รายการอ้างอิง

[1] Brian D. Davison, A Survey of Proxy Cache Evaluation Techniques, Web Caching Workshop, March 1999.

[2] Junbiao Zhang, Rauf Lzmailov, Daniel Reininger, Maximilian Ott, WebCASE: A simulation environment for web caching study, Web Caching Workshop, March 1999.

[3] Bajaj S., Breslau L., Estrin D., Fall K., Floyd S., Haldar P., Iiandley M., Helmy A., Heidemann J., Huang P., Kumar S., McCanne S., Rejaie R., Sharma P., Shenker S., Varadhan K., Yu H., Xu Y., and Zappala D., Virtual InterNetwork Testbed: Status and research agenda. Tech. Rep. 98-678, University of Sounthern California, [online] Available from: http://www.isi.edu/nsnam/ns [1998,July].

[4] David Wetherall and Christopher J. Linblad, Extending Tcl for dynamic objec-oriented programming, In Proceedings of the USENIX Tcl/Tk Workshop, page 288, Toronto, Ontario, July 1995, USENIX.

[5] Hypertext Transfer Protocol - HTTP/1.0, RFC 1945.

[6] D. Wessels, and K. Claffy, "ICP and the Squid web cache," IEEE Journal on Selected Areas in Communications, vol. 16, no. 3, pp. 345-357, Apr. 1998.

[7] V. Valloppillil, and K. W. Ross, "Cache Array Routing Protocol v1.0," Internet draft, [online] Available from: http://ircache.nlanr.net/Cache/ICP/carp.txt.

[8] S. Gadde, M. Rabinovich, and J. Chase, "Reduce, reuse, recycle: an approach to building large Internet caches," in proc. of HotOS Workshop, pp. 93-98, May. 1997.

[9] Z. Wang, J. Crowcroft, "Cachemesh: A Distributed Cache System For World Wide Web," in proc. of 2nd International WWW Caching Workshop, 1997.

[10] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, Haobo Yu, The VINT Project, Advances in Network Simulation, IEEE, May 2000.

[11] Jia Wang, A Survey of Web Caching Schemes for the Internet, 1999.

[12] Digital Equipment Corporation. Proxy cache log traces, [online] Available from: ftp://ftp.digital.com/pub/DEC/traces/proxy/webtraces.html [1996,September].

[13] James Gwertzman and Margo Seltzer. World wide web cache consistency. In Proceedings of the USENIX Technical Conference, pages 141-152. USENIX Association, January 1996.

[14] J. Gettys, J. Mogul, R. Fielding, H. Frystyk, and T. Bernes-Lee. Hypertext transfer protocol http/1.1. RFC 2068, W3C, 1997.

[15] N. Niclausse, Z. Liu, and P. Nain. A new efficient caching policy for the world wide web. In Proceedings of Workshop on Internet Server Performance (WISP'98), Madison, WI, June 1998.

[16] P. Lorenzetti and L. Rizzo. Replacement policies for a proxy cache. Technical Report LR-960731, Univ. di Pisa, 1996.

[17] Squid Internet Object Cache, [online] Available from: http://www.nlnar.net/Squid/.

ภาคผนวก

ภาคผนวก ก

สรุปคำสั่ง

1. คำสั่งพื้นฐานทั่วไปที่ใช้ในสคริปต์

- ns <otclfile> <arg> <arg> ...
  คำสั่งพื้นฐานที่ใช้รันสคริปต์ที่ใช้จำลองการทำงาน
- set ns_ [new Simulator]
  คำสั่งสร้างอินสแตนท์ของอ็อบเจกต์ตัวจำลองการทำงาน
- set now [$ns_ now]
  คำสั่งนี้จะส่งค่าเวลาปัจจุบันของการจำลองการทำงานกลับมา
- $ns_ halt
  คำสั่งหยุดการทำงาน
- $ns_ run
  คำสั่งเริ่มการทำงาน
- $ns_ at <time> <event>
  คำสั่งกำหนดเวลาที่จะให้เกิดเหตุการณ์ต่างๆ ขึ้น
- $ns_ cancel <event>
  คำสั่งยกเลิกเหตุการณ์ได้กำหนดไว้
- $ns_ create-trace <type> <file> <src> <dst> <optional arg: op>
  คำสั่งสร้างการติดตามอ็อบเจกต์ในรูปแบบ <type> ระหว่าง <src> กับ <dst> และเขียนผลของการติดตามลงใน <file> โดยรูปแบบของผลการติดตามจะกำหนดโดย op ถ้ากำหนดเป็น nam รูปแบบของผลการติดตามจะอยู่ในรูปแบบของ nam แต่ถ้าไม่กำหนดรูปแบบของผลการติดตามจะอยู่ในรูปแบบของ ns
- $ns_ flush-trace
  คำสั่งเขียนข้อมูลการติดตามอ็อบเจกต์ทั้งหมดลงบัฟเฟอร์
- $ns_ after <delay> <event>
  คำสั่งกำหนดเหตุการณ์ที่จะเกิดขึ้นหลังจากช่วงระยะเวลาหนึ่ง
- $ns_ is-started
  คำสั่งนี้จะส่งค่า True กลับมาถ้าตัวจำลองการทำงานได้เริ่มทำงานแล้ว และจะส่งค่า False กลับมาถ้าตัวจำลองการทำงานยังไม่ได้เริ่มทำงาน

- set agent [new Agent/AgentType]
  คำสั่งสร้าง agent ต่างๆ ตามรูปแบบ AgentType
- $ns_ attach-agent <node> <agent>
  คำสั่งใช้ติด <agent> บน <node>
- $ns_ connect <src> <dst>
  คำสั่งการเชื่อมต่อระหว่าง <src> และ <dst> agent
- set node [$ns node]
  คำสั่งการสร้าง node
- $ns duplex-link $node0 $node1 <bandwidth> <delay> <queue_type>
  คำสั่งการสร้างลิงค์เพื่อเชื่อมต่อระหว่างโหนดโดยมีรูปแบบ <queue_type> ต่างๆ ดังนี้ DropTail, RED, CBQ, FQ, SFQ, DRR

2. คำสั่งที่ใช้ในการจำลองการทำงานของเว็บแคช

- set server [new Http/Server <sim> <s-node>]
  คำสั่งสร้างอินสแตนท์ของ Http Server บน <s-node> ที่กำหนด และใช้อินสแตนท์ของตัวจำลองการทำงาน <sim> เป็นตัวส่งผ่านอาร์กิวเมนต์
- set client [new Http/Client <sim> <c-node>]
  คำสั่งสร้างอินสแตนท์ของ Http Client บน <c-node>
- set cache [new Http/Cache <sim> <e-node>]
  คำสั่งสร้างอินสแตนท์ของ Http Cache บน <e-node>
- set pgp [new PagePool/Math]
  คำสั่งสร้าง Math PagePool ใช้สำหรับจัดการการร้องขอที่สร้างขึ้น
- $pgp ranvar-size <rv>
  คำสั่งกำหนดตัวแปรสุ่มสำหรับขนาดของเอกสาร
- $pgp ranvar-age <rv>
  คำสั่งกำหนดตัวแปรสุ่มสำหรับอายุของเอกสาร
- $pgp ranvar-pid <rv>
  คำสั่งกำหนดตัวแปรสุ่มสำหรับเลขประจำเอกสาร
- set pgp [new PagePool/ProxyTrace]
  คำสั่งสร้าง ProxyTrace PagePool ใช้สำหรับจัดการการร้องขอจากข้อมูลจริง

- $pgp set-reqfile "reqlog"
  คำสั่งกำหนดไฟล์ที่ใช้สำหรับการร้องขอ
- $pgp set-pagefile "pglog"
  คำสั่งกำหนดไฟล์ที่ใช้เป็นข้อมูลของเอกสาร
- $pgp set-client-num <num>
  คำสั่งกำหนดจำนวนผู้เรียกขอในการจำลองการทำงานของเว็บแคช
- $pgp bimodal-ratio <ratio>
  คำสั่งกำหนดค่าของเอกสารที่มีการเปลี่ยนแปลงบ่อย โดยมีค่าเป็น <ratio>*10 เปอร์เซ็นต์
- $pgp ranvar-dp <rv>
  คำสั่งกำหนดช่วงเวลาการเปลี่ยนแปลงของเอกสารสำหรับเอกสารที่มีการเปลี่ยนแปลงบ่อย
- $pgp ranvar-dp <rv>
  คำสั่งกำหนดช่วงเวลาการเปลี่ยนแปลงของเอกสารสำหรับเอกสารที่ไม่ค่อยมีการเปลี่ยนแปลง
- $server set-page-generator <pgp>
  คำสั่งกำหนด PagePool สำหรับ Server
- $server log <handle-to-log-file>
  คำสั่งกำหนด log file สำหรับ Server
- $client set-page-generator <pgp>
  คำสั่งกำหนด PagePool สำหรับ Client
- $client set-interval-generator <ranvar>
  คำสั่งกำหนดช่วงเวลาระหว่างการร้องขอแต่ละครั้ง
- $client log <handle-to-log-file>
  คำสั่งกำหนด log file สำหรับ Client
- $cache log <log-file>
  คำสั่งกำหนด log file สำหรับ Cache
- $client connect <cache>
  คำสั่งเชื่อมต่อ Client กับ Cache

- $cache connect <server>

  คำสั่งเชื่อมต่อ Cache กับ Server

- $client start-session <cache> <server>

  คำสั่งเริ่มต้นส่งการร้องขอจาก Client ไปยัง Server โดยผ่าน Cache

- set fifo [new Http/Cache/FIFO]

  $cache attach-repl $fifo

  คำสั่งกำหนดขั้นตอนวิธีการแทนที่แบบ FIFO

- set lru [new Http/Cache/LRU]

  $cache attach-repl $lru

  คำสั่งกำหนดขั้นตอนวิธีการแทนที่แบบ LRU

- set size [new Http/Cache/SIZE]

  $cache attach-repl $size

  คำสั่งกำหนดขั้นตอนวิธีการแทนที่แบบ SIZE

- set cache [new Http/Cache/TTL $ns $node]

  คำสั่งกำหนดความต้องกันของเว็บแคชแบบ Adaptive TTL

- set cache [new Http/Cache/TTL/Plain $ns $node]

  คำสั่งกำหนดความต้องกันของเว็บแคชแบบ Plain Old TTL

- $cache set-thresh <percent>

  คำสั่งกำหนดค่าขีดแบ่งสำหรับ Adaptive TTL

- $cache set-thresh <time>

  คำสั่งกำหนดค่าขีดแบ่งสำหรับ Plain Old TTL

- set arch [new Http/Cache/ICP]

  $cache attach-arch $arch

  คำสั่งกำหนดสถาปัตยกรรมของการแคชเป็นแบบพี่น้อง

- $cache set-cachesize <size>

  คำสั่งกำหนดขนาดของเว็บแคช

- $cache get-cachesize

  คำสั่งเรียกดูค่าขนาดของเว็บแคช

- $cache set-group $group

  คำสั่งกำหนดกลุ่มของแคชในระดับเดียวกัน เพื่อใช้ในการติดต่อสื่อสารภายในกลุ่ม

- $cache1 set-parent $cache2

  คำสั่งกำหนดว่าแคชตัวใดเป็นแคชในระดับที่สูงกว่าที่เชื่อมต่อกันอยู่ ซึ่งใช้ในการเชื่อมต่อแบบลำดับชั้น

- $cache puts-output

  คำสั่งแสดงข้อมูลของ Cache เมื่อสิ้นสุดการทำงาน ดังตัวอย่าง

  -------- Cache 1 --------

  Total Requests: 55988 ----- Total Bytes: 628449987.0

  Cache Hit: 19188 ----- Bytes Cache Hit: 137423125.0

  Hit Rate: 0.343 ----- Byte Hit Rate: 0.219

  IMS Request: 0 ----- Valid: 0 ----- Invalid: 0

- $client puts-output

  คำสั่งแสดงข้อมูลของ Client เมื่อสิ้นสุดการทำงาน ดังตัวอย่าง

  -------- Client 1 --------

  Client Requests: 55988 ---- Client Stales: 18180

- $server puts-output

  คำสั่งแสดงข้อมูลของ Server เมื่อสิ้นสุดการทำงาน ดังตัวอย่าง

  -------- Server 1 --------

  Server Got Requests: 96507 ---- Total Bytes : 1067209350.0 ----- Page Modified : 7992850

ภาคผนวก ข

ตัวอย่างสคริปต์

```
# Script: wc-fifo1.tcl
#
# 1 Client, 1 Cache, 1 Server
#
# Cache Size 100 MB
#
#   c1------e1------s1
# 100Mb 40ms  10Mb 2ms
#
set ns [new Simulator]

# Create topology/routing
set node(c) [$ns node]
set node(e) [$ns node]
set node(s) [$ns node]
$ns duplex-link $node(s) $node(e) 10Mb 2ms DropTail
$ns duplex-link $node(e) $node(c) 100Mb 40ms DropTail
$ns rtproto Session

# HTTP logs
set log [open "http-fifo1.log" w]

#Open the NAM trace file
#set nf [open out.nam w]
#$ns namtrace-all $nf

# Use PagePool/Proxy Trace
set pgp [new PagePool/ProxyTrace]

# Set trace files. There are two files; one for request stream, the other for
# page information, e.g., size and id
$pgp set-reqfile "reqlog"
$pgp set-pagefile "pglog"

# Set number of clients that will use this page pool. It's used to assign
# requests to clients
$pgp set-client-num 1

# Set the ratio of hot pages in all pages. Because no page modification
# data is available in most traces, we assume a bimodal page age distribution
$pgp bimodal-ratio 0.1

# Dynamic (hot) page age generator
set tmp [new RandomVariable/Constant] ;# Age generator
$tmp set val_ 100
$pgp ranvar-dp $tmp

# Static page age generator
set tmp [new RandomVariable/Constant]
$tmp set val_ 5000
$pgp ranvar-sp $tmp
```

```
set server [new Http/Server $ns $node(s)]
$server set-page-generator $pgp
#$server log $log
puts "Server ID $server"

set cache [new Http/Cache $ns $node(e)]
set fifo [new Http/Cache/FIFO]
$cache attach-repl $fifo
$cache log $log
$cache set-cachesize 104857600
set csize [$cache get-cachesize]
puts "Cache ID $cache Size: $csize"

set client [new Http/Client $ns $node(c)]
# XXX When trace-driven, don't assign a request interval generator
$client set-page-generator $pgp
#$client log $log
puts "Client ID $client"

set startTime 0 ;# simulation start time
set finishTime 86462 ;# 86461.979102 simulation end time
$ns at $startTime "start-connection"
$ns at $finishTime "finish"

proc start-connection {} {
        global ns server cache client
        $client connect $cache
        $cache connect $server
        $client start-session $cache $server
}

proc finish {} {
        global ns log client cache server ;#nf
        $ns flush-trace
        flush $log
        close $log
#       close $nf
        puts "Finish"
        puts "---------------------------"
        $cache puts-output
        puts "---------------------------"
        $client puts-output
        puts "---------------------------"
        $server puts-output
        puts "---------------------------"
        exit 0
}

$ns run
```

```
# Script: wc-lru1.tcl
#
# 1 Client, 1 Cache, 1 Server
#
# Cache Size 100 MB
#
#   c1------e1------s1
# 100Mb 40ms  10Mb 2ms
#
set ns [new Simulator]

# Create topology/routing
set node(c) [$ns node]
set node(e) [$ns node]
set node(s) [$ns node]
$ns duplex-link $node(s) $node(e) 10Mb 2ms DropTail
$ns duplex-link $node(e) $node(c) 100Mb 40ms DropTail
$ns rtproto Session

# HTTP logs
set log [open "http-lru1.log" w]

#Open the NAM trace file
#set nf [open out.nam w]
#$ns namtrace-all $nf

# Use PagePool/Proxy Trace
set pgp [new PagePool/ProxyTrace]

# Set trace files. There are two files; one for request stream, the other for
# page information, e.g., size and id
$pgp set-reqfile "reqlog"
$pgp set-pagefile "pglog"

# Set number of clients that will use this page pool. It's used to assign
# requests to clients
$pgp set-client-num 1

# Set the ratio of hot pages in all pages. Because no page modification
# data is available in most traces, we assume a bimodal page age distribution
$pgp bimodal-ratio 0.1

# Dynamic (hot) page age generator
set tmp [new RandomVariable/Constant] ;# Age generator
$tmp set val_ 100
$pgp ranvar-dp $tmp

# Static page age generator
set tmp [new RandomVariable/Constant]
$tmp set val_ 5000
$pgp ranvar-sp $tmp

set server [new Http/Server $ns $node(s)]
$server set-page-generator $pgp
#$server log $log
puts "Server ID $server"
```

```
set cache [new Http/Cache $ns $node(e)]
set lru [new Http/Cache/LRU]
$cache attach-repl $lru
$cache log $log
$cache set-cachesize 104857600
set csize [$cache get-cachesize]
puts "Cache ID $cache Size: $csize"

set client [new Http/Client $ns $node(c)]
# XXX When trace-driven, don't assign a request interval generator
$client set-page-generator $pgp
#$client log $log
puts "Client ID $client"

set startTime 0 ;# simulation start time
set finishTime 86462 ;# 86461.979102 simulation end time
$ns at $startTime "start-connection"
$ns at $finishTime "finish"

proc start-connection {} {
        global ns server cache client
        $client connect $cache
        $cache connect $server
        $client start-session $cache $server
}

proc finish {} {
        global ns log client cache server ;#nf
        $ns flush-trace
        flush $log
        close $log
#       close $nf
        puts "Finish"
        puts "---------------------------"
        $cache puts-output
        puts "---------------------------"
        $client puts-output
        puts "---------------------------"
        $server puts-output
        puts "---------------------------"
        exit 0
}

$ns run
```

```
# Script: wc-size1.tcl
#
# 1 Client, 1 Cache, 1 Server
#
# Cache Size 100 MB
#
#   c1-----e1------s1
# 100Mb 40ms  10Mb 2ms
#
set ns [new Simulator]

# Create topology/routing
set node(c) [$ns node]
set node(e) [$ns node]
set node(s) [$ns node]
$ns duplex-link $node(s) $node(e) 10Mb 2ms DropTail
$ns duplex-link $node(e) $node(c) 100Mb 40ms DropTail
$ns rtproto Session

# HTTP logs
set log [open "http-size1.log" w]

#Open the NAM trace file
#set nf [open out.nam w]
#$ns namtrace-all $nf

# Use PagePool/Proxy Trace
set pgp [new PagePool/ProxyTrace]

# Set trace files. There are two files; one for request stream, the other for
# page information, e.g., size and id
$pgp set-reqfile "reqlog"
$pgp set-pagefile "pglog"

# Set number of clients that will use this page pool. It's used to assign
# requests to clients
$pgp set-client-num 1

# Set the ratio of hot pages in all pages. Because no page modification
# data is available in most traces, we assume a bimodal page age distribution
$pgp bimodal-ratio 0.1

# Dynamic (hot) page age generator
set tmp [new RandomVariable/Constant] ;# Age generator
$tmp set val_ 100
$pgp ranvar-dp $tmp

# Static page age generator
set tmp [new RandomVariable/Constant]
$tmp set val_ 5000
$pgp ranvar-sp $tmp

set server [new Http/Server $ns $node(s)]
$server set-page-generator $pgp
#$server log $log
puts "Server ID $server"
```

```
set cache [new Http/Cache $ns $node(e)]
set size [new Http/Cache/SIZE]
$cache attach-repl $size
$cache log $log
$cache set-cachesize 104857600
set csize [$cache get-cachesize]
puts "Cache ID $cache Size: $csize"

set client [new Http/Client $ns $node(c)]
# XXX When trace-driven, don't assign a request interval generator
$client set-page-generator $pgp
#$client log $log
puts "Client ID $client"

set startTime 0 ;# simulation start time
set finishTime 86462 ;# 86461.979102 simulation end time
$ns at $startTime "start-connection"
$ns at $finishTime "finish"

proc start-connection {} {
        global ns server cache client
        $client connect $cache
        $cache connect $server
        $client start-session $cache $server
}

proc finish {} {
        global ns log client cache server ;#nf
        $ns flush-trace
        flush $log
        close $log
#       close $nf
        puts "Finish"
        puts "---------------------------"
        $cache puts-output
        puts "---------------------------"
        $client puts-output
        puts "---------------------------"
        $server puts-output
        puts "---------------------------"
        exit 0
}

$ns run
```

```
# Script: wc-attl0.tcl
#
# 1 Client, 1 Cache, 1 Server
#
# Threshold 0 ,Unlimited Cache Size
#
#   c1------e1------s1
# 10Mb 2ms  1.5Mb 50ms
#
set ns [new Simulator]

# Create topology/routing
set node(c) [$ns node]
set node(e) [$ns node]
set node(s) [$ns node]
$ns duplex-link $node(s) $node(e) 1.5Mb 50ms DropTail
$ns duplex-link $node(e) $node(c) 10Mb 2ms DropTail
$ns rtproto Session

# HTTP logs
set log [open "http-gen0.log" w]

# Use PagePool/Math
set pgp [new PagePool/Math]
set tmp [new RandomVariable/Exponential] ;# Size generator
$tmp set avg_ 4096  ;# average page size
$pgp ranvar-size $tmp
set tmp [new RandomVariable/Exponential] ;# Age generator
$tmp set avg_ 14400 ;# average page age
$pgp ranvar-age $tmp
set tmp [new RandomVariable/Exponential] ;# Page ID generator
$tmp set avg_ 300
$pgp ranvar-pid $tmp

set server [new Http/Server $ns $node(s)]
$server set-page-generator $pgp
#$server log $log

set cache [new Http/Cache/TTL $ns $node(e)]
$cache set-thresh 0
$cache log $log

set client [new Http/Client $ns $node(c)]
set tmp [new RandomVariable/Exponential] ;# Poisson process
$tmp set avg_ 0.1 ;# average request interval
$client set-interval-generator $tmp
$client set-page-generator $pgp
#$client log $log

set startTime 0 ;# simulation start time
set finishTime 14400 ;# simulation end time
$ns at $startTime "start-connection"
$ns at $finishTime "finish"

proc start-connection {} {
        global ns server cache client
        $client connect $cache
```

```
        $cache connect $server
        $client start-session $cache $server
}

proc finish {} {
        global ns client cache server log
        $ns flush-trace
        flush $log
        close $log
        puts "Finish"
        puts "------------------------------"
        $cache puts-output
        puts "------------------------------"
        $client puts-output
        puts "------------------------------"
        $server puts-output
        exit 0
}

$ns run
```

```
# Script: wc-pttl0.tcl
#
# 1 Client, 1 Cache, 1 Server
#
# Threshold 0 ,Unlimited Cache Size
#
#   c1------e1------s1
# 10Mb 2ms  1.5Mb 50ms
#
set ns [new Simulator]

# Create topology/routing
set node(c) [$ns node]
set node(e) [$ns node]
set node(s) [$ns node]
$ns duplex-link $node(s) $node(e) 1.5Mb 50ms DropTail
$ns duplex-link $node(e) $node(c) 10Mb 2ms DropTail
$ns rtproto Session

# HTTP logs
set log [open "http-gen0.log" w]

# Use PagePool/Math
set pgp [new PagePool/Math]
set tmp [new RandomVariable/Exponential] ;# Size generator
$tmp set avg_ 4096  ;# average page size
$pgp ranvar-size $tmp
set tmp [new RandomVariable/Exponential] ;# Age generator
$tmp set avg_ 14400 ;# average page age
$pgp ranvar-age $tmp
set tmp [new RandomVariable/Exponential] ;# Page ID generator
$tmp set avg_ 300
$pgp ranvar-pid $tmp

set server [new Http/Server $ns $node(s)]
$server set-page-generator $pgp
#$server log $log

set cache [new Http/Cache/TTL/Plain $ns $node(e)]
$cache set-thresh 0
$cache log $log

set client [new Http/Client $ns $node(c)]
set tmp [new RandomVariable/Exponential] ;# Poisson process
$tmp set avg_ 0.1 ;# average request interval
$client set-interval-generator $tmp
$client set-page-generator $pgp
#$client log $log

set startTime 0 ;# simulation start time
set finishTime 14400 ;# simulation end time
$ns at $startTime "start-connection"
$ns at $finishTime "finish"

proc start-connection {} {
        global ns server cache client
        $client connect $cache
```

```
        $cache connect $server
        $client start-session $cache $server
}

proc finish {} {
        global ns client cache server log
        $ns flush-trace
        flush $log
        close $log
        puts "Finish"
        puts "------------------------------"
        $cache puts-output
        puts "------------------------------"
        $client puts-output
        puts "------------------------------"
        $server puts-output
        exit 0
}

$ns run
```

```
# Script: wc-h1.tcl
#
# 3 Clients, 3 Caches, 1 Server
#
# Cache1 -> FIFO, Cache Size -> 200 MB
# Cache2 -> LRU + Plain Old TTL, Threshold 100, Cache Size -> 200 MB
# Cache3 -> SIZE + Adaptive TTL, Threshold 0.1, Cache Size -> 200 MB
#
#                          s1
#                        / 10Mb 2ms
#                      e3
#       100Mb 50ms /   | 100Mb 50ms
#                  e2   c3
#  100Mb 50ms /   | 100Mb 50ms
#             e1  c2
#100Mb 50ms |
#             c1

set ns [new Simulator]

# Create topology/routing
for {set i 1} {$i <= 3} {incr i} {
        set node(c$i) [$ns node]
}
for {set i 1} {$i <= 3} {incr i} {
        set node(e$i) [$ns node]
}
set node(s1) [$ns node]
$ns duplex-link $node(c1) $node(e1) 100Mb 50ms DropTail
$ns duplex-link $node(c2) $node(e2) 100Mb 50ms DropTail
$ns duplex-link $node(c3) $node(e3) 100Mb 50ms DropTail
$ns duplex-link $node(e1) $node(e2) 100Mb 50ms DropTail
$ns duplex-link $node(e2) $node(e3) 100Mb 50ms DropTail
$ns duplex-link $node(e3) $node(s1) 10Mb 2ms DropTail
$ns duplex-link-op $node(c1) $node(e1) orient down
$ns duplex-link-op $node(c2) $node(e2) orient down
$ns duplex-link-op $node(c3) $node(e3) orient down
$ns duplex-link-op $node(e1) $node(e2) orient left-down
$ns duplex-link-op $node(e2) $node(e3) orient left-down
$ns duplex-link-op $node(e3) $node(s1) orient left-down
$ns rtproto Session

# HTTP logs
set log1 [open "http-h1-log1.log" w]
set log2 [open "http-h1-log2.log" w]
set log3 [open "http-h1-log3.log" w]

# Use PagePool/ProxyTrace
set pgp1 [new PagePool/ProxyTrace]

# Set trace files. There are two files; one for request stream, the other for
# page information, e.g., size and id
$pgp1 set-reqfile "reqlog"
$pgp1 set-pagefile "pglog"

# Set number of clients that will use this page pool. It's used to assign
# requests to clients
```

```
$pgp1 set-client-num 3

# Set the ratio of hot pages in all pages. Because no page modification
# data is available in most traces, we assume a bimodal page age distribution
$pgp1 bimodal-ratio 0.1

# Dynamic (hot) page age generator
set tmp [new RandomVariable/Exponential] ;# Age generator
$tmp set avg_ 5 ;# average page age
$pgp1 ranvar-dp $tmp

# Static page age generator
set tmp [new RandomVariable/Exponential] ;# Age generator
$tmp set avg_ 1000 ;# average page age
$pgp1 ranvar-sp $tmp

# --- Set Client(s) Cache(s) Server(s) ----
# ------ Server(s) ------
set server1 [new Http/Server $ns $node(s1)]
$server1 set-page-generator $pgp1

# ------ Cache(s) ------
set cache1 [new Http/Cache $ns $node(e1)]
set fifo [new Http/Cache/FIFO]
$cache1 attach-repl $fifo
$cache1 set-cachesize 209715200 ;# 200 MB
set c1size [$cache1 get-cachesize]
puts "Cache No.1 $cache1 Size: $c1size"
$cache1 log $log1

set cache2 [new Http/Cache/TTL/Plain $ns $node(e2)]
set lru [new Http/Cache/LRU]
$cache2 attach-repl $lru
$cache2 set-thresh 100
$cache2 set-cachesize 209715200 ;# 200 MB
set c2size [$cache2 get-cachesize]
puts "Cache No.2 $cache2 Size: $c2size"
$cache2 log $log2

set cache3 [new Http/Cache/TTL $ns $node(e3)]
set size [new Http/Cache/SIZE]
$cache3 attach-repl $size
$cache3 set-thresh 0.1
$cache3 set-cachesize 209715200 ;# 200 MB
set c3size [$cache3 get-cachesize]
puts "Cache No.3 $cache3 Size: $c3size"
$cache3 log $log3

# ------ Client(s) ------
set client1 [new Http/Client $ns $node(c1)]
$client1 set-page-generator $pgp1

set client2 [new Http/Client $ns $node(c2)]
$client2 set-page-generator $pgp1

set client3 [new Http/Client $ns $node(c3)]
$client3 set-page-generator $pgp1
```

```
set startTime 0 ;# simulation start time
set finishTime 10000 ;#86379 simulation end time
$ns at $startTime "start-connection"
$ns at $finishTime "finish"

proc start-connection {} {
        global ns server1 cache1 cache2 cache3 client1 client2 client3

        $client1 connect $cache1
        $client2 connect $cache2
        $client3 connect $cache3
        $cache1 connect $cache2
        $cache2 connect $cache3
        $cache3 connect $server1

        # Set Parent Cache
        $cache1 set-parent $cache2
        $cache2 set-parent $cache3

        # Start Sending Request
        $client1 start-session $cache1 $server1
        $client2 start-session $cache2 $server1
        $client3 start-session $cache3 $server1
}

proc finish {} {
        global ns client1 client2 client3 cache1 cache2 cache3 server1 log1 log2 log3
        $ns flush-trace
        flush $log1
        flush $log2
        flush $log3
        close $log1
        close $log2
        close $log3
        puts "Finish"
        puts "-------- Cache 1 --------"
        $cache1 puts-output
        puts "-------- Cache 2 --------"
        $cache2 puts-output
        puts "-------- Cache 3 --------"
        $cache3 puts-output
        puts "-------- Client 1 --------"
        $client1 puts-output
        puts "-------- Client 2 --------"
        $client2 puts-output
        puts "-------- Client 3 --------"
        $client3 puts-output
        puts "-------- Server 1 --------"
        $server1 puts-output
        exit 0
}

$ns run
```

```
# Script: wc-icp1.tcl
#
# 3 Clients, 3 Caches, 1 Server
#
# Cache1 -> FIFO, Cache Size -> 200 MB
# Cache2 -> LRU + Plain Old TTL, Threshold 100, Cache Size -> 200 MB
# Cache3 -> SIZE + Adaptive TTL, Threshold 0.1, Cache Size -> 200 MB
#
#                  s1
#   10Mb 2ms  /   |   \ 10Mb 2ms
#            e1 -- e2 -- e3
#100Mb 50ms |     |      | 100Mb 50ms
#            c1    c2     c3

set ns [new Simulator]

# Create topology/routing
for {set i 1} {$i <= 3} {incr i} {
        set node(c$i) [$ns node]
}
for {set i 1} {$i <= 3} {incr i} {
        set node(e$i) [$ns node]
}
set node(s1) [$ns node]
$ns duplex-link $node(c1) $node(e1) 100Mb 50ms DropTail
$ns duplex-link $node(c2) $node(e2) 100Mb 50ms DropTail
$ns duplex-link $node(c3) $node(e3) 100Mb 50ms DropTail
$ns duplex-link $node(e1) $node(e2) 100Mb 50ms DropTail
$ns duplex-link $node(e2) $node(e3) 100Mb 50ms DropTail
$ns duplex-link $node(e1) $node(e3) 100Mb 50ms DropTail
$ns duplex-link $node(e1) $node(s1) 10Mb 2ms DropTail
$ns duplex-link $node(e2) $node(s1) 10Mb 2ms DropTail
$ns duplex-link $node(e3) $node(s1) 10Mb 2ms DropTail
$ns duplex-link-op $node(c1) $node(e1) orient down
$ns duplex-link-op $node(c2) $node(e2) orient down
$ns duplex-link-op $node(c3) $node(e3) orient down
$ns duplex-link-op $node(e1) $node(e2) orient left
$ns duplex-link-op $node(e2) $node(e3) orient left
$ns duplex-link-op $node(e1) $node(e3) orient left
$ns duplex-link-op $node(e1) $node(s1) orient left-down
$ns duplex-link-op $node(e2) $node(s1) orient down
$ns duplex-link-op $node(e3) $node(s1) orient right-down
$ns rtproto Session

# HTTP logs
set log1 [open "http-icp1-log1.log" w]
set log2 [open "http-icp1-log2.log" w]
set log3 [open "http-icp1-log3.log" w]

# Use PagePool/ProxyTrace
set pgp1 [new PagePool/ProxyTrace]

# Set trace files. There are two files; one for request stream, the other for
# page information, e.g., size and id
$pgp1 set-reqfile "reqlog"
$pgp1 set-pagefile "pglog"
```

```
# Set number of clients that will use this page pool. It's used to assign
# requests to clients
$pgp1 set-client-num 3

# Set the ratio of hot pages in all pages. Because no page modification
# data is available in most traces, we assume a bimodal page age distribution
$pgp1 bimodal-ratio 0.1

# Dynamic (hot) page age generator
set tmp [new RandomVariable/Exponential] ;# Age generator
$tmp set avg_ 5 ;# average page age
$pgp1 ranvar-dp $tmp

# Static page age generator
set tmp [new RandomVariable/Exponential] ;# Age generator
$tmp set avg_ 1000 ;# average page age
$pgp1 ranvar-sp $tmp

# --- Set Client(s) Cache(s) Server(s) ----
# ------ Server(s) ------
set server1 [new Http/Server $ns $node(s1)]
$server1 set-page-generator $pgp1

# ------ Cache(s) ------
set cache1 [new Http/Cache $ns $node(e1)]
set fifo [new Http/Cache/FIFO]
$cache1 attach-repl $fifo
set arch1 [new Http/Cache/ICP]
$cache1 attach-arch $arch1
$cache1 set-cachesize 209715200 ;# 200 MB
set c1size [$cache1 get-cachesize]
puts "Cache No.1 $cache1 Size: $c1size"
$cache1 log $log1

set cache2 [new Http/Cache/TTL/Plain $ns $node(e2)]
set lru [new Http/Cache/LRU]
$cache2 attach-repl $lru
set arch2 [new Http/Cache/ICP]
$cache2 attach-arch $arch2
$cache2 set-thresh 100
$cache2 set-cachesize 209715200 ;# 200 MB
set c2size [$cache2 get-cachesize]
puts "Cache No.2 $cache2 Size: $c2size"
$cache2 log $log2

set cache3 [new Http/Cache/TTL $ns $node(e3)]
set size [new Http/Cache/SIZE]
$cache3 attach-repl $size
set arch3 [new Http/Cache/ICP]
$cache3 attach-arch $arch3
$cache3 set-thresh 0.1
$cache3 set-cachesize 209715200 ;# 200 MB
set c3size [$cache3 get-cachesize]
puts "Cache No.3 $cache3 Size: $c3size"
$cache3 log $log3

# ------ Client(s) ------
```

```
set client1 [new Http/Client $ns $node(c1)]
$client1 set-page-generator $pgp1

set client2 [new Http/Client $ns $node(c2)]
$client2 set-page-generator $pgp1

set client3 [new Http/Client $ns $node(c3)]
$client3 set-page-generator $pgp1

set startTime 0 ;# simulation start time
set finishTime 10000 ;# 86379 simulation end time
$ns at $startTime "start-connection"
$ns at $finishTime "finish"

proc start-connection {} {
        global ns server1 cache1 cache2 cache3 client1 client2 client3

        $client1 connect $cache1
        $client2 connect $cache2
        $client3 connect $cache3
        $cache1 connect $cache2
        $cache2 connect $cache3
        $cache1 connect $cache3
        $cache1 connect $server1
        $cache2 connect $server1
        $cache3 connect $server1

        # Set Group of Neighbour
        lappend group1 $cache1
        lappend group1 $cache2
        lappend group1 $cache3
        $cache1 set-group $group1
        $cache2 set-group $group1
        $cache3 set-group $group1

        # Start Sending Request
        $client1 start-session $cache1 $server1
        $client2 start-session $cache2 $server1
        $client3 start-session $cache3 $server1
}

proc finish {} {
        global ns client1 client2 client3 cache1 cache2 cache3 server1 log1 log2 log3
        $ns flush-trace
        flush $log1
        flush $log2
        flush $log3
        close $log1
        close $log2
        close $log3
        puts "Finish"
        puts "-------- Cache 1 --------"
        $cache1 puts-output
        puts "-------- Cache 2 --------"
        $cache2 puts-output
        puts "-------- Cache 3 --------"
        $cache3 puts-output
```

```
        puts "-------- Client 1 --------"
        $client1 puts-output
        puts "-------- Client 2 --------"
        $client2 puts-output
        puts "-------- Client 3 --------"
        $client3 puts-output
        puts "-------- Server 1 --------"
        $server1 puts-output
        exit 0
}

$ns run
```

ประวัติผู้เขียนวิทยานิพนธ์

นายนพดล จ.จิตต์เจริญชัย เกิดเมื่อวันที่ 27 กันยายน พ.ศ. 2520 สำเร็จการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2542 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2542