Towards improving the efficiency of ICN packet-caches

Yannis Thomas and George Xylomenos

Mobile Multimedia Laboratory, Department of Informatics

School of Information Sciences and Technology

Athens University of Economics and Business, Greece

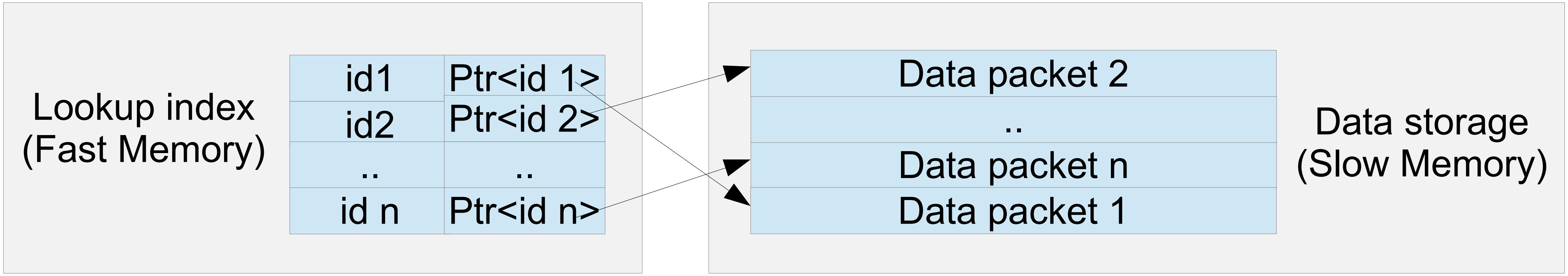
Email: thomasi,xgeorge@aueb.gr

Web caching in ICN

- ICN is a novel network architecture which offers in-network packet-level caching as a native feature.
  - Data packets are self-verified (redundancy detection at packet level)
  - Routers exploit their queuing buffers as caching units (seamless in-network caching)
  - Receiver-driven communication model with one-to-one request-response correlation (supporting off-path and on-path caching)

A common packet-cache module

- Uses both the router's fast (SRAM) and slower memory of the device
- The lookup index is a HashTable stored in the SRAM with the:
  - *key* being the packet's identifier
  - *value* being a pointer to the actual data in the slower memory
- Usually exploit the LRU replacement policy at packet level



Lookup index
(Fast Memory)

id1	Ptr<id 1>
id2	Ptr<id 2>
..	..
id n	Ptr<id n>

Data packet 2
..
Data packet n
Data packet 1

Data storage
(Slow Memory)

Issues of ICN packet-caches

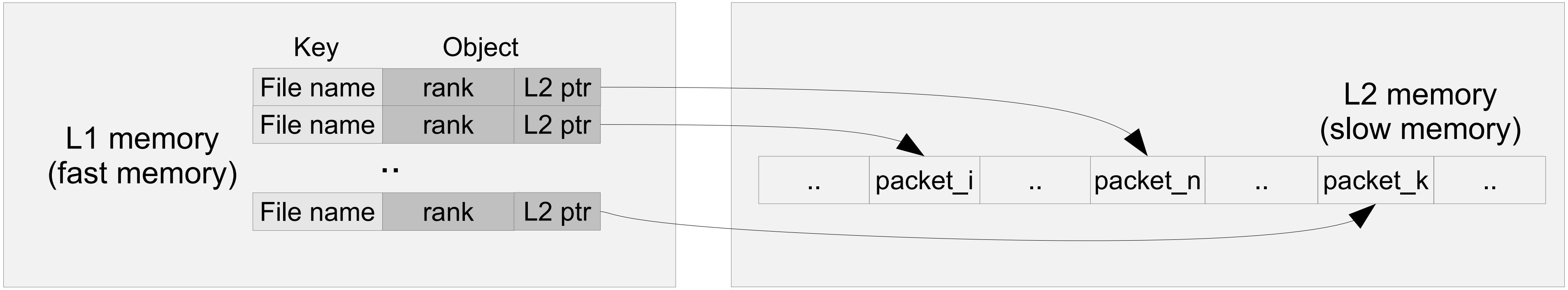
- Huge lookup indexes (*1 entry per packet*) mitigate caching dynamic
  - Each entry in SRAM costs 48bytes and maps to just 9000bytes (Ethernet jumbo frame)
- Small cache size causes poor hit-ratio
  - The largest single-chip SRAM can hold 72Mbit, that is less than 300K packets (≈2.5Gbytes)

Object-oriented Packet Cache (OPC)

- OPC is a novel management policy for ICN packet-caches.
- Main design goal is the minimization of the indexing costs, that is **shrink the lookup index**
- OPC creates a object-level lookup index (*1 entry per content*) by deploying two layers of operation
  - **L1**: serves as lookup index for all the packets of a content
  - **L2**: allows access to specific packet of a content

OPC layers of operation

- OPC stores only a **compact** part of **consequent** packets of each file, ranging from the **first** packet to the *n*-th packet of that file.
  - The lookup index (L1) maps the content's name with the rank (sequence id) of the **last stored** packet for each file
  - The data storage (L2) stores in order the packets of each content



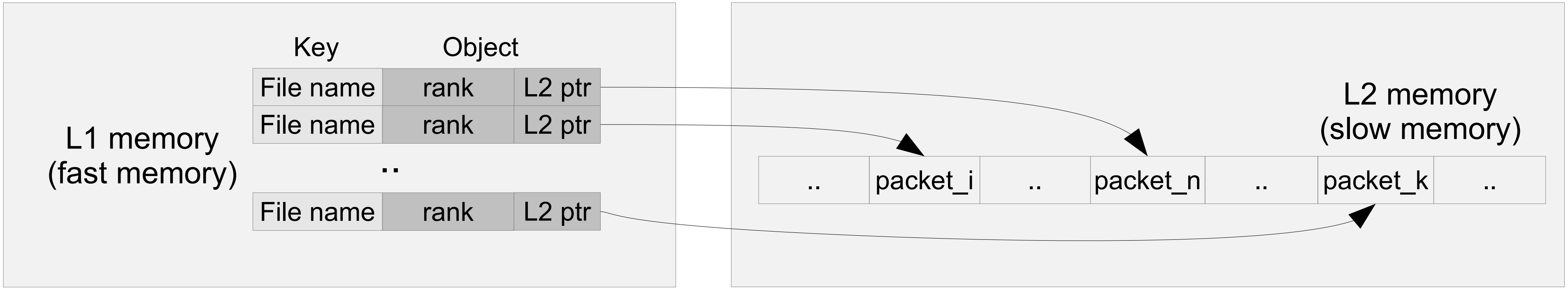
Key	Object	
File name	rank	L2 ptr
File name	rank	L2 ptr
..		
File name	rank	L2 ptr

..	packet_i	..	packet_n	..	packet_k	..

OPC lookup process

Upon the arrival of a request

1. OPC parses the content's identifier (c_id) and the packet's sequence id (seq_id).
2. inquires L1 using *c_id*.
3. If $rank \geq seq\_id$ then the packet is stored in L2 at position $L2\_ptr * seq\_id * packet\_size$



L1 memory
(fast memory)

Key	Object	
File name	rank	L2 ptr
File name	rank	L2 ptr
..		
File name	rank	L2 ptr

L2 memory
(slow memory)

..	packet_i	..	packet_n	..	packet_k	..

OPC replacement process

- Insertion policy
  - Insert an arrived packet only when
    - Its the **first** packet of the content, OR
    - The **previous** packet is also inserted
- Eviction policy
  - Remove the last packet of the least recently used* content

LRU management at content-level

LIFO management at packet-level

*Currently using Least Recently Used (LRU) file organization but could be anything.

Thank you!