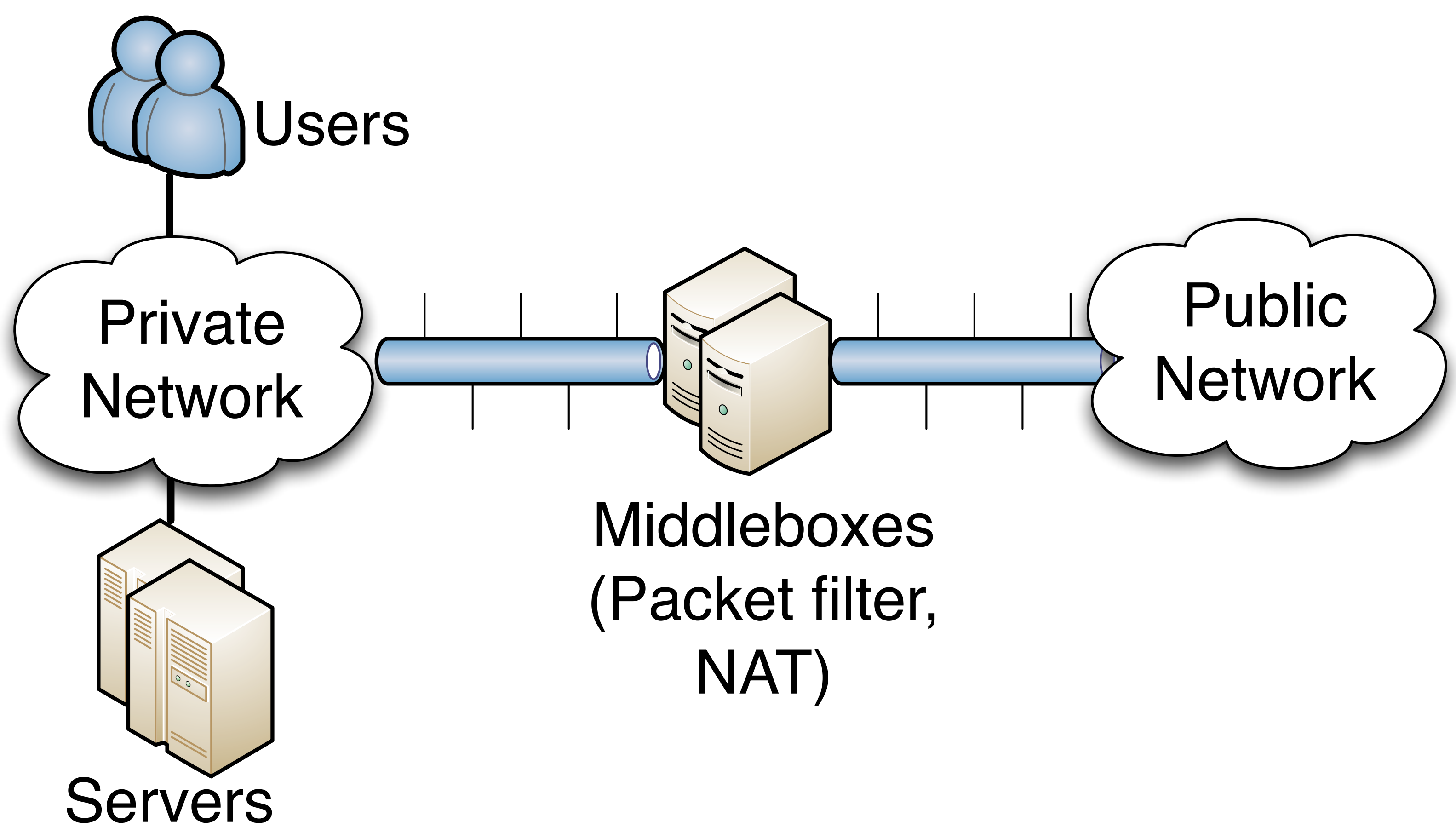
Low Complexity Multi-Resource Fair Queueing with Bounded Delay

Wei Wang, Ben Liang, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

May 1, 2014

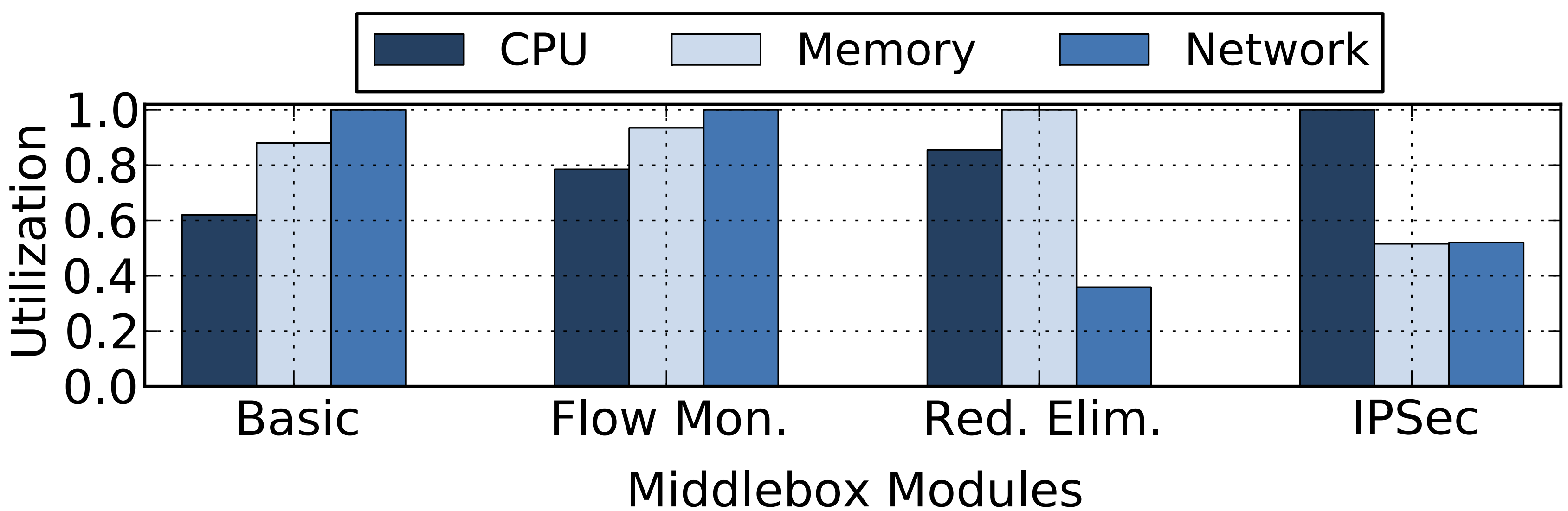
Background

- Middleboxes are widely deployed in today's network
  - IPsec, Monitoring, Firewalls, WAN optimization, etc



Background

- Performing complex network functions requires **multiple** middlebox resources
  - CPU, memory b/w, link b/w



Ghodsi SIGCOMM'12

How to **fairly** share multiple resources among flows?

Desired Fair Queueing Algorithm

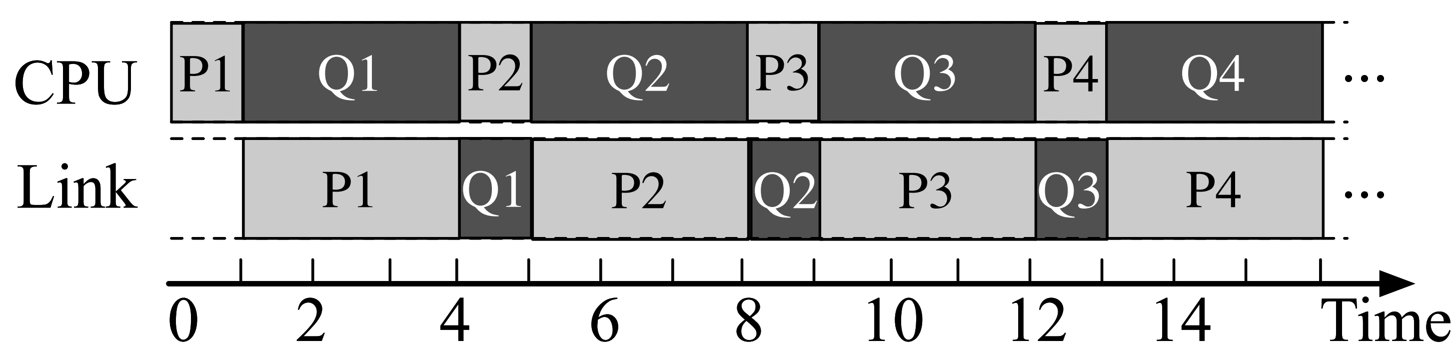
- **Fairness**
- Bounded scheduling delay
- Low complexity

Dominant Resource Fairness (DRF)

- **Dominant resource**: The resource that requires the most **processing time**
  - A packet *p* requires 1 ms of CPU processing, and 3 ms of link transmission
  - Link bandwidth is its dominant resource

Dominant Resource Fairness (DRF)

- Max-min fairness on flow's processing time of the dominant resource
  - Flows receive the same processing time on their respective dominant resources



Desired Fair Queueing Algorithm

- Fairness
- **Bounded scheduling delay**
- Low complexity

Scheduling Delay

- Scheduling delay of packet p
  - D(p) = t2 - t1
  - t1: time when p reaches the head of its queue
  - t2: time when p finishes service on all resources

Bounded Scheduling Delay

- Scheduling delay is bounded by a small constant factor
  - Inversely proportional to a flow's weight

$$D_i(p) \leq C/w_i$$

Desired Fair Queueing Algorithm

- Fairness
- Bounded scheduling delay
- **Low complexity**

Low Complexity

- Make scheduling decisions at O(1) time
  - Independent of the number of flows
  - Easy to implement

The State-of-the-art

- Dominant Resource Fair Queueing (DRFQ) [Ghodsi12]
  - High complexity O(log n)
- Multi-resource round robin (MR^3) [ICNP13]
  - O(1) time
  - May incur **unbounded delay** for weighted flows

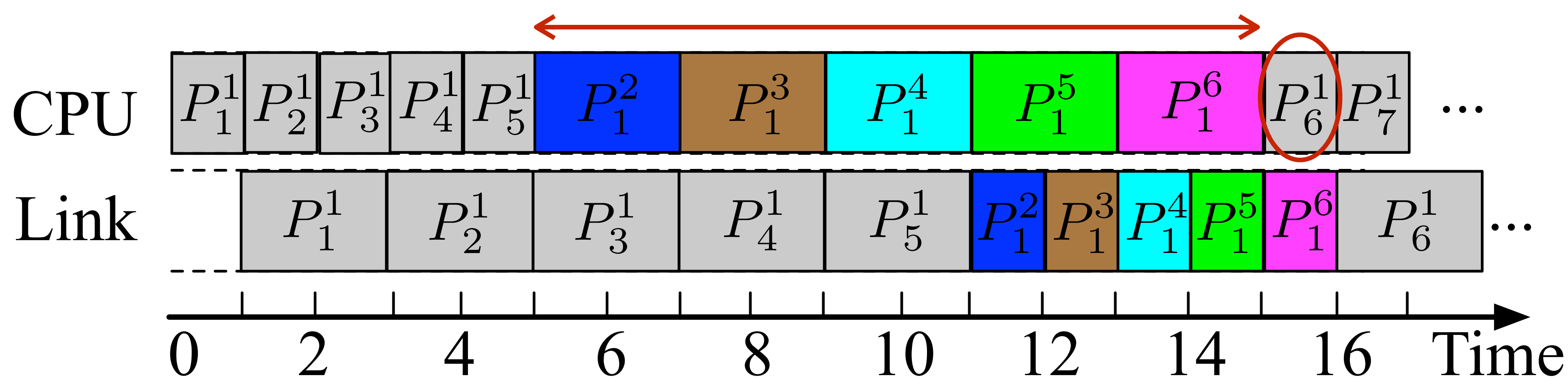
We propose Group Multi-Resource Round Robin (GMR^3)

GMR^3

- O(1) time
- Bounded scheduling delay
- Near-perfect fairness

Delay Problem of Multi-Resource Round Robin

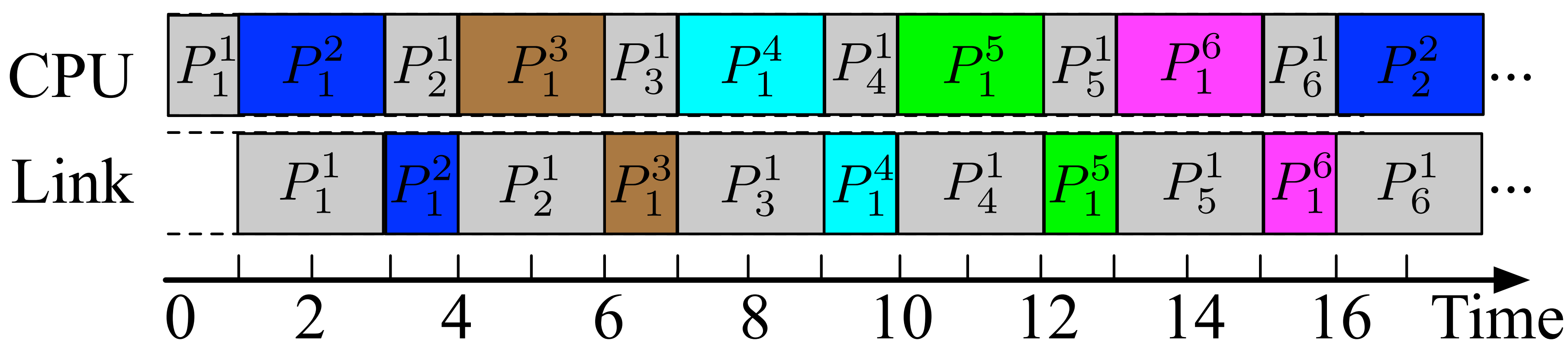
- Flow 1 weighs 1/2, while flow 2 to 6 each weighs 1/10



- Flows with large weights are served in a “burst” mode
  - Some packets have to wait for an entire round to be scheduled

An Improvement

- Spread the scheduling opportunities over time, in proportion to flows' respective weights



- Packets do not need to wait for a long round to get scheduled

Flow Grouping

- Normalized flow weights $\sum_{i=1}^{n} w_i = 1$.
- Flow group *k*

$$G_k = \{i : 2^{-k} \leq w_i < 2^{-k+1}\}, \quad k = 1, 2, \ldots$$

  - Flows with approximately the same weights
  - A small number of flow groups $n_g \leq \log_2 W$
  - *W* — $\max_i w_i / \min_j w_j$

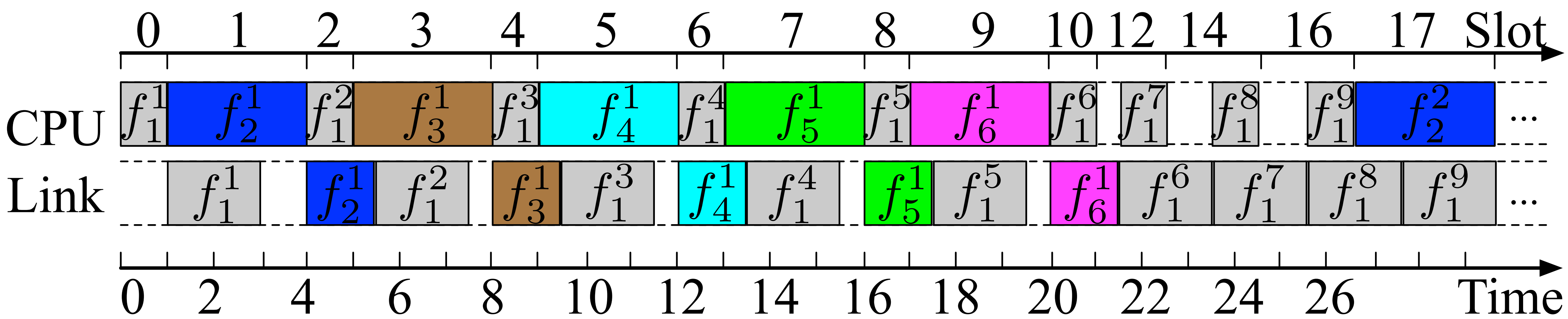
Distributing Scheduling Opportunities

- Virtual slot 0, 1, 2, …, each representing a scheduling opportunity of a flow
- Each flow i of flow group G_k is assigned to **exactly one slot** every 2^k slots, roughly matching its weight

$$G_k = \{i : 2^{-k} \le w_i < 2^{-k+1}\}, \quad k = 1, 2, \ldots$$

An example

- Flow group G1 — flow 1 (weight = 1/2)
- Flow group G4 — flow 2 to 6 (weight = 1/10)



Fine tune the dominant service a flow receives at each scheduling opportunity

Credit System

- Each flow maintains a credit account
  - Credit balance represents the deserved dominant service in the current round
  - Deposit credits upon a scheduling opportunity
  - Withdraw credits at the end of a scheduling opportunity
    - credits = the dominant services received due to this scheduling opportunity

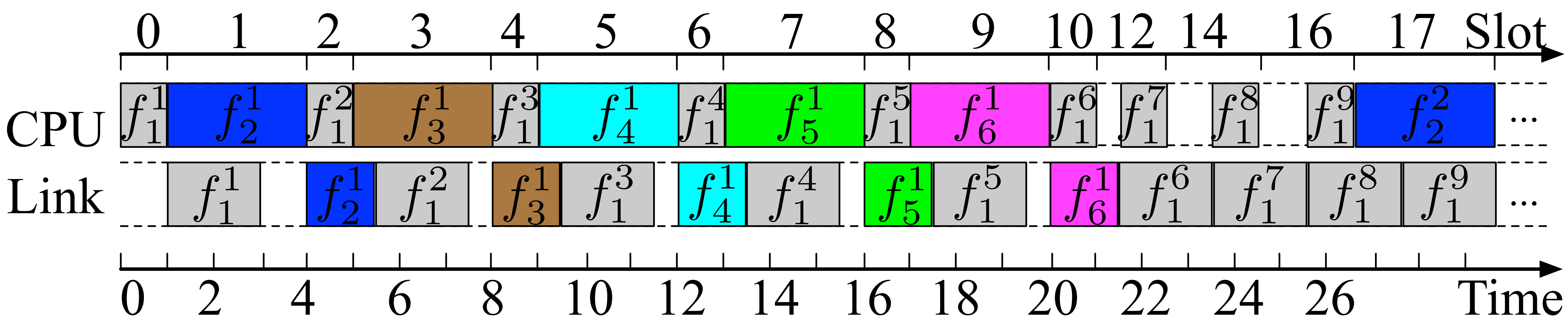
Depositing Credits

- Flow i belonging to flow group G_k: $2^{-k} \le w_i < 2^{-k+1}$,
- Credits deposited upon a scheduling opportunity

$$c_i = 2^k L w_i \ ,$$

  - L — Maximum packet processing time
  - Roughly the same amount of credits $L \le c_i < 2L$

Potential Progress Gap



- A flow may not receive dominant services in the assigned virtual slot
- Potential progress gap may lead to arbitrary unfairness

Progress Control Mechanism

- Enforce roughly consistent progress across all resources
- Upon the k^{th} scheduling opportunity, defer flow *i*'s service until it has already received service on the last resource due to the previous opportunity (k-1)
  - Work progress on any two resources will not differ too much

Two-Level Hierarchical Scheduling

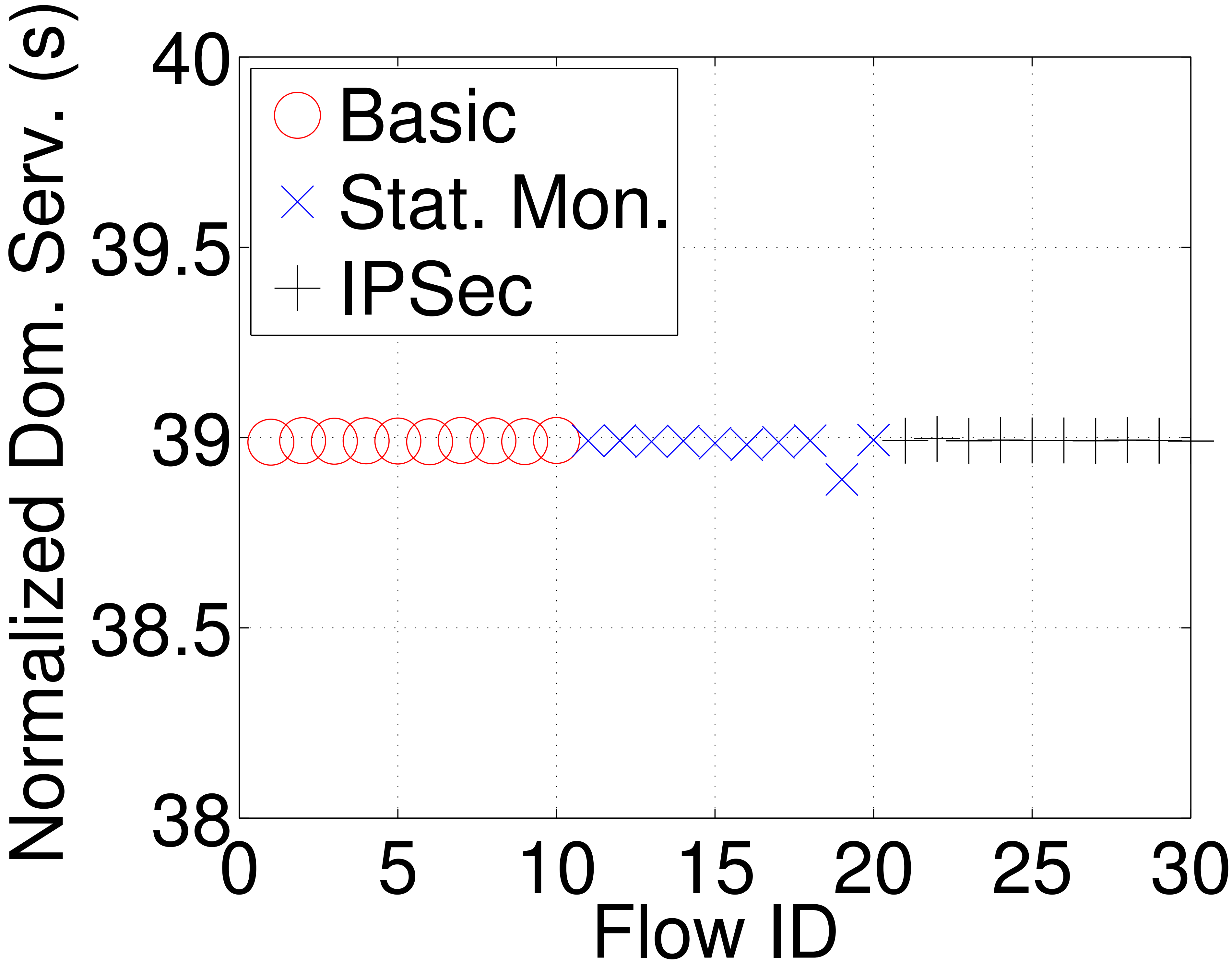
- Combine flows with similar weights into a **flow group**
- Inter-group scheduling — determine which flow group to choose
- Intra-group scheduling — determine which flow to choose from the selected flow group
  - Round robin
  - Credit system + Progress control mechanism

Performance Analysis

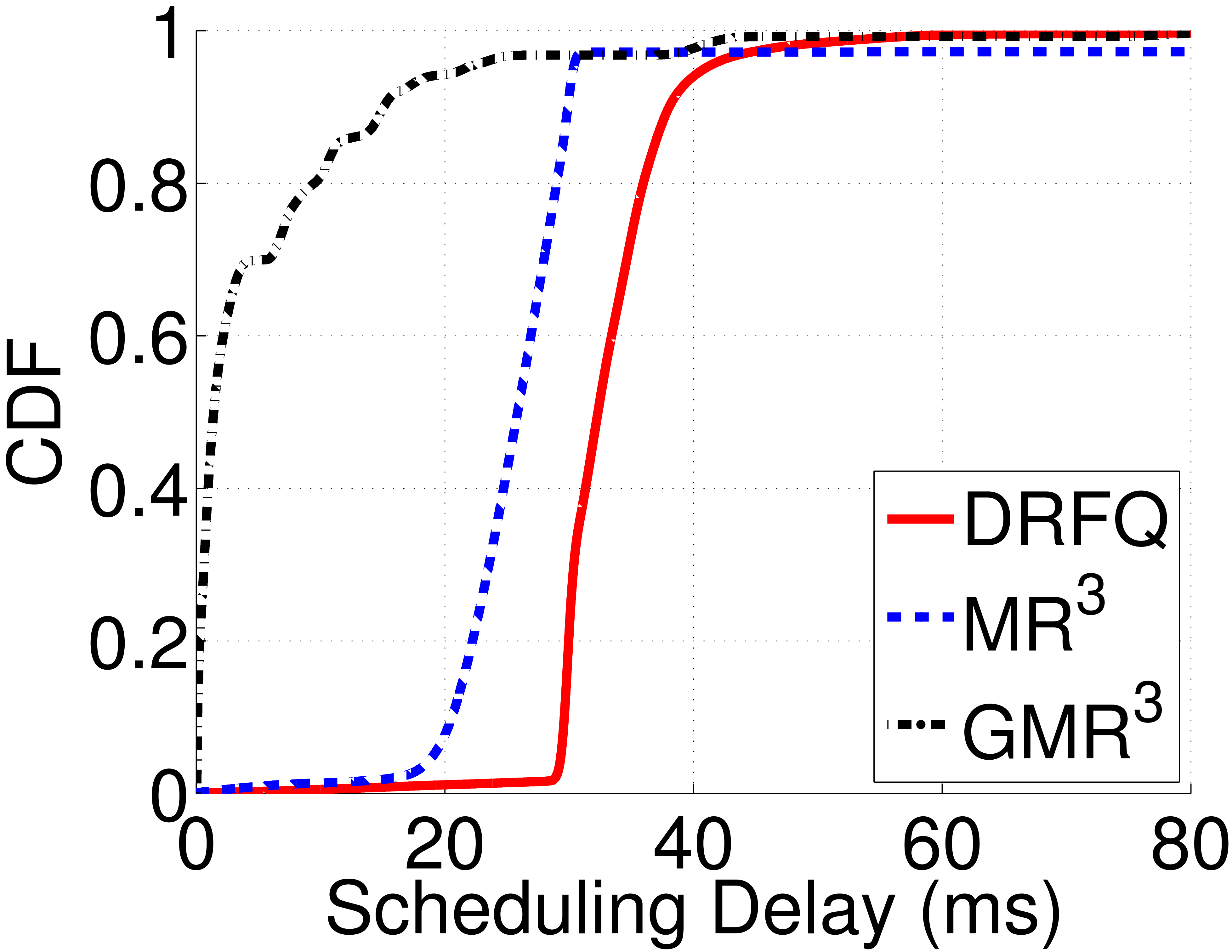
- n — # of flows m — # of resources
- W — $\max_i w_i / \min_j w_j$ L — Max pkt proc time

Scheme	**Complexity**	**Fairness**[1]	**Scheduling Delay**
DRFQ [10]	$O(\log n)$	$L(1/w_i + 1/w_j)$	Unknown
MR^3 [17]	$O(1)$	$2L(1/w_i + 1/w_j)$	$4(m+W)^2 L/w_i$
GMR^3	$O(1)$	$9L(1/w_i + 1/w_j)$	$24mL/w_i$

Simulation Results



(a) Normalized dominant service.



(b) CDF of the scheduling delay.

Conclusions

- GMR^3, a two-level hierarchical scheduling algorithm
- The *first* multi-resource fair queueing of
  - O(1) complexity
  - near-perfect fairness
  - bounded scheduling delay