Network Technology and Cloud Storage

Dept. of Information Technology & Comm.,
Shih Chien University
Name: Dr. Jui-Pin Yang (楊瑞賓 博士)
Date: 2015/05/04

Portfolio

- Personal information
  - Name: Jui-Pin Yang
  - Ph.D. degree: Network & Communication, Dept. of Electrical Engineering, National Chung Cheng University
- Now
  - Associate professor, Dept. of ITC, SCU
- Past
  - 跨領域科技管理碩士學分班,國立政治大學
  - Visiting scholar, Dept. of CS, NUS
  - Visiting scholar, IIC, Hokkaido University
  - Assistant professor, Dept. of ITC, SCU
  - Project leader/Engineer, ICL/South, ITRI (2004/1~2008/1)

Outline

- Network Technology
  - **Fair Queueing
  - OpenFlow Protocol
- Cloud Storage
  - **Load Balancing
- Conclusion

Fair Queueing

- Introduction
- Background
- CSFQ Algorithm
- Simulations
- Evaluations of CSFQ
- Rotating Preference Queues (RPQ)

(2) Ion Stoica, Scott Shenker and Hui Zhang. “Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks.” IEEE/ACM Transactions on Networking, 2003.

Introduction

- **Main Idea**:
  - Achieve fair bandwidth allocations at the router with low implementation complexity
- **Goals**:
  - Achieve fair allocation close to DRR and comparable or better than RED and FRED under most scenarios
  - Reduce complexity by not having the core node maintain per flow state

Fair Queueing

- Introduction
- Background
- CSFQ Algorithm
- Simulations
- Evaluations of CSFQ
- Rotating Preference Queues (RPQ)

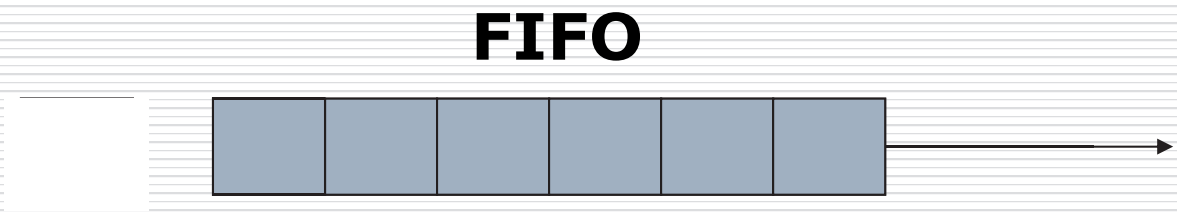
(1) *Nagaraj Shirali and Choong-Soo Lee slides*

(2) Ion Stoica, Scott Shenker and Hui Zhang. “Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks.” IEEE/ACM Transactions on Networking, 2003.

Previous Work

- FIFO queueing with Drop Tail
- Random Early Drop (RED)
- Flow Random Early Drop (FRED)
- Deficit Round Robin (DRR)

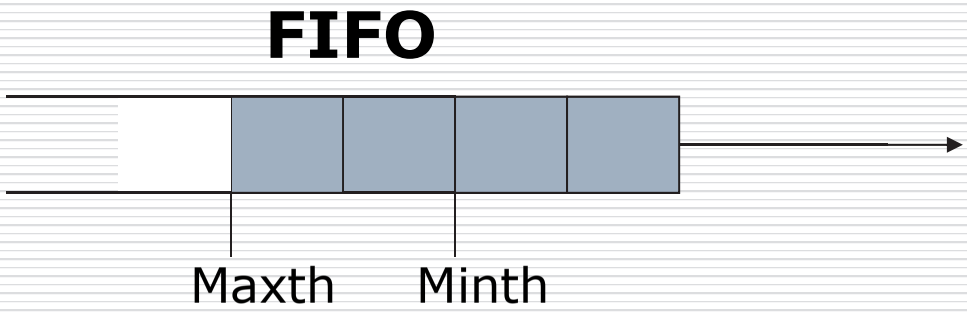
FIFO queueing with Drop Tail

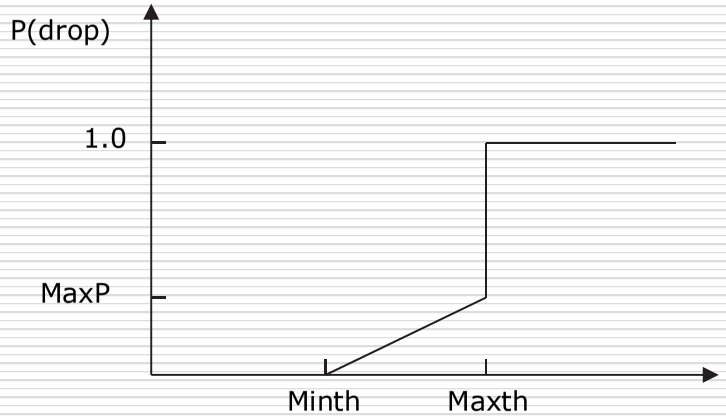


Disadvantages:

- Pushes congestion control out to end hosts (TCP)
- Introduces *global synchronization* when packets are dropped from several connections

Random Early Drop (RED)

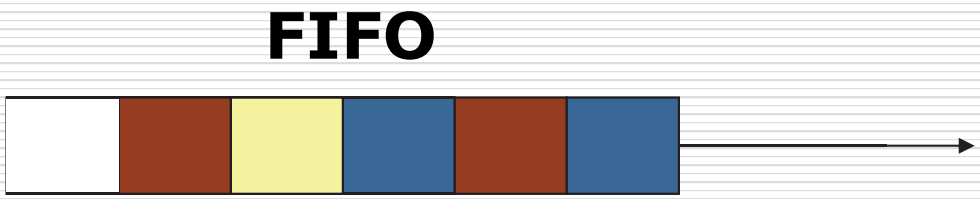




Disadvantage:

- For heterogeneous traffic, RED provides no clear advantage over tail-drop FIFO for fairness

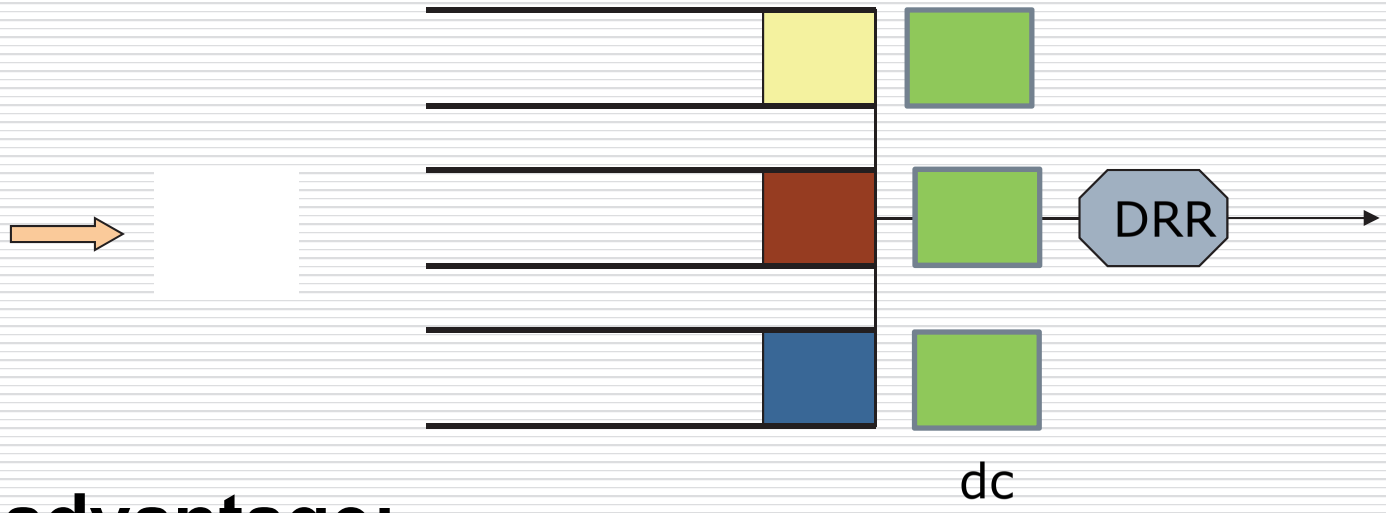
Flow Random Early Drop (FRED)



Disadvantage:

Complex to implement – maintain state on per-flow basis, i.e., queue length of each flow

Deficit Round Robin (DRR)



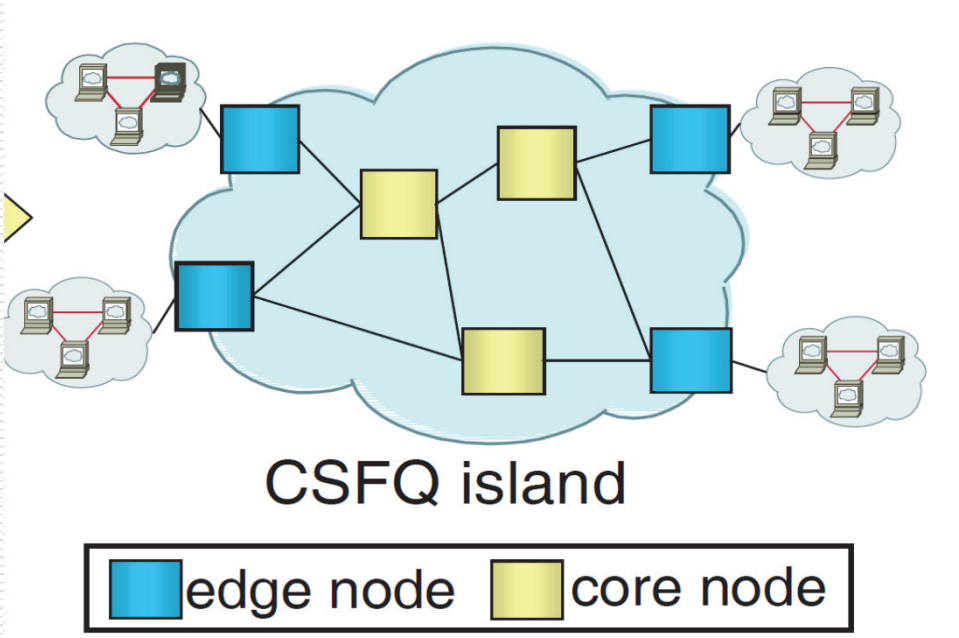
Disadvantage:

- Need to perform packet classification and maintain state and buffers on per-flow basis and perform operations on per-flow basis
- Need buffer management, i.e., PO

Definitions

- ***Island of routers*** – a contiguous portion of the network with well defined interior and edges
- ***Edge Router*** – computes per-flow rate estimates and *labels* the packets with these estimates
- ***Core Router*** – uses FIFO queueing and keeps no per-flow state, employs a probabilistic dropping algorithm that uses the packet label and its own measurement of aggregate traffic
- ***Stateless*** – absence of per-flow state at the core routers

Island of Routers



Fair Queueing

- Introduction
- Background
- CSFQ Algorithm
- Simulations
- Evaluations of CSFQ
- Rotating Preference Queues (RPQ)

(2) Ion Stoica, Scott Shenker and Hui Zhang. “Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks.” IEEE/ACM Transactions on Networking, 2003.

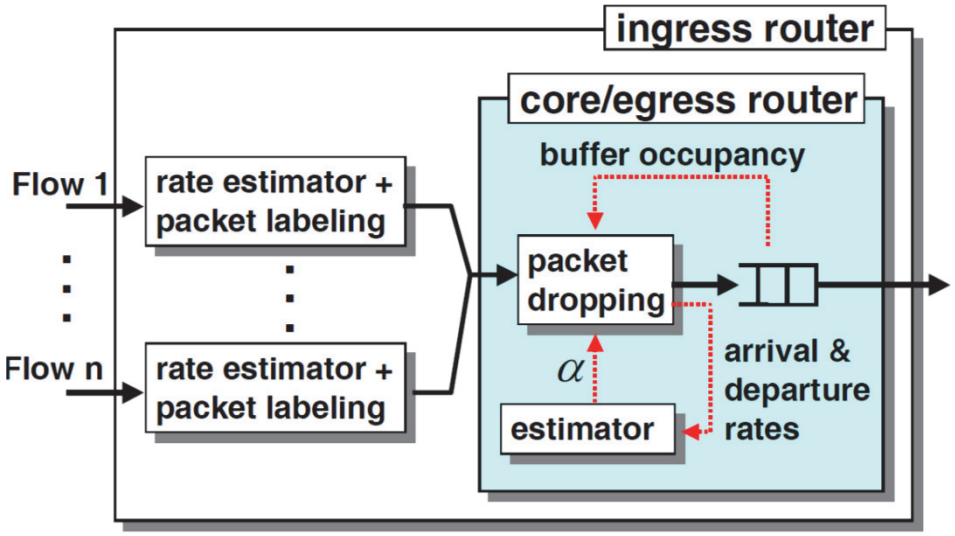
CSFQ

- Edge routers compute per-flow rate estimates and label the packets with these estimates
- Core routers use FIFO queueing and keep no per-flow state, they employ a probabilistic dropping algorithm based on packet labels and own aggregate traffic estimates

CSFQ

- Bandwidth allocations using this method are approximately fair
- Core routers keep no per-flow state and avoid using complicated packet scheduling and buffering algorithms, hence are easier to adopt

CSFQ



- Assume that flow *i* has arrival rate $r_i(t)$ and the fair rate is $a(t)$.
- If $r_i(t) < a(t)$, all of its traffic is forwarded
- If $r_i(t) > a(t)$, then a fraction $(r_i(t) - a(t))/ r_i(t)$ will be dropped; each packet of the flow is dropped with probability $(1-a(t)/r_i(t))$. Thus the output rate of any flow *i* will be $max(r_i(t) ,a(t))$

CSFQ

- The problem now becomes how to calculate the flow rate $r_i(t)$ values and the fair rate a(t), without keeping per flow state in the core routers
- Flow rates $r_i(t)$, are calculated at edge routers which keep per flow state and then insert the rate value inside the packet header of packets belonging to that flow

CSFQ

- To estimate the fair rate a(t), an iterative procedure is used: core routers estimate aggregate arrival rate A and the aggregate rate of accepted traffic F (arrival rate – dropped packets).
- Based on these, the fair rate a is computed periodically as:

  * if there is no congestion (A<=C where C is the link's capacity), then a is set to the maximum $r_i(t)$

  * if the links are congested, then $a_{new} = a_{old} * C/F$

CSFQ - Example

- Assume we have two flows f_1 and f_2, with rates $r_1 = 20$ and $r_2 = 30$ and the link's capacity is C = 30. Initially let's say that only r_1 is active and the link is not congested, so $a_1 = 20$. Then r_2 becomes active.

  Since no packets were dropped, F = 50.
- Since A = 50>C, $a_2 = a_1 * C/F = 20 * 30/50 = 12$, Therefore, for f_1 (1-12/20 = 40%) of its packets are dropped while for f_2 (1-12/30 = 60%) of its packets are dropped and F = 12+12 = 24

  Since A>C, $a_3 = a_2 * C/F = 12 * 30/24 = 15$
- Now F = 30, and $a_4 = a_3 * C/F = 15 * 30/30 = 15$. Therefore, a has converged to the right fair rate.

CSFQ

- Estimation of flow arrival rates:

$$R_{new} = (1-e^{-T/K})*l/T + e^{-T/K}*R_{old}$$

where T = packet interarrival time

l = packet size

K = constant

- To summarize, Edge routers needs to

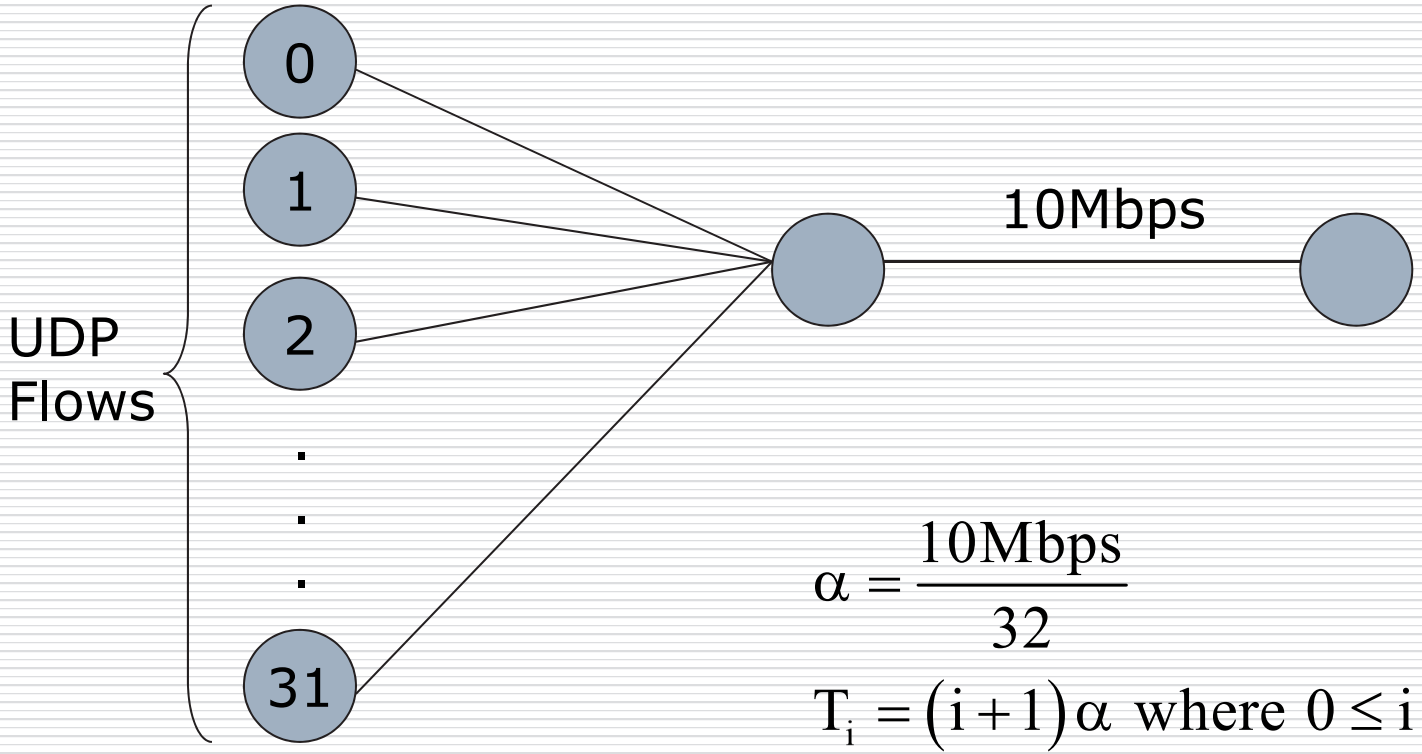
1) Classify the packet to a flow
2) Update the fair share rate estimation for the outgoing link
3) Update the flow rate estimation
4) Label the packet

Fair Queueing

- Introduction
- Background
- CSFQ Algorithm
- Simulations
- Evaluations of CSFQ
- Rotating Preference Queues (RPQ)

(2) Ion Stoica, Scott Shenker and Hui Zhang. “Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks.” IEEE/ACM Transactions on Networking, 2003.

Simulations – Single Congested Link

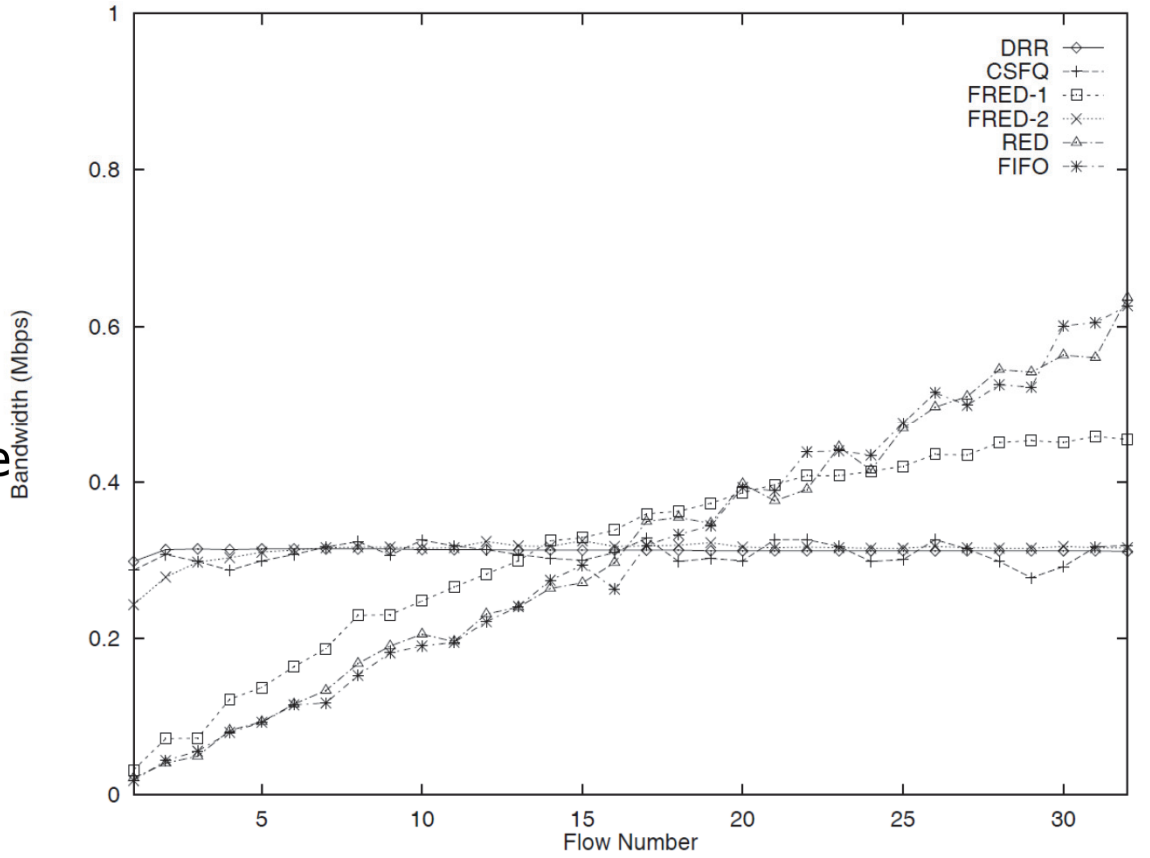


$$\alpha = \frac{10\text{Mbps}}{32}$$

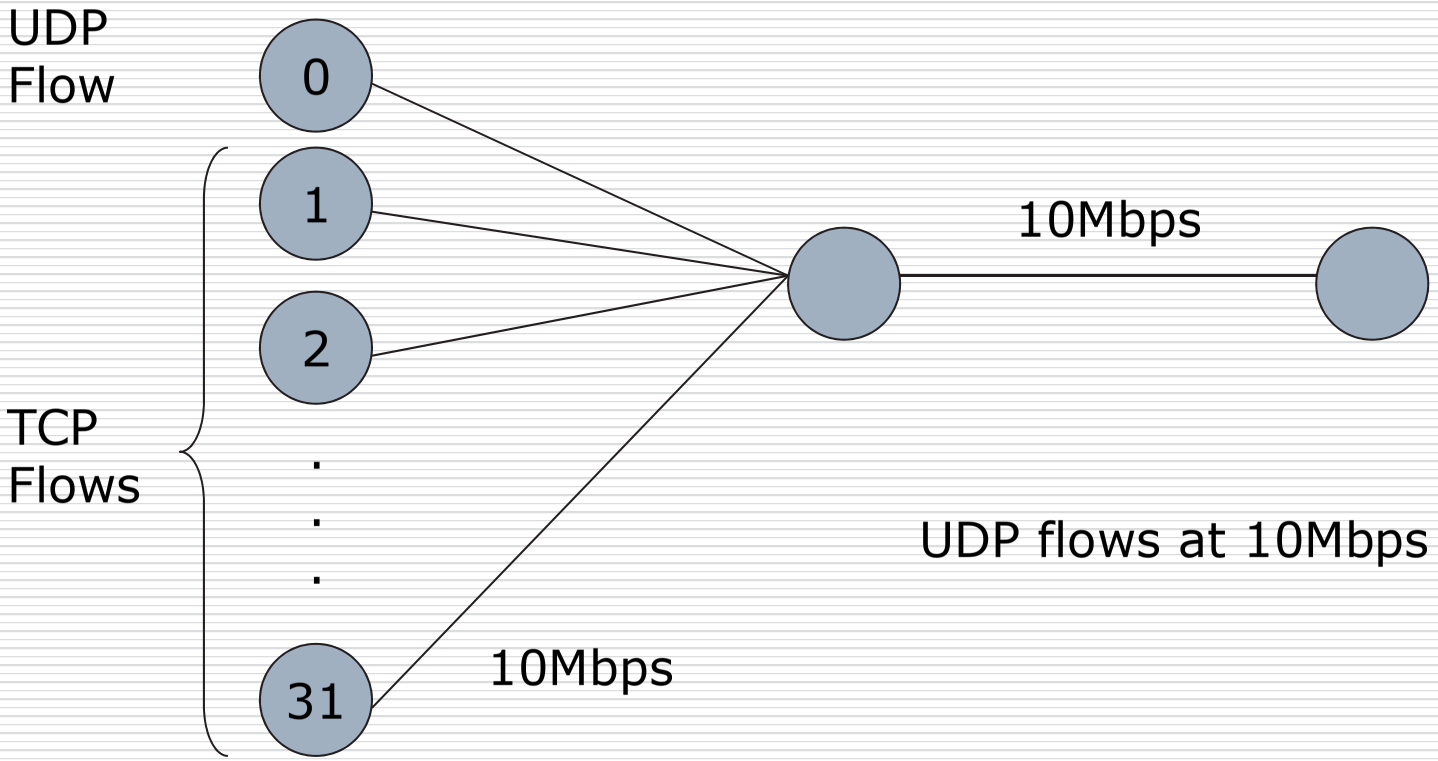
$$T_i = (i+1)\alpha \text{ where } 0 \le i \le 31$$

Simulations – Single Congested Link

*FRED preferentially drops a packet of a flow that has either (1) had many packets dropped in the past, or (2) a queue larger than average queue size.
*Main difference between the two is that FRED-2 guarantees to each flow a minimum number of buffers.

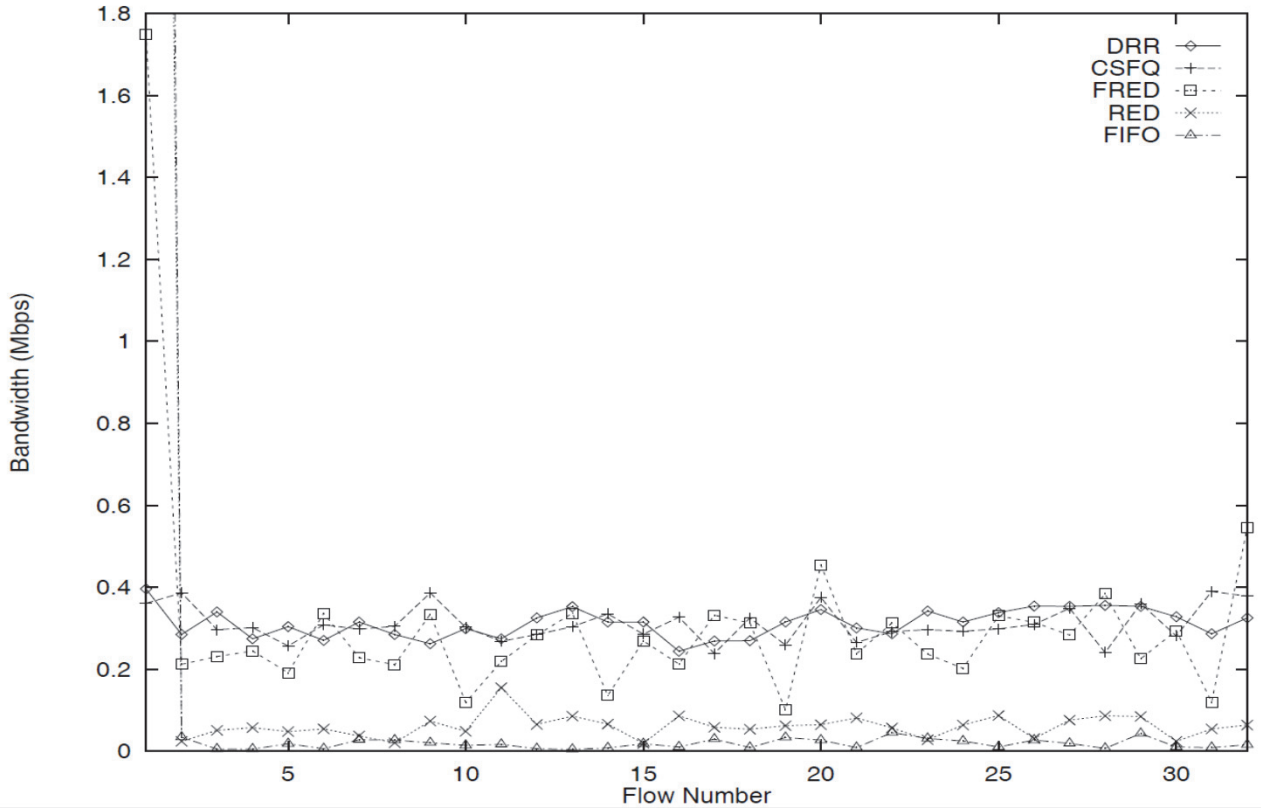


Simulations – Single Congested Link

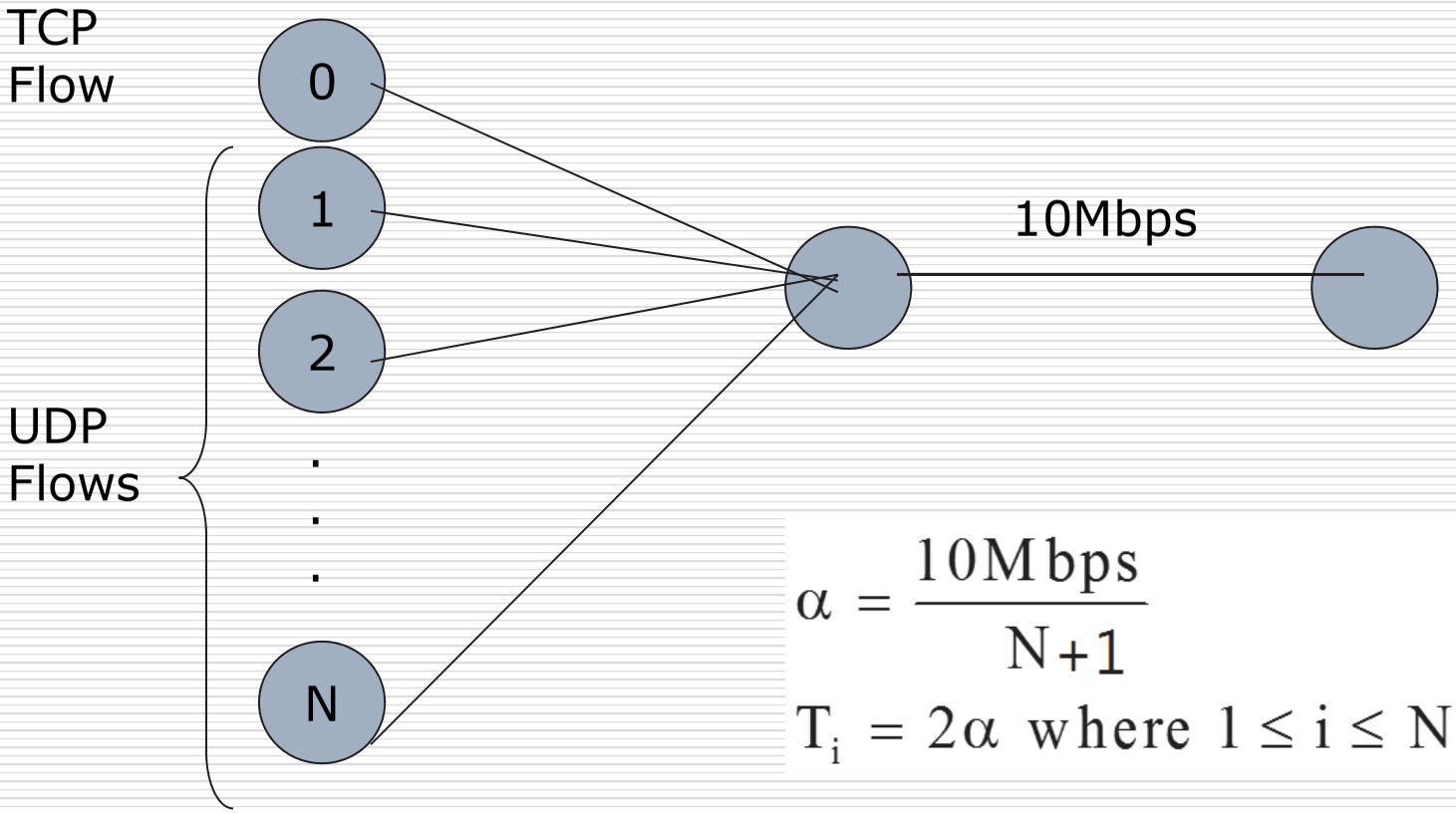


Simulations – Single Congested Link

*The throughputs of one CBR flow (0 indexed) sending at 10 Mbps, and of 31 TCP flows sharing a 10 Mbps link.



Simulations – Single Congested Link

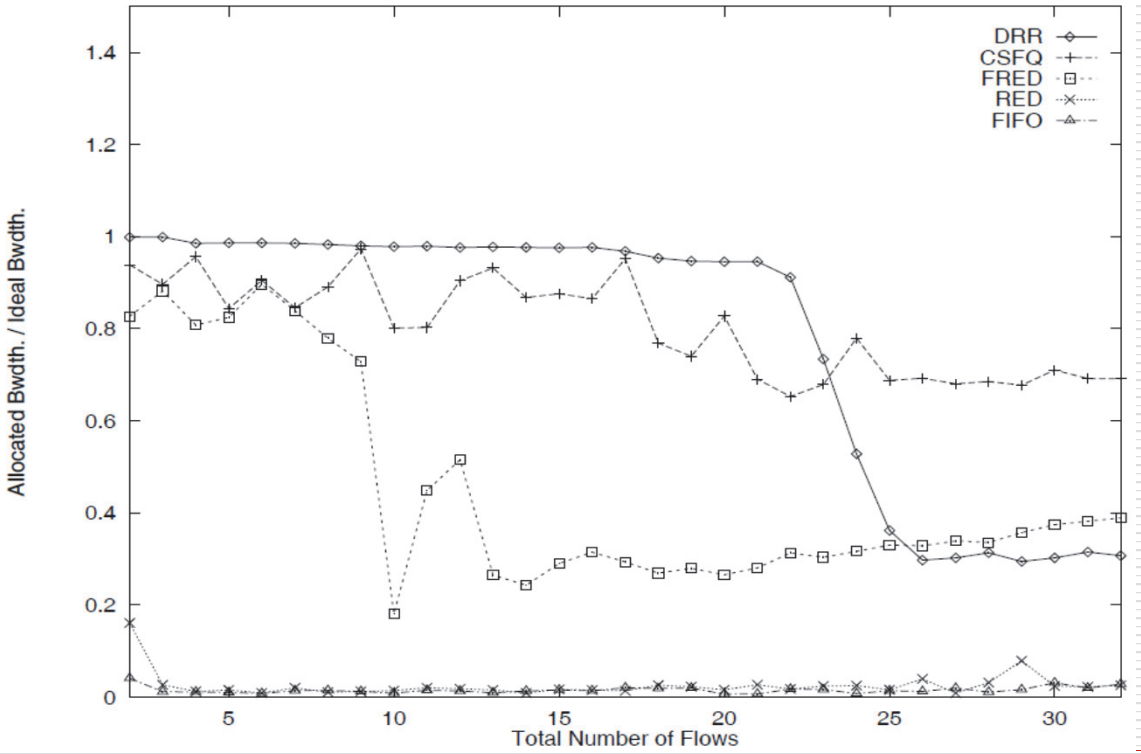


$$\alpha = \frac{10\text{Mbps}}{N+1}$$

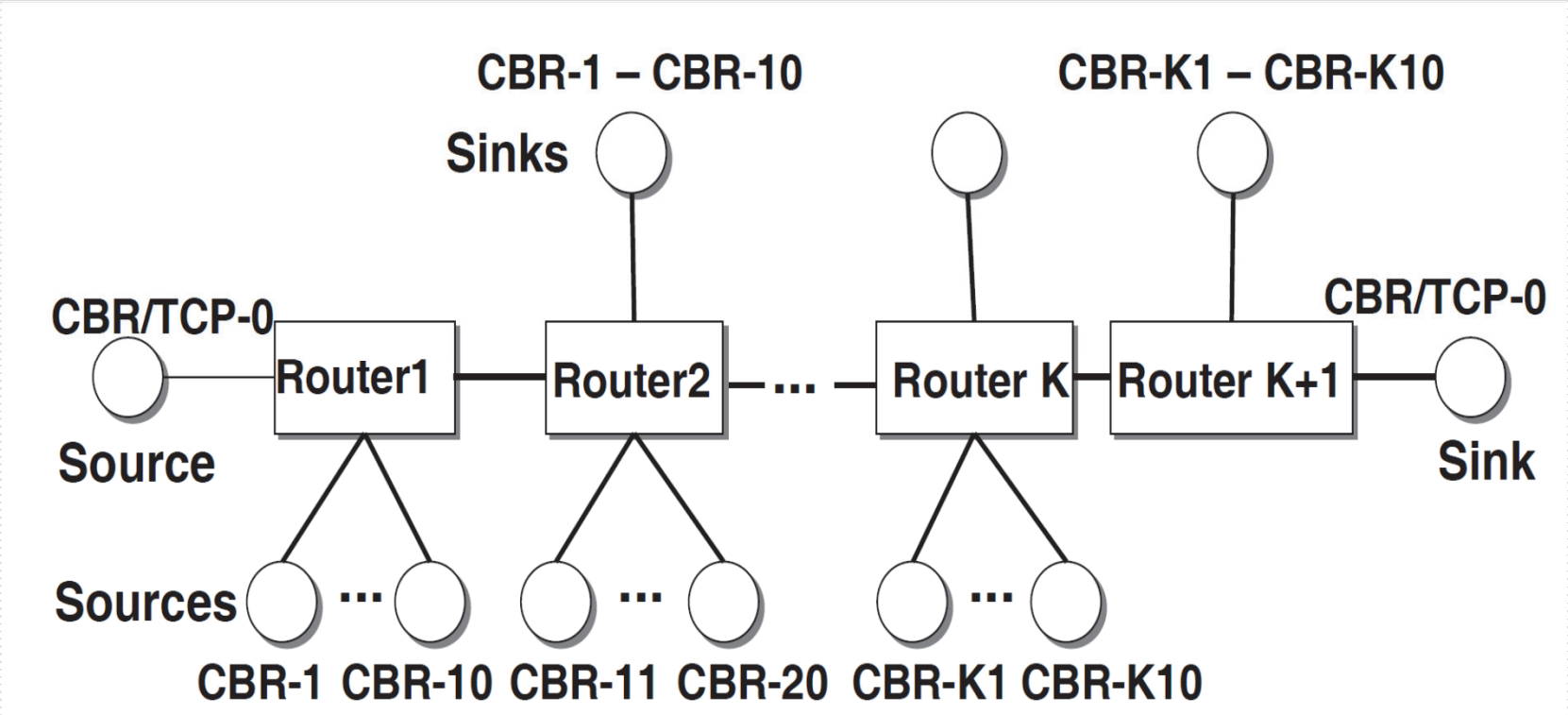
$$T_i = 2\alpha \text{ where } 1 \le i \le N-1$$

Simulations – Single Congested Link

*The normalized bandwidth of a TCP flow that competes with N CBR flows sending at twice their allocated rates, as a function of N
*buffer size=64 KB

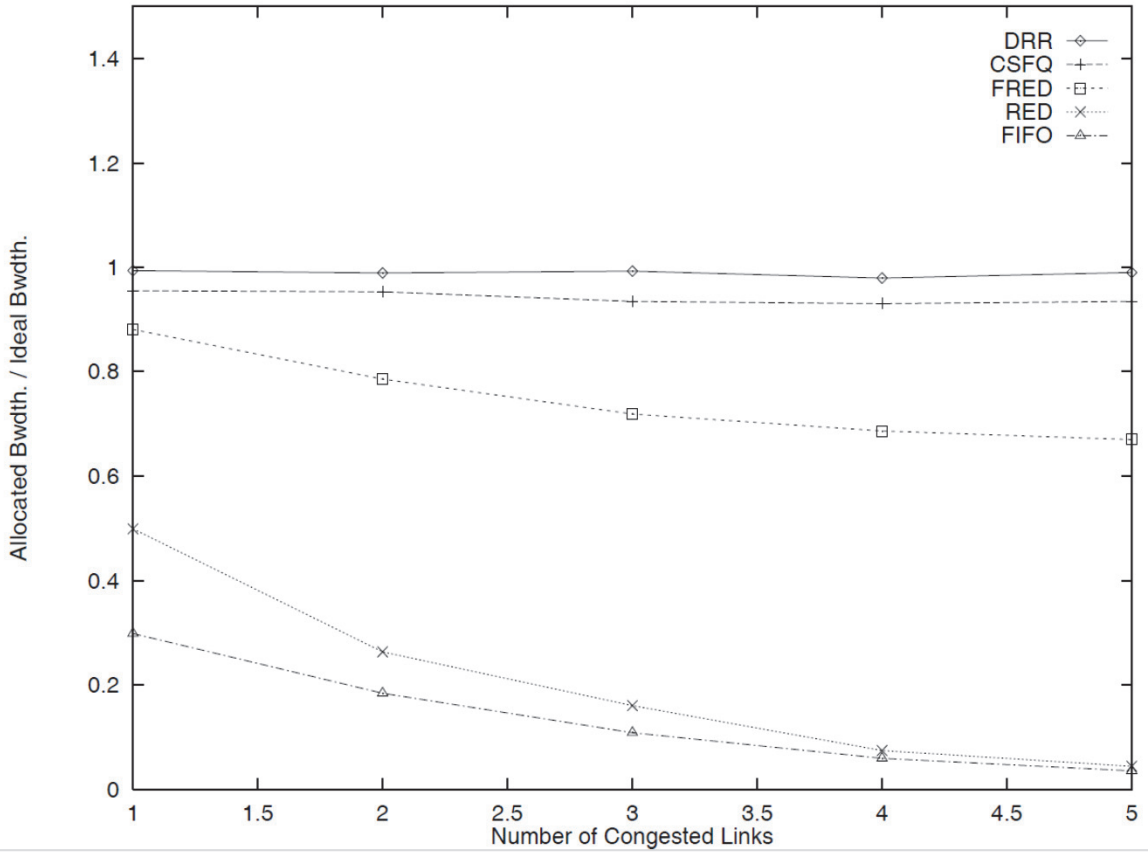


Simulations – Multiple Congested Links



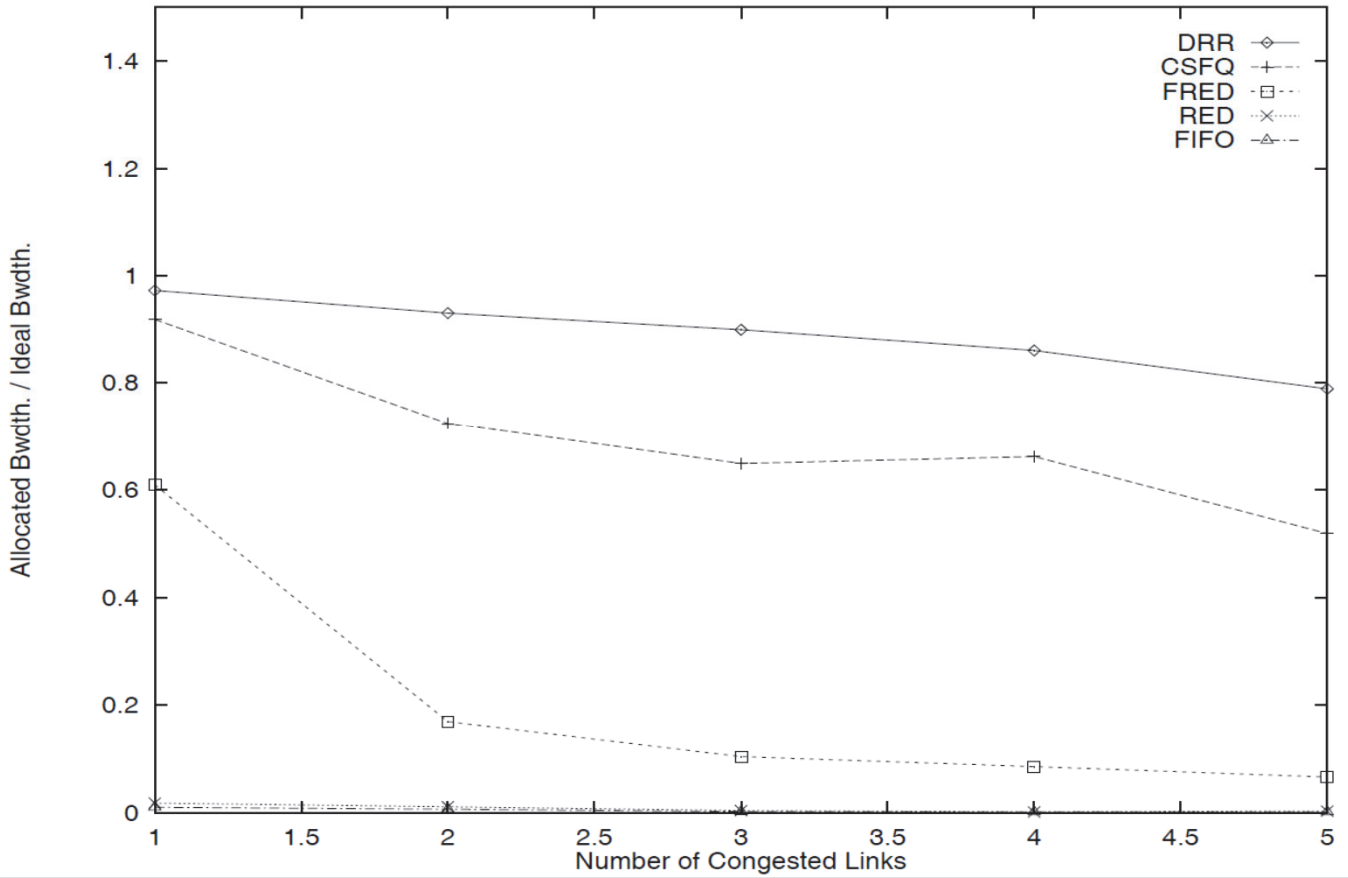
Simulations – Multiple Congested Links

UDP-0 =0.909Mbps Others=2 Mbps



Simulations – Multiple Congested Links

TCP-0



Fair Queueing

- Introduction
- Background
- CSFQ Algorithm
- Simulations
- Evaluations of CSFQ
- Rotating Preference Queues (RPQ)

(2) Ion Stoica, Scott Shenker and Hui Zhang. “Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks.” IEEE/ACM Transactions on Networking, 2003.

Evaluations of CSFQ

- Reasonable approximation of fair share
- Roughly comparable performance to FRED
  - Sometimes much better than FRED
  - FRED has per-packet overhead
- Not quite as fair as DRR

Fair Queueing

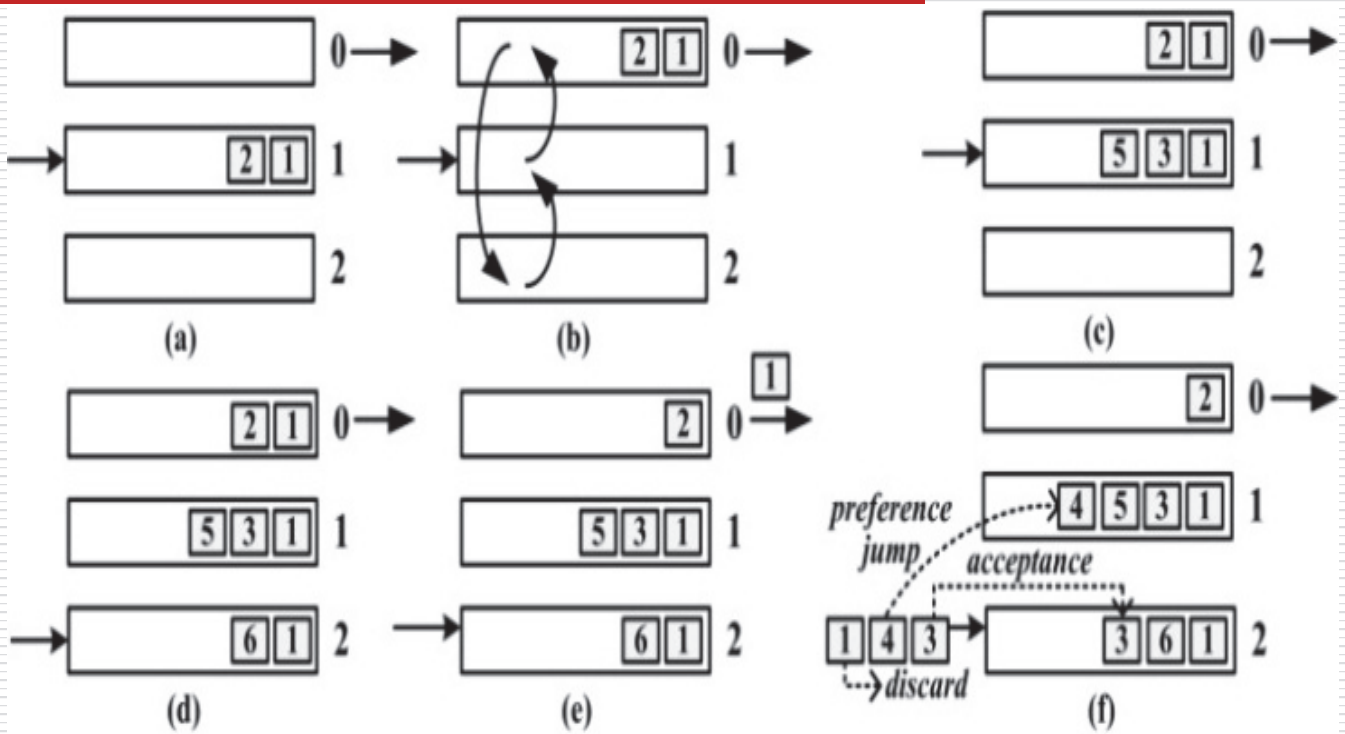
- Introduction
- Background
- CSFQ Algorithm
- Simulations
- Evaluations of CSFQ
- Rotating Preference Queues (RPQ)

(2) Ion Stoica, Scott Shenker and Hui Zhang. “Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks.” IEEE/ACM Transactions on Networking, 2003.

RPQ Introduction

- RPQ consists of a set of FIFO output queues that are dynamically rotated
- RPQ selects accepted packets and dispatches them to adequate output queues depending on packet distribution and preference

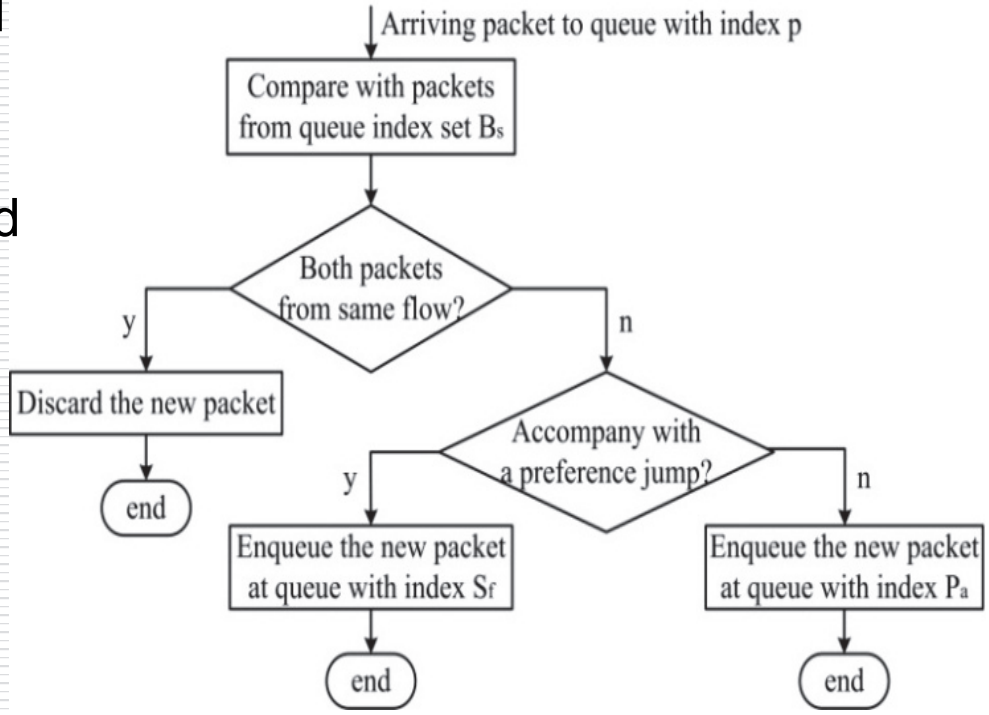
RPQ Example



RPQ example (with 3 FIFO output queues) (a) t = 0 (b) t = 0^+ (c) $0^+ < t < \Delta$ (d) $t = \Delta$ (e) $t = \Delta +$ (f) $\Delta + < t < 2\Delta$

RPQ

- N FIFO output queues, one special queue (indexed by 0) and N-1 general queues (indexed by 1 to N-1)
- Each general queue is associated with a range of arrival times if the packets that arrive during time interval [(p-1)Δ, pΔ) belong to the output queue with index p where Δ denotes a time unit
- When a packet arrives at any time t_a, the arrival belongs to the output queue with an index of $1 + \lfloor (t_a \bmod (N-1)\Delta)/\Delta \rfloor$



RPQ

- RPQ first estimates the amount of discarded packets $\widehat{D}_{new}$ during a constant time interval K_d by (1)

$$\widehat{D}_{new} = \alpha D_{now} + (1-\alpha)\widehat{D}_{old} \qquad (1)$$

- $\widehat{D}_{old}$ is the value that exists before the updating of $\widehat{D}_{new}$
- D_{now} represents the amount of packets discarded during the current time interval
- α is a parameter that determines dependence on short-term or long-term packet discard behavior, where $0 \leq \alpha \leq 1$

RPQ

- Determine which arriving packets are qualified for acceptance. RPQ estimates $\widehat{\mathrm{Q}}_{\mathrm{new}}$ by (2) where $\widehat{\mathrm{Q}}_{\mathrm{new}}$ denotes the number of output queues to be compared

  $$\widehat{\mathrm{Q}}_{\mathrm{new}} = (\widehat{\mathrm{A}}_{\mathrm{new}} - \mathrm{CK_d})\widehat{\mathrm{Q}}_{\mathrm{old}}/\widehat{\mathrm{D}}_{\mathrm{new}} \qquad (2)$$

- $\widehat{\mathrm{Q}}_{\mathrm{old}}$ be the value before the updating of $\widehat{\mathrm{Q}}_{\mathrm{new}}$
- $\widehat{\mathrm{A}}_{\mathrm{new}}$ be the amount of arriving packets
- C denotes the output link capacity of a router

RPQ

- Equation (3) is revised from (2) leading to a simple $\widehat{\mathrm{Q}}_{\mathrm{new}}$
- The upper bound of the compared queues is N-1

$$\widehat{\mathrm{Q}}_{\mathrm{new}} = \begin{cases} \min\left(\left\lceil \widehat{\mathrm{Q}}_{\mathrm{new}} \right\rceil, \mathrm{N}-1\right), & \mathrm{rand}(0,1) \le \left(\widehat{\mathrm{Q}}_{\mathrm{new}} - \left\lfloor \widehat{\mathrm{Q}}_{\mathrm{new}} \right\rfloor\right) \\ \min\left(\left\lfloor \widehat{\mathrm{Q}}_{\mathrm{new}} \right\rfloor, \mathrm{N}-1\right), & \text{else} \end{cases} \quad (3)$$

RPQ

- After $\widehat{Q}_{new}$, RPQ begins comparing the arriving packet with residing packets that belong to the queue index set B_s

$$B_s = \sum_{i=1}^{\widehat{Q}_{new}} (p - i + N + 1) \bmod N \quad (4)$$

RPQ

- Without a preference jump, the new packet will be enqueued at a queue with an index of p_a according to (5)
- p_a denotes the queue with the current minimum index

$$p_a = \min(p, \max(1, (p - \hat{Q}_{new} + N + 1) \bmod N) \quad (5)$$

RPQ

- If a preference jump occurs, the new packet will be enqueued at the queue with an index of S_f according to (6)
- S_f denotes the queue with the least minimum index related to the new packet
- $q_{p,j}$ denotes the current queue length of flow j related to the queue with an index of p, R_p denotes the residual buffer size of the queue with an index of p, and pkt_{size} denotes the packet size of the new packet.

$$S_f = \left\{ \min(p) \middle| q_{p,j} = 0 \text{ and } R_p \geq pkt_{size},\ p \in [1, Pa) \right\} (6)$$

RPQ

- RPQ uses FIFO scheduling and only transmits residing packets from a queue with an index of 0; hence, it can prevent packets residing in larger indexes from experiencing bandwidth starvation
- Whenever a queue with an index of 0 is empty, RPQ dynamically rotates the indexes of all the output queues by (7)

$$p = (N + p - 1) \bmod N,\ p \in [0, N - 1] \quad (7)$$

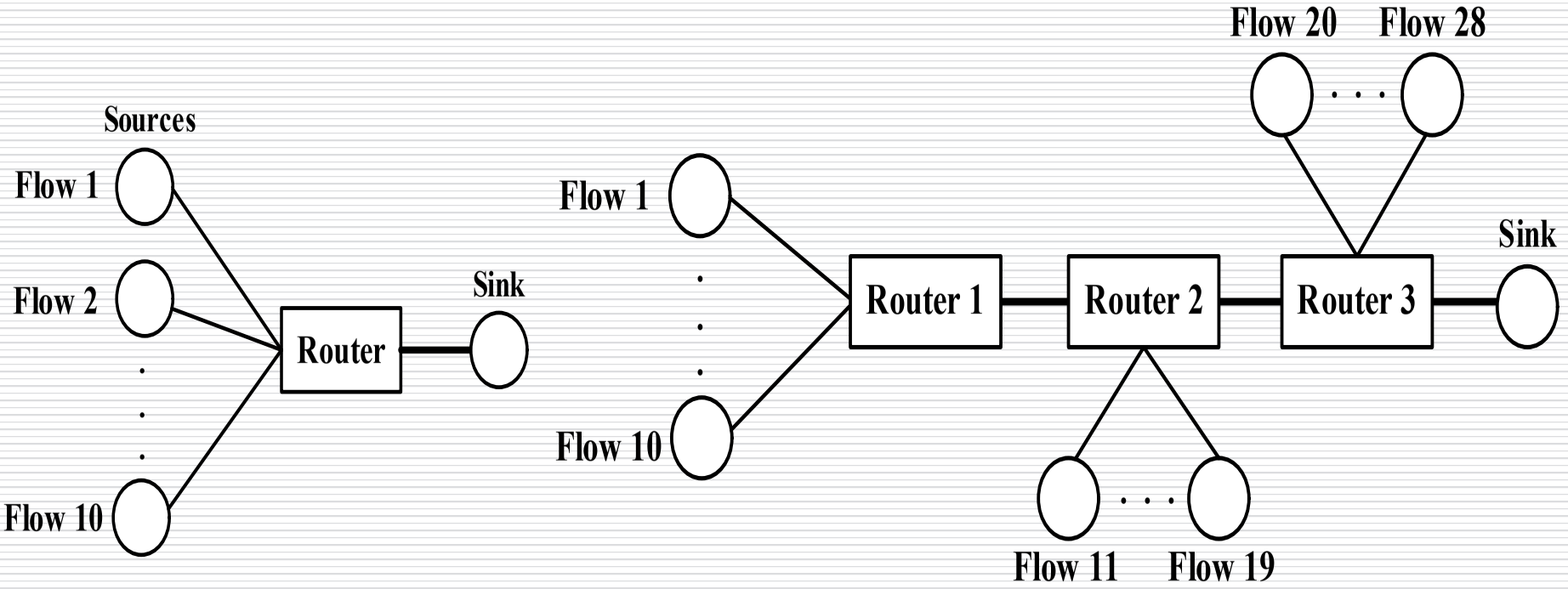
Definition of Fairness

- NBR, D_j denotes the mean departure rate, and r_j denotes the mean arrival rate related to flow j
- f denotes the max-min fair share rate

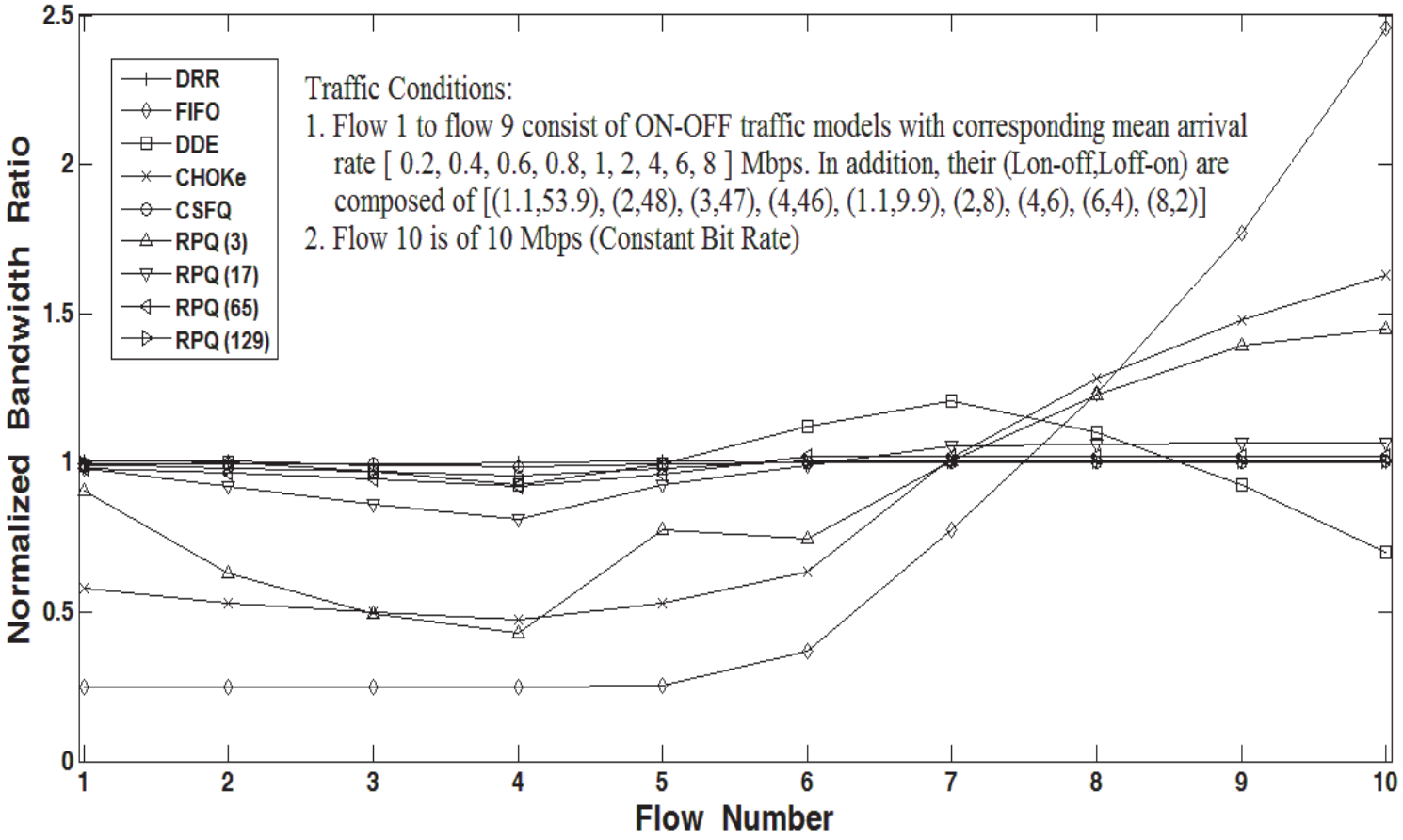
$$NBR_j = D_j / \min(r_j, f) \quad (8)$$

$$\sum_{j=1}^{n} \min(r_j, f) = C \quad (9)$$

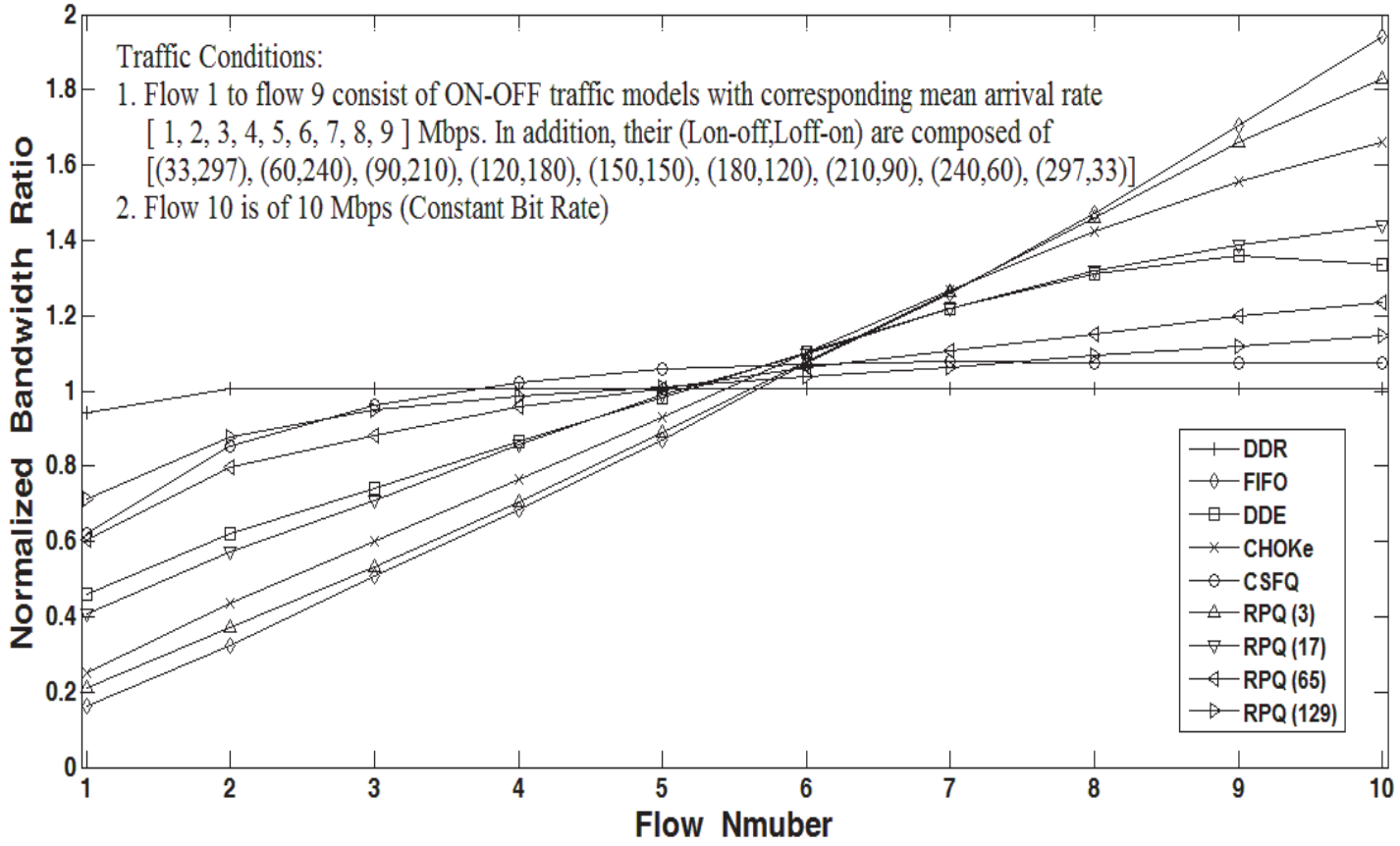
Network Topologies



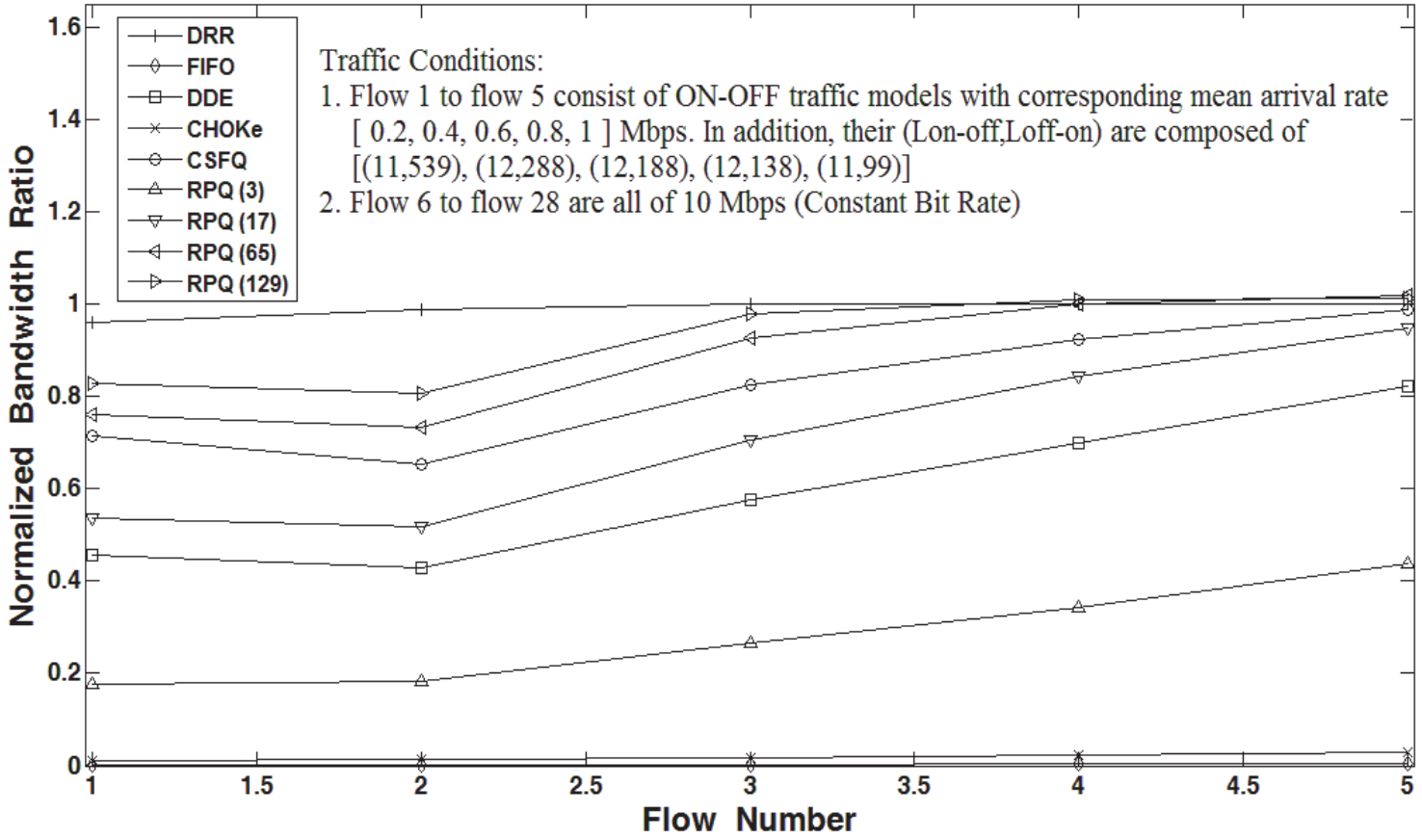
A single congested link



A single congested link



Multiple Congested Links



RPQ-Conclusion

- RPQ provides a significant degree of fairness in our simulations. While not precisely matching the fairness benchmark of DRR, RPQ is comparable or superior to CSFQ, and it works much better than DDE, CHOKe and FIFO
- we would like to extend the RPQ by considering packets with different loss priorities and by studying the impact on throughput and packet order by a router that is composed of multiple network processors

Outline

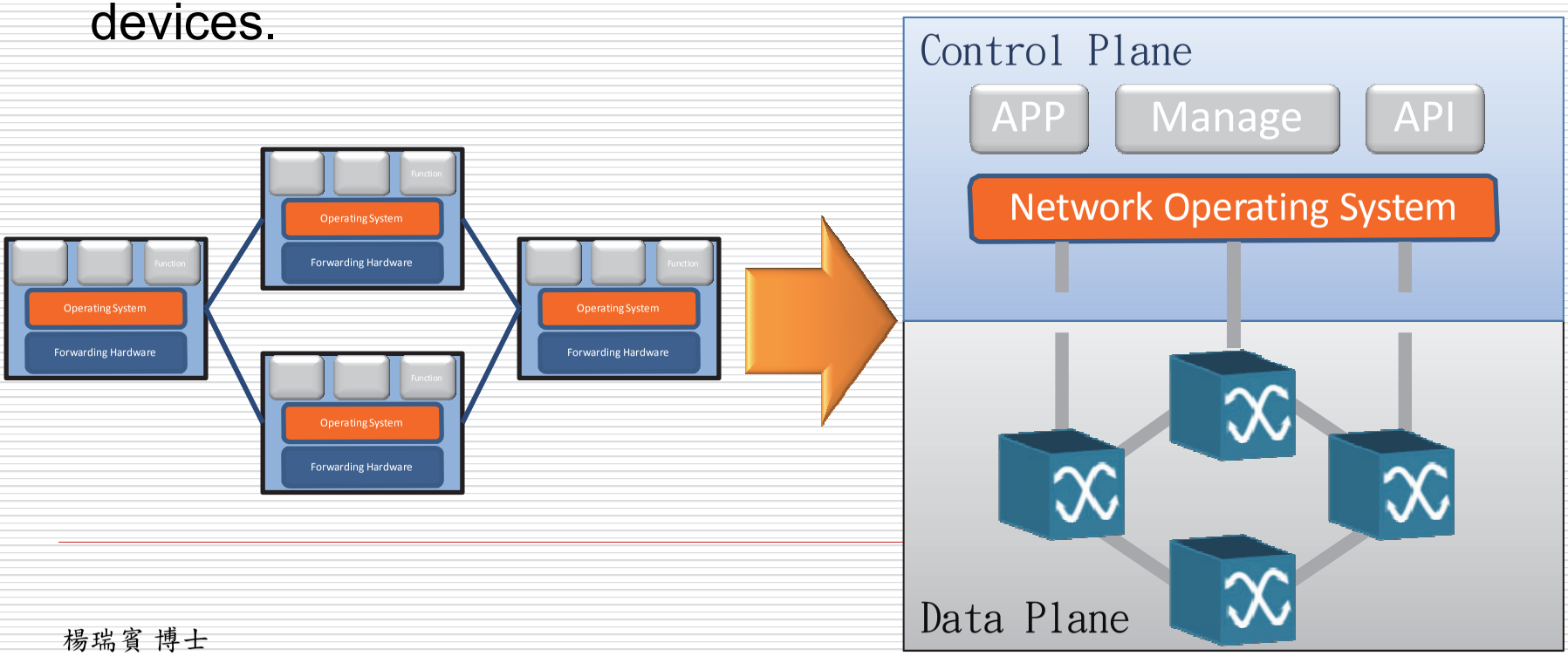
- Network Technology
  - Fair Queueing
  - OpenFlow Protocol
- Cloud Storage
  - Load Balancing
- Conclusion

Software Defined Network (SDN)

- SDN是一個新的網路模式用來解決現有網路的問題

  SDN定義:

  – The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices.



SDN 優勢

Programmability

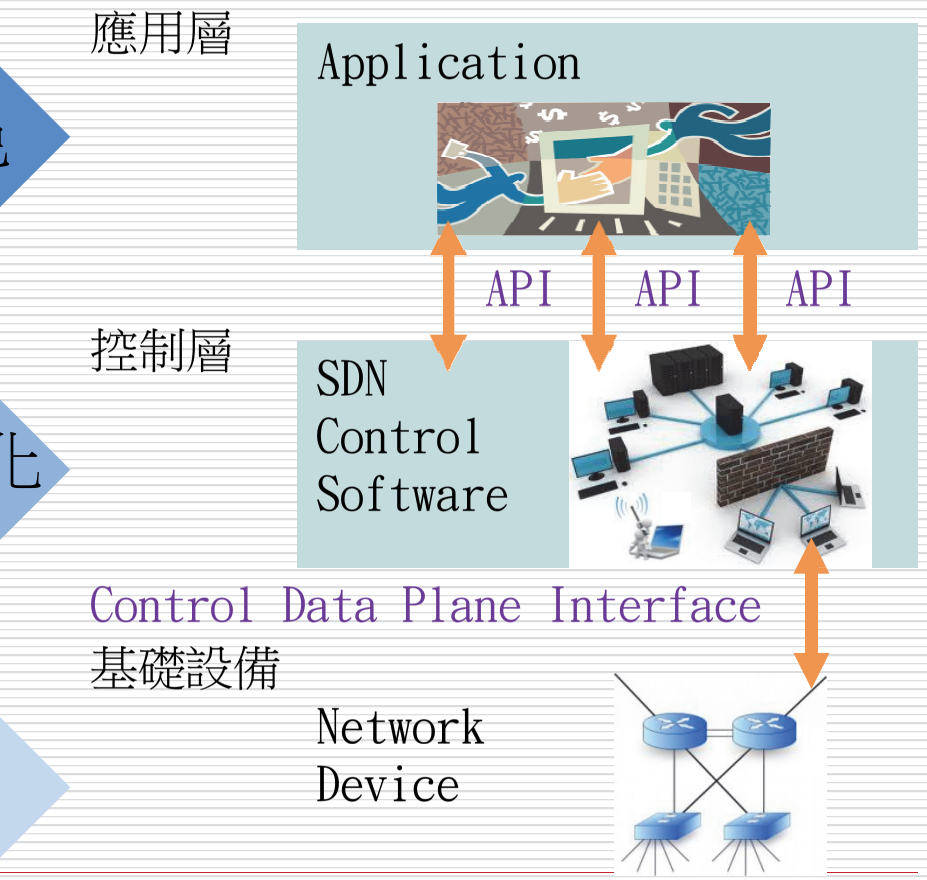
發揮創意、增加服務與特色

Centralized Intelligence

容易供裝/維運、效能最佳化

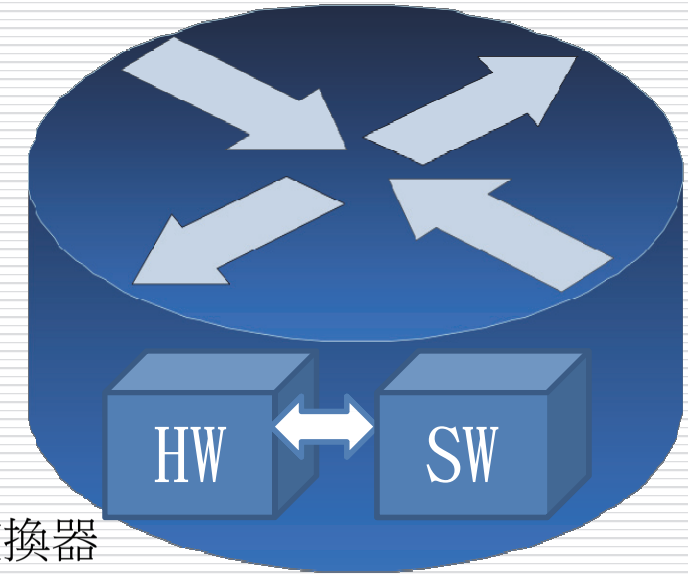
Abstraction

軟/硬體分離、抽象化配置



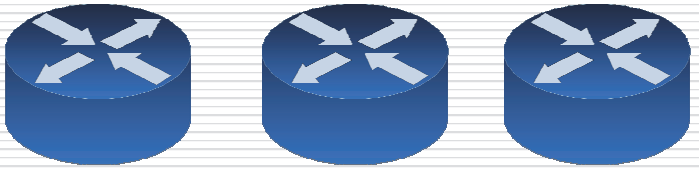
OpenFlow (OF)

- Data plane的控制協定
- OF是實現SDN的方法並不能完全代表SDN
- OF Controller + OF Switch



傳統交換器

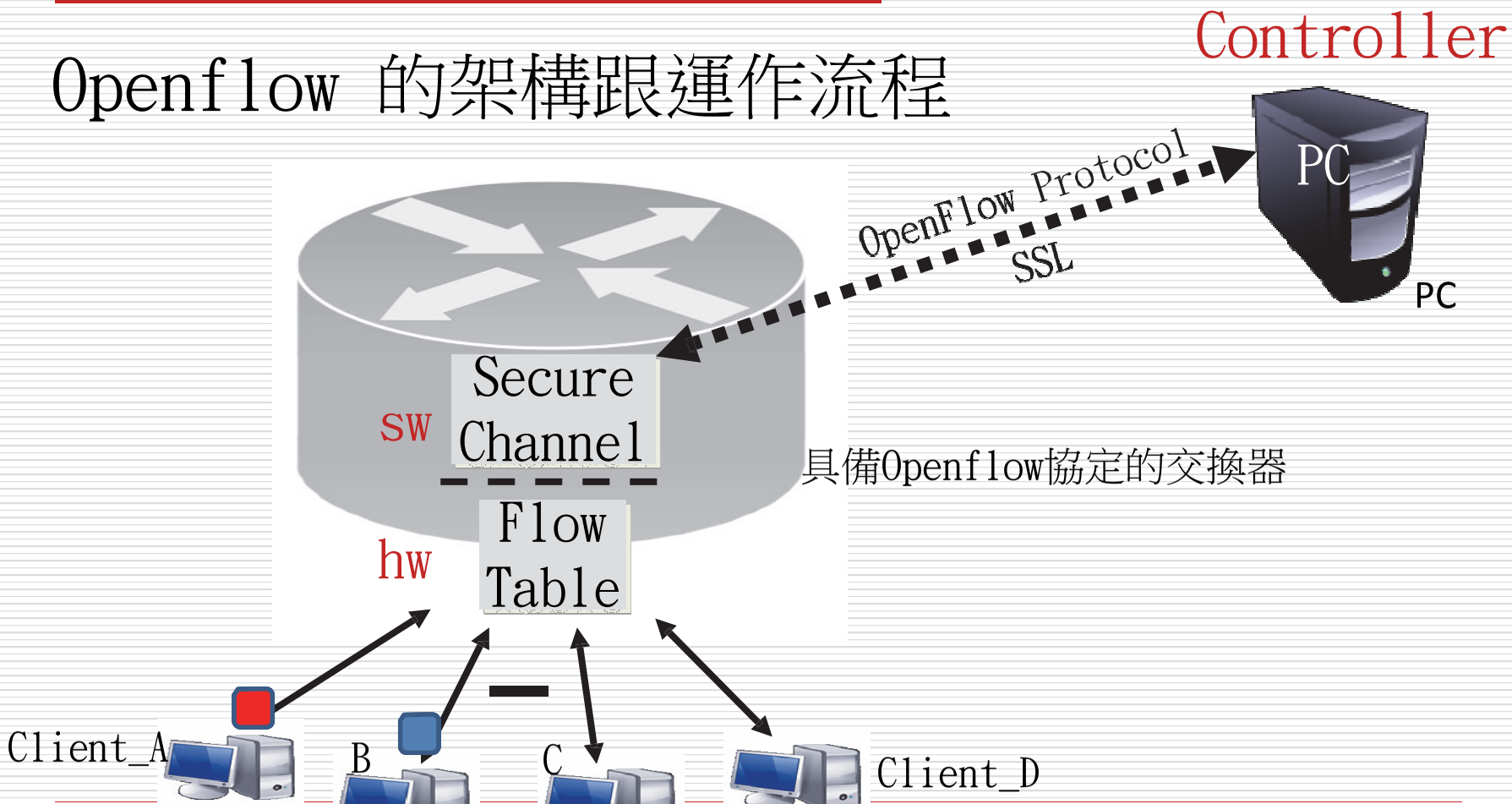
Openflow控制器



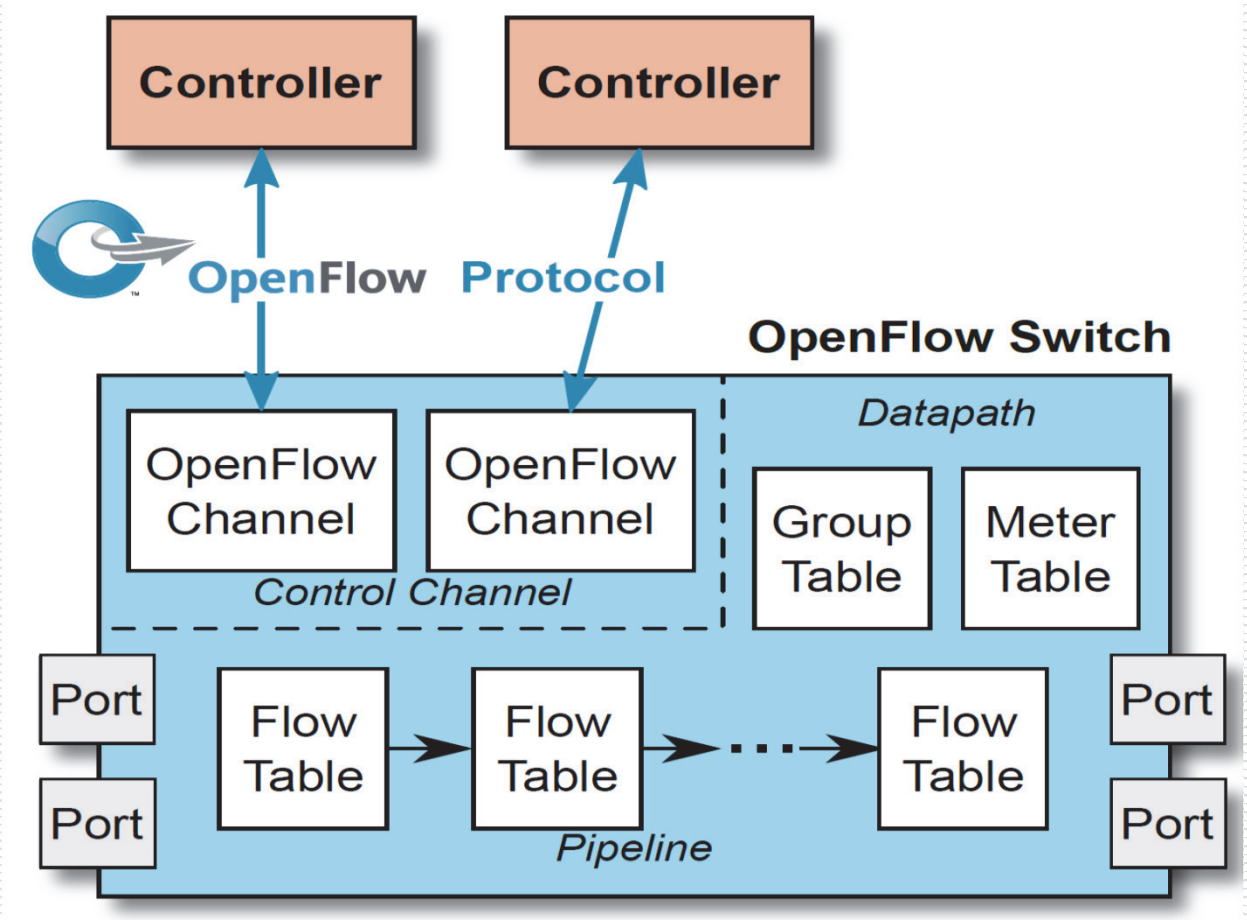
Openflow交換器

OpenFlow

Openflow 的架構跟運作流程



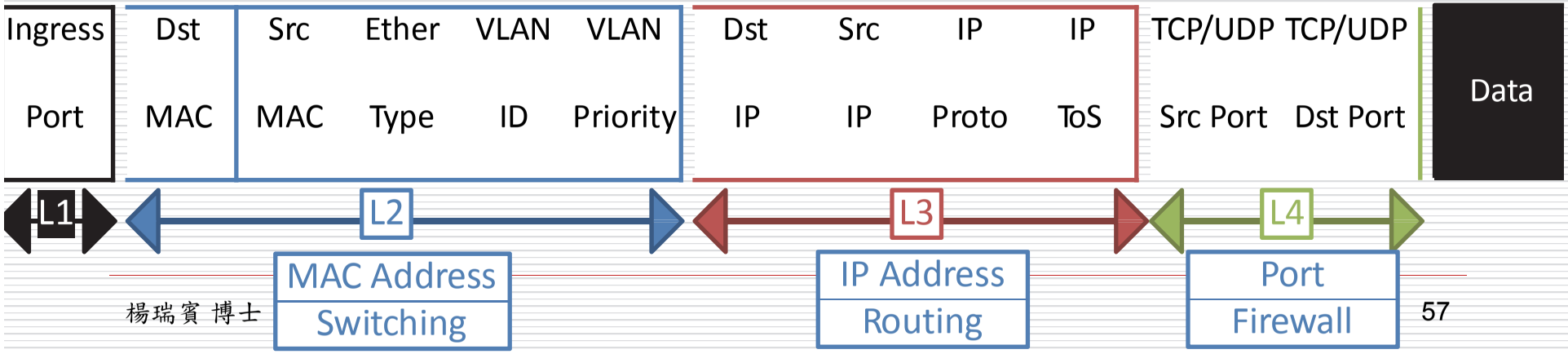
OpenFlow Switch



Flow Table

- Flow Table
  -v1.5

- IP網路：透過辨識L2/L3位址傳遞封包
- Openflow：透過Flow傳遞



Examples

Flow Name	Switch Port	MAC Src	MAC Dst	Eth type	VLAN	IP Src	IP Dst	TCP Src	TCP Dst	Priority	Action
SwitchA_D	1	00:A1..	00:D1..	0800	1	*	*	*	*	32768	Output =4
SwitchD_A	4	00:D1..	00:A1..	0800	1	*	*	*	*	32768	Output =1
Firewall	*	*	*	*	*	**	*	22	*	1	Drop
Routing	*	*	*	*	*	*	1.1.1.1	*	*	32768	Output =5
VLAN Switching	*	*	00:B1..	*	2	*	*	*	*	32768	Output =2,3

OF控制器種類

Name	Programming language	License	Comment
NOX	C++	GPL	Initially developed at Stanford University
POX	Python	Apache	Forked from the NOX controller. POX is written in Python and runs under various platforms.
Beacon	Java	BSD	Initially developed at Stanford.
Floodlight 中華電信	Java	Apache	Forked from the Beacon controller and sponsored by Big Switch Networks.
Maestro	Java	LGPL	Multi-threaded OpenFlow controller developed at Rice University.
NodeFLow	JavaScript	MIT	JavaScript OpenFlow controller based on Node.JS.
Trema	C and Ruby	GPL	Plugins can be written in C and in Ruby. Trema is developed by NEC.
OpenDaylight	Java	EPL	OpenDaylight is hosted by the Linux Foundation, but has no restrictions on the operating system.

Mainstream

Outline

- Network Technology
  - Fair Queueing
  - OpenFlow Protocol
- Cloud Storage
  - Load Balancing
- Conclusion

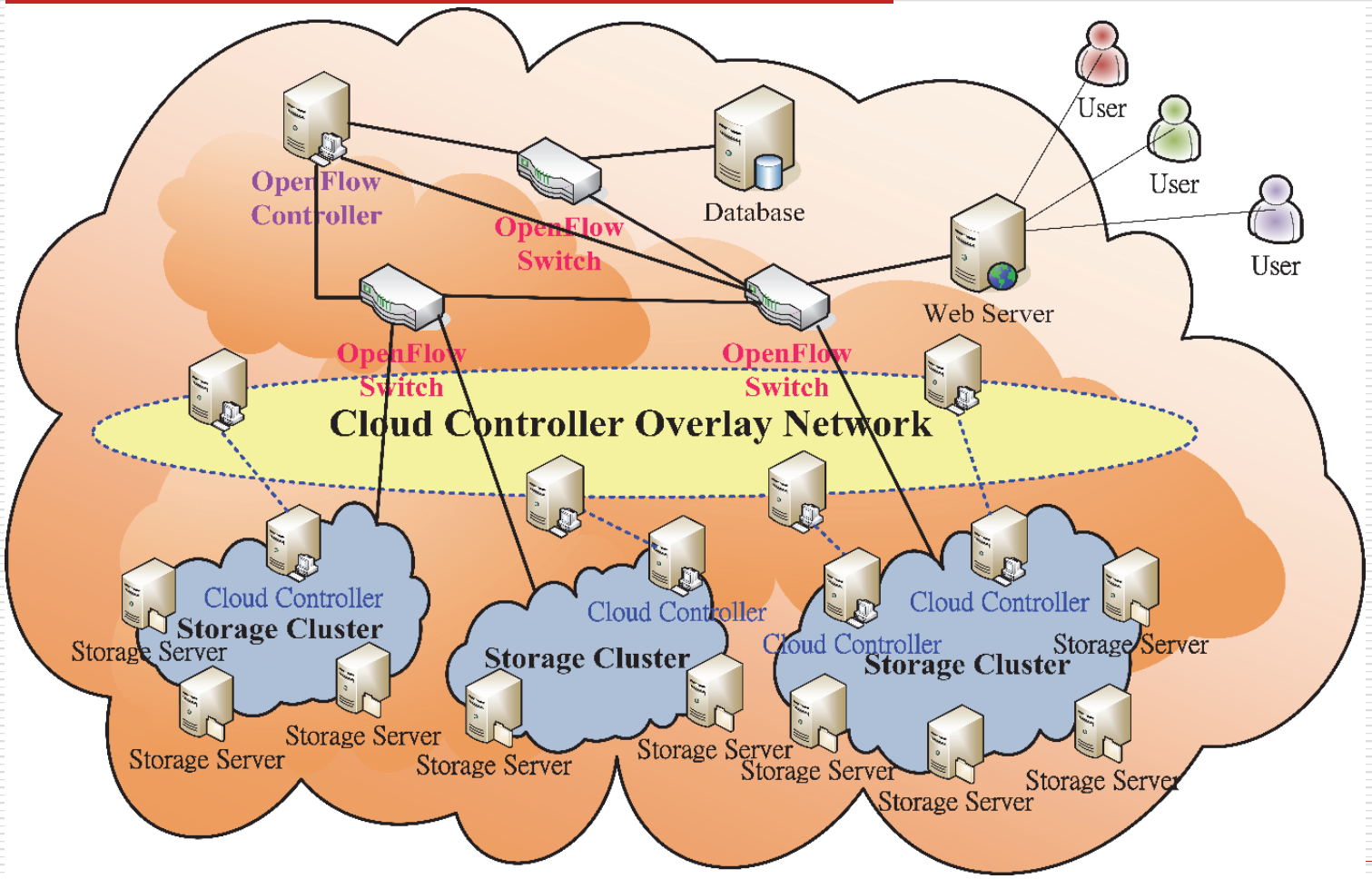
雲端儲存服務供應商之儲存服務、特性與介面

供應商	Amazon	Microsoft	Nirvana	AT&T	Sun Micro	Google
儲存服務	Amazon Simple Storage Service	Azure storage services	Storage Delivery Network	Synaptic Storage	Sun Cloud Storage Service	Google Cloud Drive
特性	Scalable Reliable, Low-latency	Scalable storage type	Secure, Reliable	Secure, Availability	Availability, Detailed metering, Fast Cloning	Secure, Reliable, Share
介面	HTTP, REST/SOAP API	HTTP, REST API	HTTP	HTTP, REST API	HTTP, REST API	HTTP, REST API

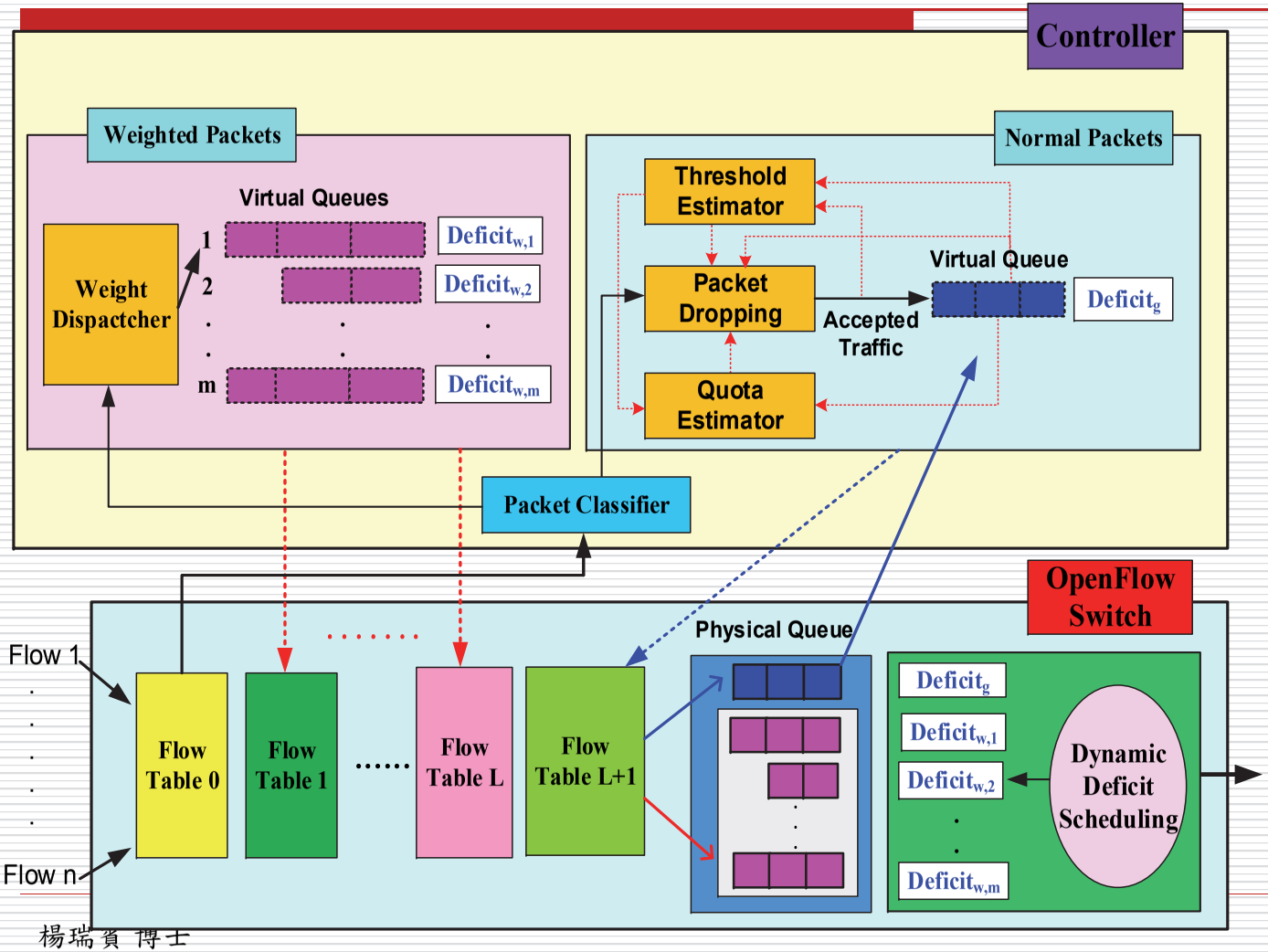
Problems

- 雲端儲存服務供應商的服務等級協定(Service Level Agreement，SLA)，包括Amazon，Microsoft以及Google。僅提供簡易的服務等級協定能力，那就是保證服務正常運行時間，例如；95%，99%或是99.9%
- 發生違約情況，給予使用者不同比例的服務費用折扣(10%~25%)或是免費延長服務天數

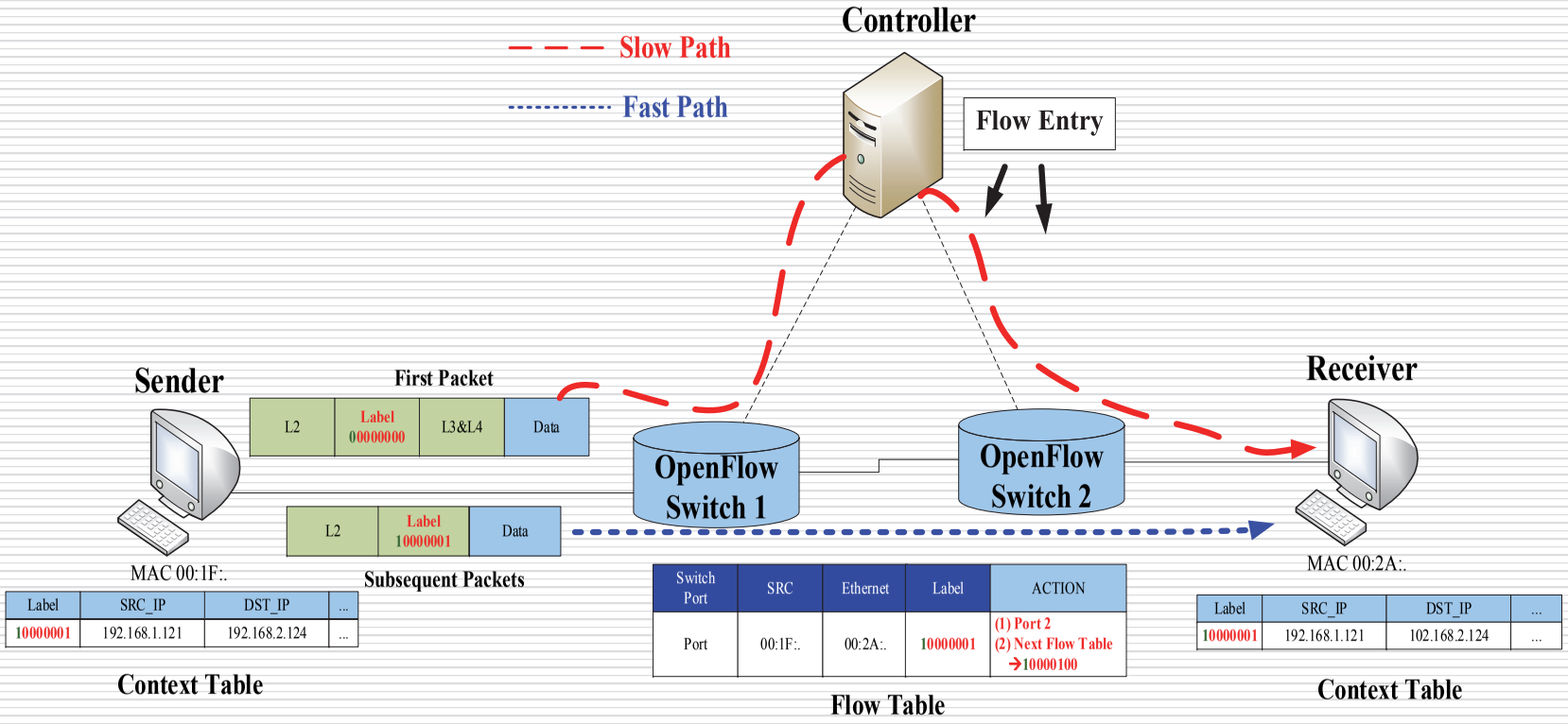
雲端儲存平台架構


OpenFlow Controller
OpenFlow Switch
Database
User
User
User
Web Server
OpenFlow Switch
OpenFlow Switch
Cloud Controller Overlay Network
Cloud Controller
Storage Cluster
Storage Server
Storage Server
Storage Server
Cloud Controller
Storage Cluster
Storage Server
Storage Server
Cloud Controller
Cloud Controller
Storage Cluster
Storage Server
Storage Server
Storage Server

OpenFlow研究(一)

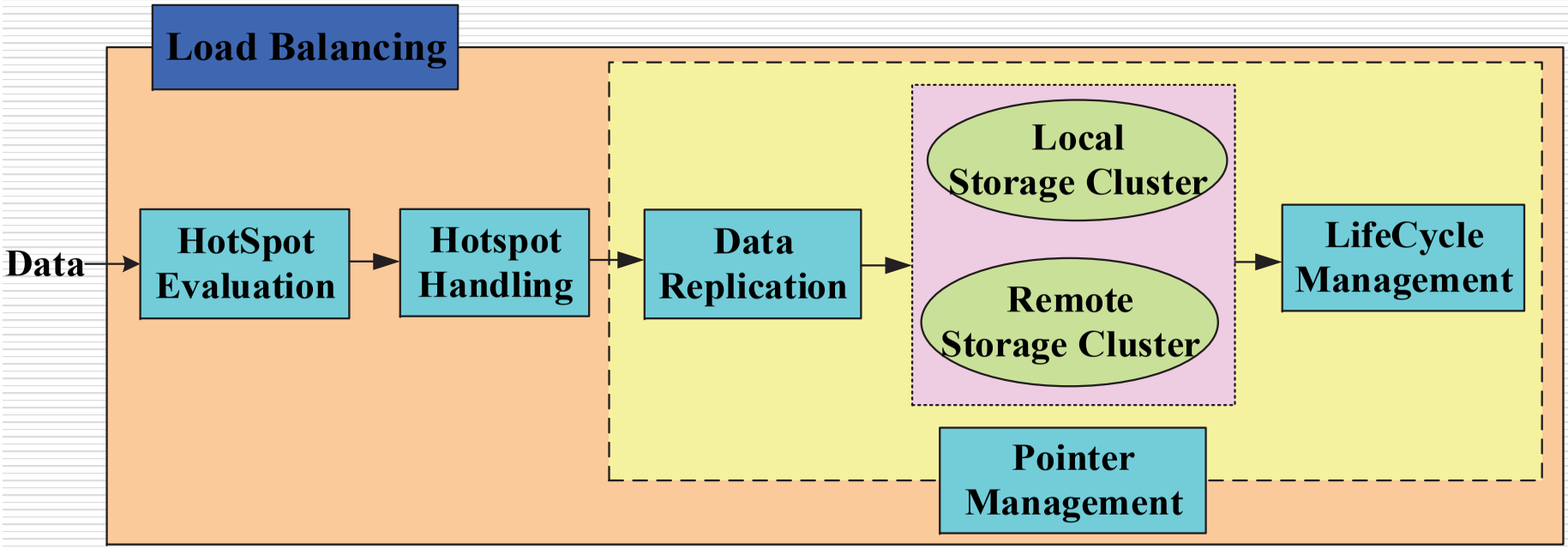


OpenFlow研究(二)



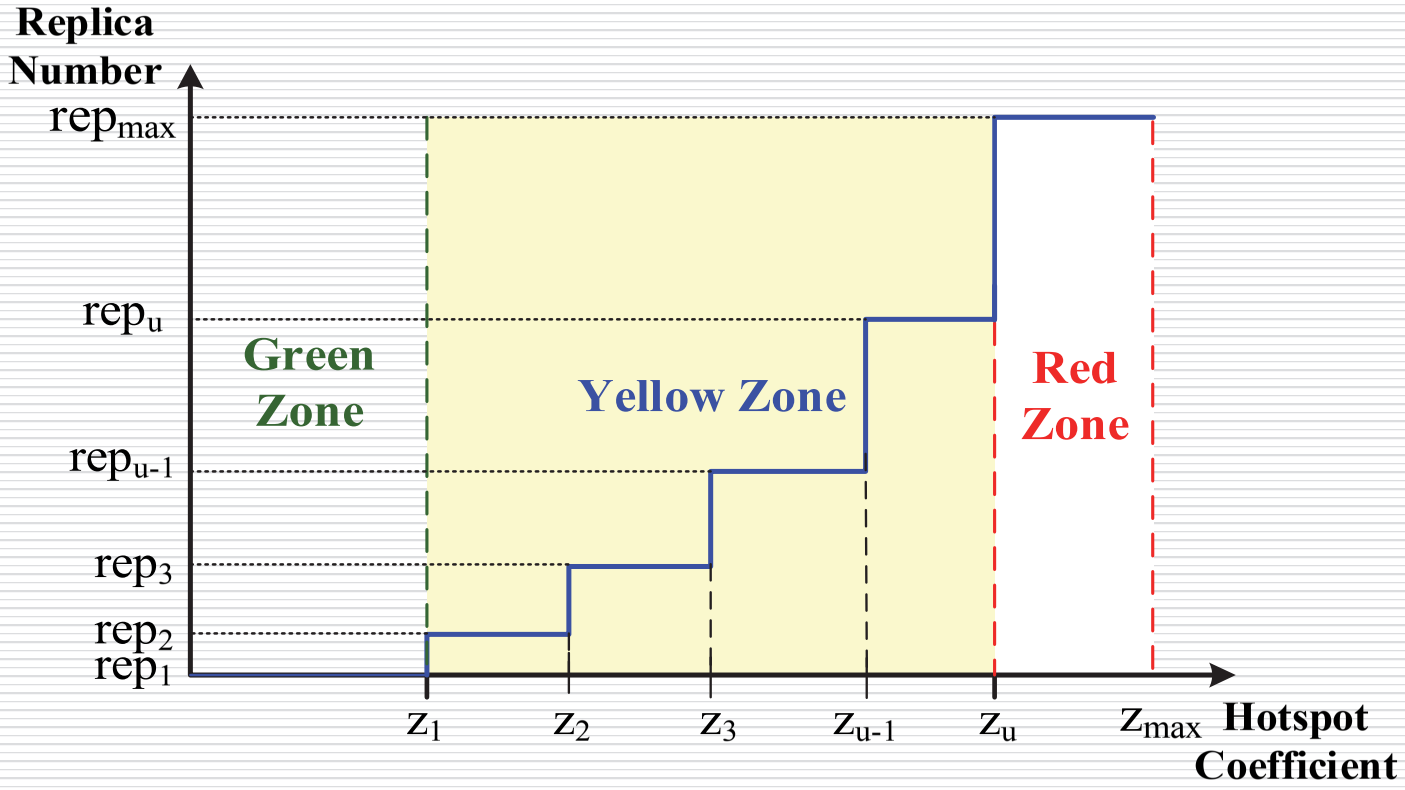
快速封包篩選機制

Load Balancing

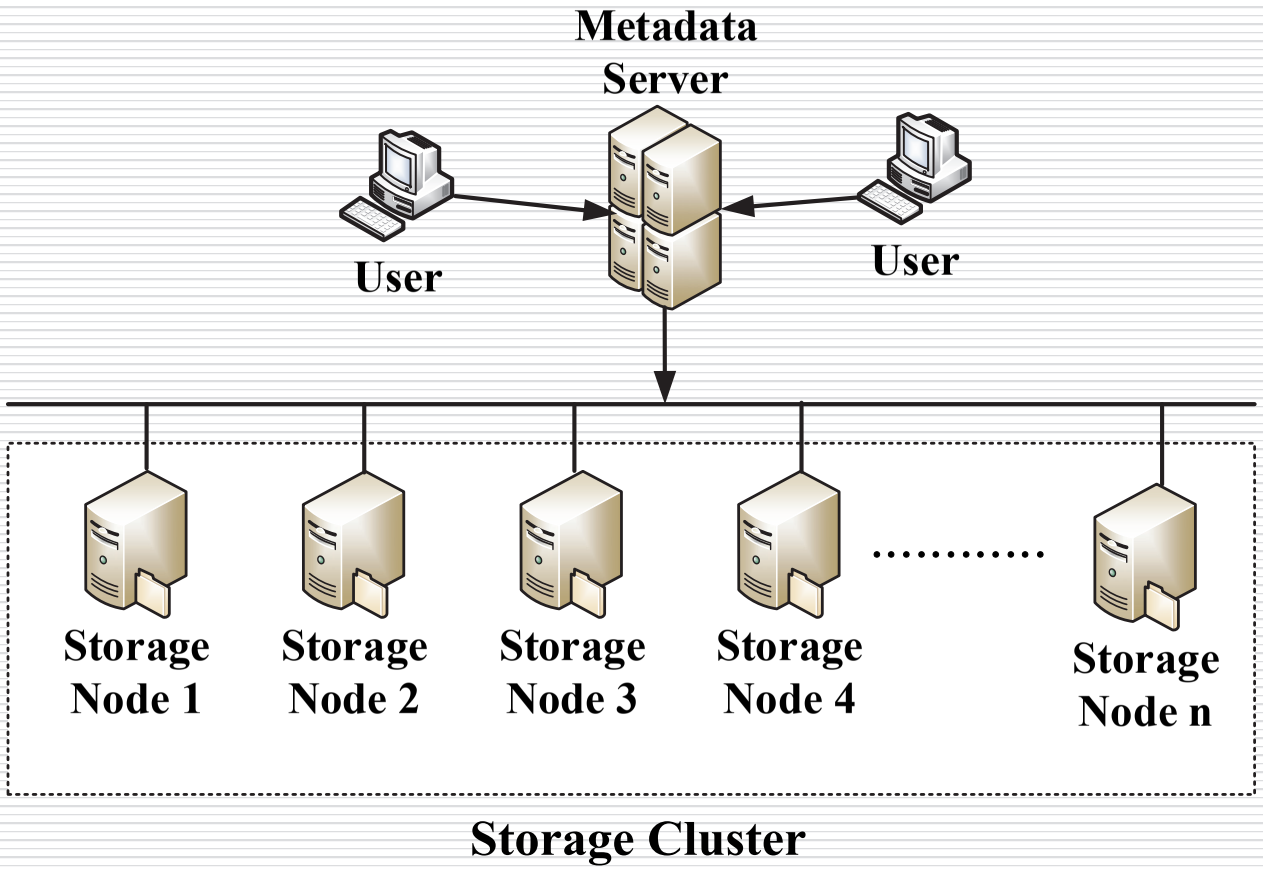


Load Balancing Research

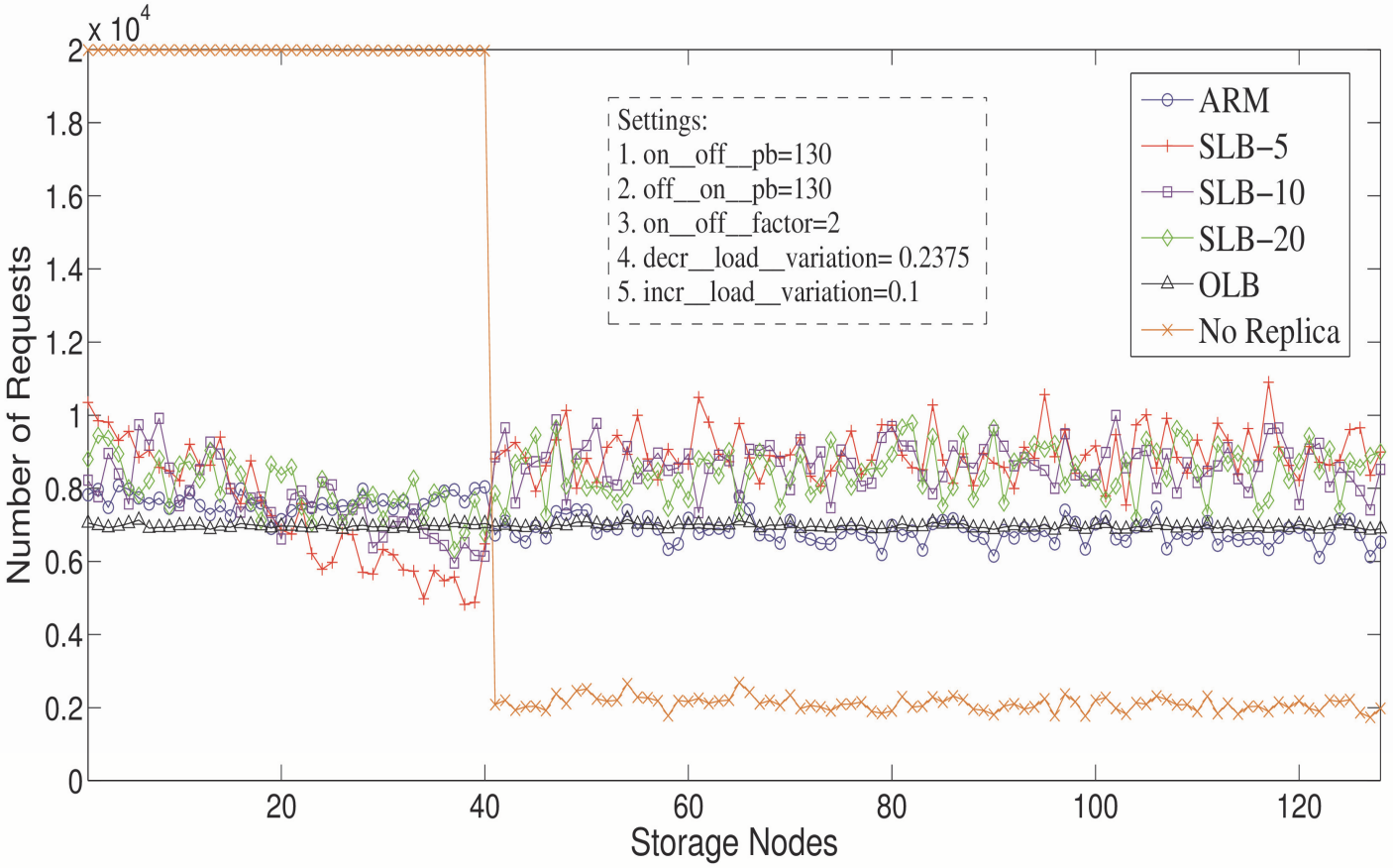
Active Hotspot detection→dynamic replica allocations



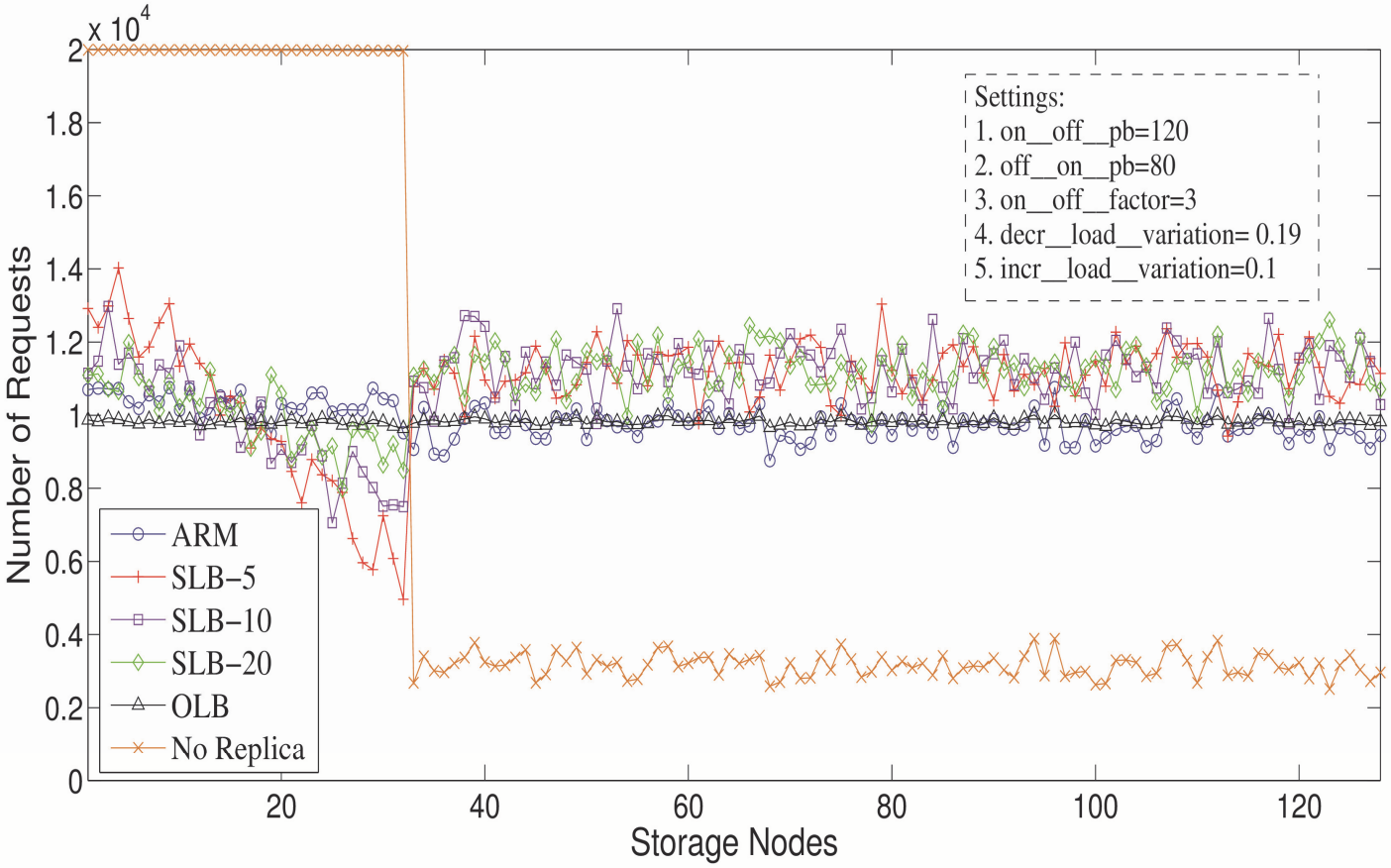
Storage cluster architecture



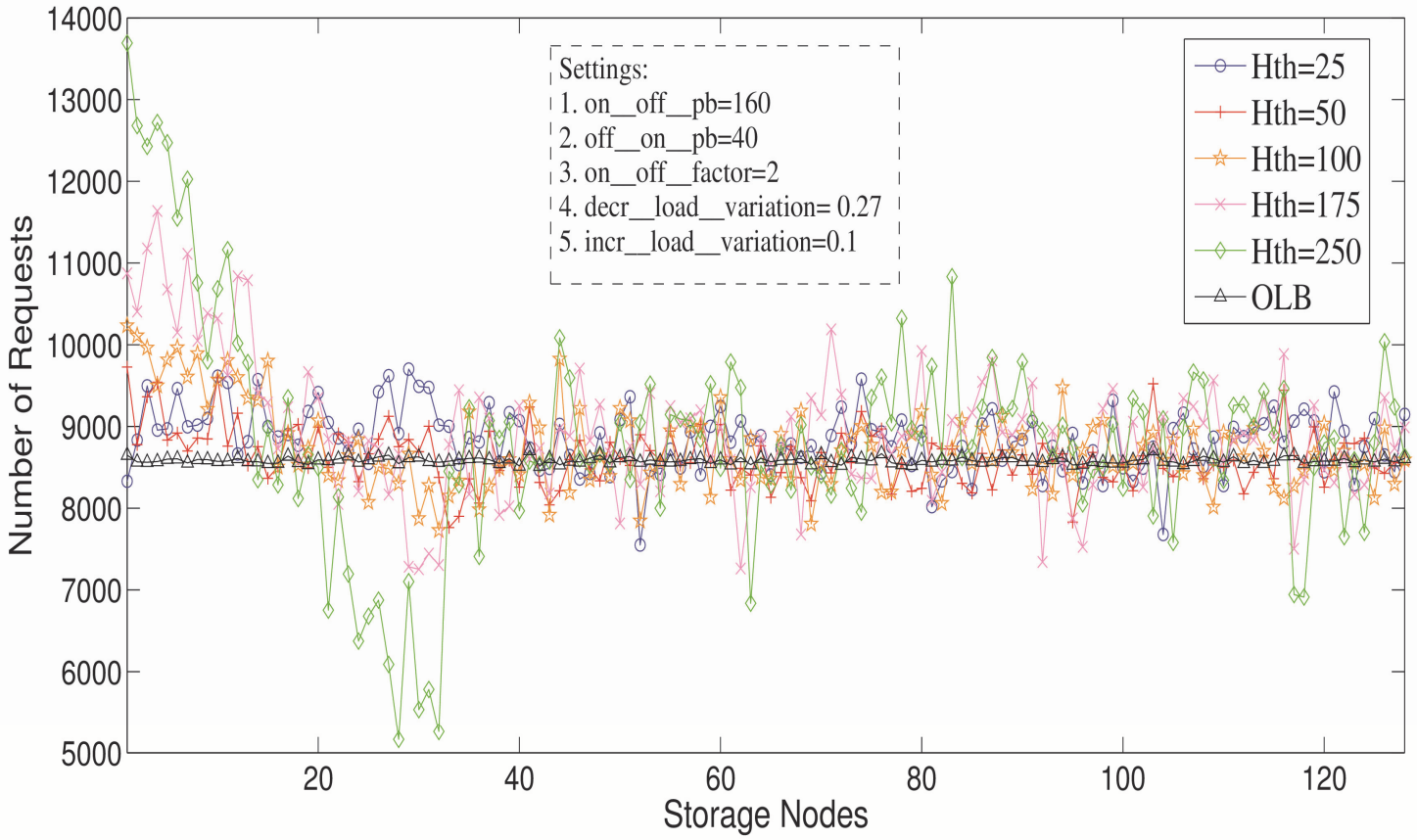
Simulation Results



Simulation Results



Simulation Results



Outline

- Network Technology
  - Fair Queueing
  - OpenFlow Protocol
- Cloud Storage
  - Load Balancing
- Conclusion

Conclusion-Research

Service-Level Agreement

- Resource Metrics and SLA Parameters Planning (Resource Utilization)
- SLA Template Design (Web SLA Language)
- SLA Mapping (SLA Negotiation between Users and Providers)
- SLA Monitoring (QoS Guarantee)

Dynamic Resource Management

- Request Scheduling (QoS Guarantee)
- Load Balancing (Cache Allocation, Hotspot Handling, Data Migration and LifeCycle Management, different storage clusters)
- Data Replication (Data Reliability and Hotspot Effect)

System Development

- SMI-S (Storage Management Initiative Specification)
- CDMI (Cloud Data Management Interface)
- OpenFlow

Conclusion-競賽

- 2015經濟部技術處搶先大賽
  1. 智慧停車位系統(室內定位設備,立體停車場業者)
  2. 結合Game與AR技術旅遊導覽服務(手機與AR軟體,店家,旅行社以及各縣市旅遊單位)
  3. 老人居家照護管理系統(Server,社福單位,老人保健藥商)
  4. APP集點系統(PC與NFC設備,多那之pos廠,seven)
  5. Face to Face FB(手機與Server,大眾,廣告商)

Q&A