CS120: Computer Networks

Lecture 20. RTP and RPC

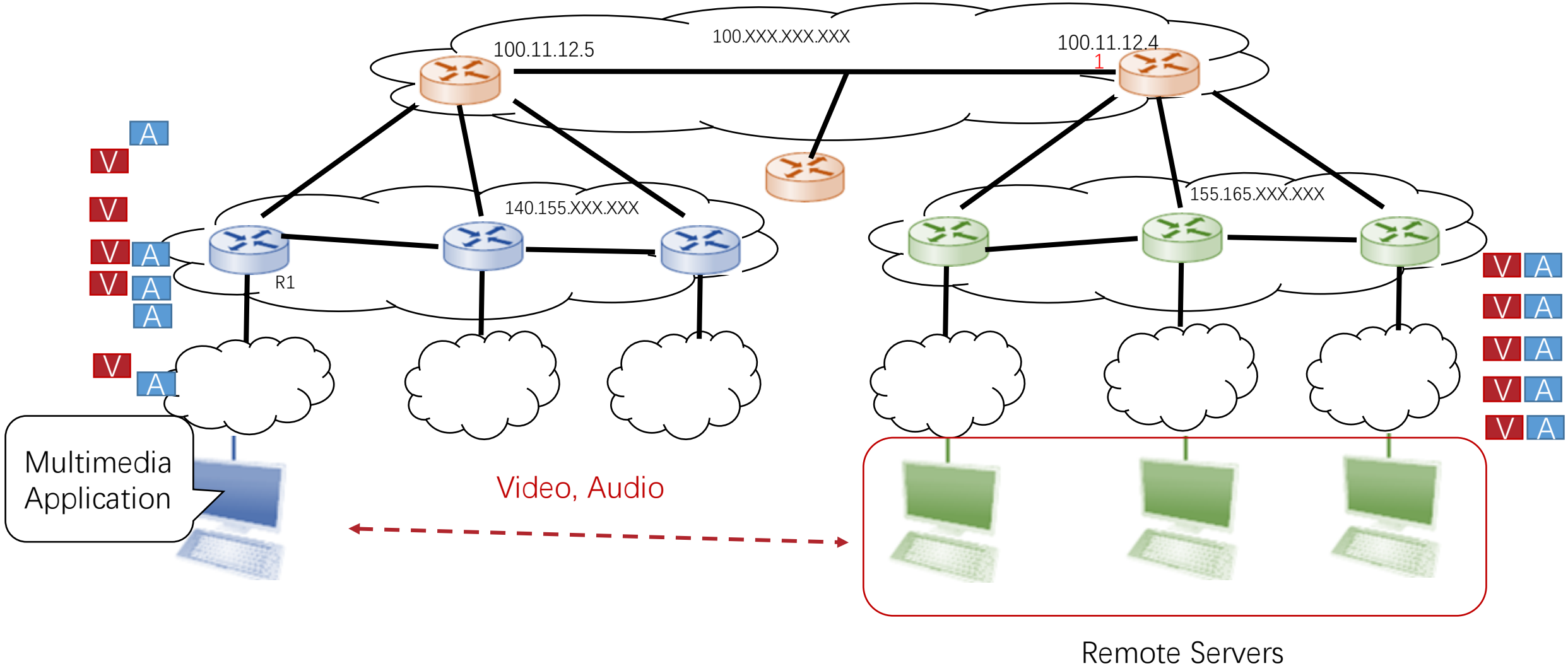
Haoxian Chen

Slides adopted from: Zhice Yang

Other End-to-end Protocols

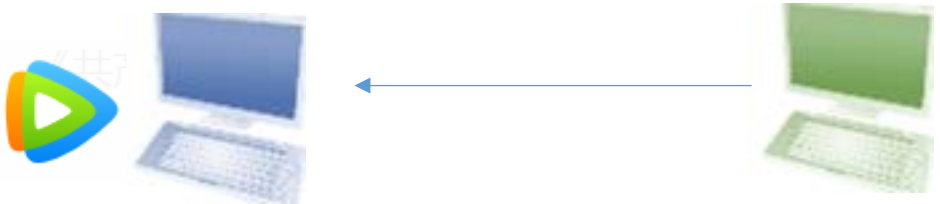
- Transportation Protocols
  - UDP
  - TCP
  - Remote Procedure Call (RPC)
  - Realtime Transport Protocol (RTP)
  - Others

Realtime Network Applications: Challenges

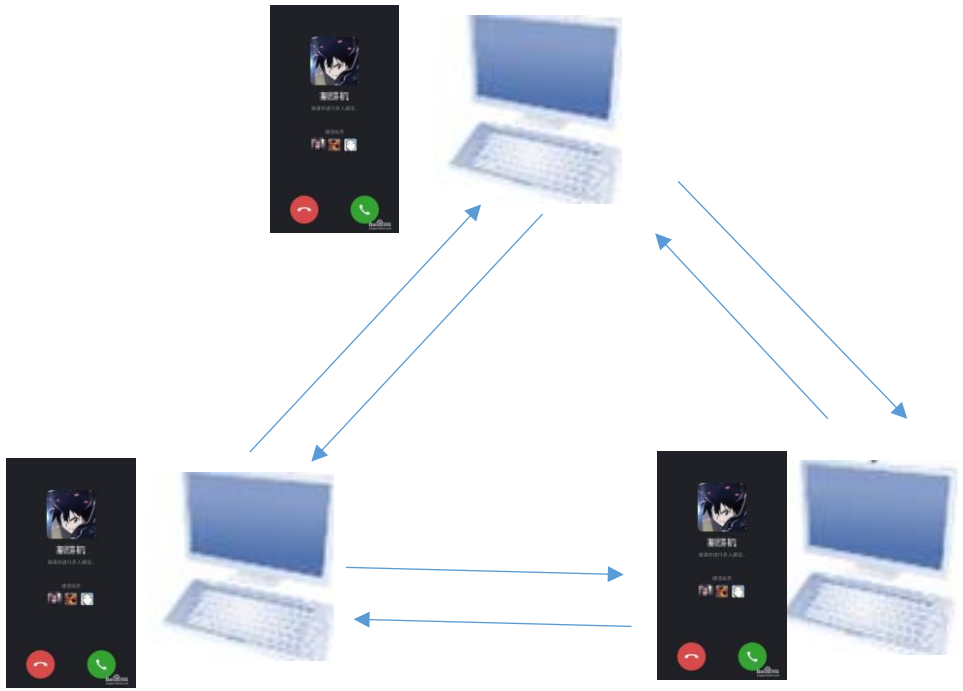


Realtime Network Applications

- Multimedia Applications
  - Applications involve video, audio, and data.
  - Two Classes:
    - Streaming application
      - TV broadcast, music broadcast
    - Interactive application
      - VoIP



Server



Realtime Network Applications: Solution

- Increase Network Capacity
- Upgrade Network Infrastructure/Protocol
  - Resource Reservation: RSVP
- Host Buffer

➢End-to-end Transport Protocol for Realtime Applications

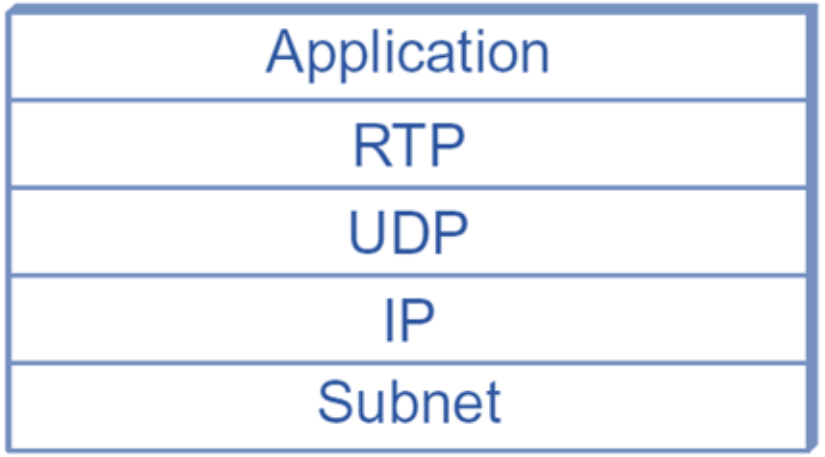
Why a New Transport Protocol ?

- TCP is not enough
  - TCP retransmissions introduce latency
  - Multimedia applications have their own needs
    - Have the information of video encoding
      - Better framing, error handling, etc.

➢ A new transport protocol

A New Transport Protocol

- Data Plane: Realtime Transport Protocol (RTP)
  - Data Carrier
- Control Plane: Realtime Transport Control Protocol (RTCP)
  - If RTP data is sent to the UDP-port P (should be even) RTCP messages should be sent on port P+1
  - Control Messages
    - Control Rate
    - Synchronization
  - Measurement Messages
    - Feedback Congestions/Qualities

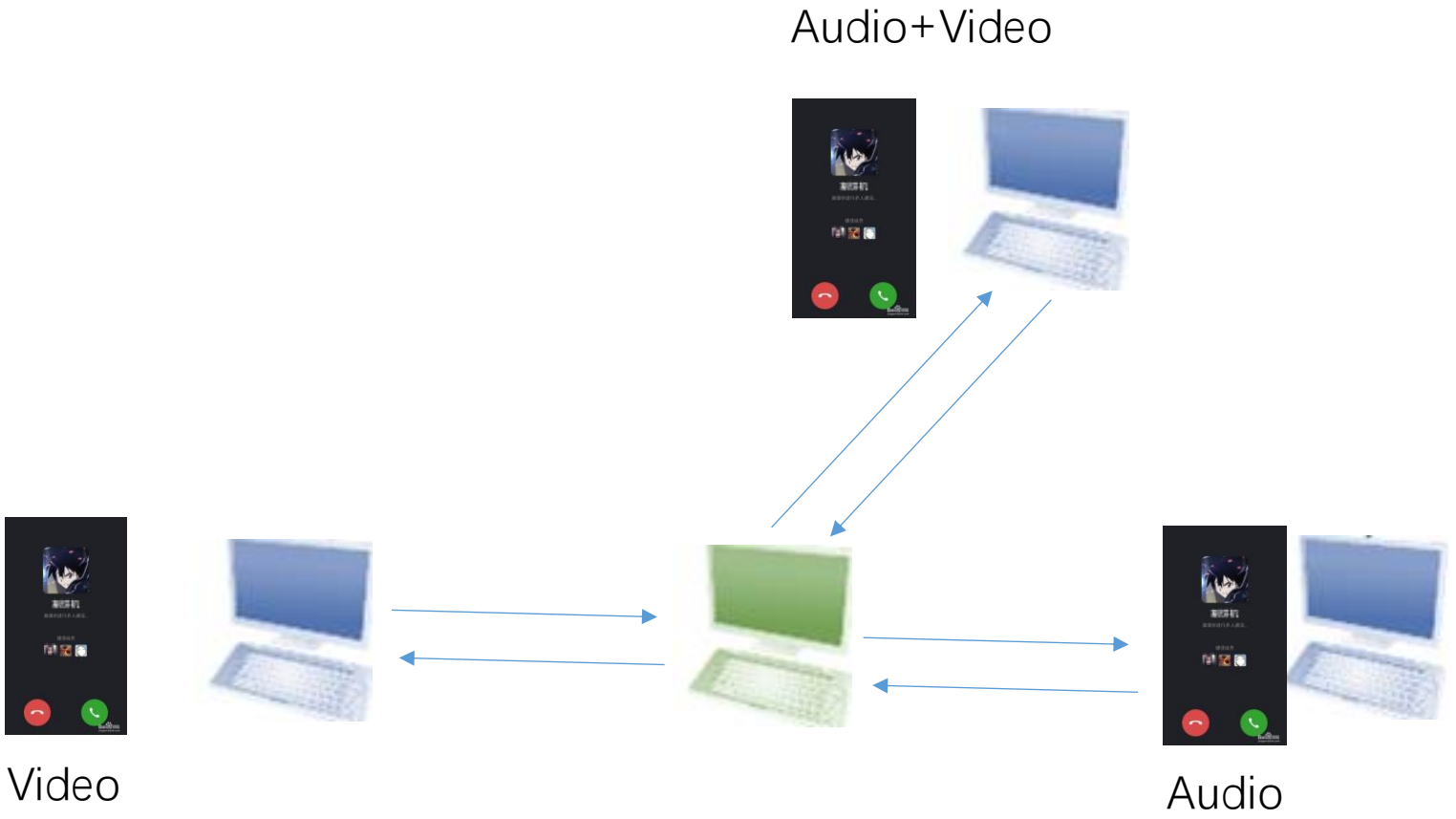


RTP and RTCP do not provide QoS

RTP Basic Design

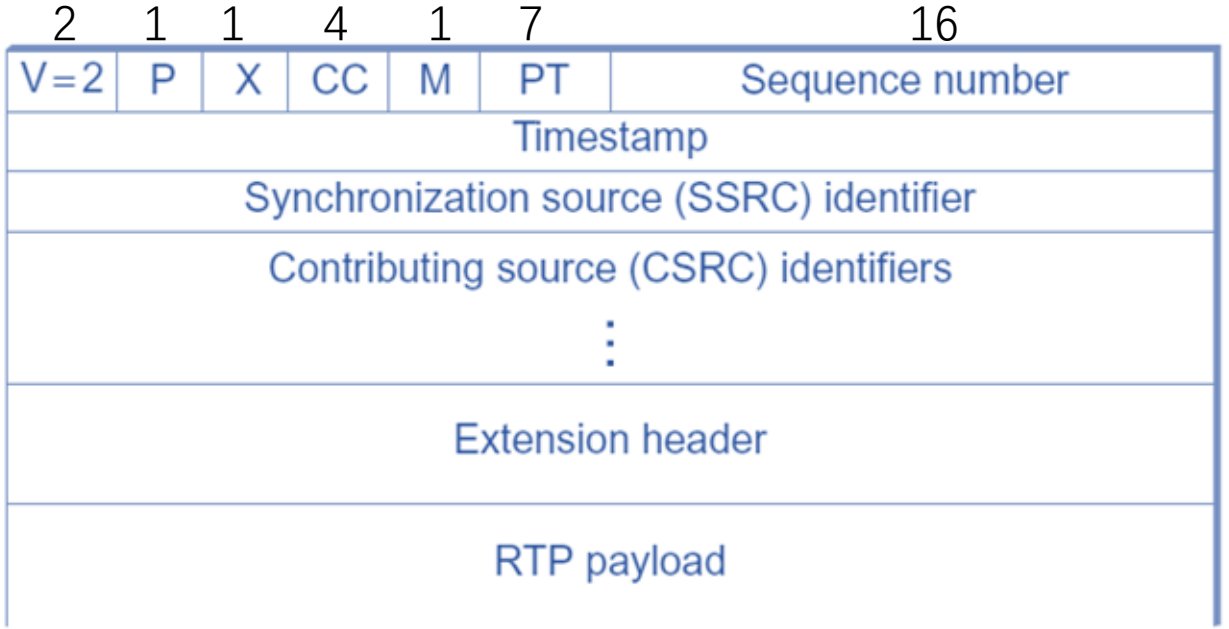
- To support various multimedia applications
  - Protocol is partially determined by application
  - Profile
    - Specify common information
      - Data rate, delay, etc.
  - Format
    - Format for the RTP payload

RTP Use Case



RTP Header

- V: Version
- P: Padding
- X: Extension
- CC: # of CSRC
- M: Marker
- PT: Payload Type
  - Encoding schemes
- Sequence number
  - For loss detection and reordering
- SSRC: Source ID
- CSRC List: List of Contributing Source ID
- Timestamp



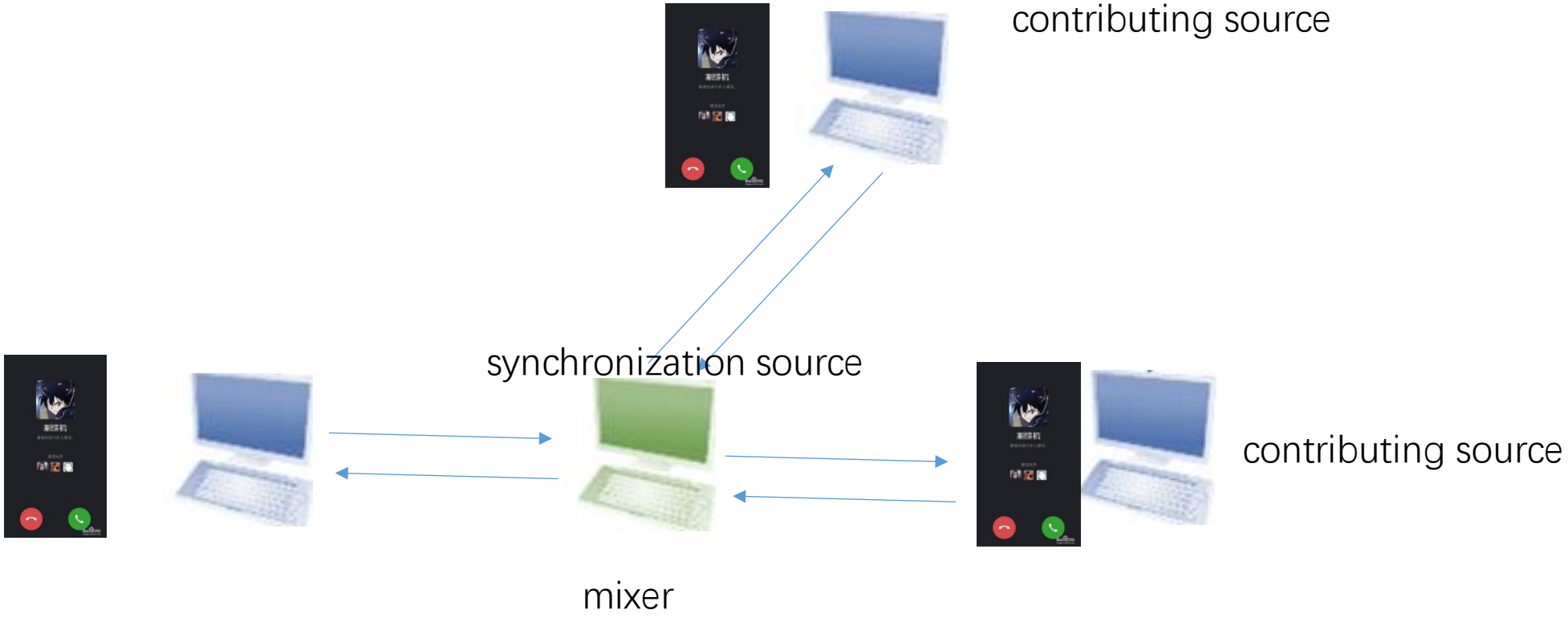
RTP Header

- Payload type (7 bits): indicates type of encoding currently being used. If sender changes encoding during call, sender
  - informs receiver via payload type field
  - Payload type 0: PCM mu-law, 64 kbps
  - Payload type 3: GSM, 13 kbps
  - Payload type 7: LPC, 2.4 kbps
  - Payload type 26: Motion JPEG
  - Payload type 31: H.261
  - Payload type 33: MPEG2 video

RTP Header

- Timestamp field (32 bits long): sampling instant of first byte in this RTP data packet
  - for audio, timestamp clock increments by one for each sampling period (e.g., each 125 us for 8 KHz sampling clock)
  - if application generates chunks of 160 encoded samples, timestamp increases by 160 for each RTP packet when source is active. Timestamp clock continues to increase at constant rate when source is inactive.
- SSRC field (32 bits long): identifies source of RTP stream. Each stream in RTP session has distinct SSRC

RTP Use Case


contributing source
synchronization source
contributing source
mixer

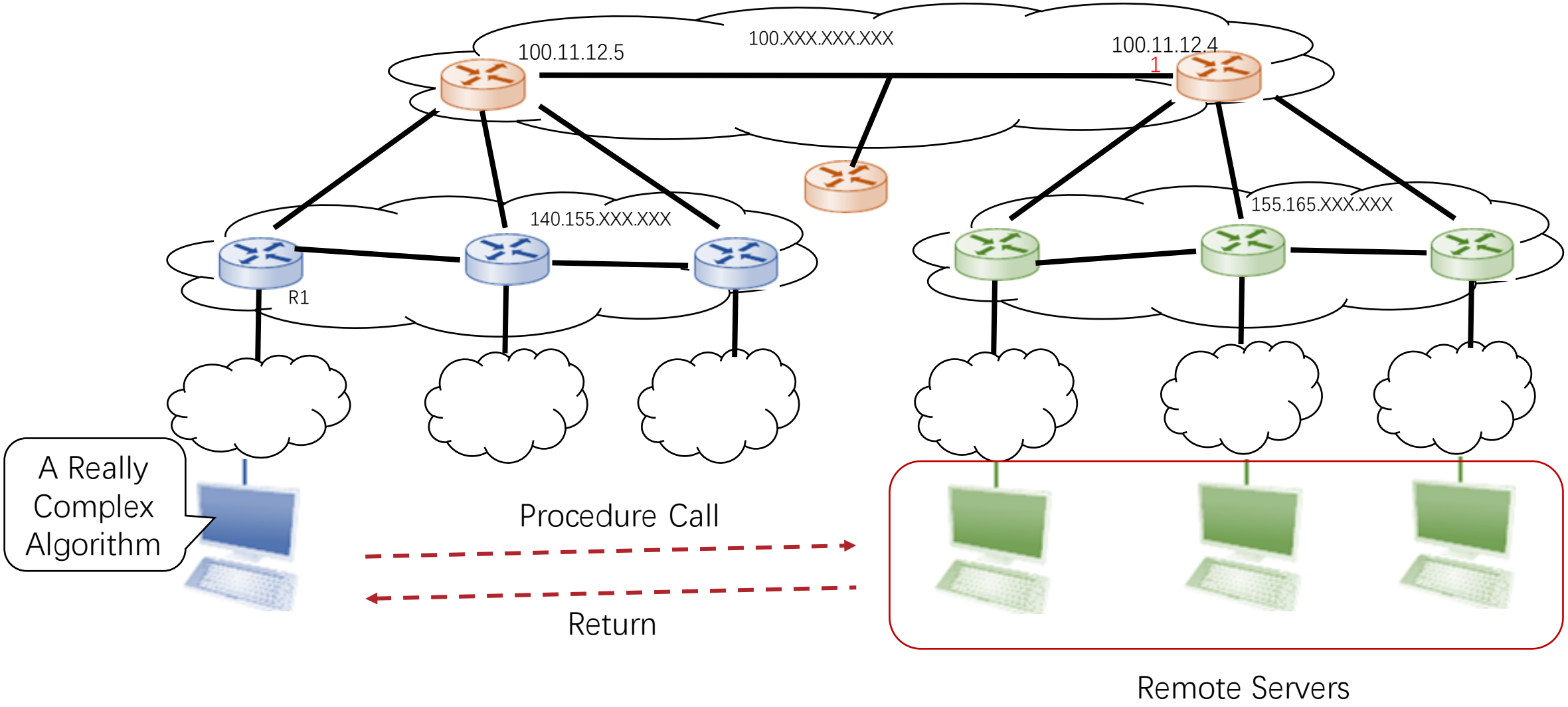
RTP Timestamp

- Relative Timestamp
  - Real time of each tick is defined in profile
- Difference of timestamps of consecutive packets may differ
  - Due to video encoding
- Consecutive timestamps may have same value
  - From coupled source (video and audio)

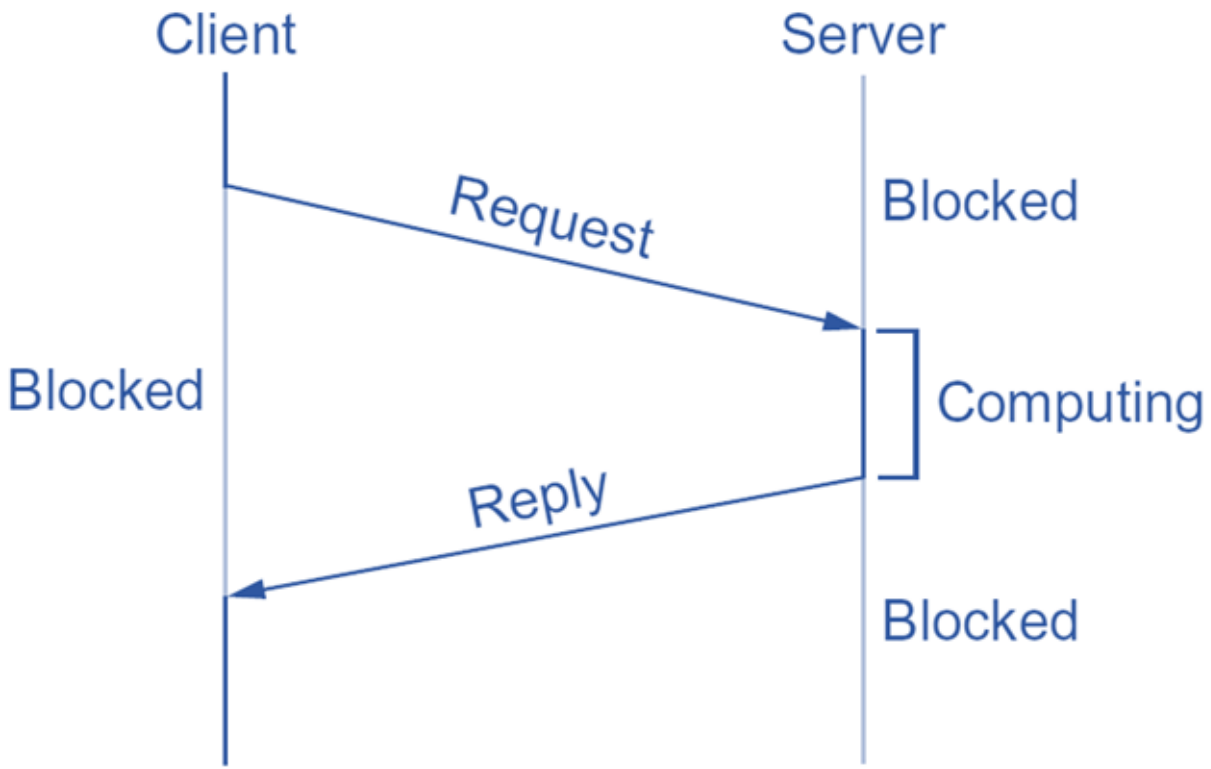
RTP Demo

- VLC RTP Broadcast
  - https://www.bogotobogo.com/VideoStreaming/VLC/How_to_Streaming_Live_Network_rtp.php
- Audio Broadcast via RTP
  - http://www.radioparadise.com/rp_2.php?#

Remote Procedure Call (RPC)



Remote Procedure Call (RPC)



Why RPC ?

- Computation Limitation
  - e.g., phone, wearables, UAVs, etc.
- Hide the Implementation
  - Similar as libs.
  - e.g., protect proprietary algorithms
- Functions that just can’t run locally
  - e.g., different architecture
- Super Computing
- Local Procedure Call
  - Special RPC runs in local machine, used for cross domain function calls
- And more

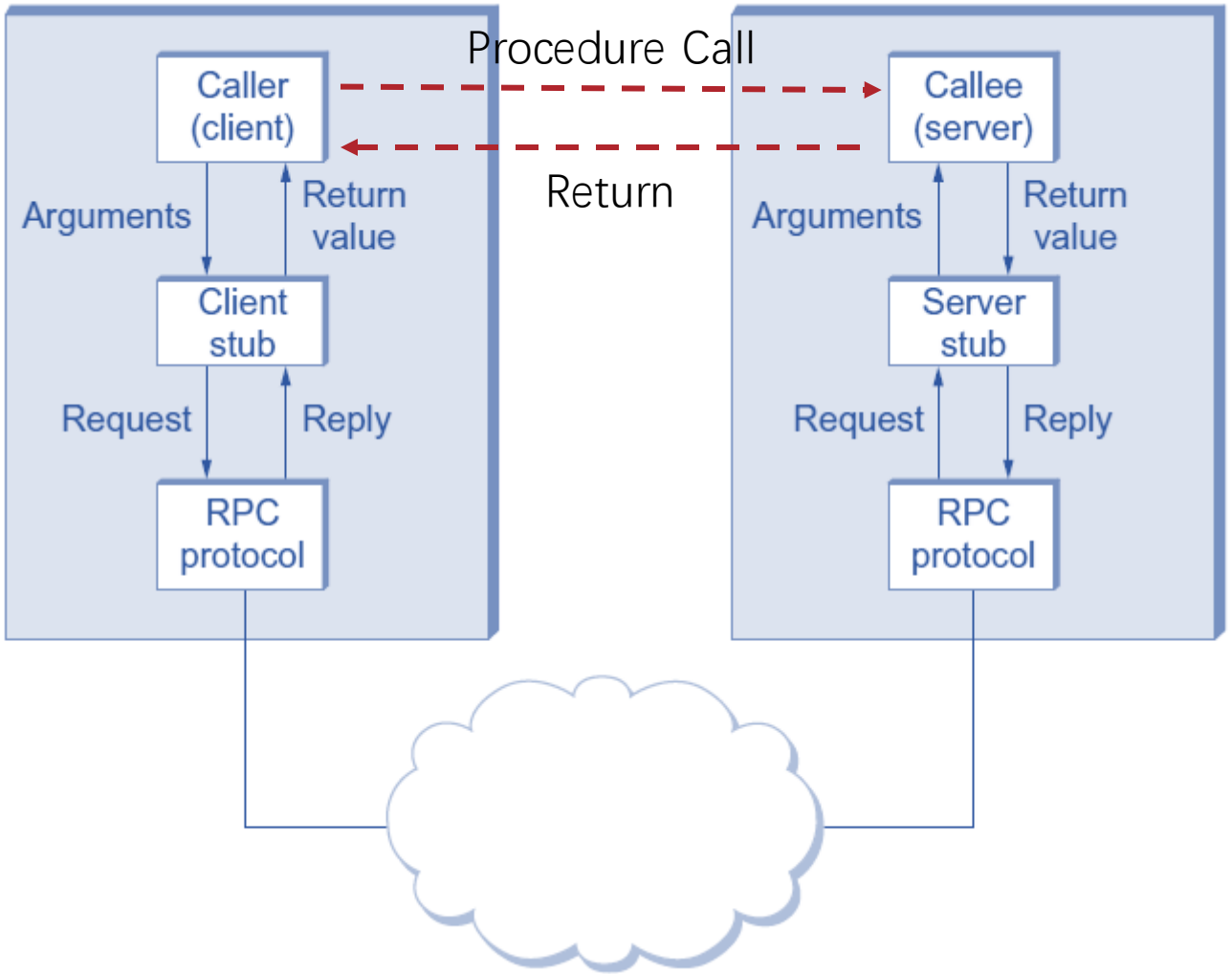
Why TCP/UDP is not Enough ?

- UDP does not maintain correctness, order, etc. of messages
  - Remote Procedure Call does require such guarantee
- TCP is data oriented
  - Not so convenient for programming function call
  - Would like to invoke remote function seamlessly just like a local function
    - Almost transparent for the programmer

RPC Challenges

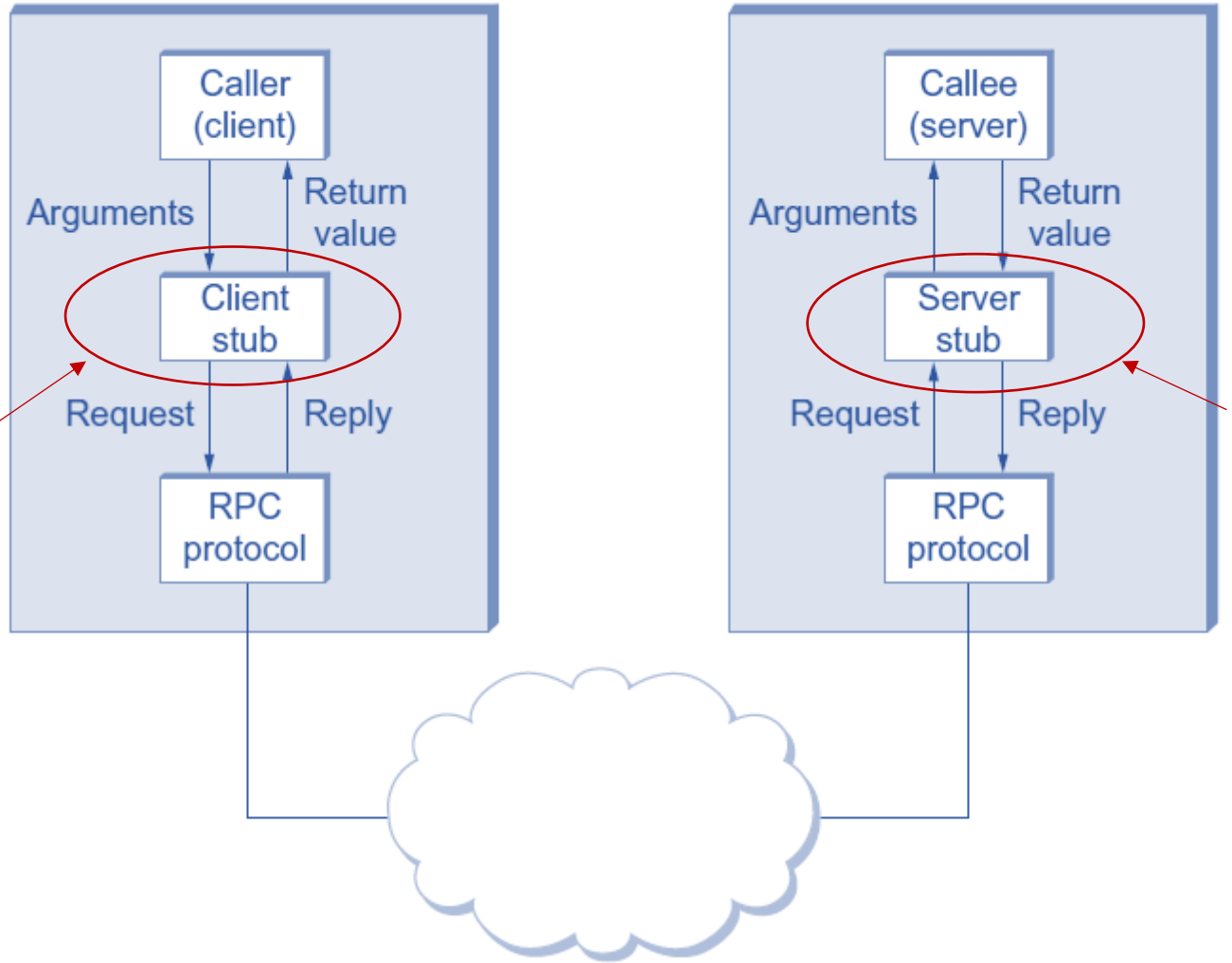
- Network Issues
  - ➢ An end-to-end RPC protocol to deal with the potentially undesirable properties of the underlying network
- Heterogeneity
  - Client and Server might have different:
    - OS versions
    - Languages
    - Endian-ness
    - Hardware architectures
    - etc.
  - ➢ Programming language and compiler support to package local functions and arguments into request messages (unpackage on the server)

RPC Mechanisms



RPC Mechanism – Language and Compiler

Stub is like a proxy to translates procedure calls between network transmissions



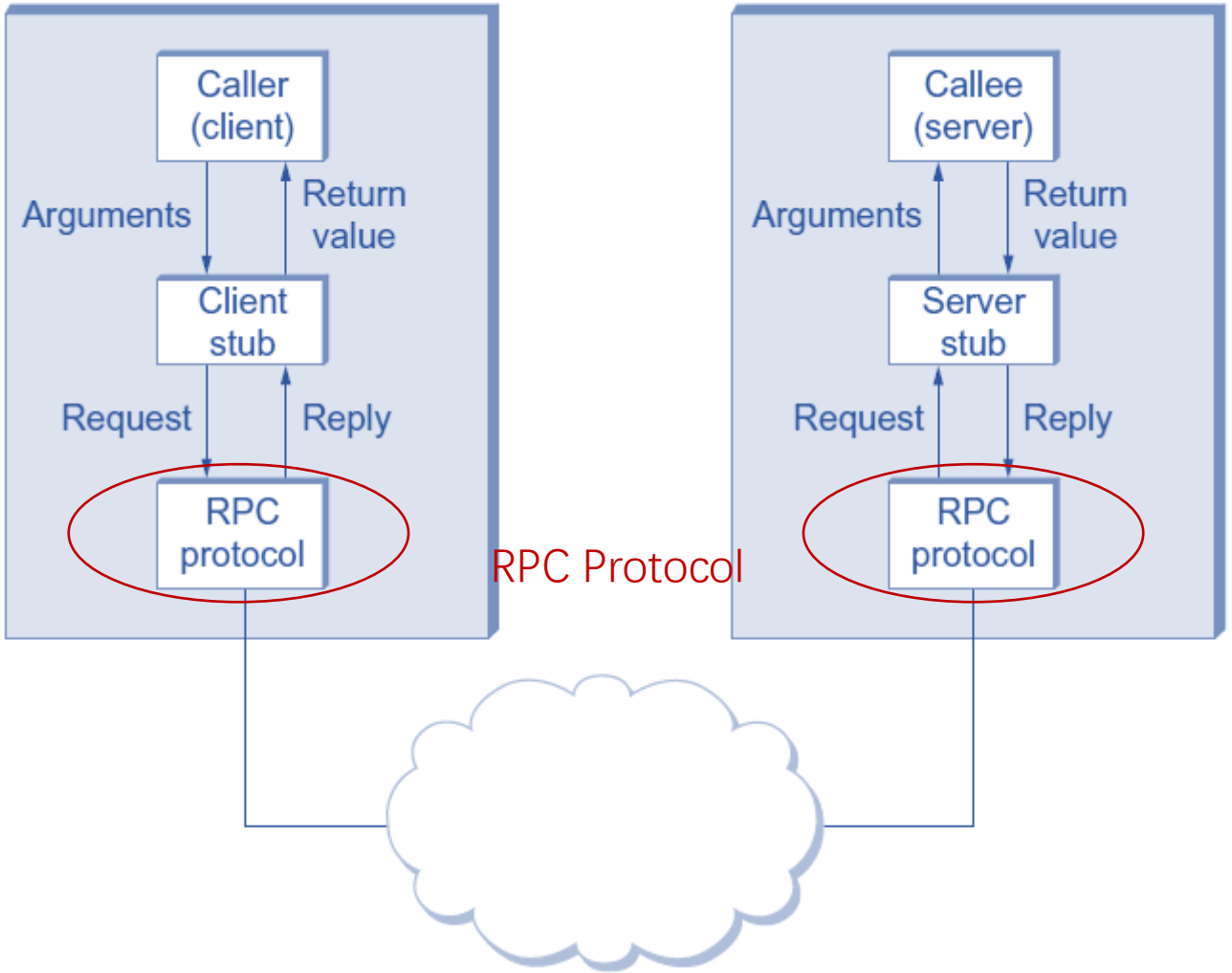
Marshals parameters and calls the server

Unmarshals parameters and calls the local function

Generating Stub

- Language-level Support
  - Complier generates stub
  - e.g., Java, Python, Haskell, Go
- Higher level Support
  - Through additional libs, applications, compliers.
  - e.g., C, C++

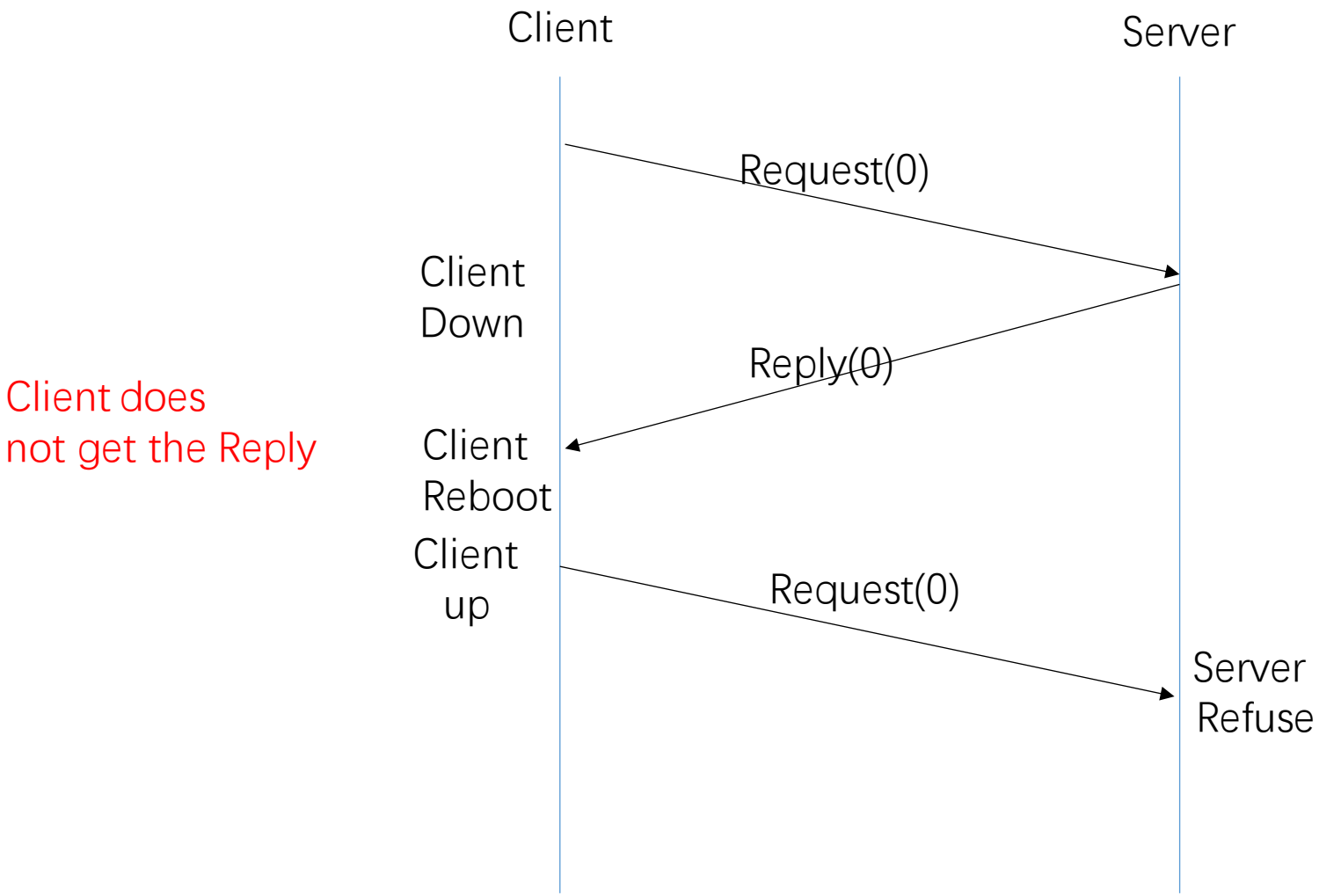
RPC Mechanism – Protocol



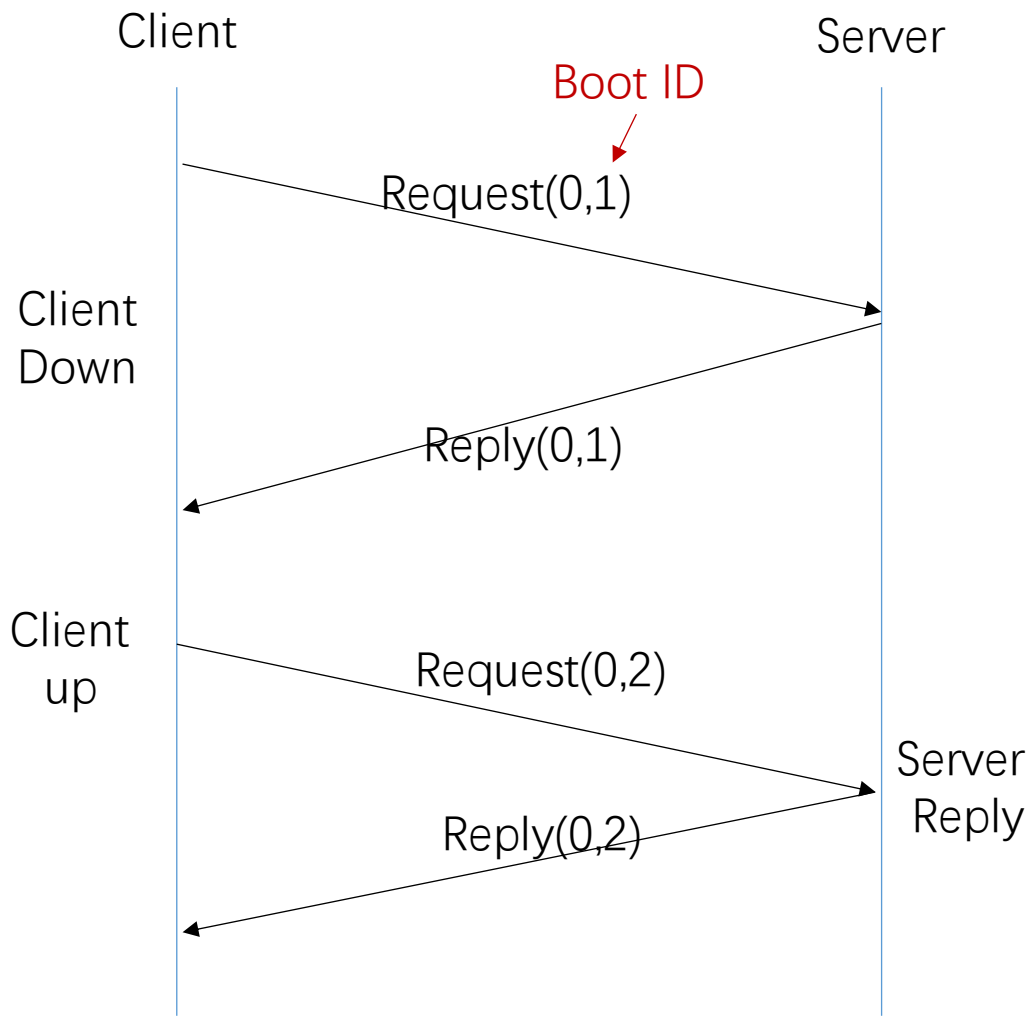
RPC Protocol: Identifiers in RPC

- Identifying Remote Function/Methods
  - Similar as IP address
  - IP address + port + function name
- Identifying Each Message
  - Message ID
- Identifying Unexpected Response
  - Boot ID

Identifying Unexpected Response



Identifying Unexpected Response



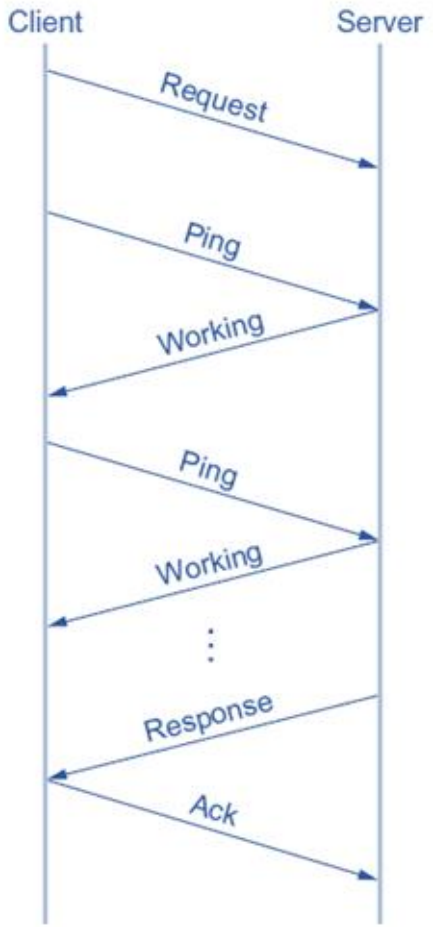
RPC Protocol: Reliable Transmission

- Similar to TCP
  - Depending on the requirement
    - Stop-wait or sliding window
  - Use multiple “channel” for improving efficiency
  - Use “ping” messages to detect liveness

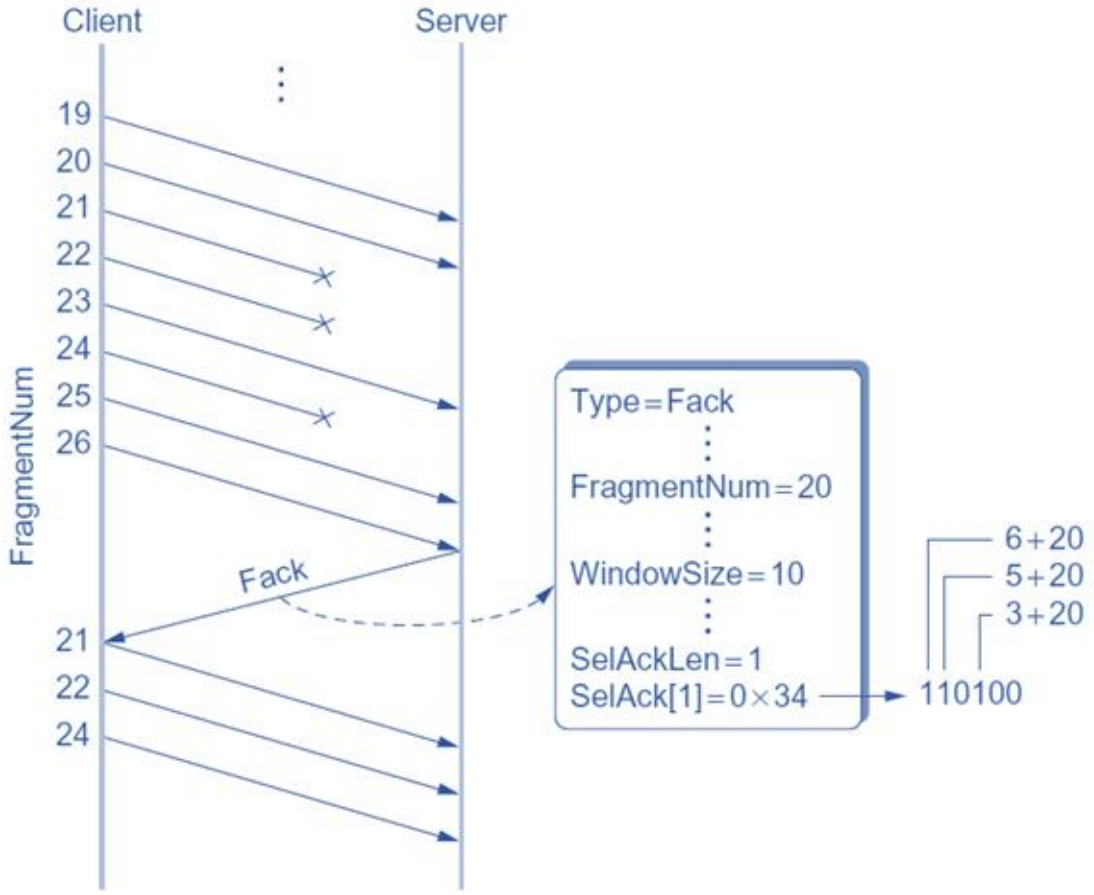
RPC Protocol: RPC Semantics

- At-most-once
  - Must recognize duplicate requests
    - Maintain identifiers of past requests
  - e.g., payment, launch a missile
- Zero-or-more
  - e.g., HTTP GET, Hash, etc.

RPC Protocol: Examples from DCE RPC



Liveliness Checking



Fragmentation and Fragment ACK

RPC Implementations

- Sun PRC
- DCE-PRC
- Java RMI
- DCOM
- etc.

RPC Demo

- Java RMI (Remote Method Invocation)
  - To capture loopback traffic https://www.netresec.com/index.ashx?page=RawCap

Reference

- Textbook 5.3 & 5.4
- https://docs.oracle.com/javase/tutorial/rmi/