Minimizing Delay in Shared Pipelines

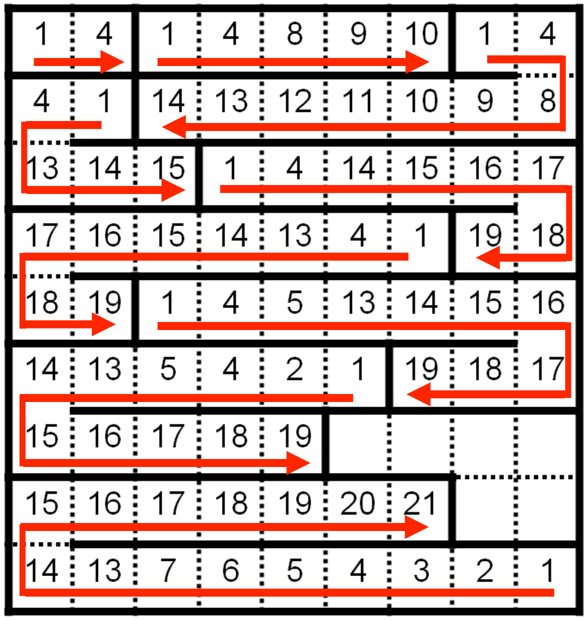
Ori Rottenstreich (Technion, Israel)

Joint work with

Isaac Keslassy (Technion, Israel)

Yoram Revah, Aviran Kadosh (Marvell Israel)

Pipeline Sharing for H.264 video-compression standard

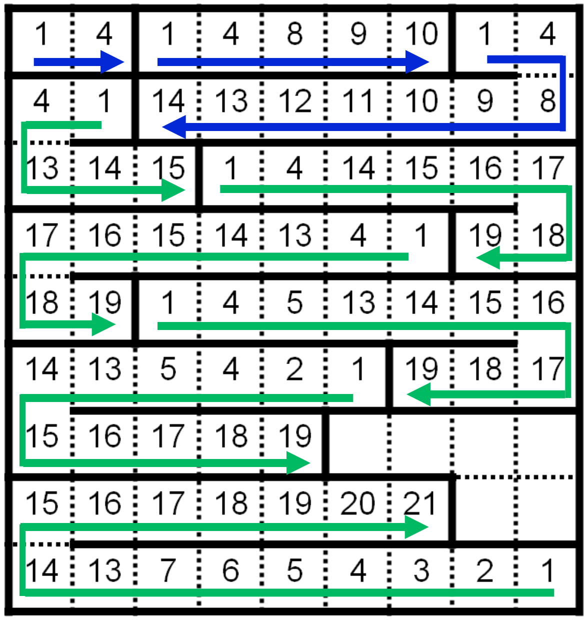


1	4	1	4	8	9	10	1	4
4	1	14	13	12	11	10	9	8
13	14	15	1	4	14	15	16	17
17	16	15	14	13	4	1	19	18
18	19	1	4	5	13	14	15	16
14	13	5	4	2	1	19	18	17
15	16	17	18	19				
15	16	17	18	19	20	21		
14	13	7	6	5	4	3	2	1

9 profiles with 21 features are organized in 9 pipelines in a 9x9 multicore

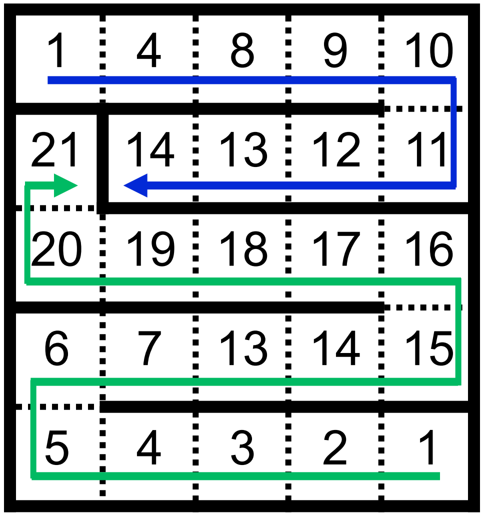
scalability problem:
more profiles, more features
→ more cores

Pipeline Sharing for H.264 video-compression standard



1	4	1	4	8	9	10	1	4
4	1	14	13	12	11	10	9	8
13	14	15	1	4	14	15	16	17
17	16	15	14	13	4	1	19	18
18	19	1	4	5	13	14	15	16
14	13	5	4	2	1	19	18	17
15	16	17	18	19				
15	16	17	18	19	20	21		
14	13	7	6	5	4	3	2	1

9 profiles with 21 features are organized in 9 pipelines in a 9x9 multicore

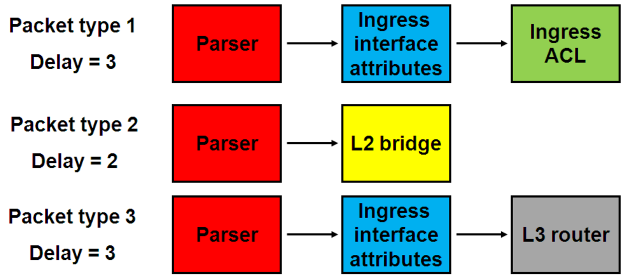


1	4	8	9	10
21	14	13	12	11
20	19	18	17	16
6	7	13	14	15
5	4	3	2	1

Pipeline Sharing: The 9 profiles are serviced by one of two possible pipelines in a smaller 5x5 multicore

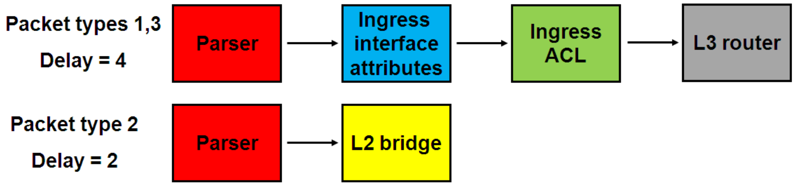
Another
Pipeline Sharing Example

- *Regular architecture, without pipeline sharing:*



  - ○ 3 pipelines for the *k*=3 (uniformly distributed) packet types with *N*=8 cores
  - ○ Average delay of *T*=(3+2+3)/3≈2.67 time slots

- *With pipeline sharing:*

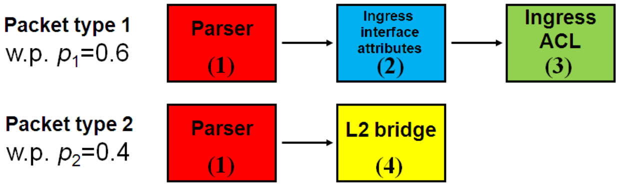


  - ○2 pipelines for the *k*=3 packet types with only *N*=6 cores
  - ○Average delay of *T*=(4+2+4)/3≈3.33 time slots

○Tradeoff: Less cores, larger average delay

Model and Problem Definition

- *Traffic*:
  - There are *k* packet types with known probabilities, each requires to perform tasks among {1, …, *r* } possible tasks
  - Example: Type 1 w.p. p_1=0.6 requires tasks S_1={1,2,3} and
    Type 2 w.p. p_2=0.4 requires tasks S_2={1,4}



Packet type 1
w.p. p_1=0.6
Parser (1) → Ingress interface attributes (2) → Ingress ACL (3)

Packet type 2
w.p. p_2=0.4
Parser (1) → L2 bridge (4)

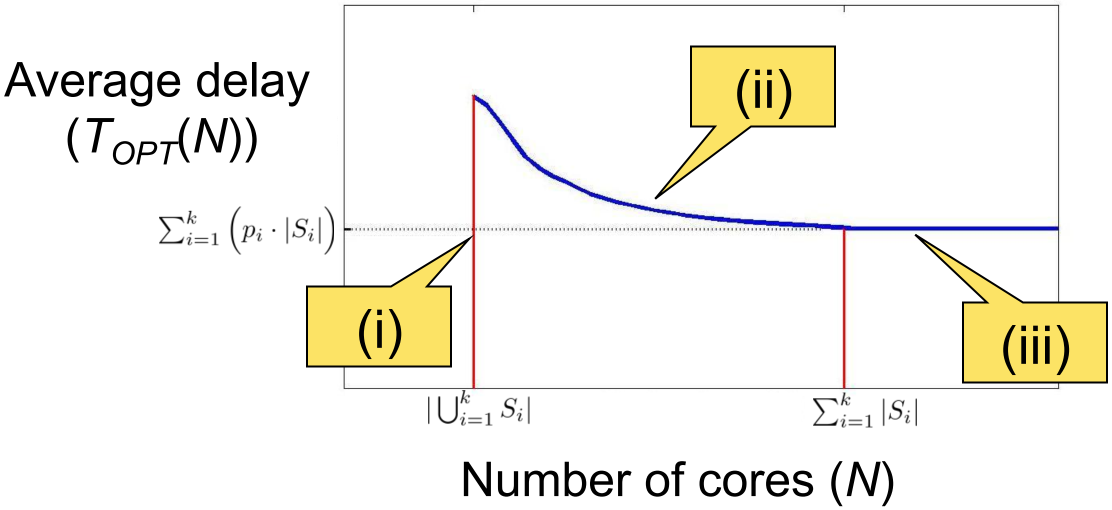
- *Pipeline sharing*:
  - A limited number of *N* homogeneous cores is given. Each core can serve any task
  - The cores should be divided into pipelines, each serves one or more packet types
  - The delay of a packet equals the length of its pipeline

- *Optimization Problem:*
  - Divide the *N* cores into pipelines, such that the average delay is minimized
  - For a given *N*, we denote by $T_{OPT}(N)$ the minimal possible average delay

Outline

- Introduction and Problem Definition
- ➡ General Properties
- Optimal Algorithm for a Special Case
- Greedy Algorithm
- Experimental Results
- Summary

General Properties



- *Property:*

  (i) At least $N = |\bigcup_{i=1}^{k} S_i|$ cores are required

  (ii) For all $N \geq 0,$ the optimal average delay $T_{OPT}(N)$ satisfies
  $T_{OPT}(N) \geq \sum_{i=1}^{k}\left(p_i \cdot |S_i|\right)$

  (iii) $T_{OPT}(N) = \sum_{i=1}^{k}\left(p_i \cdot |S_i|\right)$ for $N \geq \sum_{i=1}^{k} |S_i|$

General Properties

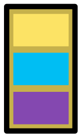
- *Property:* Let *k* be the number of packet types. Then,

(i) Given an unlimited number of cores, the number of solutions with *d* pipelines $Q_1, \cdots, Q_d$ is given by $S(k,d)$, where $S(k,d) = \frac{1}{d!} \cdot \sum_{j=1}^{d} \left((-1)^{d-j} \cdot \binom{d}{j} \cdot j^k \right)$, is the *Stirling number of the second kind of k,d*

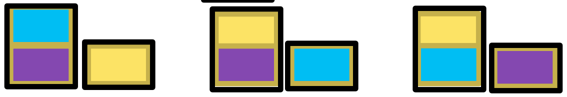
(ii) The total number of solutions is $G(k) = \sum_{d=1}^{k} S(k,d)$

- *Example:* Consider *k*=3 packet types 🟨 🟦 🟪

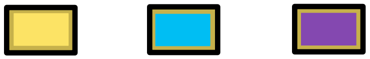
There is a single $(S(3,1) = 1)$ solution with 1 pipeline:



There are $S(3,2) = 3$ solutions with 2 pipelines:



There is $S(3,3) = 1$ solution with 3 pipelines:



The total number of solutions is $G(3) = 1 + 3 + 1 = 5$

Separability

- *Property:* Assume that packet types $S_1, \cdots, S_k$ can be partitioned into two disjoint sets, i.e. they can be ordered s.t. $(\exists m \in [1,k])\left(\bigcup_{i=1}^{m} S_i\right) \bigcap \left(\bigcup_{i=(m+1)}^{k} S_i\right) = \emptyset.$
  Then, $(\exists N_0 \in [0,N])$ s.t. an optimal solution given *N* cores can be obtained as the union of the two sets of pipelines in the optimal solutions for packet types [1,*m*] with N_0 cores and for packet types [(*m*+1),*k*] with (N-N_0) cores.

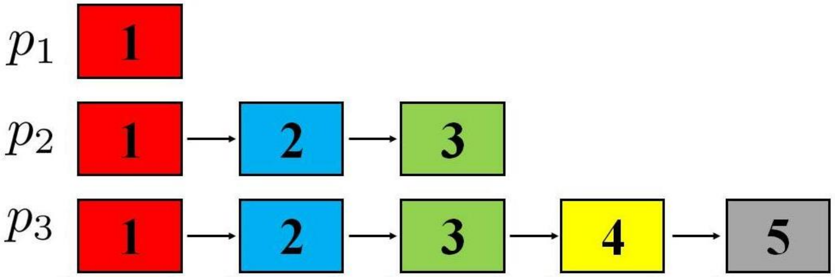
- *Proof Outline:* Any pipeline in an optimal solution cannot serve tasks from both sets $\left(\bigcup_{i=1}^{m} S_i\right), \left(\bigcup_{i=(m+1)}^{k} S_i\right).$
  Otherwise, it could be partitioned into two smaller pipelines to reduce the average delay.

Outline

- Introduction and Problem Definition
- General Properties
- Optimal Algorithm for a Special Case
- Greedy Algorithm
- Experimental Results
- Summary

Simple Case of the Required Tasks: $S_i = [1,X_i]$

- We suggest an optimal algorithm for a special case of the sets of required tasks: $S_i = [1, X_i]$ for $X_i \in [1, r]$
- *Example:* k=3, $([1], p_1), ([1,3], p_2),\ ([1,5], p_3)$



- The condition is equivalent to the requirement $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_k$
- Assume that $S_1 \cdots S_k$ are ordered s.t. $X_i \leq X_j$ for $i < j$

- Let $Q_1, \cdots, Q_d$ be the sets of tasks served by the pipelines in an optimal solution

- *Proposition:* The pipelines in an optimal solution $Q_1, \cdots, Q_d$ satisfy $(\forall j \in [1, d])\ Q_j \in \{S_1, \cdots, S_k\}$, i.e. the pipelines in the solution are among the pipelines in the input

Simple Case of the Required Tasks: *S*$_i$ = [1,*X*$_i$]

- *Proposition:* Assume that $Q_1, \cdots, Q_d$ are ordered such that $Q_1 \subsetneqq Q_2 \subsetneqq \cdots \subsetneqq Q_{d-1} \subsetneqq Q_d$. Then,

  (i) The packet types are served by an increasing order of pipelines.

  In particular, the packet types served by each pipeline form a subset of consecutive packet types from the input.

  (ii) The last packet type is served by the last pipeline Q_d .

$$\underbrace{S_1, S_2, S_3}_{Q_1} \quad \underbrace{S_4, S_5}_{Q_2} \quad \underbrace{S_6, \ldots}_{Q_3} \qquad \underset{\cdots}{\cdots} \qquad \underbrace{\ldots, S_k}_{Q_d}$$

Simple Case of the Required Tasks: *S*$_i$ = [1,*X*$_i$]

- Let $T^i(n)$ (for $i \in [0,k]$, $n \in [0,N]$) be the minimal possible average delay obtained in the service of the first *i* types with at most *n* cores. Let $Q^i(n)$ be the corresponding set of pipelines.

- *Proposition:*

  (i) For $i \geq 1$, $T^i(n) = \min_{j \in [1,i]} \left(T^{i-j}(n - |S_i|) + \left(\sum_{m=(i-(j-1))}^{i} p_m \right) \cdot |S_i| \right)$

  (ii) For *j* that minimizes (i), $Q^i(n) = Q^{i-j}(n - |S_i|) \bigcup \{S_i\}$

- *Algorithm:*

  ○ In step *i* (for $i \in [1,k]$), calculate $T^i(n)$ for $n \in [0,N]$

  ○ Return: $T^k(N)$ (optimal delay), $Q^k(N)$ (set of pipelines)

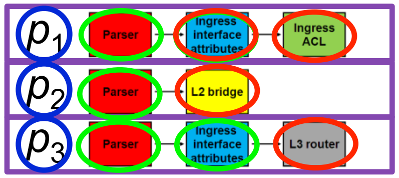
  ○ Complexity: $O(k \cdot N \cdot k) = O(k^2 \cdot N)$

Outline

- Introduction and Problem Definition
- General Properties
- Optimal Algorithm for a Special Case
- Greedy Algorithm
- Experimental Results
- Summary

Pipeline Merging Algorithm

- *Idea:* Start with a pipeline for each type, merge pairs of pipelines with common cores



- **Intuition:** For each pair with common cores, we prefer to merge
  - More common cores
  - Less non-common cores
  - Low probability to be served by a pipeline in the pair

- Marginal cost of a possible merging operation
  - x – expected increment in the average delay
  - y – decreased number of cores
  - Marginal cost of $R=x/y$

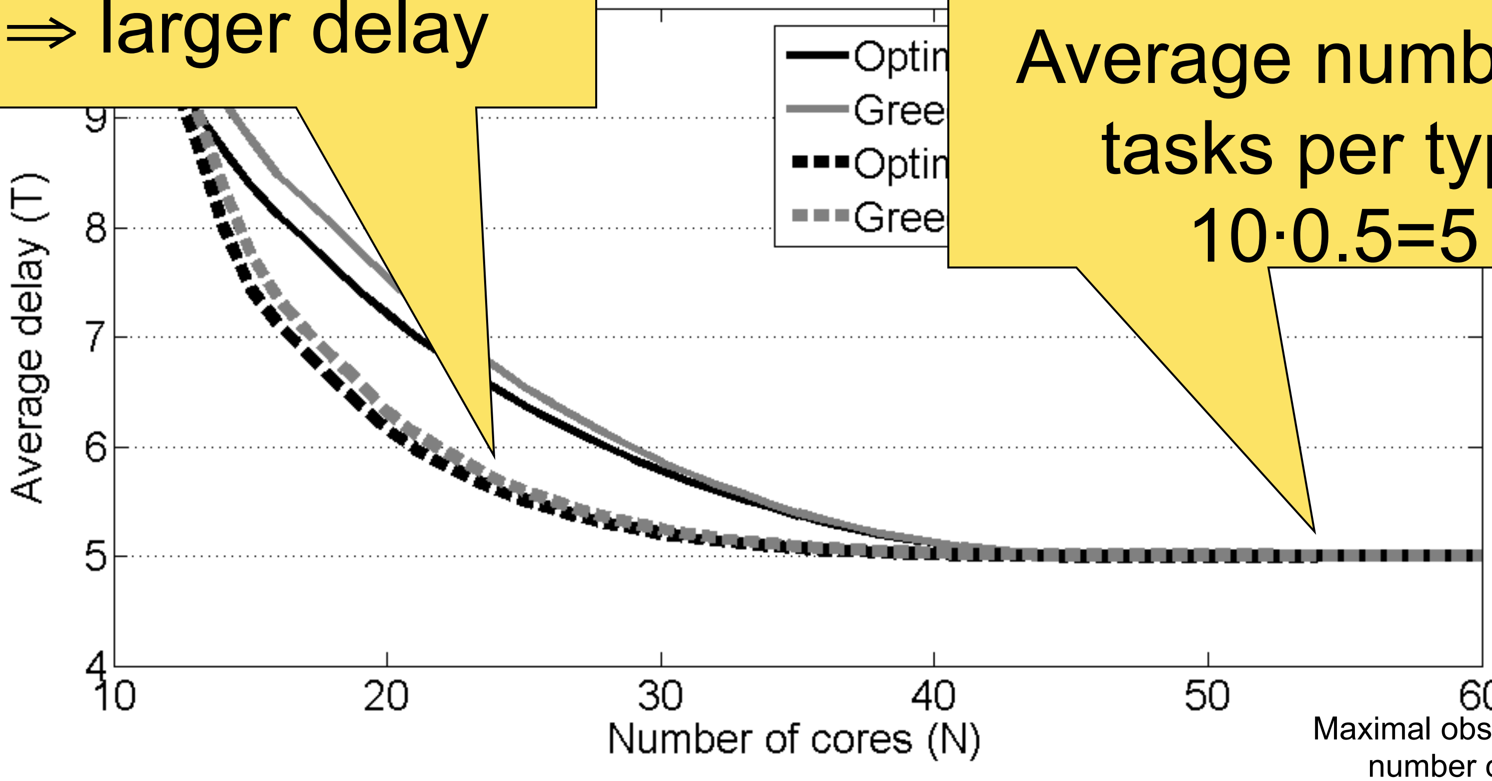
- *Algorithm:* Until the required number of cores is obtained, merge pairs of pipelines with minimal marginal cost $R=x/y$

- Not necessarily optimal but very efficient on synthetic and real-life applications

Experimental Results

- *Synthetic Simulation Parameters*:
  - We consider k=8 packet types
  - Tasks selected among r =10 possible tasks {1,…,r =10}
  - Each type requires a specific task w.p. 0.5 without any dependency between different types or tasks
  - Two options for the packet types probabilities:
    - ❑ *fixed prob.* – all types w.p. 1/k = 0.125
    - ❑ *variable prob.* – geometrically decreasing probabilities of 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}, 2^{-7} and 2^{-7}
  - Results are based on the average of 10^3 iterations

Experimental Results



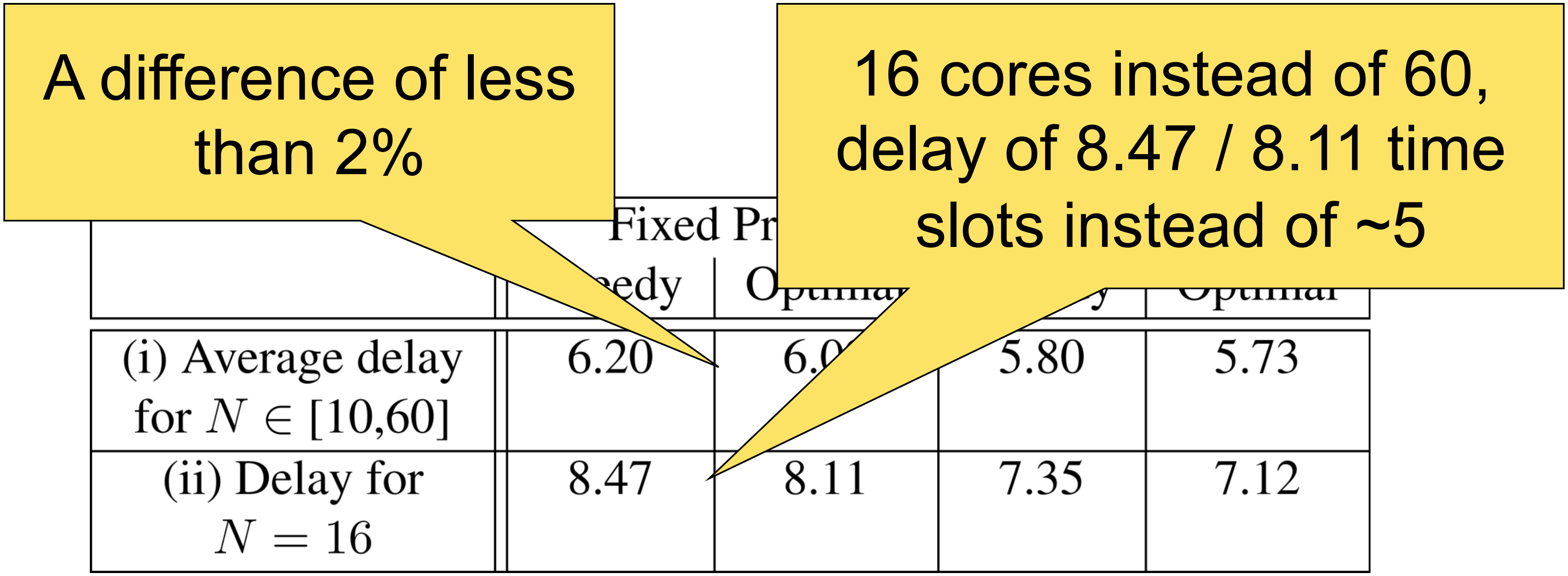
Less cores
⇒ larger delay

A lower bound:
Average number of
tasks per type
10·0.5=5

Maximal observed total
number of tasks

- Effectiveness of the number of cores on the average delay in time slots. *k*=8 packet types with *r* =10 possible tasks, each required w.p. 0.5 by each type. 10^3 iterations.

Experimental Results



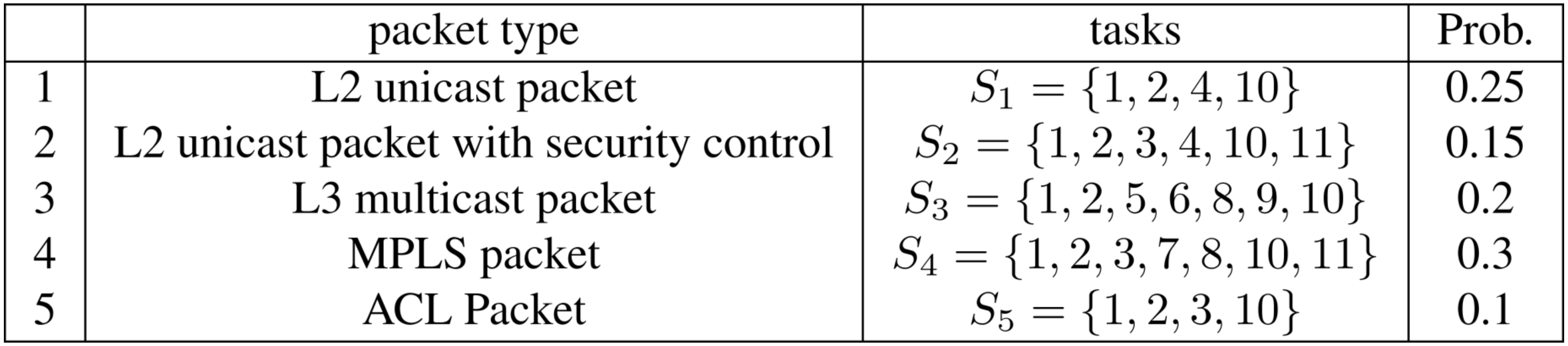
A difference of less than 2%

16 cores instead of 60, delay of 8.47 / 8.11 time slots instead of ~5

	Fixed Pr			
	edy	Optimal	y	Optimal
(i) Average delay for $N \in [10,60]$	6.20	6.0	5.80	5.73
(ii) Delay for $N = 16$	8.47	8.11	7.35	7.12

- Summary of the synthetic simulations. k=8 packet types with r =10 possible tasks, each required w.p. 0.5 by each type. 10^3 iterations.

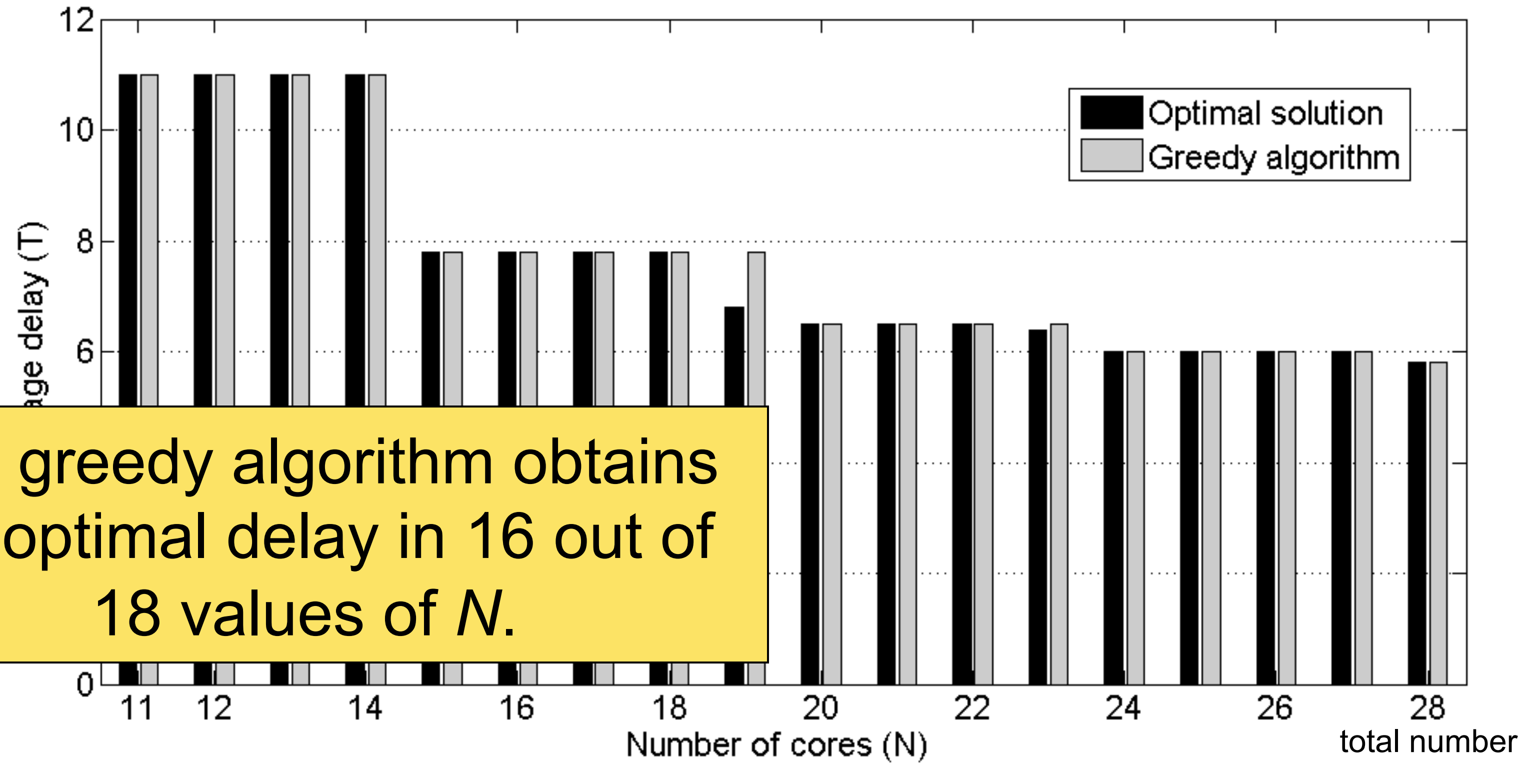
Packet-processing Application



	packet type	tasks	Prob.
1	L2 unicast packet	$S_1 = \{1, 2, 4, 10\}$	0.25
2	L2 unicast packet with security control	$S_2 = \{1, 2, 3, 4, 10, 11\}$	0.15
3	L3 multicast packet	$S_3 = \{1, 2, 5, 6, 8, 9, 10\}$	0.2
4	MPLS packet	$S_4 = \{1, 2, 3, 7, 8, 10, 11\}$	0.3
5	ACL Packet	$S_5 = \{1, 2, 3, 10\}$	0.1

- *Possible Tasks:* (1) Parsing, (2) Ingress interface attributes, (3) Ingress ACL, (4) L2 bridging, (5) L3 routing, (6) L3 replication, (7) MPLS switching, (8) header modification, (9) L2 replication, (10) Egress interface resolution, (11) Egress ACL

Packet-processing Application

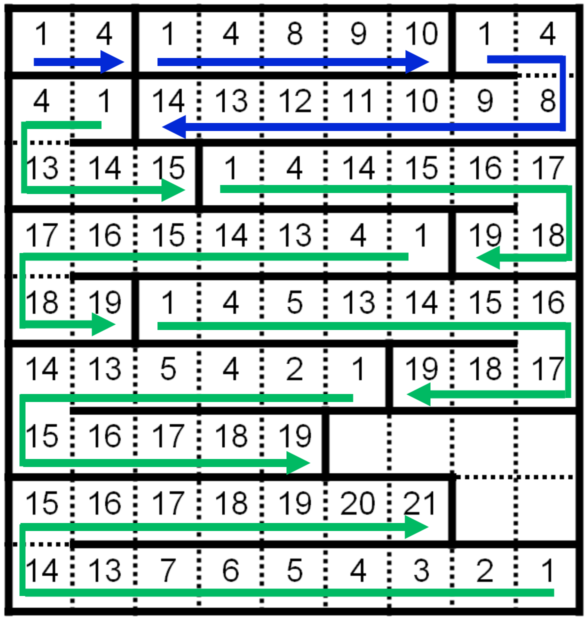


total number of tasks

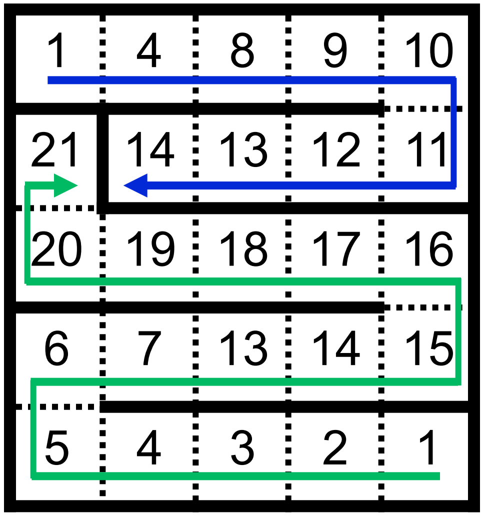
The greedy algorithm obtains the optimal delay in 16 out of 18 values of *N*.

- *k*=5 packet types w.p. (0.25, 0.15, 0.2, 0.3, 0.1) and *r* =11 possible tasks. The total number of solutions is G(5) = 52. The greedy algorithm starts with 28 cores, then reduces it to 24, 20, 15 and finally 11. It obtains the optimal delay in 16 out of 18 values of *N*.

H.264 video-compression standard

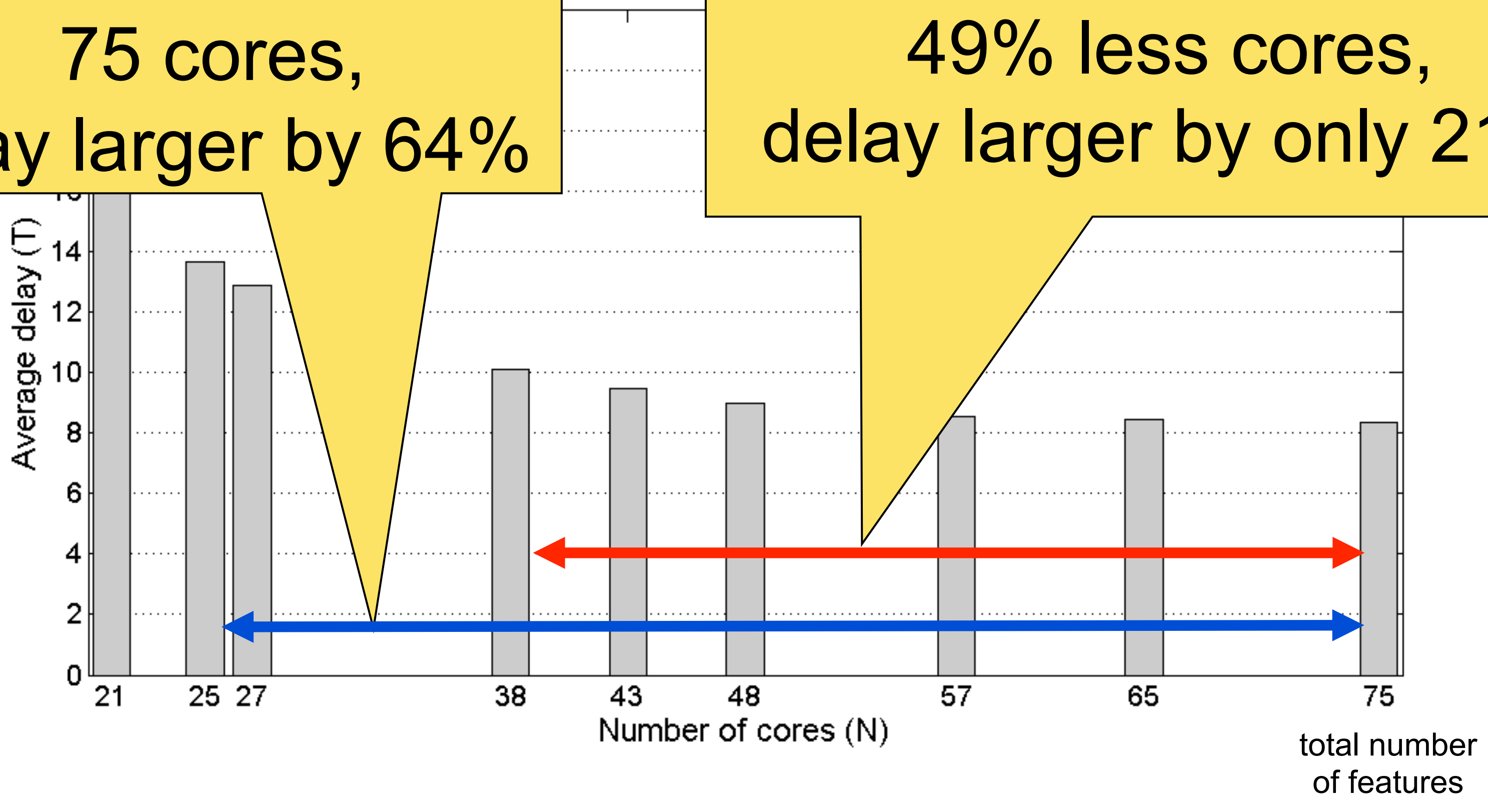


9 profiles with 21 features are organized in 9 pipelines in a 9x9 multicore



Pipeline Sharing: The 9 profiles are serviced by one of two possible pipelines in a smaller 5x5 multicore

H.264 video-compression standard



25 instead of 75 cores, delay larger by 64%

49% less cores, delay larger by only 21%

- *k*=9 popular (uniformly-distributed) profiles (types) of the H.264 standard with *r* =21 supported features (tasks). Results are based on the greedy algorithm.

Concluding Remarks

- **New approach of sharing pipelines to reduce the number of required cores**

- Analysis of the optimal average delay

- Optimal solution for $S_i = [1, X_i]$

- Greedy algorithm for the general case

Thank You