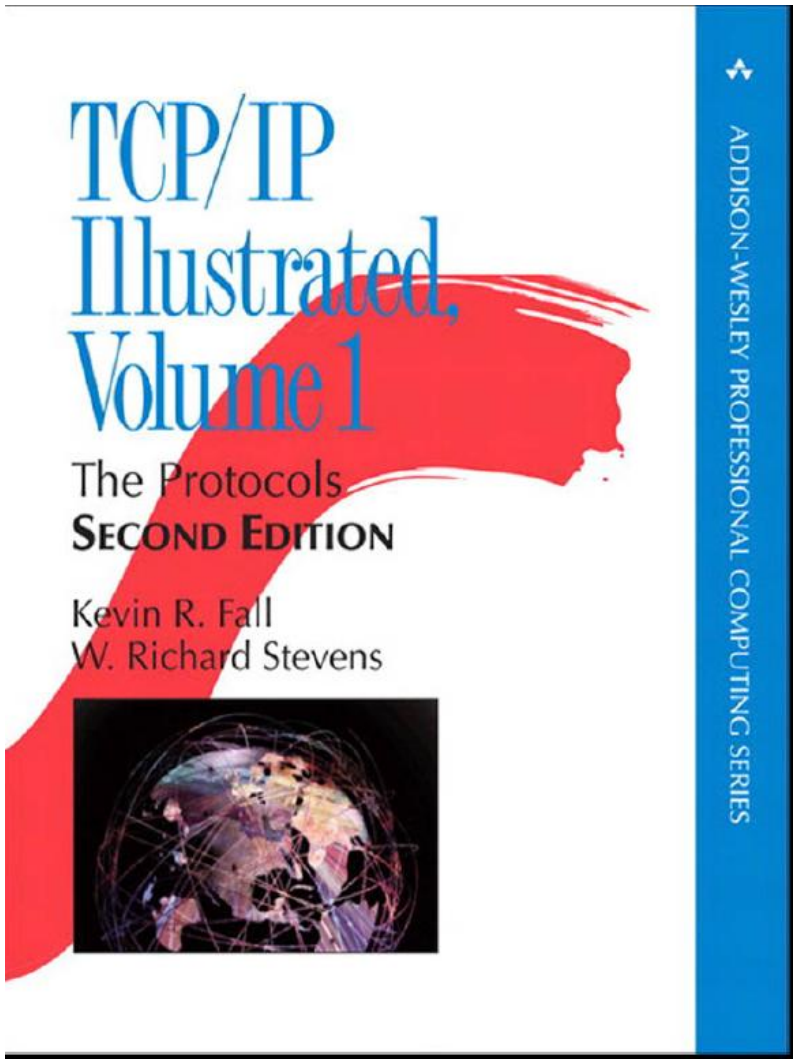


Textbook 2



References

Ref 1



Ref 2

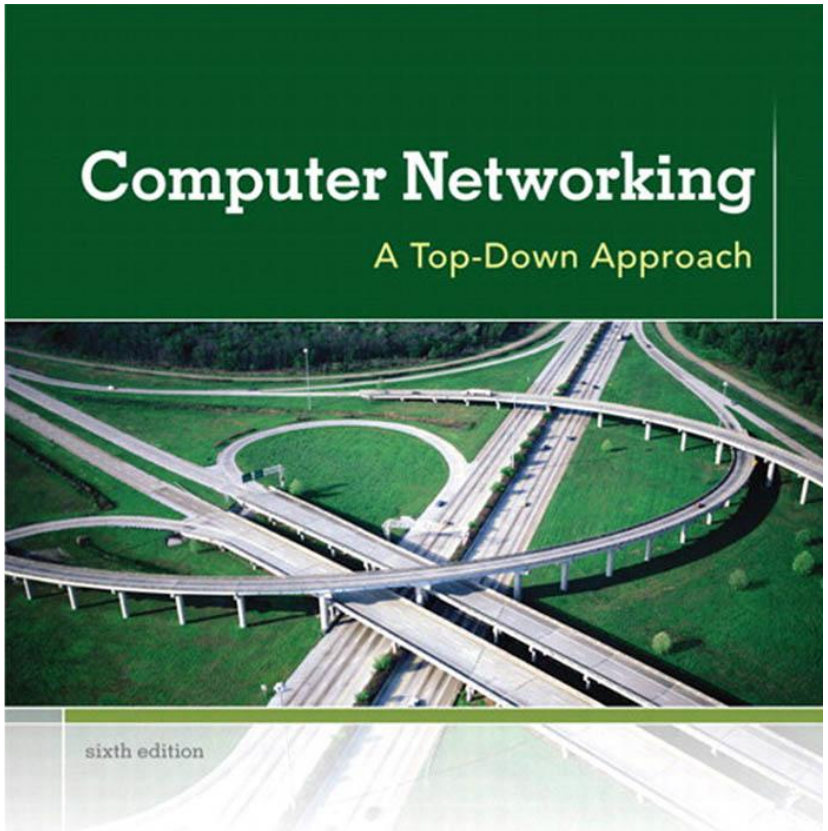
TCP Congestion Control: A Systems Approach

TCP Congestion Control: A Systems Approach

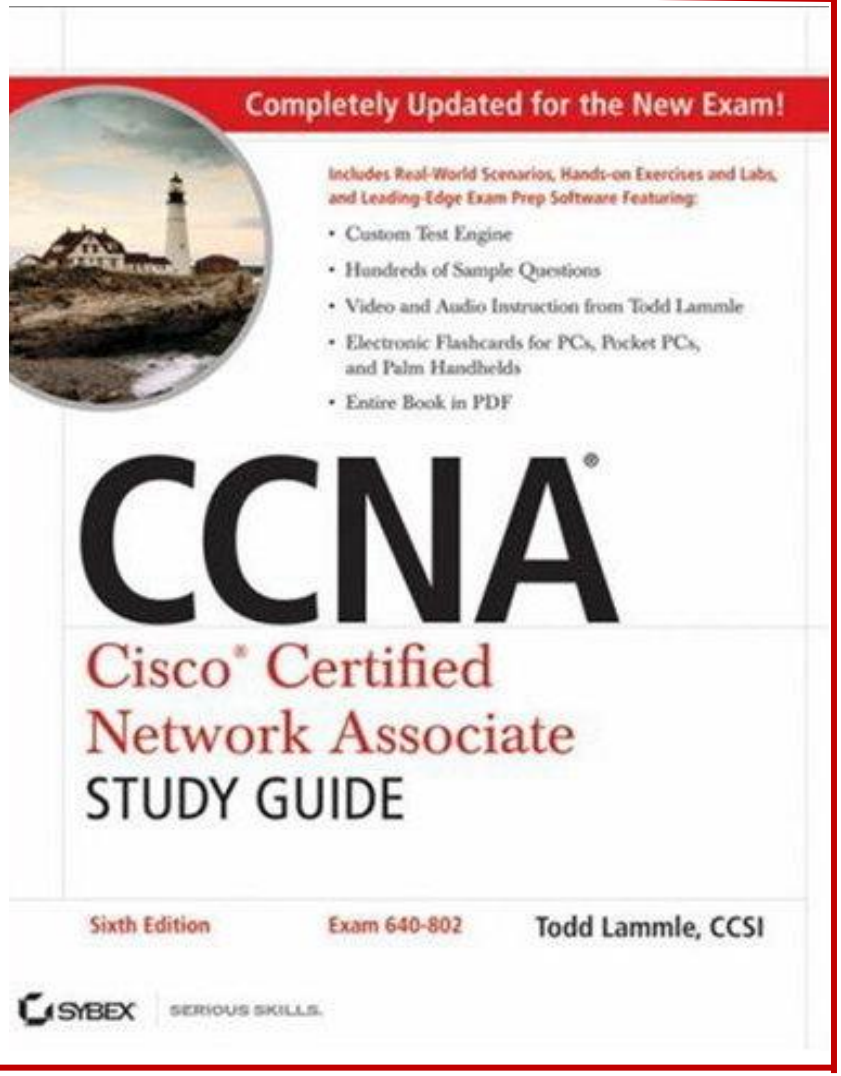
Peterson, Brakmo, and Davie

References

Ref 3



Ref 4



References

Ref 5