TI Developer Conference
February 15-17, 2005 • Houston, TX

SEE THE **FUTURE.**
CREATE YOUR OWN.

A Multimedia Streaming Server/Client Framework for DM64x

Bhavani GK
Senior Engineer
Ittiam Systems Pvt Ltd
bhavani.gk@ittiam.com

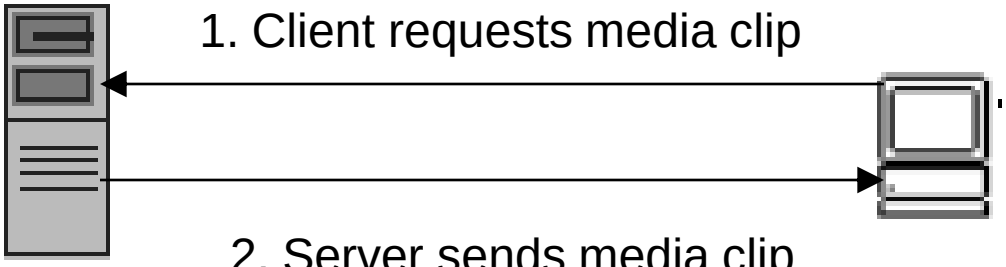
Ittiam

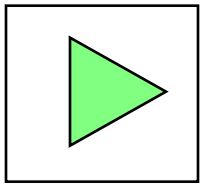
Agenda

- Overview of Streaming
- Application Scenarios
- TI DM642 Capability and Features
- Streaming Server Framework
- Streaming Client Framework
- Conclusions

Media Delivery Modes - Downloading

- **Media is downloaded from the web-server, stored locally on the client and then played out**

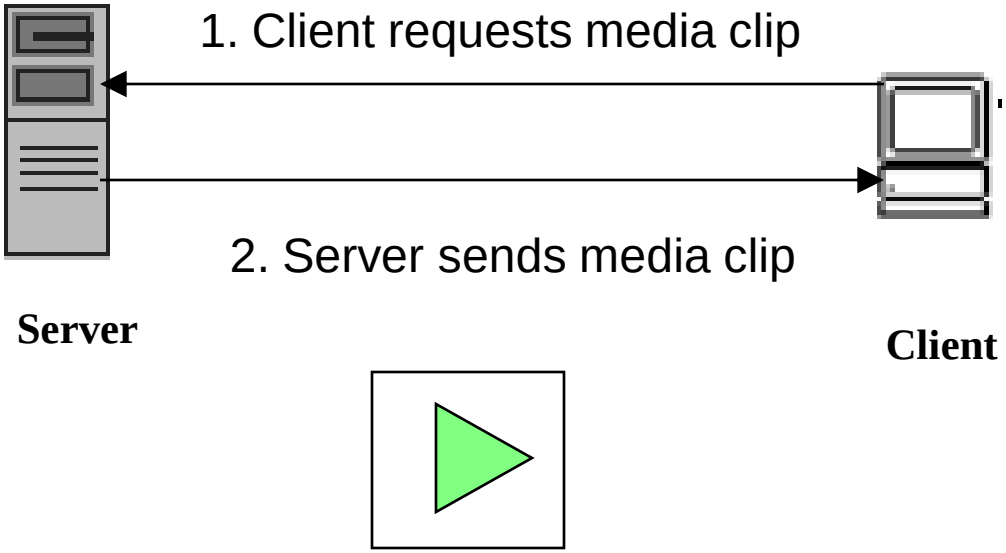




3. Client plays downloaded clip

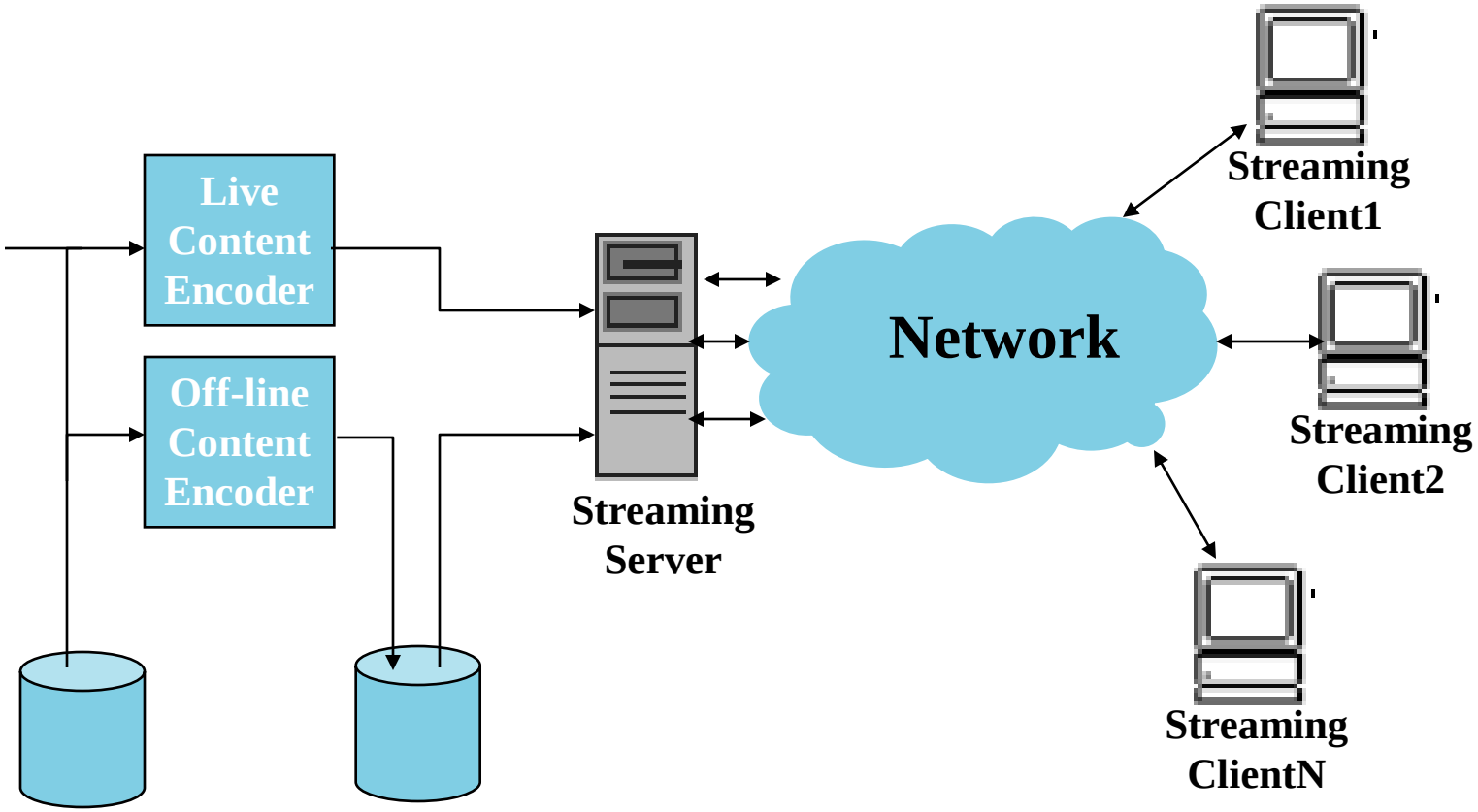
Media Delivery Modes - Streaming

- **When media is requested, the server starts "Streaming" data, the client starts playing data after buffering a few seconds of data**

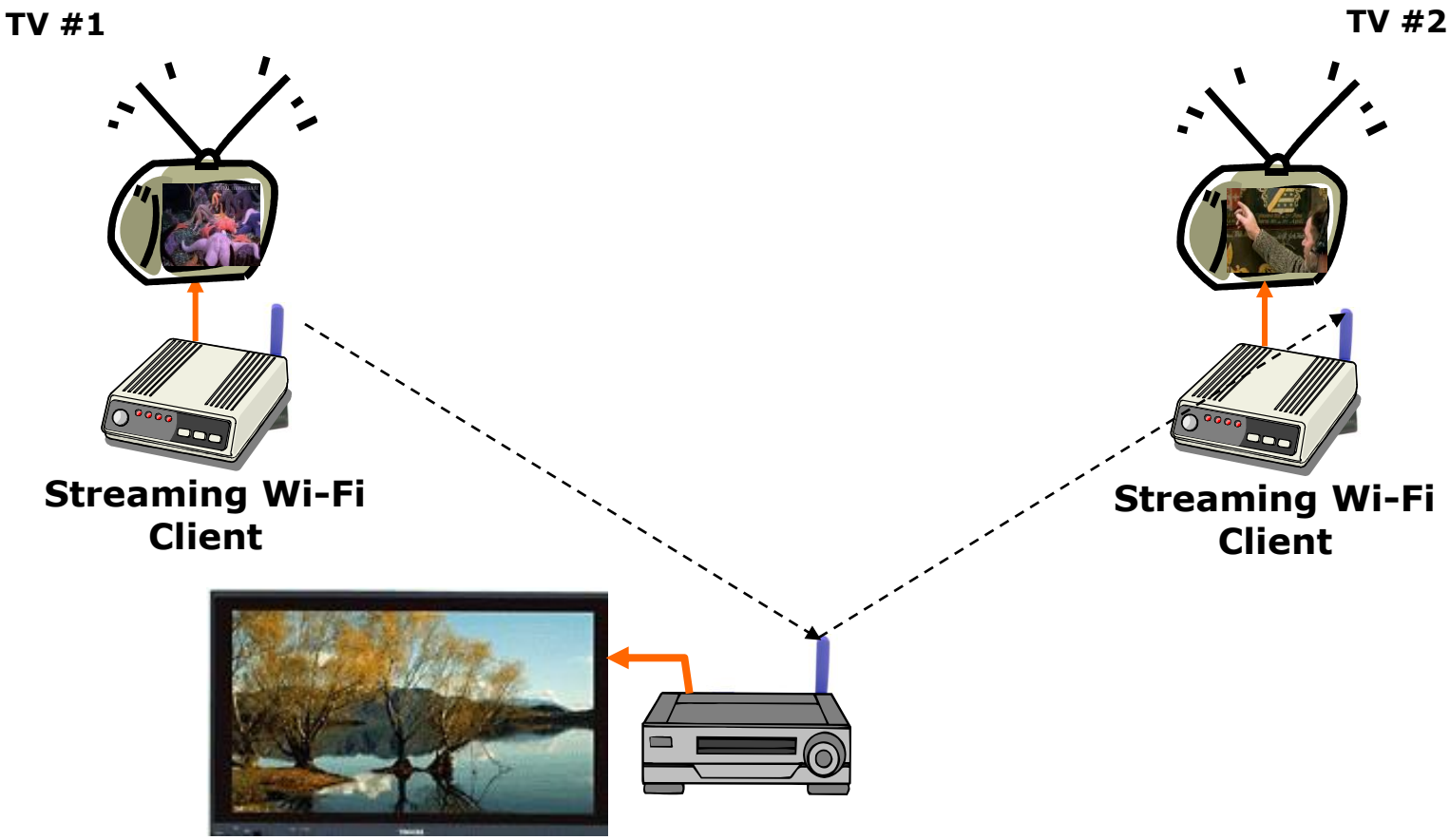


Streaming

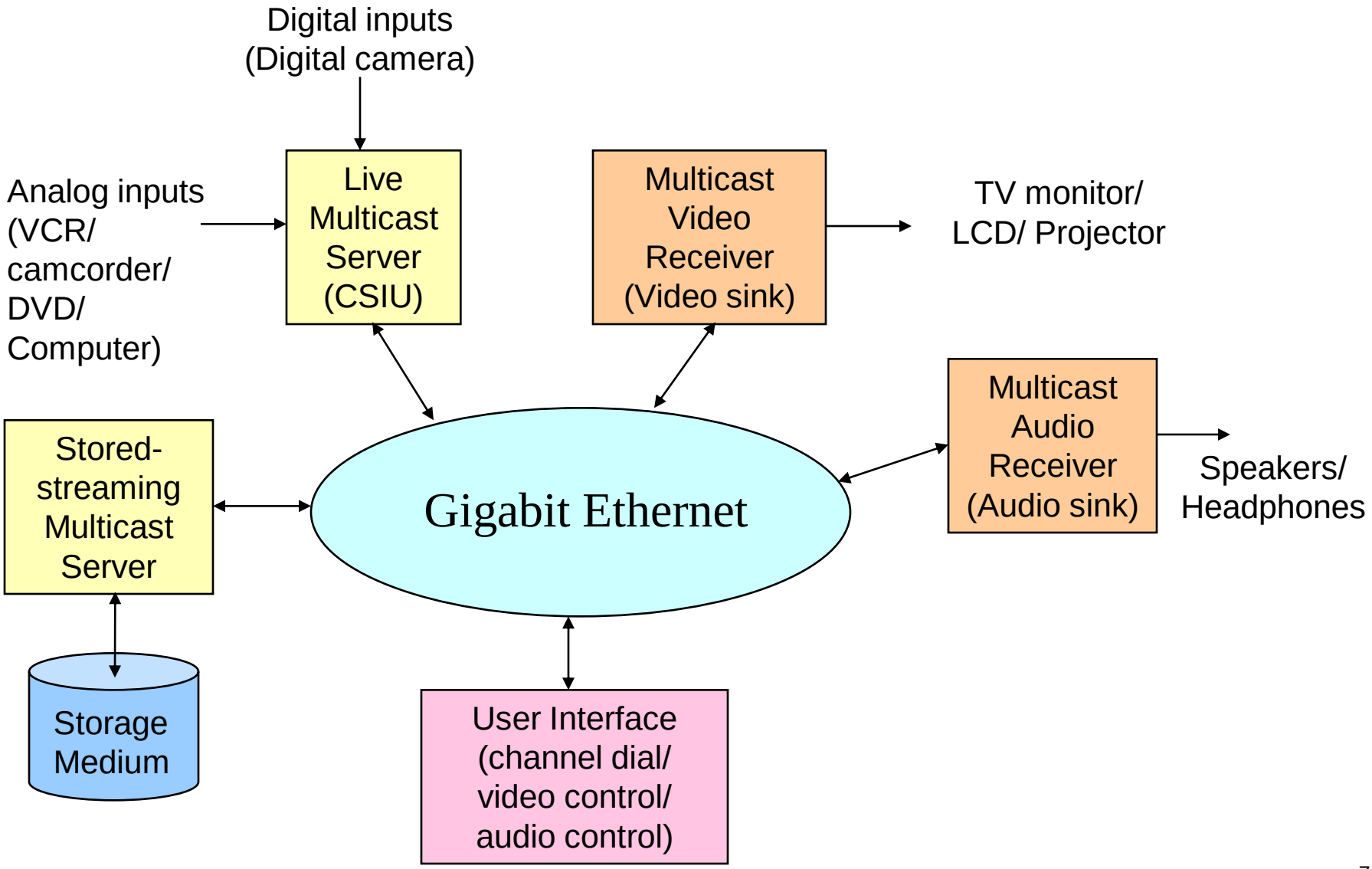
- Anytime and/or Anywhere access to media data over a network



Application Scenario 1

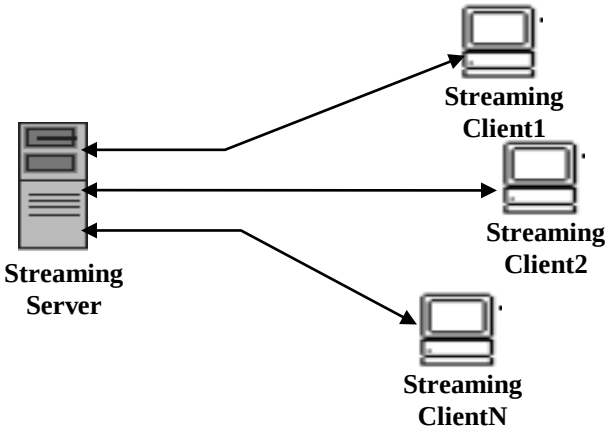


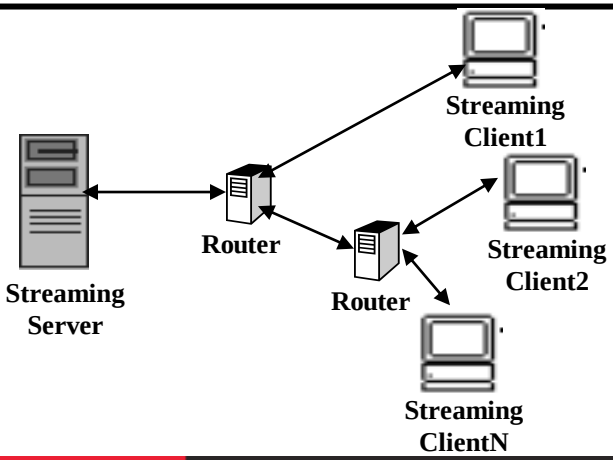
Application Scenario 2



Streaming Modes

Unicast	Multicast
Each client has a distinct streaming session	All clients have the same streaming session
Server capacity bottlenecks	Routers handle copies; only one stream from server per session. Not all routers support multicast
Suited for on-demand delivery	Suited for Live delivery
Interactivity is possible	Interactivity is not possible
Server can adapt bit-rate	Global bit-rate adaptation only





Interoperable Streaming - Protocols

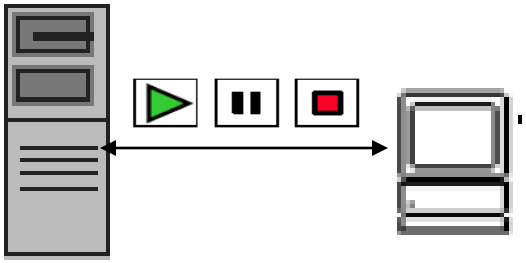
- ISMA - Internet Streaming Media Alliance
  - Accelerate the adoption and deployment of open standards for streaming rich media content such as video, audio, and associated data, over Internet protocols.
- RTP – Real Time Protocol
- RTSP – Real Time Streaming Protocol
- RTCP – Real Time Control Protocol

Interoperable Streaming - Protocols

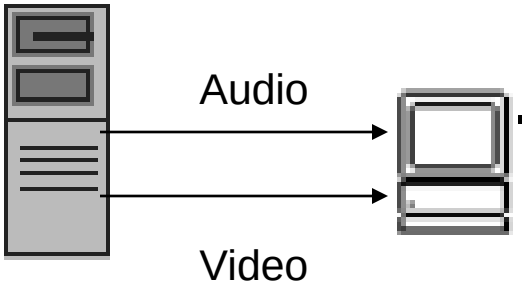
- RTSP
  - "Internet VCR remote control"
  - Communication protocol between the server and the client - Initiates and controls delivery of data, negotiates codecs
  - Uses TCP
- RTP
  - Uses for streaming Audio and Video over UDP
  - Provides timing reconstruction, loss detection, media identification through timestamps, sequence numbers
  - Separate RTP channels for each medium
- RTCP
  - Control protocol for RTP
  - Provides feedback from client to server on quality of streaming
  - Uses UDP

Interoperable Streaming - Summary

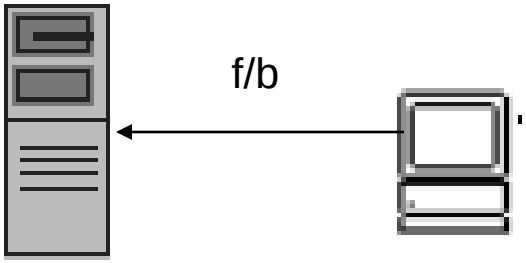
RTSP

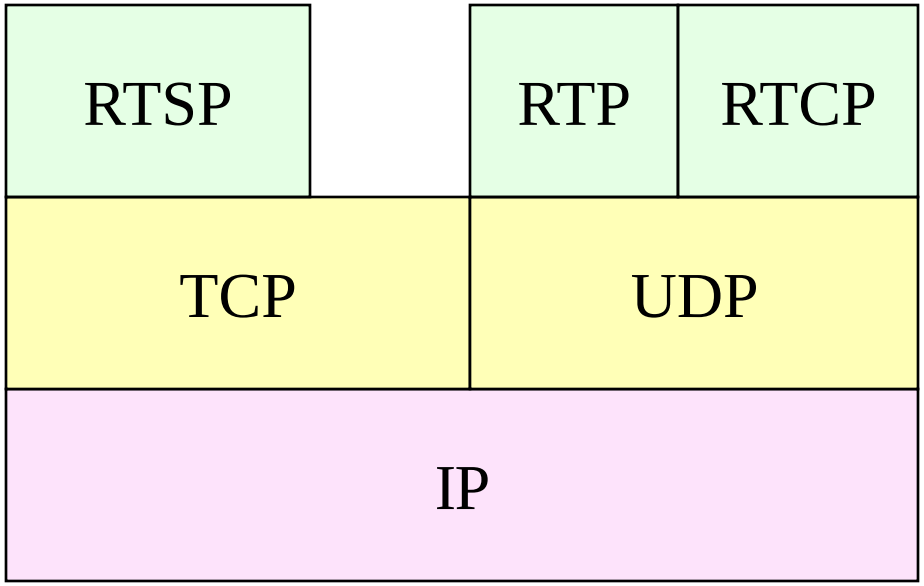


RTP

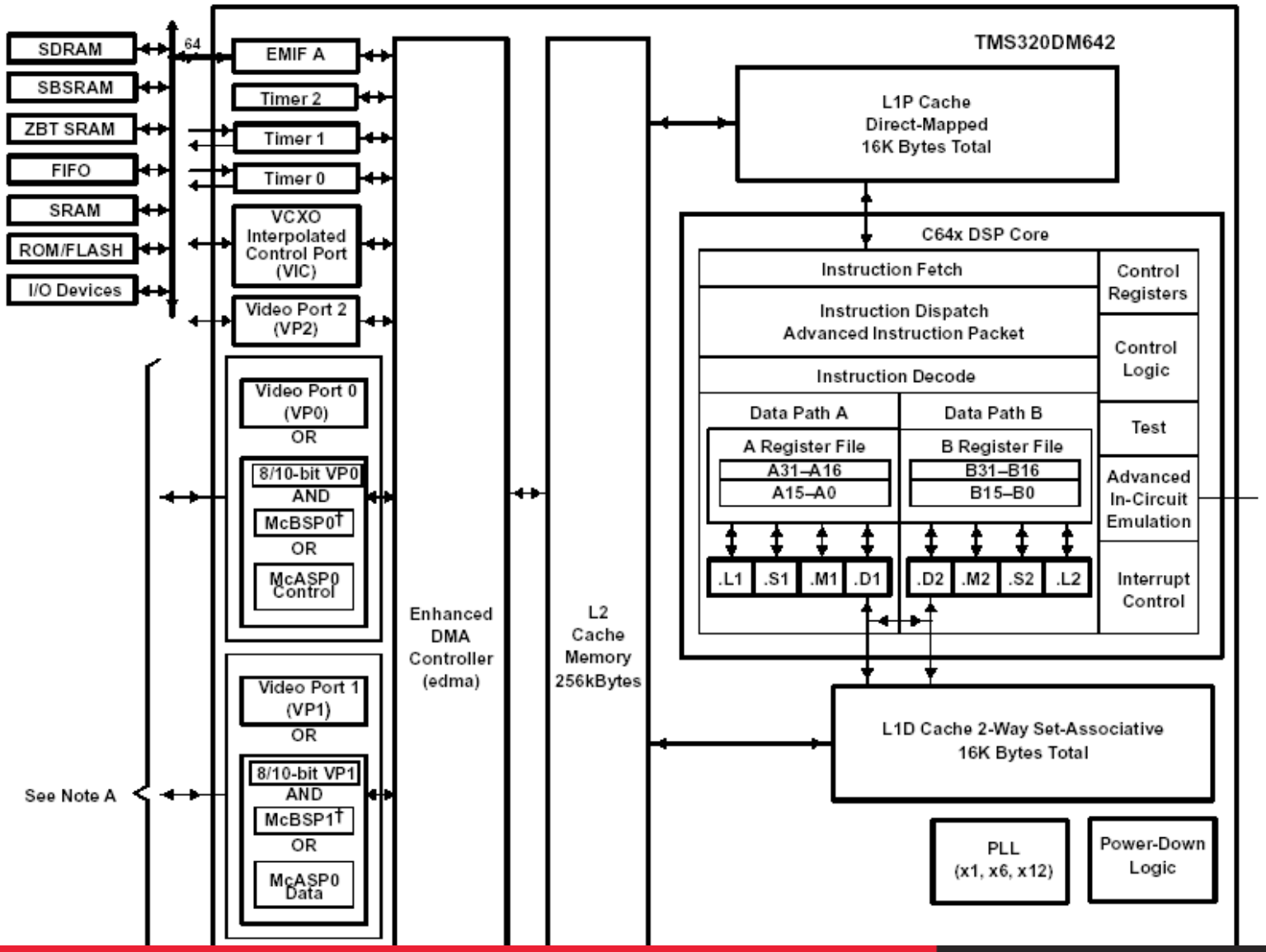


RTCP





TI TMS320DM642



TI C64x DSP Core

- VLIW architecture (VelociTI)
  - 256-bit instructions (8 32-bit)
  - Dual data paths
    - 4 functional units (L, M, S, D) each
      - L, S – arithmetic/logical; M – multiply/shift; D – Data load/store
    - 32 32-bit registers each
    - Cross path access to registers
  - Can load and store up to 64-bits
    - Non-aligned load/stores are supported
  - All instructions can be conditionally executed
  - Packed 8-bit and 16-bit operations
    - E.g. AVGU4, DOTPU4, ADD2, MIN2, MAX2, etc.
  - Instructions to pack and unpack
- Transfer crossbar with 4 queues of varying priority
- 64-channel enhanced DMA (TCInt, Chaining, linking)
- 2-level cache (16 kB I and D – L1; 256 kB L2)

TI DM642 Features

- External Memory Interfaces
  - 1 64-bit (EMIFA); 1 16-bit (EMIFB)
- 3 video ports
  - Can be configured as video input, video output, or transport stream interface
- Multi-channel Audio Serial Port (Data/Control)
- 2 Multi-channel Buffered Serial Ports
- Ethernet MAC
- PCI interface
- Host Post Interface
- I2C Interface

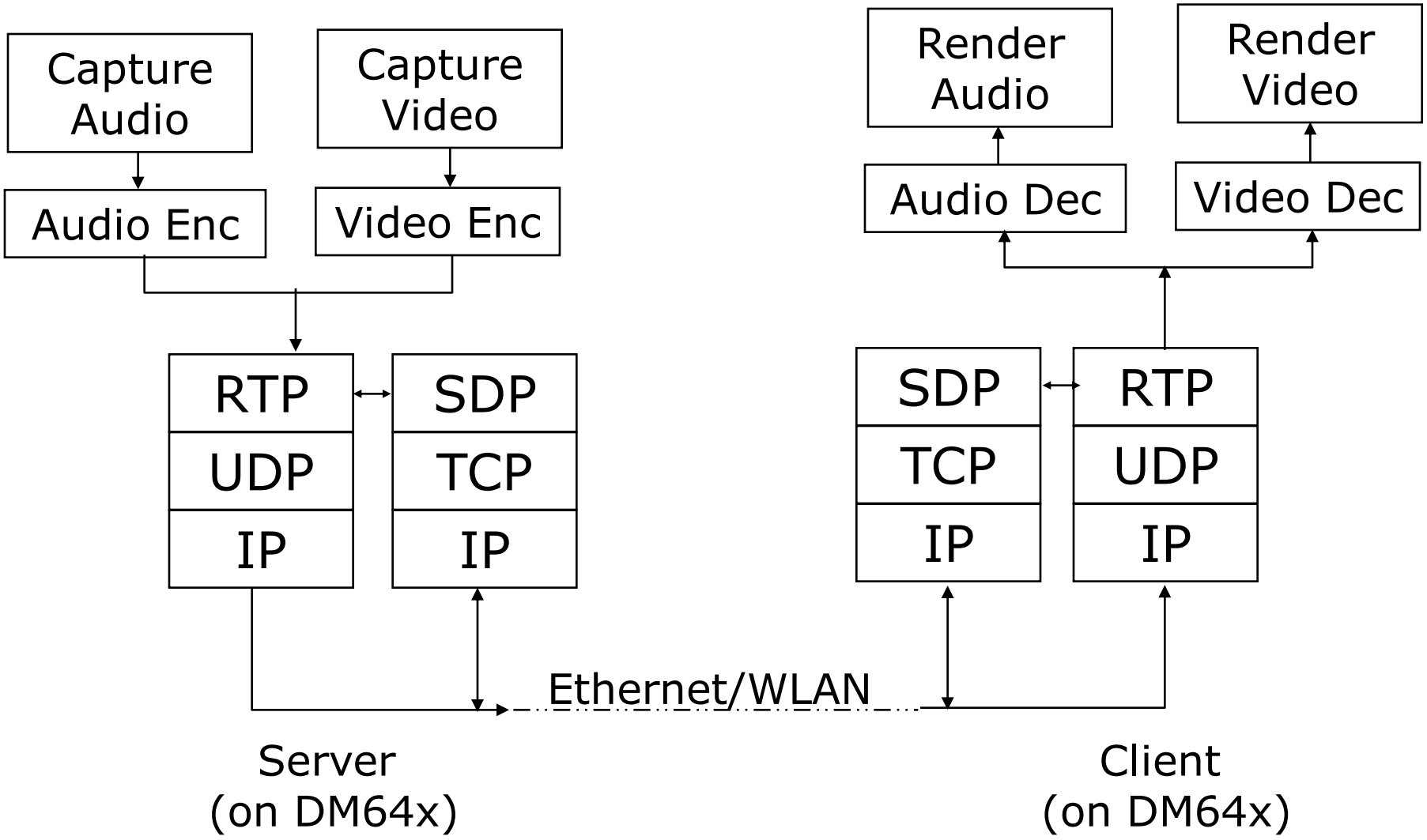
DSP BIOS

- Scalable real-time kernel
- Provides pre-emptive multi-threading, hardware abstraction
- Easy to use configuration tools to create and configure instances of various modules and objects
- Provides for real-time analysis and debugging
- Useful DSP-Bios Objects
  - **PRD, MBX, CLK, LOG, SEM, LCK**
- Useful DSP-Bios Modules
  - **QUE, SIO, GIO**

Network Development Kit - NDK

- Platform to develop network enabled applications
- Supports standard socket API, with enhanced no-copy operations for increased performance
- Multicast supported
- Provides network statistics, which aids debugging and performance analysis.
- Well documented API
- Layered architecture separates the OS and Hardware layers – eases porting on different hardware.
- 100 Mbps throughput (with automatic EDMA transfers)

Live Streaming Set-up



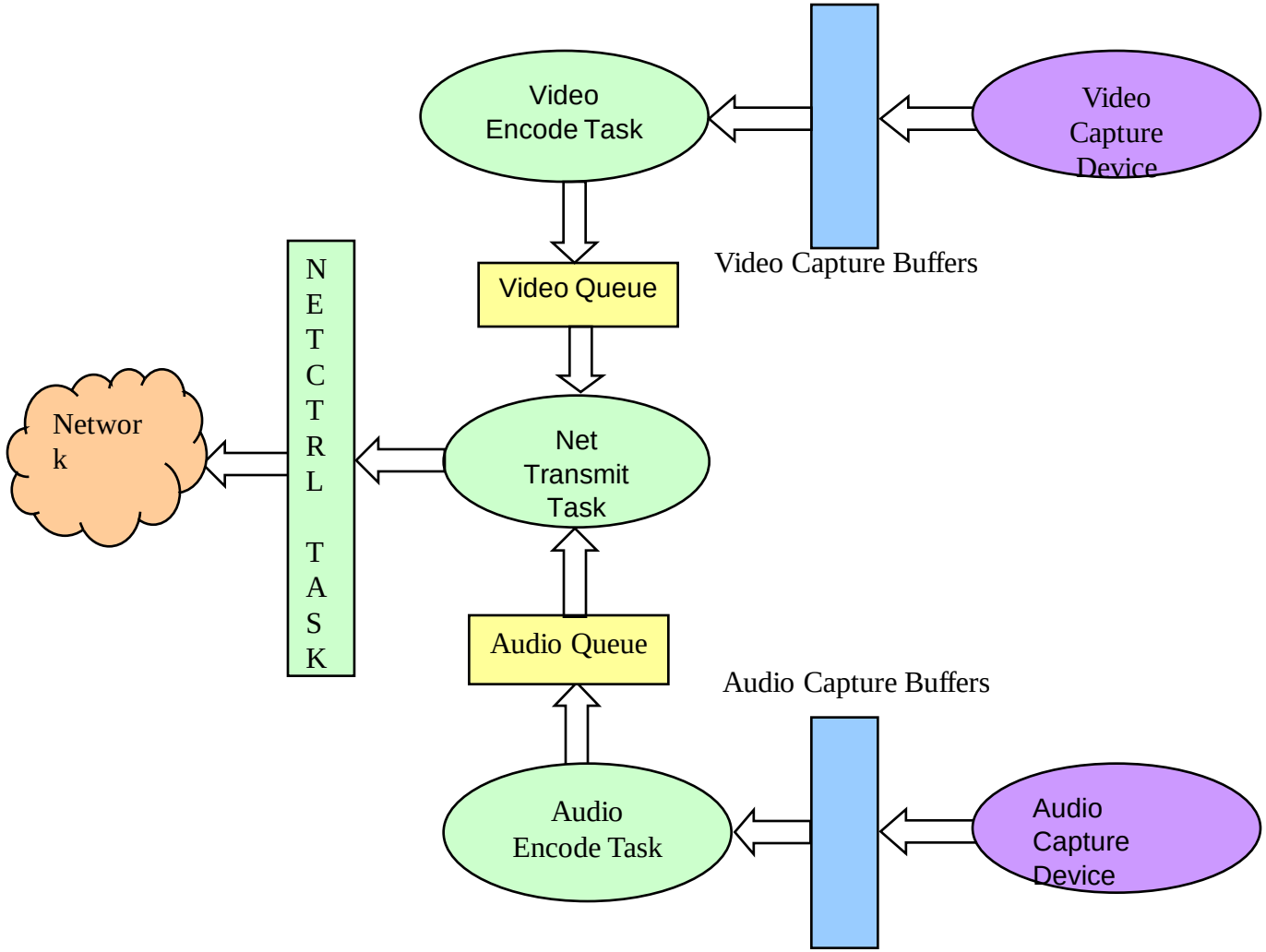
Live Server Framework Requirements

- Components
  - High Quality, full frame-rate, moderate Bit rate MPEG4 compliant Video Encoder
  - High Quality high Sampling-rate AAC compliant Audio Encoder
  - Network thread to packetize and transmit encoded data
- Mechanism to timestamp encoded data to ensure synchronised rendering of media on the client

Live Server Framework Challenges

- Running all the components in the available MCPS budget
- Fitting the memory requirements of both the encoders into the available L2 memory
- Distribution of the EDMA load
  - Video Encoder and video capture require a large DMA bandwidth
  - Transmission of data requires time critical transfers to be carried out
- Scheduling of various threads at the right time
  - Delay in scheduling the encode threads may result in dropping of captured data
  - Delay in scheduling NetTransmit thread may result in overflow of Audio and Video buffers

Live Server Framework



Live Server Framework Solutions

- Three primary threads of operation
  - Audio Encoder task
  - Video Encoder Task
  - NetTransmit task
- Task Scheduling
  - Scheduling of the encoder tasks are governed by availability of captured data
  - Buffers designed to handle worst case execution times of the encoders
  - NetTransmit task scheduled at regular intervals
  - A fixed percentage of CPU time provided to an external command and control (CAC) task.
  - The CAC task controls tasks by posting messages to individual task mailboxes

Live Server Framework Solutions

- **Timestamping of encoded data**
  - **Time stamps derived using on-chip timers associated with media data at the time of capture in the respective drivers**
  - **Real-time clock packets included in transmission to enable drift free AV sync in Client**
- **Memory/MCPS Management**
  - **Usage of scratch buffers in L2 to minimise usage of external memory by the encoders**
  - **Sharing L2 between the encoding threads**
    - **Reusing scratch memory of the encoders**
    - **Dynamic overlay of code with data**

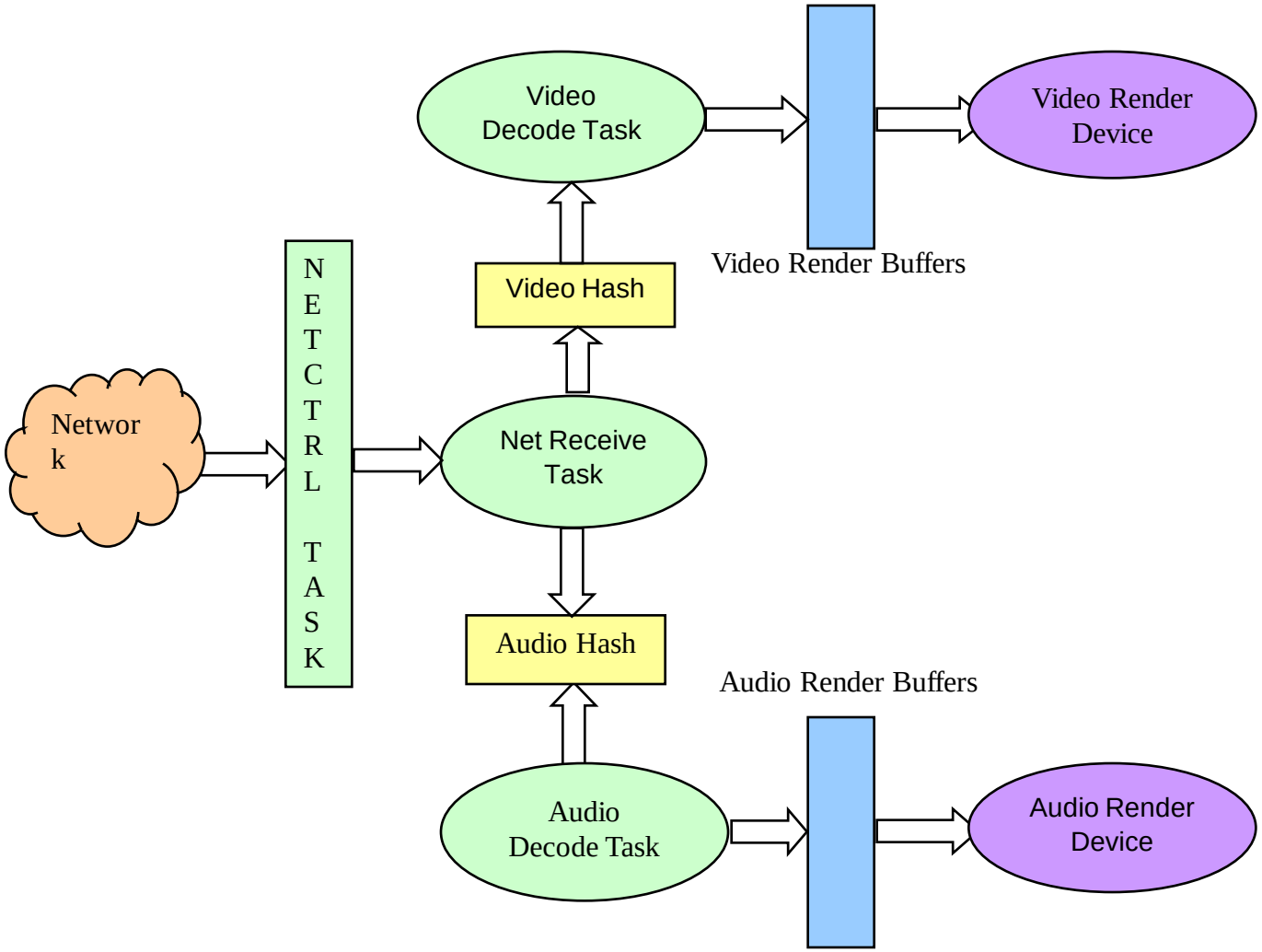
Client Framework Requirements

- Components
  - Video Decoder
  - Audio Decoder
  - Network Controller to receive data
  - User Interface
- Support to operate Audio client and video client on separate h/w units
  - Drifts between the client and server clocks can result in noticeable loss in sync over time
  - Drift can cause data to be consumed at a rate different from the rate of production (at the server) – Leads to overflow/ underflow of buffers

Client Framework Challenges

- Handling of packets arriving out of order
- Prioritizing various data transfers required by different tasks
  - Video Decoder requires a large DMA bandwidth
  - Transferring data received on the network and rendering the media data requires time critical data transfers

Client Framework



Client Framework Solutions

- **Three threads of operation**
  - **Audio Decoder task – Decodes and queues audio data for rendering**
  - **Video Decoder task - Decodes and queues video data for rendering**
  - **Network Receive Task – Receives data on the network, reorders received data, partitions data into decodable units**

Client Framework Solutions

- **Task Scheduling**
  - Scheduling of the Decoder tasks are governed by availability of rendering buffers to hold decoded data
  - No coupling (using semaphores etc) between the Audio and video decoder tasks – enables audio–only and video-only modes of operation
  - Buffers designed to handle worst case execution times of the Decoders
  - NetRecv task scheduled at regular intervals
  - A fixed percentage of CPU time provided to an external command and control(CAC) task.
  - The CAC task controls the other tasks by posting messages to individual task mailboxes

Client Framework Solutions

- **Management of EDMA resources**
  - **Usage of QDMA for short bursts within decoders**
  - **Distribute the EDMA load across queues to minimize stalls and priority inversion**
    - **for instance, requests from network controller and video display drivers should be on separate EDMA queues**

Client Framework - Drift Free AV Sync

- AV Sync Mechanism
  - The system time serves as the master clock.
  - Both the decoder tasks queue decoded data for rendering according to presentation time stamp and the current system time
- Real-time clock packets used to handle drift between the server and the client clocks
  - RTC packet used to update the client's system time wrt the server time
  - When the client's system time lags that indicated by RTC, the client catches up by not presenting a few decoded data units
  - When the client's system time is ahead of that indicated by RTC, the client slows down by inserting silence(Audio) or holding a frame (video)

Media Streamer & Distribution System

Video Input from a DVD Player

DM642 running **Ittiam MPEG-4 Video + AAC Encoder** & **Multicast Streaming Server**

Network

DM642 running **Ittiam MPEG-4 Video + AAC Decoder** & **Streaming Client**

TV Display

Bhavani GK
Senior Engineer
Ittiam Systems
bhavani.gk@ittiam.com

Get to market faster with TI products, support and partners.

SEE THE **FUTURE.**
CREATE YOUR OWN.

Ittiam