Tapping into the Fountain of CPUS- On Operating System Support for Programmable Devices

Yaron Weinsberg[1], Danny Dolev[1], Tal Anker[1],
Muli Ben-Yehuda[2] and Pete Wyckoff[3]

muli@il.ibm.com

[1] The Hebrew University Of Jerusalem (HUJI)
[2] IBM Haifa Research Lab
[3] Ohio Supercomputer Center

The Elevator Pitch

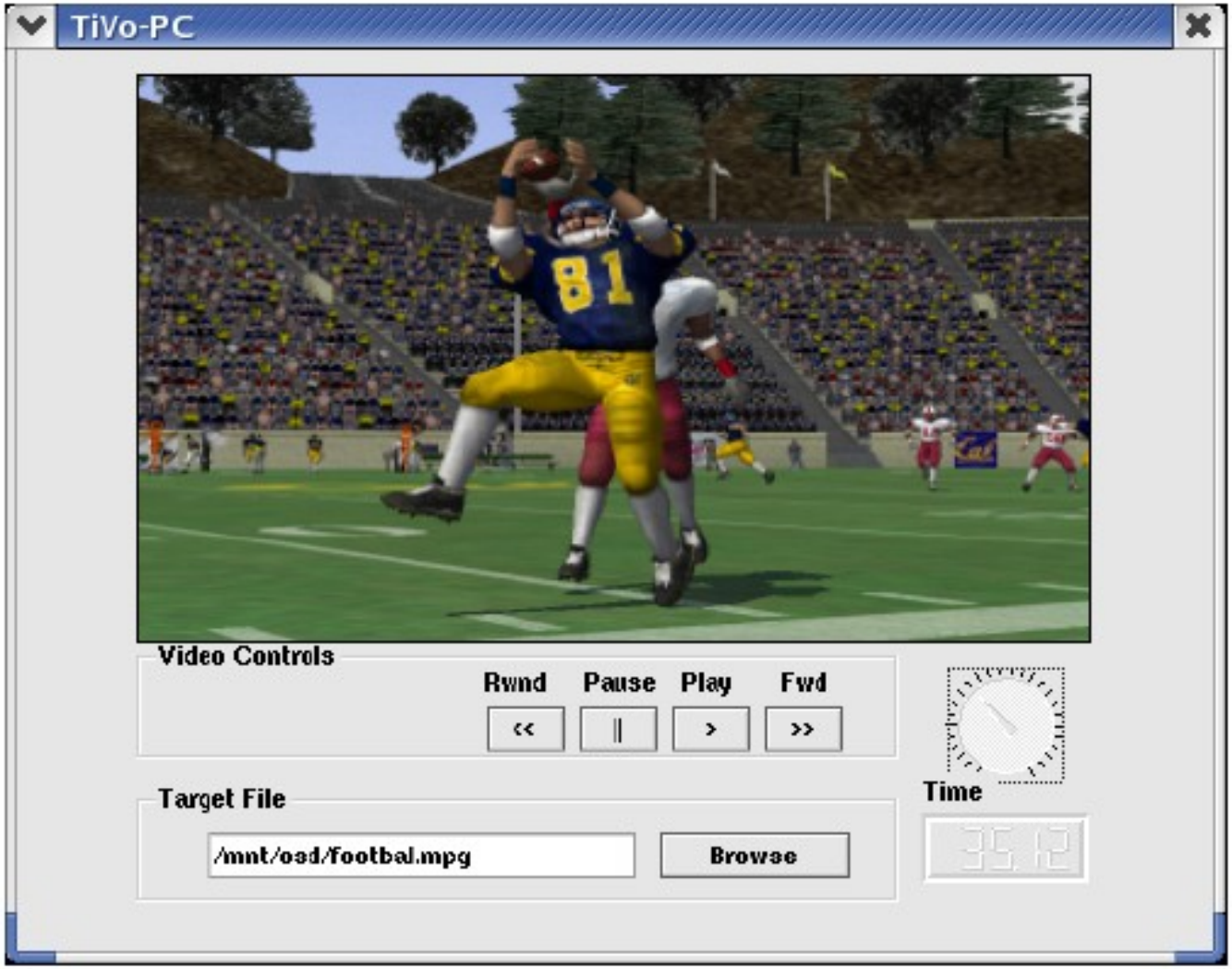
- Today's peripheral devices are *very* powerful
  - Contain general purpose CPUs, memory, specialized hardware
  - Programmable - more flexible than ASIC solutions
- Can we use them in order to execute parts of our OS and user-level applications?
  - Yes, but...
  - There is no generic framework that enables this...

The Elevator Pitch

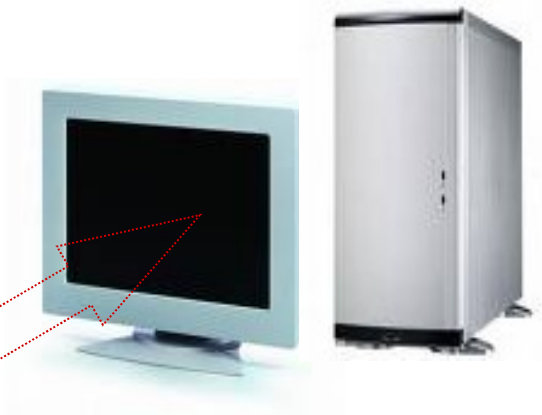
- Hydra is a generic “offloading framework”
  - Provides a programming model and runtime support that enables one to develop *Offload-Aware (OA) Applications*
  - “Aware” of any available (programmable) computing resource
- Enables a developer to define the offloading aspects of the application during design time

“TivoPC”

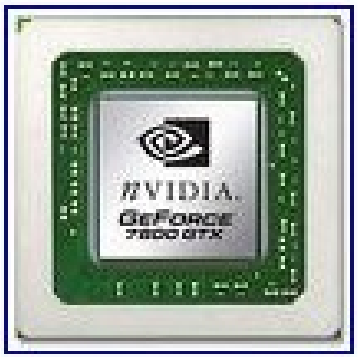
You can now compile your kernel while watching the Superbowl…



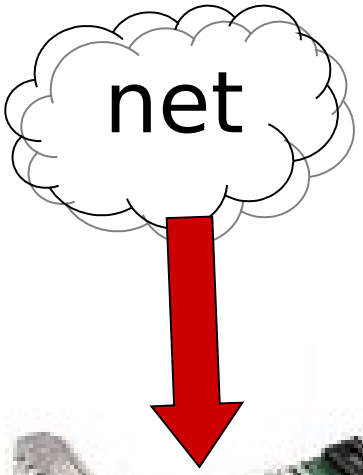
TivoPC Information Flow



GPU



net



NIC

Disk Controller

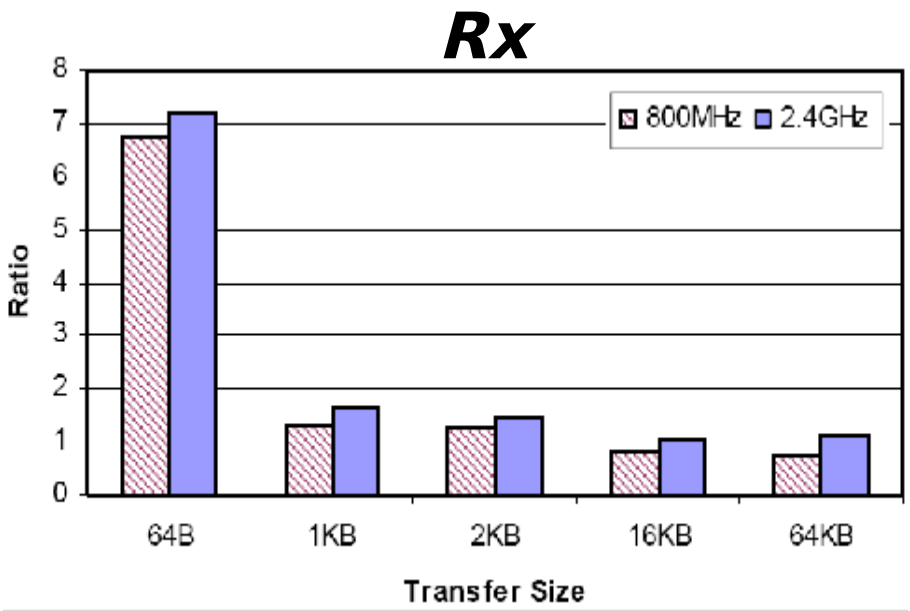
Why should we deal with offloading when a typical host is full of CPUs ?

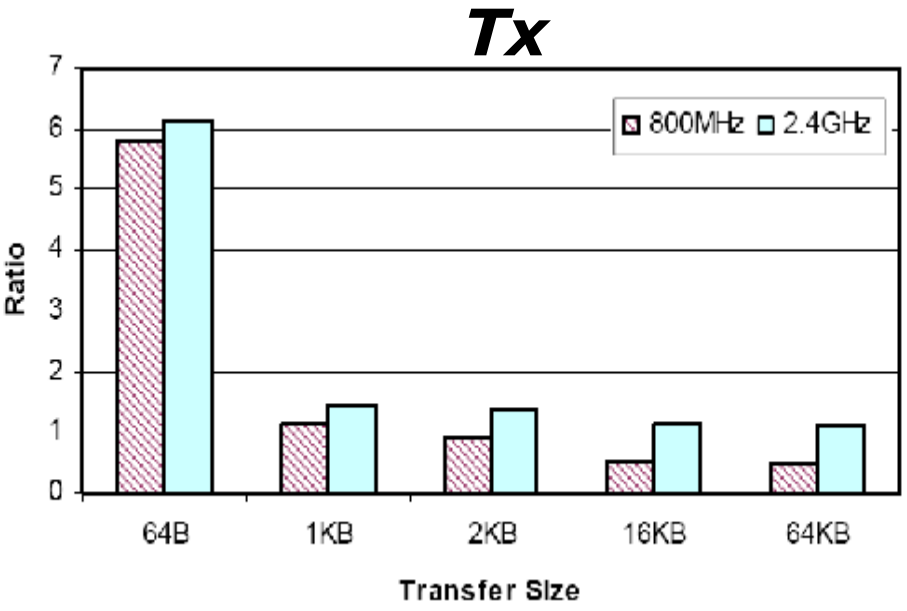
Reasons for Offloading

- **Memory Bottlenecks**
  - reduce memory pressure and cache-misses (due to filtering done at the device)
- **Timeliness guarantees**
  - GPOS ↔ Embedded OS (RTOS)
  - avoiding “OS noise” (interrupts, context switches, timers etc.)
- **Reduced power consumption**
  - Pentium 4 2.8Ghz: 68Watt
  - Intel XScale 600Mhz: 0.5Watt

Reasons for Offloading

- **Security**
  - harder to attack
- **Increased Throughput**





*The graphs appear in the paper: “TCP performance revisited”by Foong et. al. (ISPASS’03) and are used with the authors‘ permission.

Outline

- Motivation
- HYDRA Programming Model
- HYDRA Architecture
- Evaluation
- Future Work

The Current Gap

- **Not** many applications **DO** take advantage of the available processing power…
- Using programmable devices has traditionally been very difficult:
  - Requires experienced embedded engineers
  - Requires customization of each particular design for each peripheral device

→ **HYDRA to the rescue…**

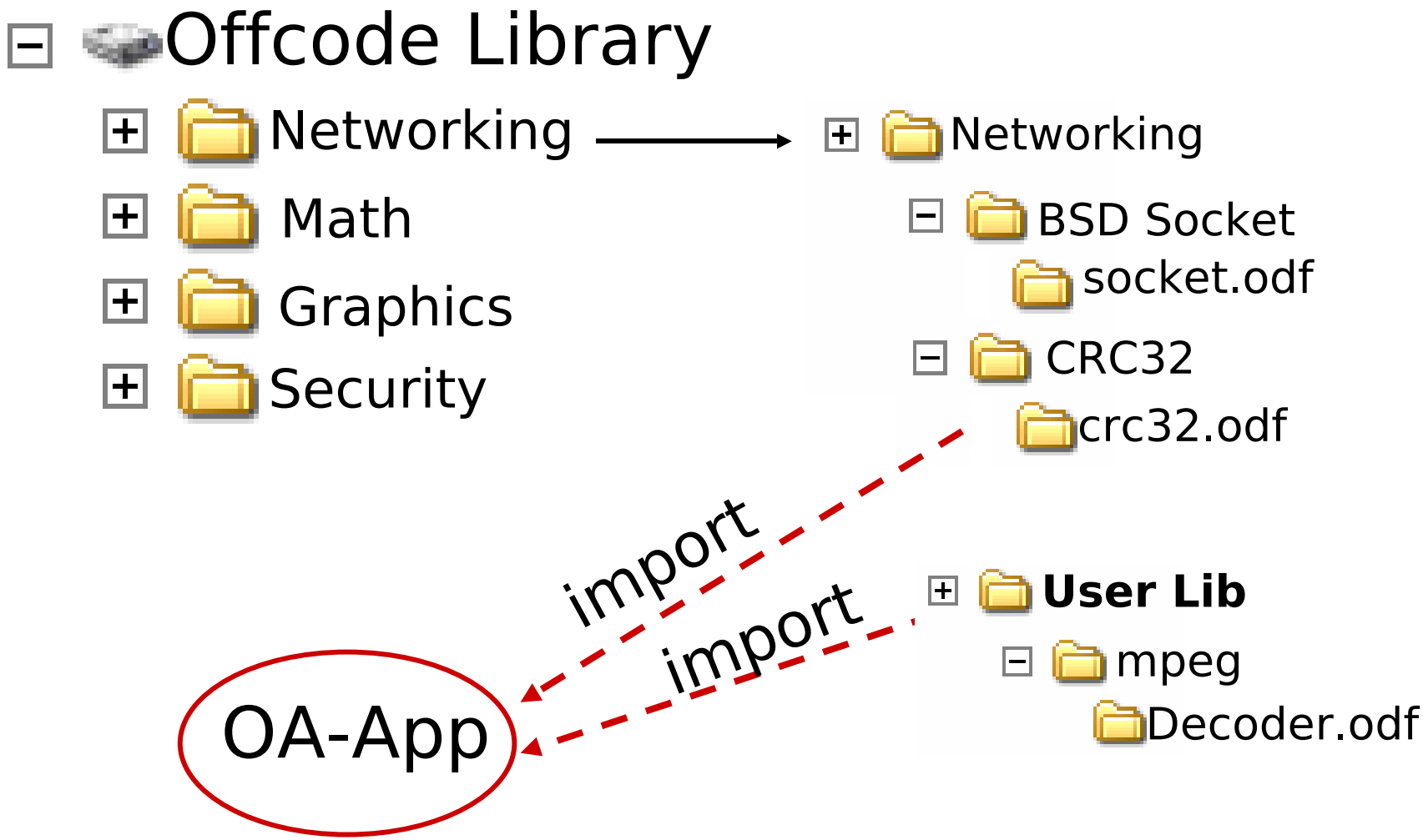
Outline

- Motivation
- HYDRA Programming Model
- HYDRA Architecture
- Evaluation
- Future Work

HYDRA Programming Model

- Hydra defines "*Offcodes*" (Offloaded-Code)
  - The minimal unit for offloading
  - Exports a well defined interface (like COM objects)
  - Given as open source or as compiled binaries
  - Described by Offcode Description File (ODF)
    - Exposes the offcode's functionality (interfaces)
    - Defined the offcode's dependencies

Offcode Libraries

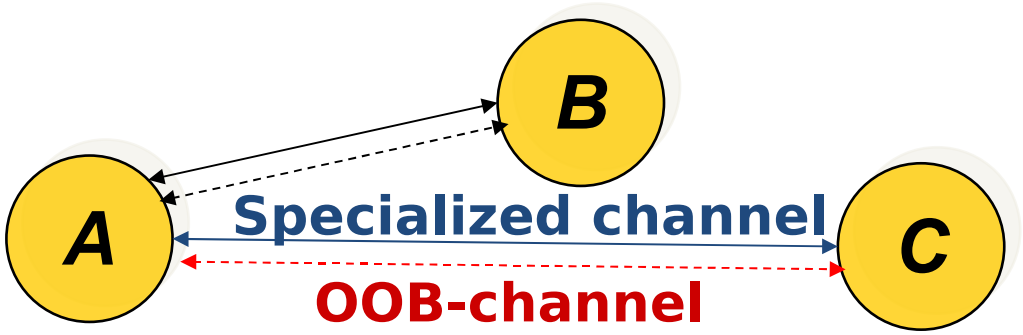


Channels

- Offcodes are interconnected via *Channels*
  - Determines various communication properties between offcodes

(I) An Out-Of-Band Channel, *OOB-channel*, is attached to every OA-application and Offcode

- Not performance critical (uses memory copies)
- Used for initialization, control and events dissemination



Channels

(II) A specialized channel is created for performance critical communication

- Hydra provides several channel types:
  - Unicast / Multicast
  - Reliable / Unreliable
  - Synchronized / Asynchronous
  - Buffered / Zero-Copy R/W/Both

Design Methodology

- OA-applications are designed by two orthogonal aspects:
  1. Basic logic design:

     Design the application logic and define the components to be offloaded

  2. Offloading Layout design:

     Define the channels of communication between offcodes and their location constraints

1. Logical Design

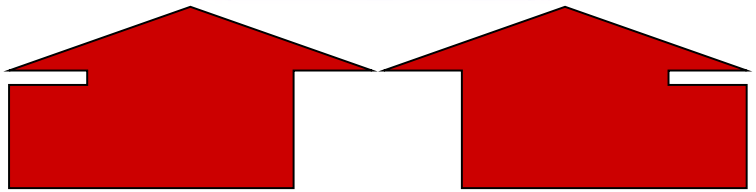
Component	Description
GUI	Provides the viewing area and user controls (play, pause, rewind and resume)
Streamer	Process the media stream (either from network or storage)
Decoder	Decodes the MPEG stream
Display	Displays the movie on screen
File	Reads/Writes data from storage

2. Offloading Layout Design

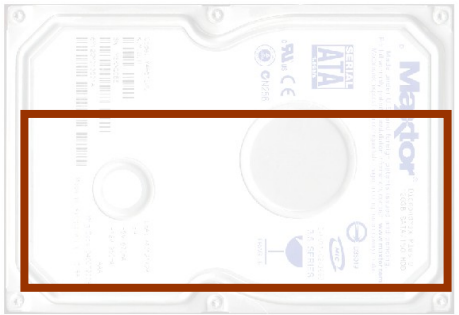
- ***GUI***
- ***Streamer***
- ***File***
- ***Display***
- ***Decoder***

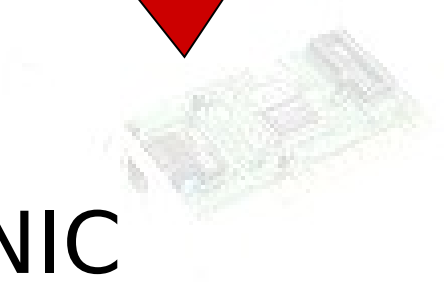
CPU

GPU



net

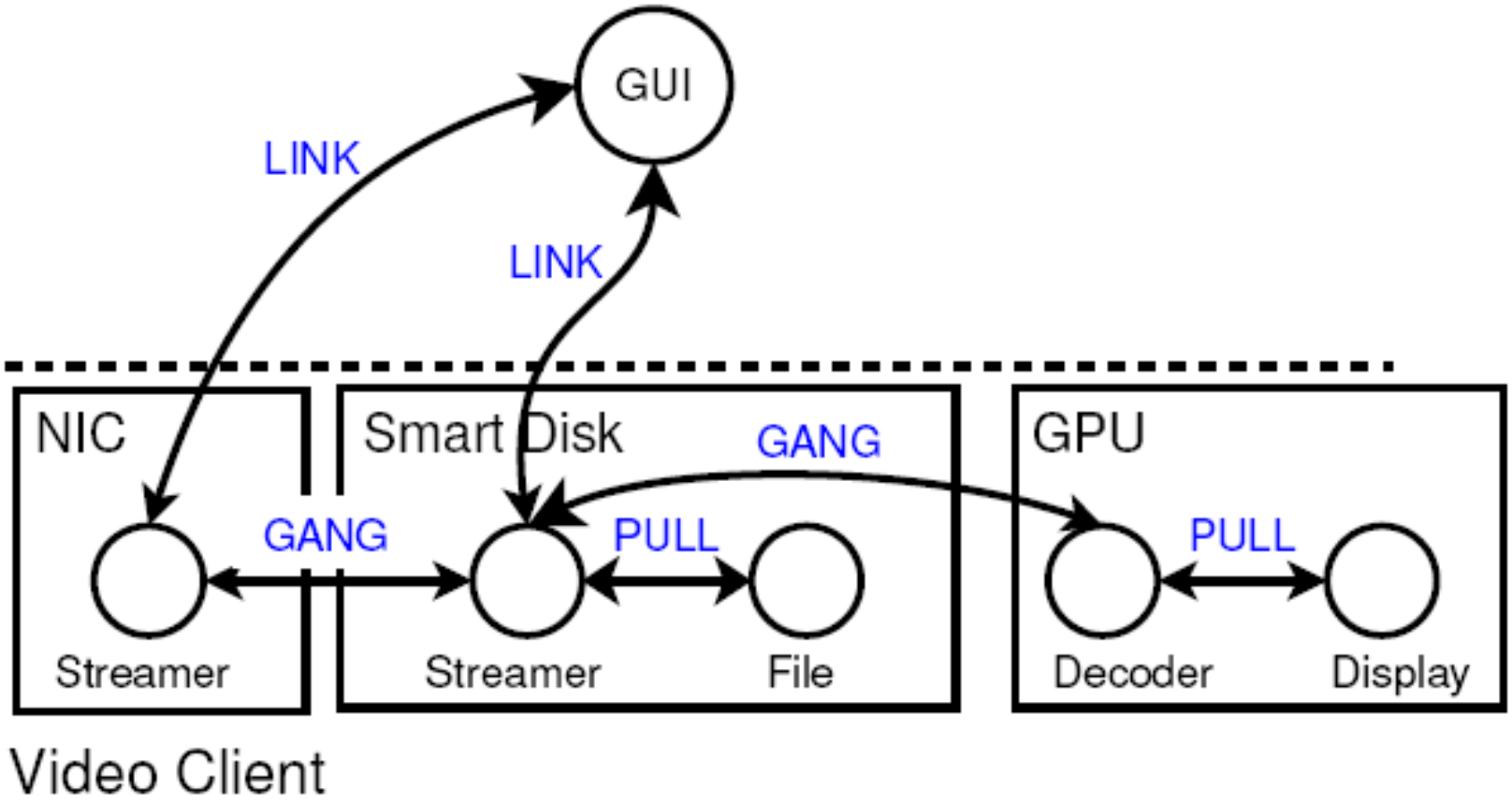




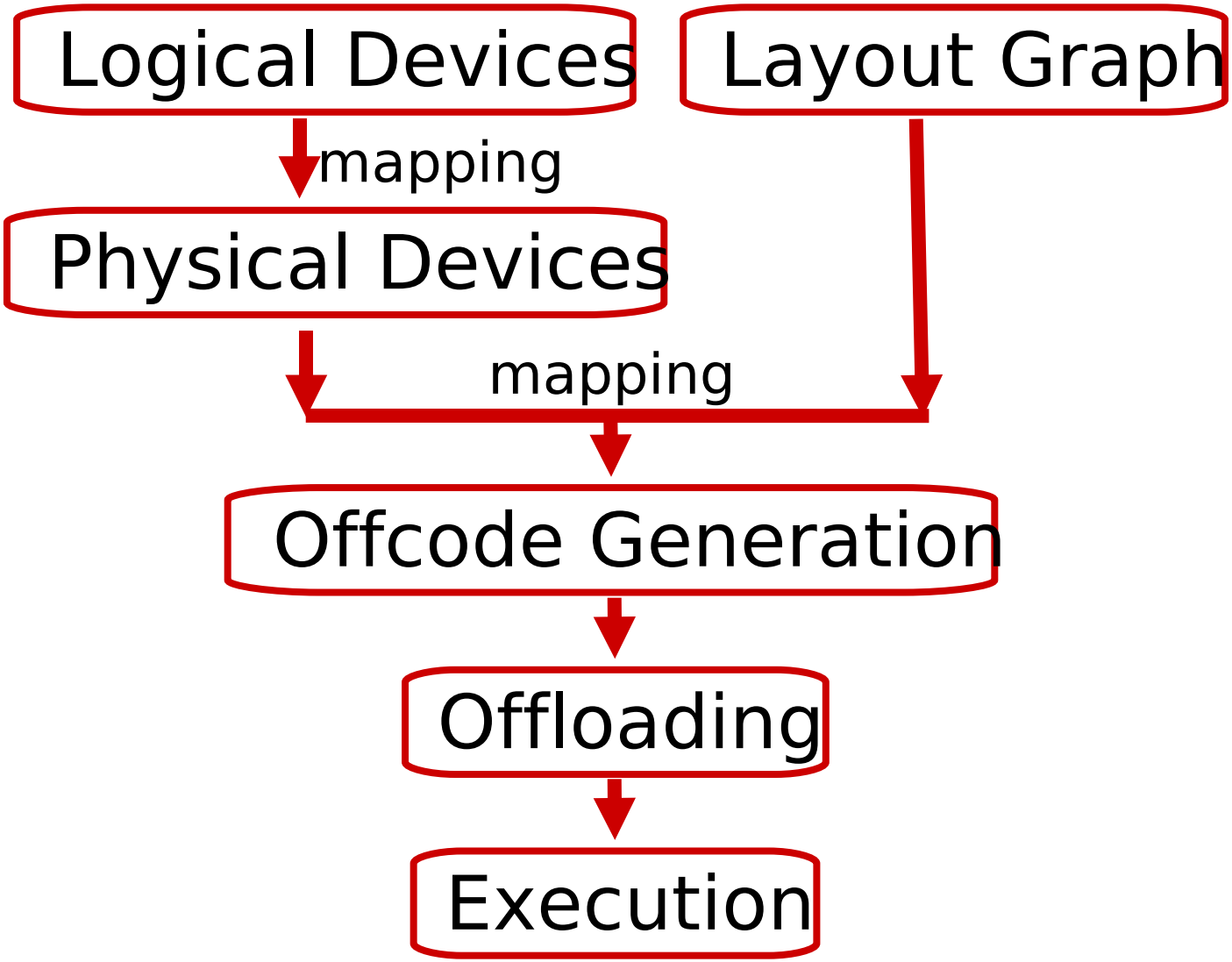
NIC

Disk Controller

TivoPC – Layout Graph



Finally: Application Deployment



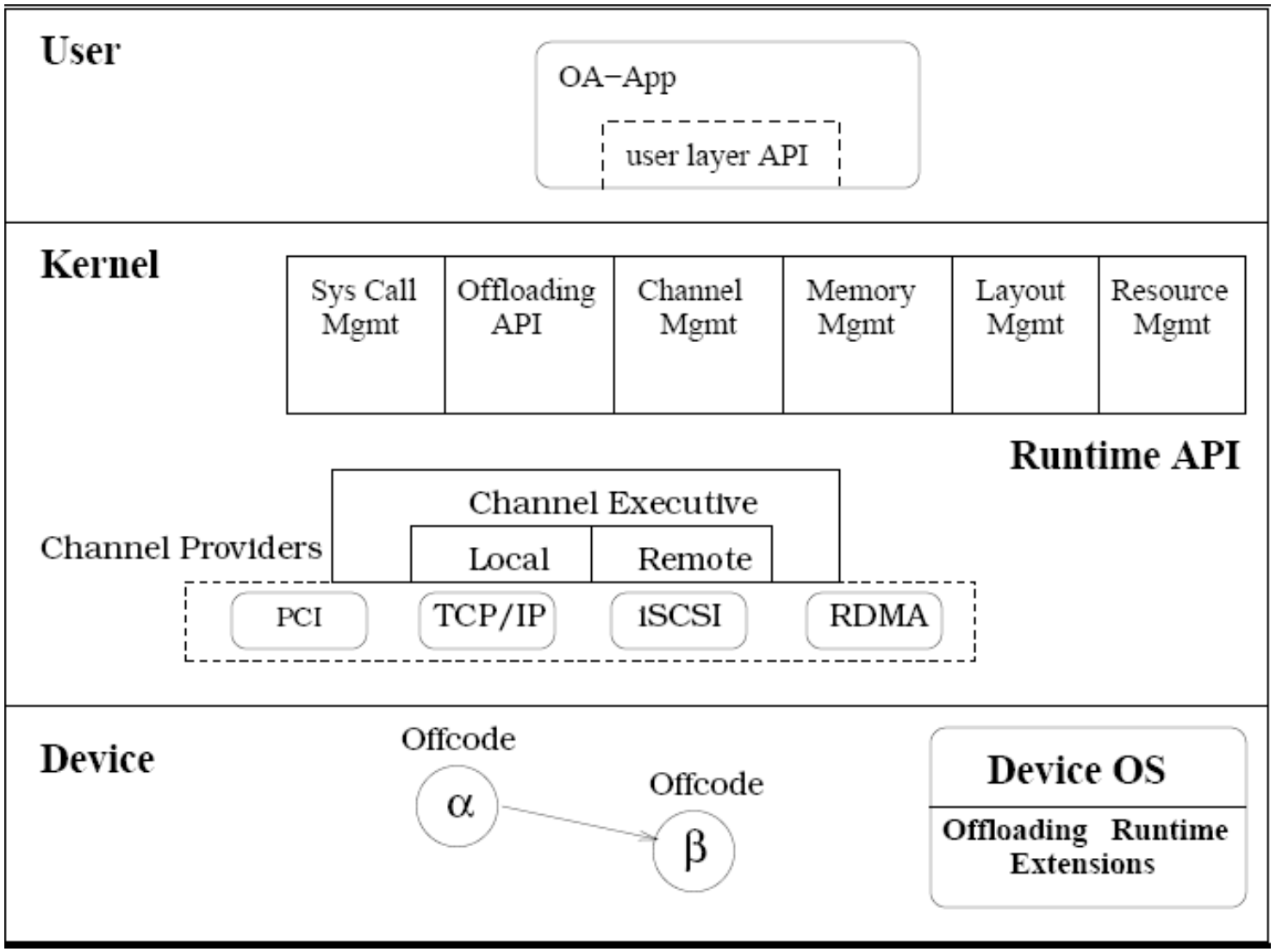
Outline

- Motivation
- HYDRA Programming Model
- HYDRA Architecture
- Evaluation
- Future Work

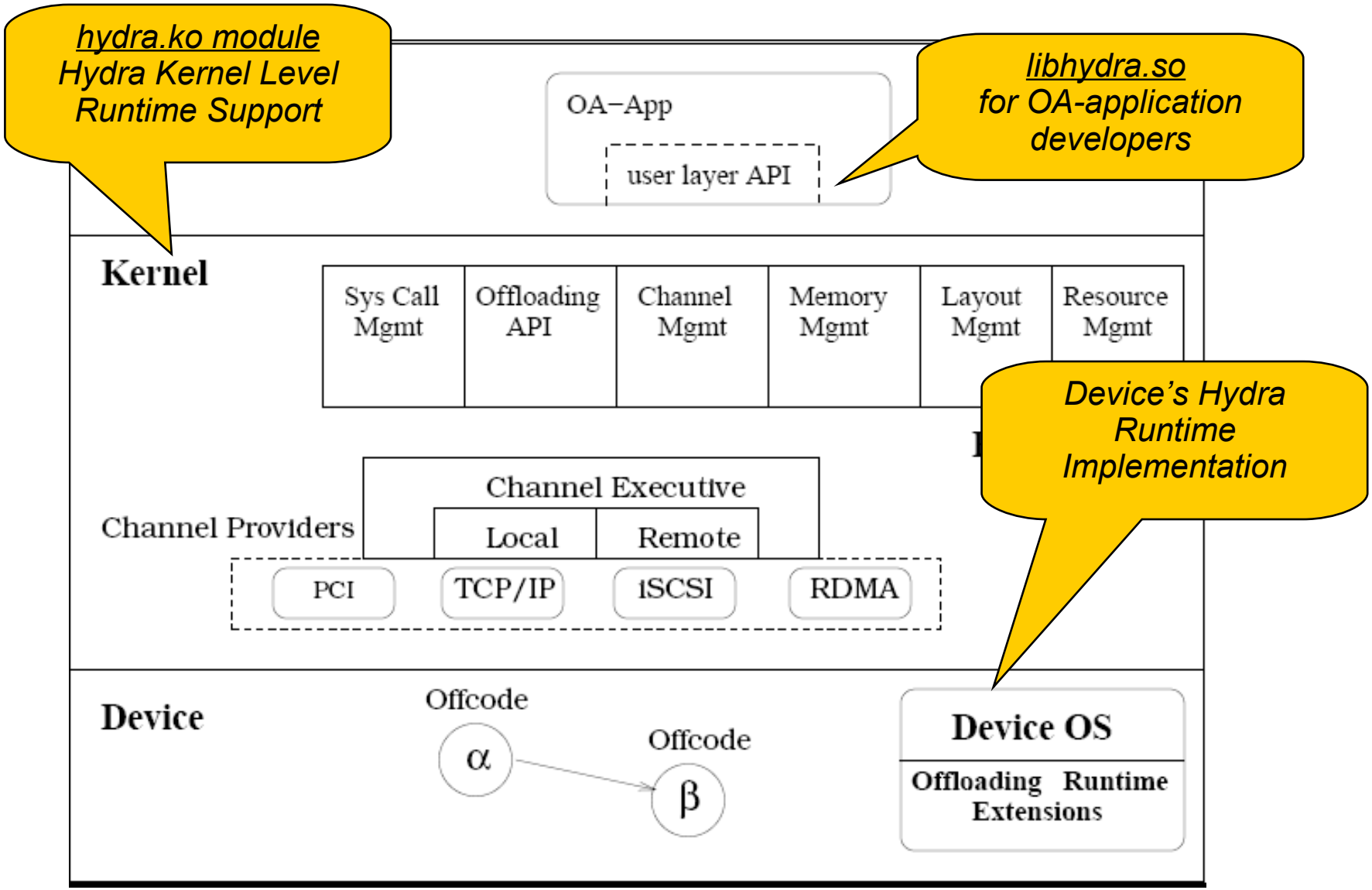
HYDRA Architecture

- The runtime system implements the programming model
- Both the host OS and target devices must implement the runtime functionality

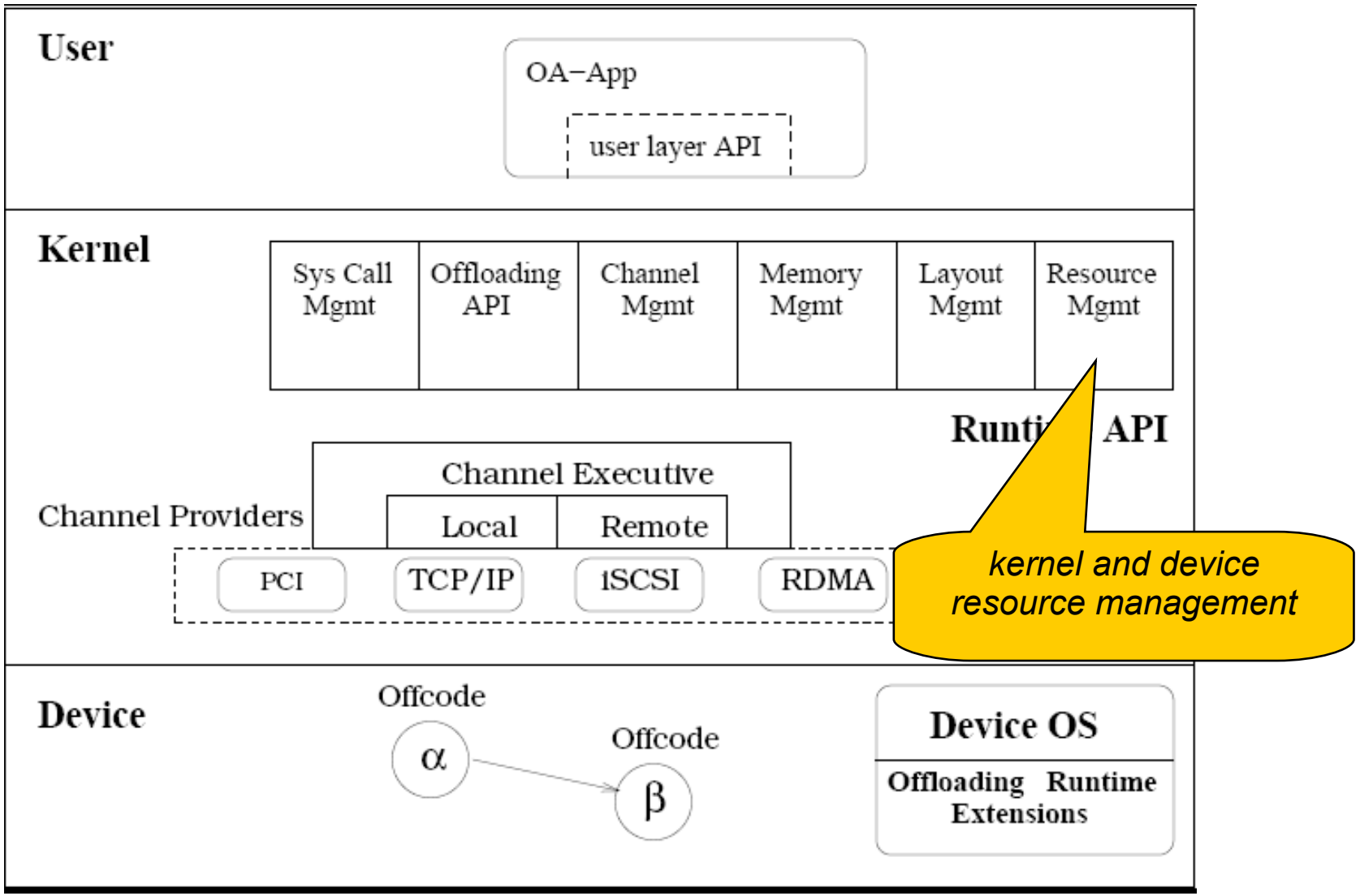
HYDRA Architecture



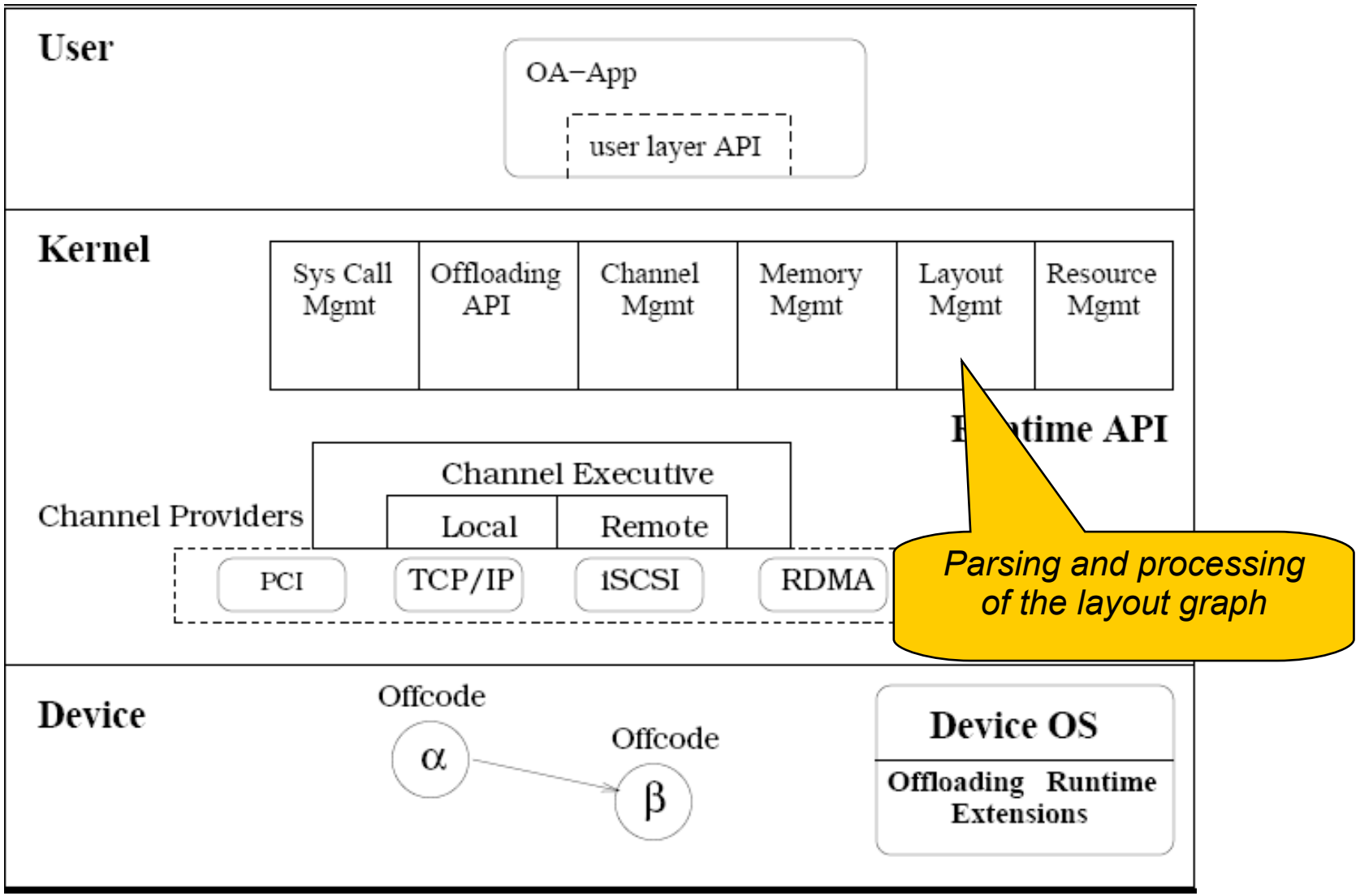
HYDRA Architecture



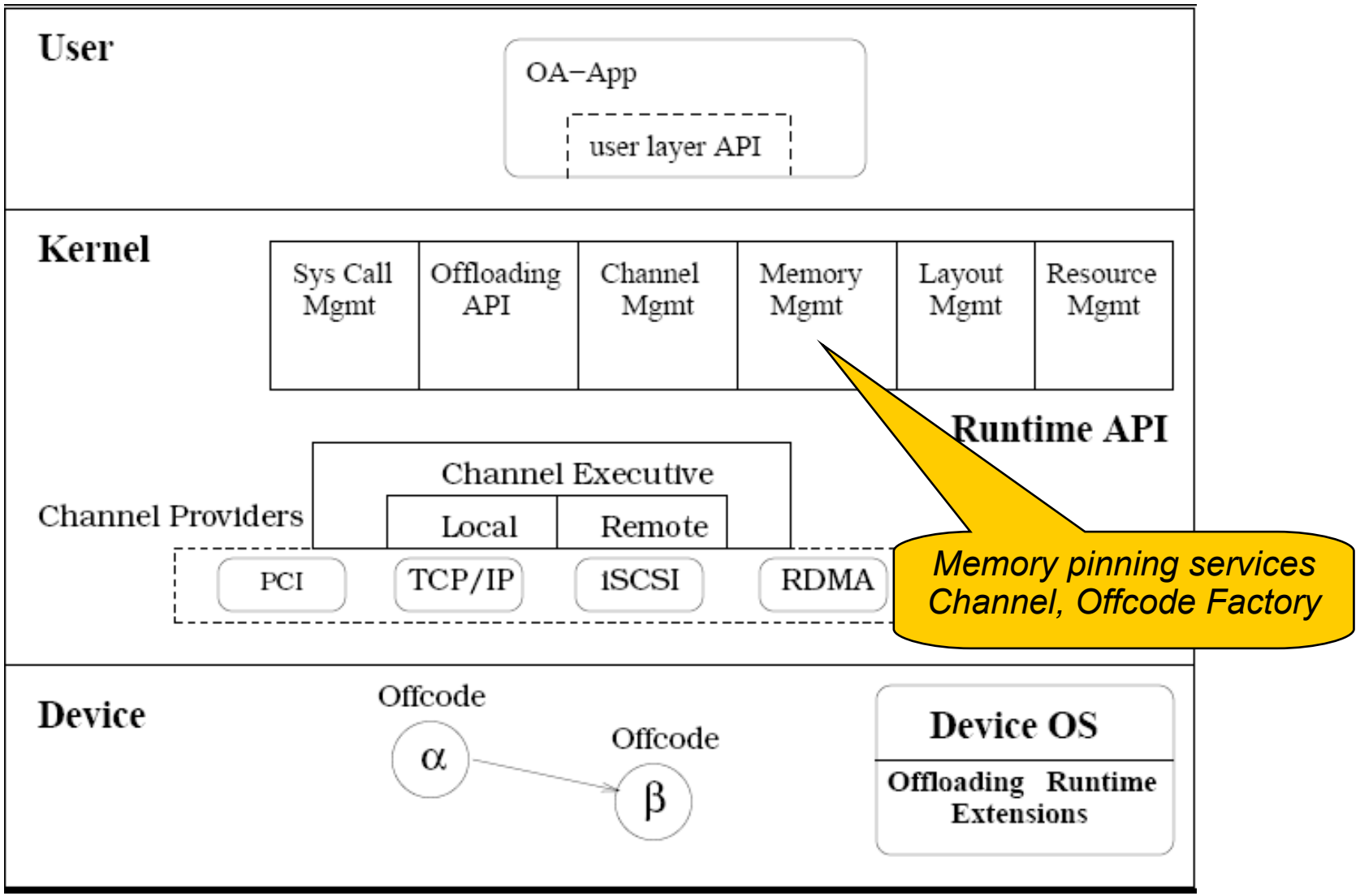
HYDRA Architecture



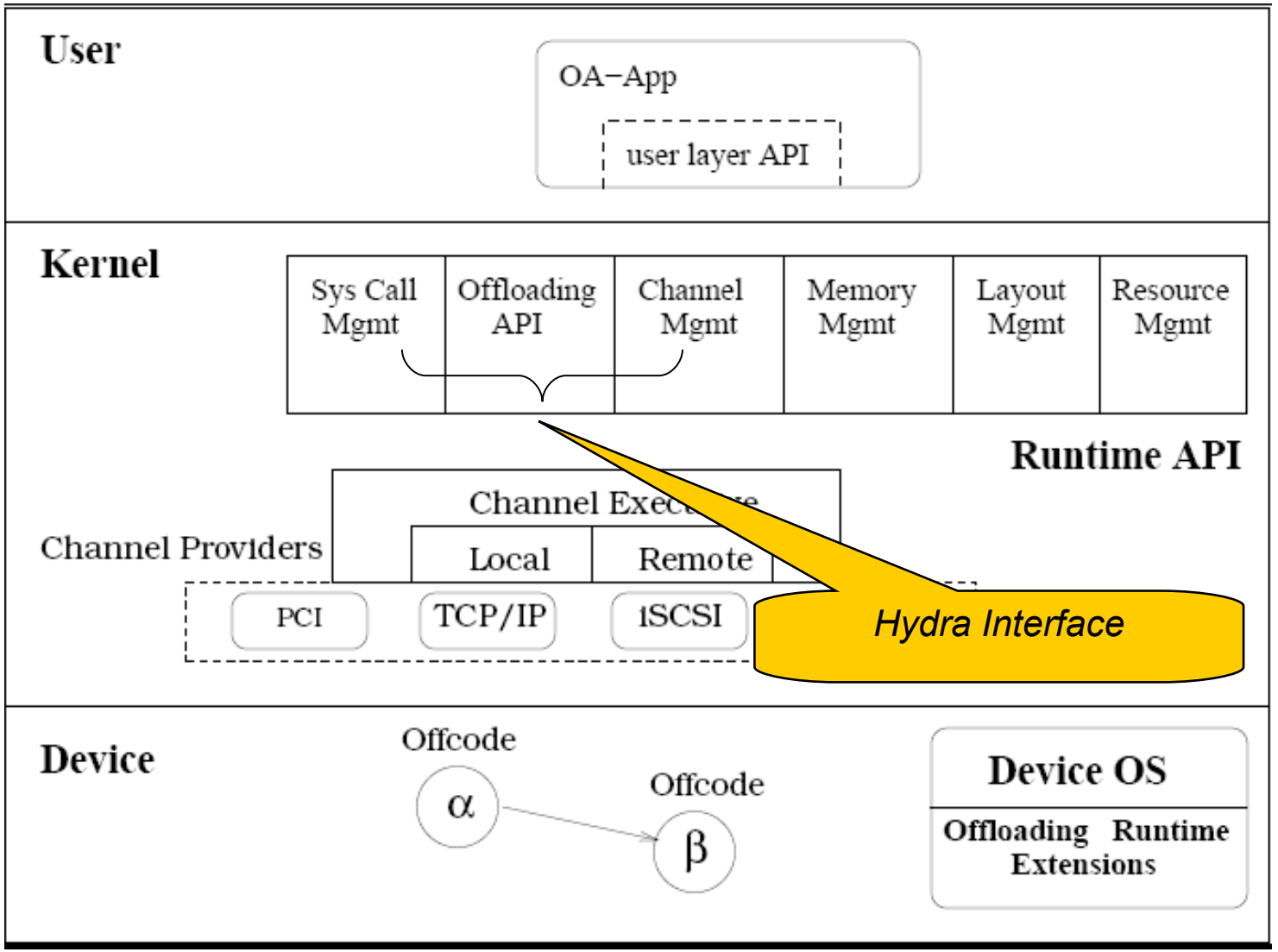
HYDRA Architecture



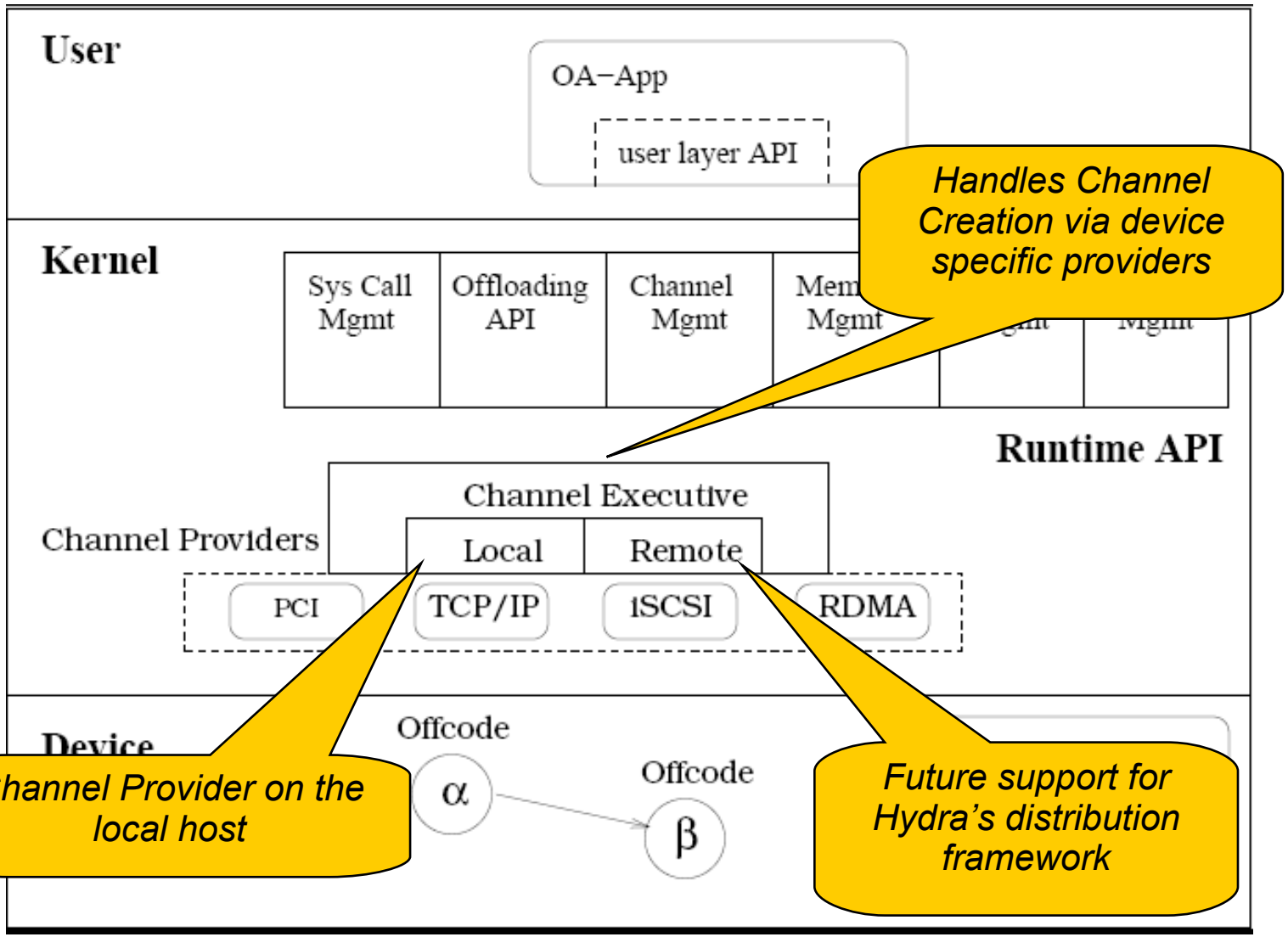
HYDRA Architecture



HYDRA Architecture



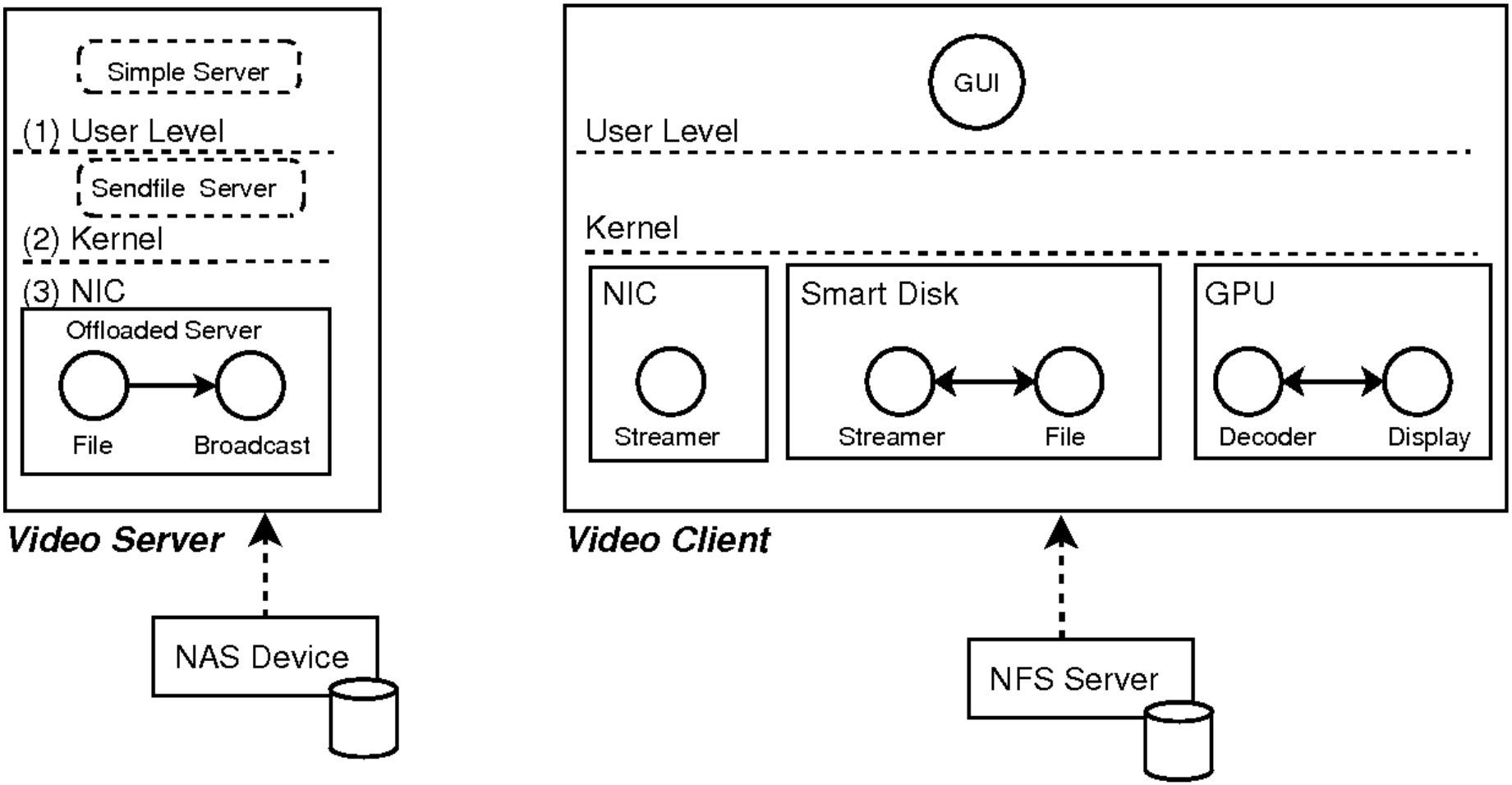
HYDRA Architecture



Outline

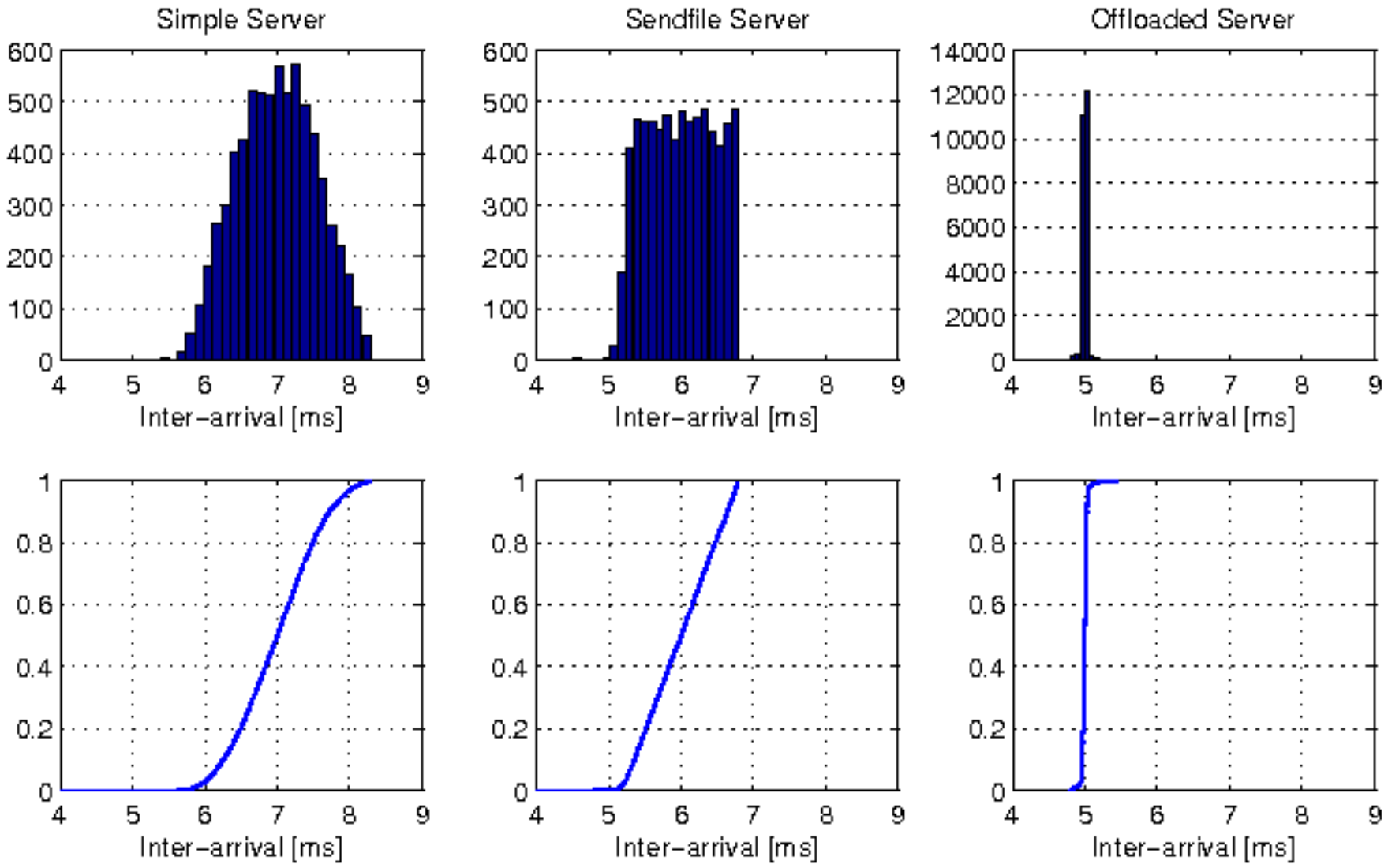
- Motivation
- HYDRA Programming Model
- HYDRA Architecture
- Evaluation
- Future Work

Evaluation – TiVo-PC

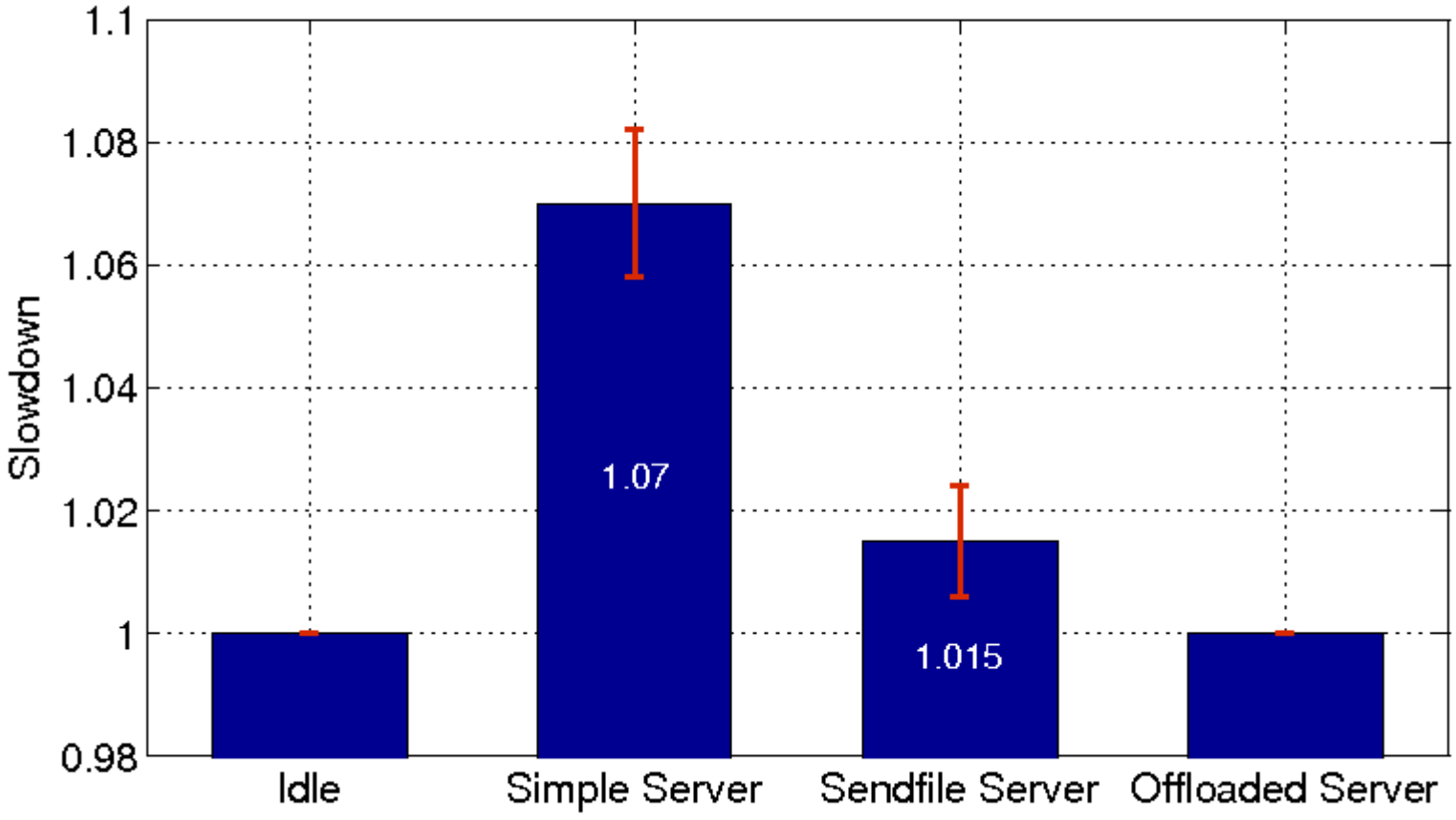


- **The server streams 1KB packets, every 5 msec (200KB MPEG movie)**

Evaluation - TiVo-PC
Packets Jitter (at the video client)



Evaluation - TiVo-PC
L2 Cache Miss Ratio (Server)



Outline

- Motivation
- HYDRA Programming Model
- HYDRA Architecture
- Evaluation
- Future Work

Future Directions

- OS Offloading
  - File system (NFS, indexing, caching, buffer cache…)
  - Device drivers offload
- Multi-core support
  - CMPs, SMPs
- Security
  - RNGs, En/Decryption, tripwire, IDS/IPS, firewall
- I/O for virtualized systems, IOMMUs, pinning

Thanks!