Network Simulator (NS-2)

p7893113@mail.ncku.edu.tw

丁義偉 (Yi-Wei Ting)

Computer Science and Information Engineering.
Computer and Internet Architecture Lab.

Why use Network Simulator?

- 1. New protocol is not easily implemented.
  (development, money, time, device, people)
- 2. New protocol is not easily verified in the world.
  (mobile IP, ad hoc…etc, more nodes or network topology)
- 3. New protocol is not easily compared others
  (How to convince other people?)
- 4. Network Simulator simulates protocol layers
  (physical, datalink , network, transport, applicatoin)

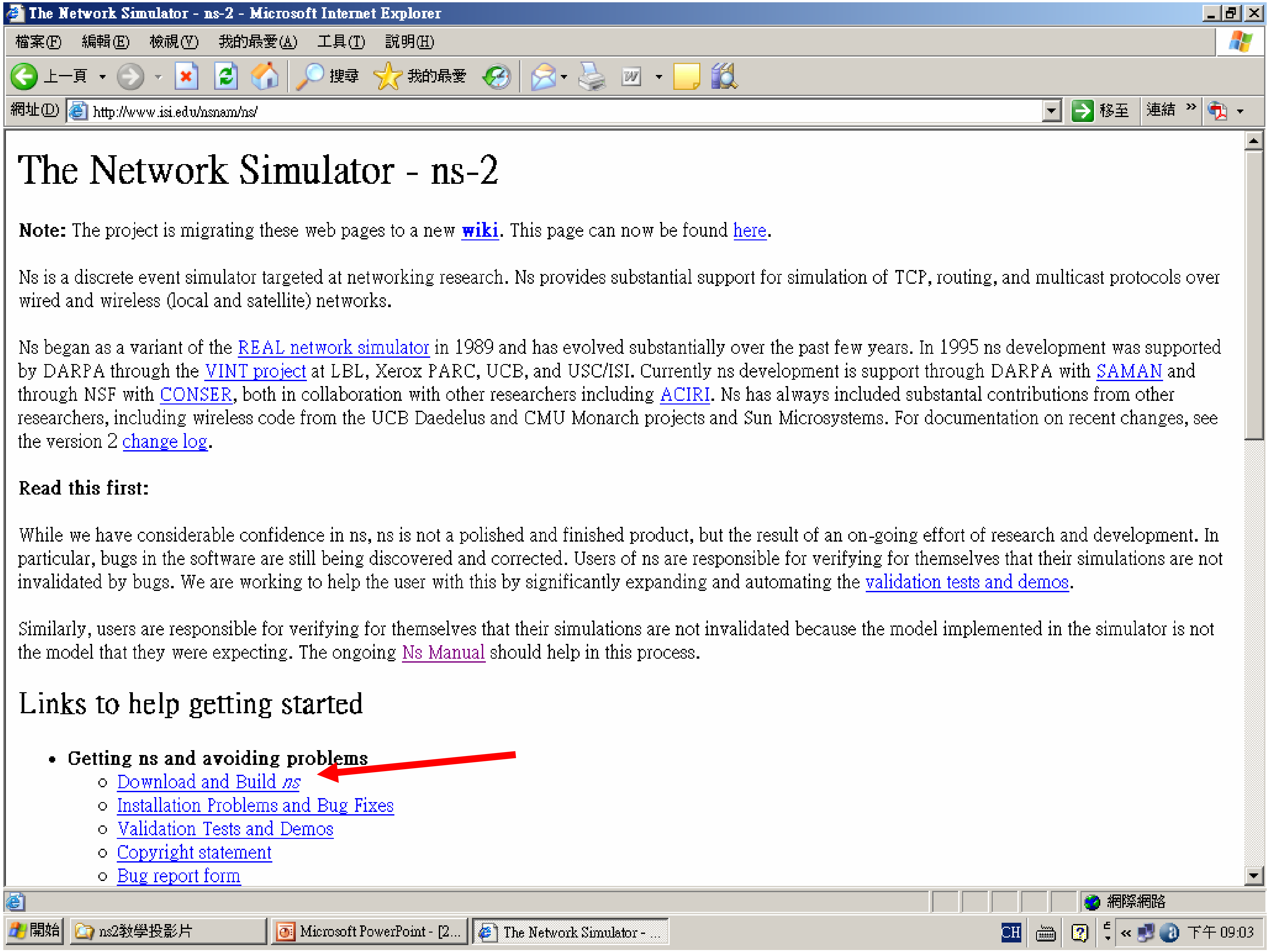
Development

- Ns began as the REAL network simulator in 1989 and has evolved over the past few years.
- In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI.
- In 1996, the first version of ns2 was release.
  - **27K lines of C++ code**
  - **12K lines of OTcl support code**
  - **18K lines of test suites examples**
  - **5K lines of documentation**

Setup

- http://www.isi.edu/nsnam/ns <-----首頁
- http://nile.wpi.edu/NS/   (tutorial site)

- download
  - Operation system
    - (FreeBSD, Linux, SunOS, Solaris, Windows)



The Network Simulator - ns-2 - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
網址(D) http://www.isi.edu/nsnam/ns/
移至
連結
網際網路
開始
ns2教學投影片
Microsoft PowerPoint - [2...
The Network Simulator - ...
CH
下午 09:03

The Network Simulator - ns-2

Note: The project is migrating these web pages to a new **wiki**. This page can now be found here.

Ns is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.

Ns began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. Currently ns development is support through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. Ns has always included substantal contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems. For documentation on recent changes, see the version 2 change log.

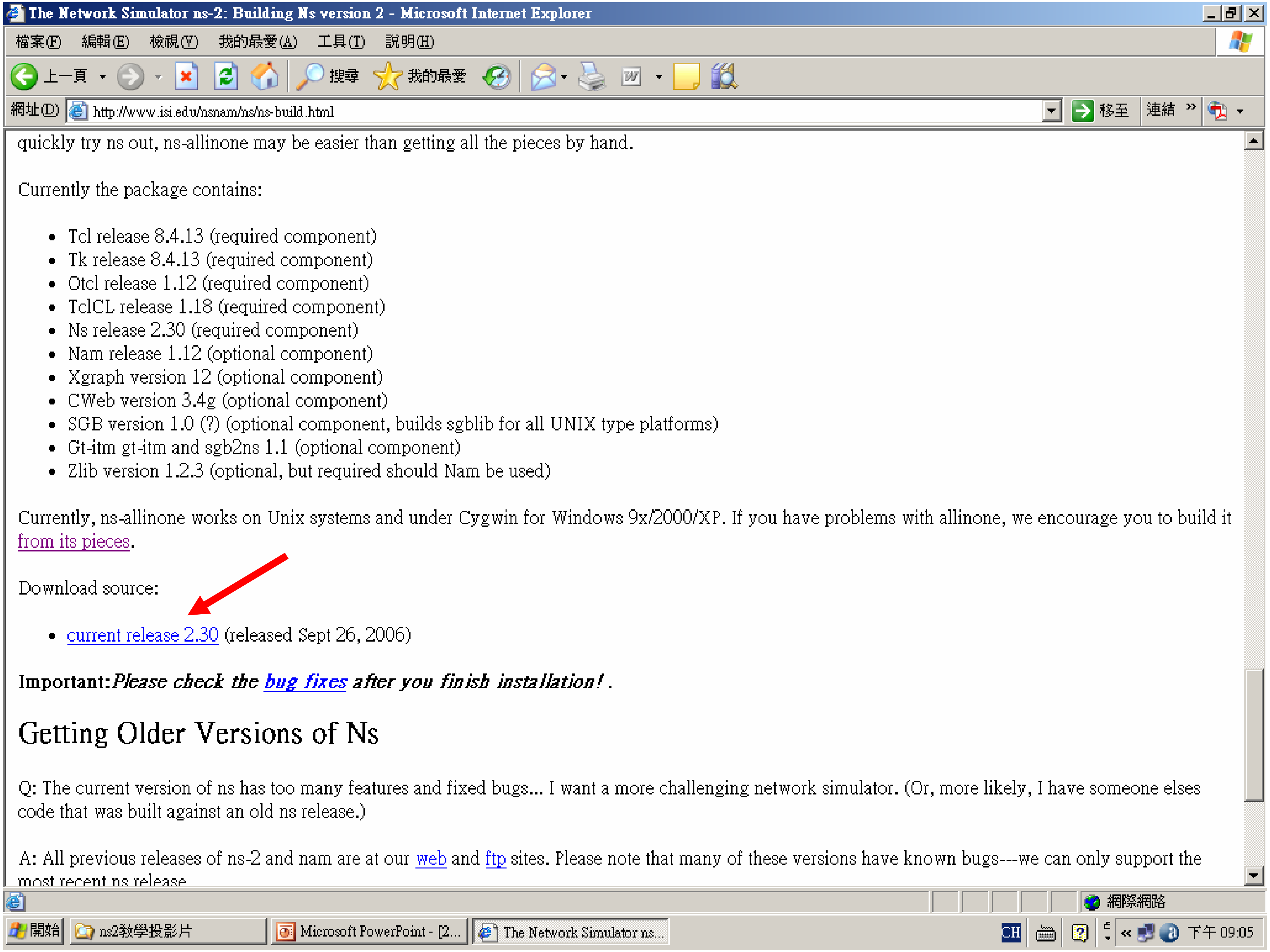
Read this first:

While we have considerable confidence in ns, ns is not a polished and finished product, but the result of an on-going effort of research and development. In particular, bugs in the software are still being discovered and corrected. Users of ns are responsible for verifying for themselves that their simulations are not invalidated by bugs. We are working to help the user with this by significantly expanding and automating the validation tests and demos.

Similarly, users are responsible for verifying for themselves that their simulations are not invalidated because the model implemented in the simulator is not the model that they were expecting. The ongoing Ns Manual should help in this process.

Links to help getting started

- **Getting ns and avoiding problems**
  - Download and Build *ns*
  - Installation Problems and Bug Fixes
  - Validation Tests and Demos
  - Copyright statement
  - Bug report form


The Network Simulator ns-2: Building Ns version 2 - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
網址(D) http://www.isi.edu/nsnam/ns/ns-build.html
移至
連結
網際網路
開始
ns2教學投影片
Microsoft PowerPoint - [2...
The Network Simulator ns...
CH
下午 09:05

quickly try ns out, ns-allinone may be easier than getting all the pieces by hand.

Currently the package contains:

- Tcl release 8.4.13 (required component)
- Tk release 8.4.13 (required component)
- Otcl release 1.12 (required component)
- TclCL release 1.18 (required component)
- Ns release 2.30 (required component)
- Nam release 1.12 (optional component)
- Xgraph version 12 (optional component)
- CWeb version 3.4g (optional component)
- SGB version 1.0 (?) (optional component, builds sgblib for all UNIX type platforms)
- Gt-itm gt-itm and sgb2ns 1.1 (optional component)
- Zlib version 1.2.3 (optional, but required should Nam be used)

Currently, ns-allinone works on Unix systems and under Cygwin for Windows 9x/2000/XP. If you have problems with allinone, we encourage you to build it from its pieces.

Download source:

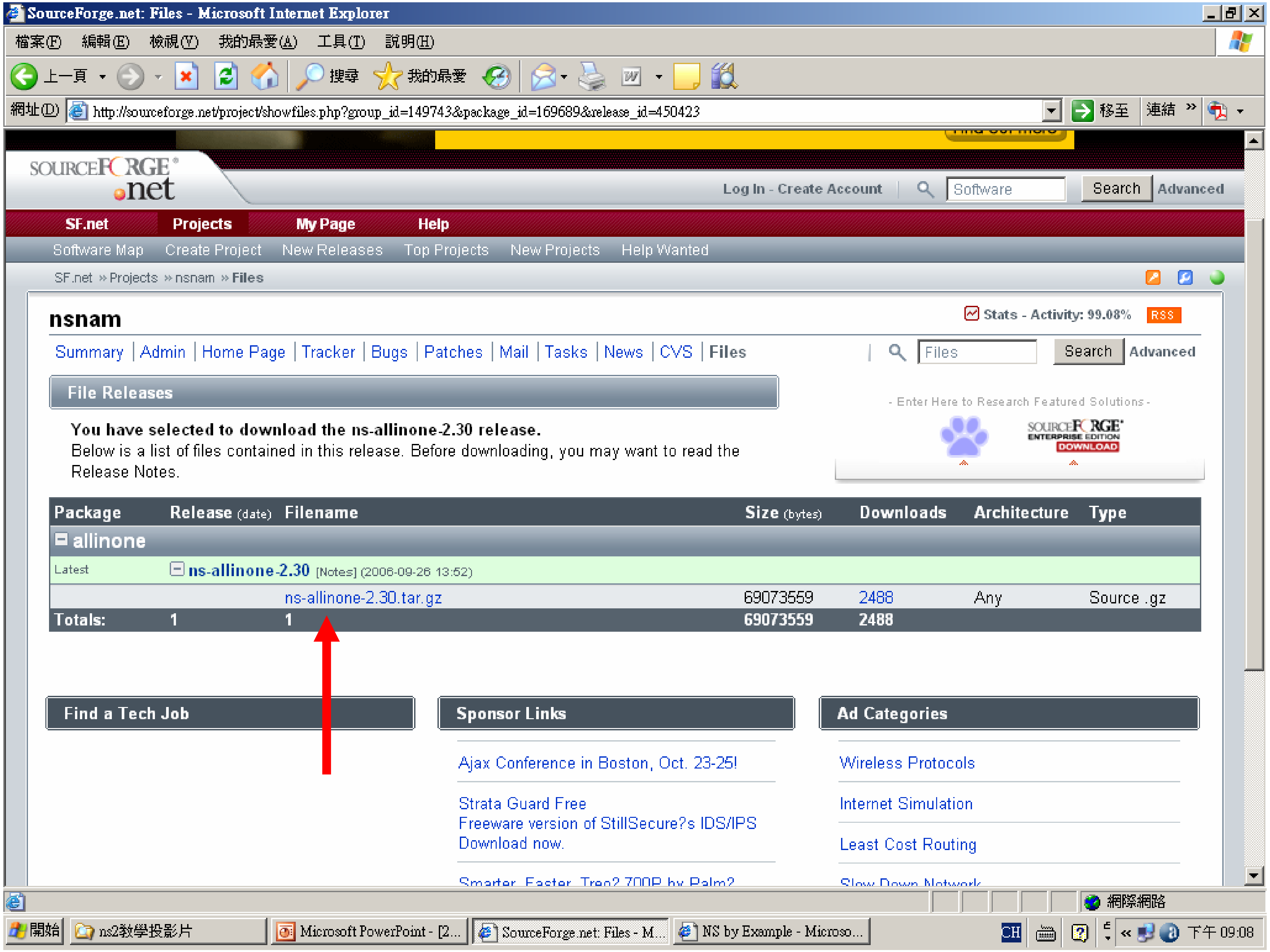
- current release 2.30 (released Sept 26, 2006)

Important: ***Please check the bug fixes after you finish installation!*** .

Getting Older Versions of Ns

Q: The current version of ns has too many features and fixed bugs... I want a more challenging network simulator. (Or, more likely, I have someone elses code that was built against an old ns release.)

A: All previous releases of ns-2 and nam are at our web and ftp sites. Please note that many of these versions have known bugs---we can only support the most recent ns release


SourceForge.net: Files - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛
網址(D) http://sourceforge.net/project/showfiles.php?group_id=149743&package_id=169689&release_id=450423
移至 連結
Log In - Create Account
Software
Search
Advanced
SF.net
Projects
My Page
Help
Software Map
Create Project
New Releases
Top Projects
New Projects
Help Wanted
SF.net » Projects » nsnam » Files
Stats - Activity: 99.08%
RSS
Summary | Admin | Home Page | Tracker | Bugs | Patches | Mail | Tasks | News | CVS | Files
Files
Search
Advanced
- Enter Here to Research Featured Solutions -
DOWNLOAD
Find a Tech Job
Sponsor Links
Ad Categories
網際網路
開始
ns2教學投影片
Microsoft PowerPoint - [2...
SourceForge.net: Files - M...
NS by Example - Microso...
下午 09:08

nsnam

File Releases

You have selected to download the ns-allinone-2.30 release.
Below is a list of files contained in this release. Before downloading, you may want to read the Release Notes.

Package	Release (date)	Filename	Size (bytes)	Downloads	Architecture	Type
allinone						
Latest	ns-allinone-2.30 [Notes] (2006-09-26 13:52)					
		ns-allinone-2.30.tar.gz	69073559	2488	Any	Source .gz
Totals:	1	1	69073559	2488		

Ajax Conference in Boston, Oct. 23-25!

Strata Guard Free
Freeware version of StillSecure?s IDS/IPS
Download now.

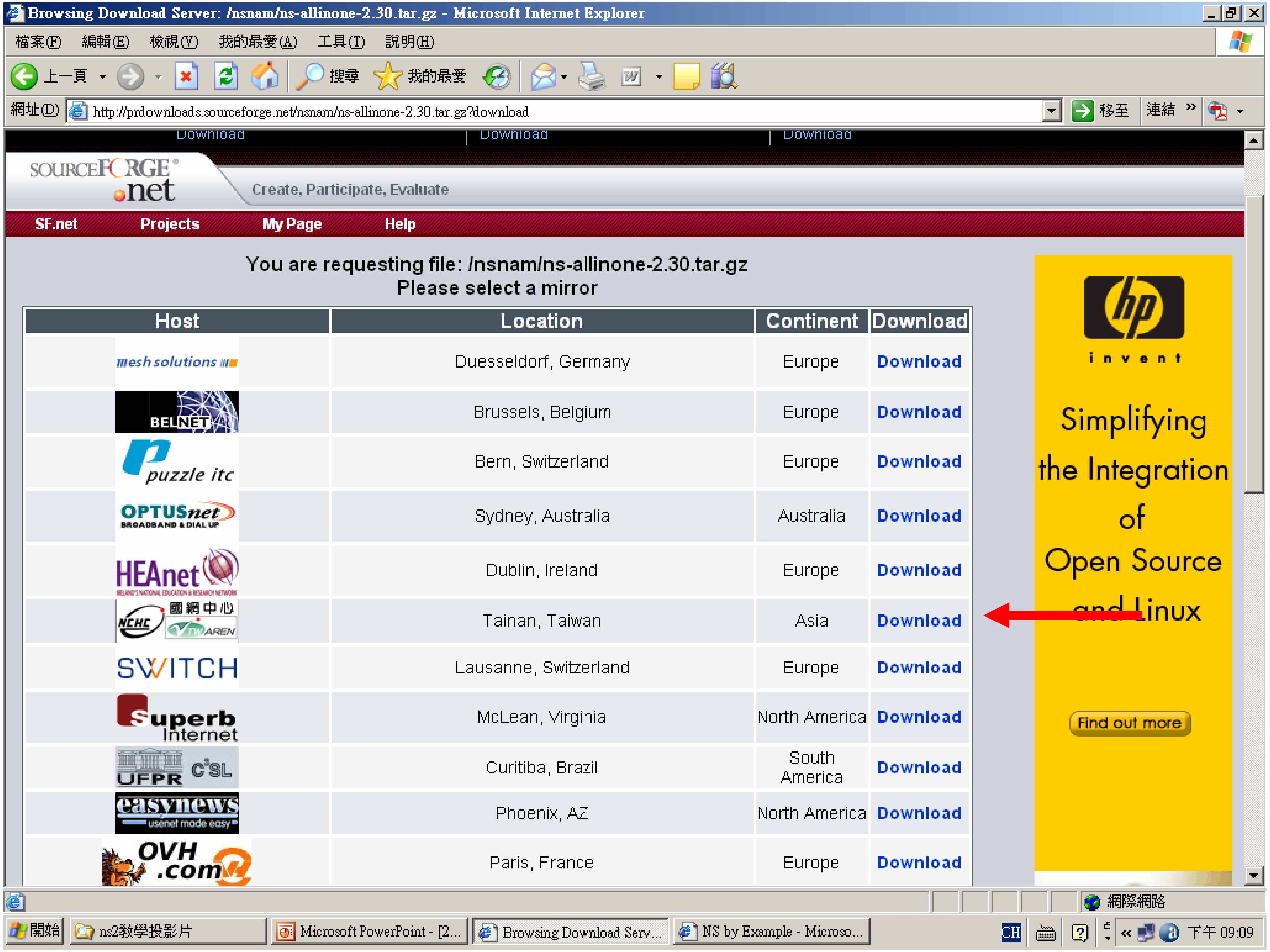
Smarter. Faster. Treo? 700P by Palm?

Wireless Protocols

Internet Simulation

Least Cost Routing

Slow Down Network



Browsing Download Server: /nsnam/ns-allinone-2.30.tar.gz - Microsoft Internet Explorer

http://prdownloads.sourceforge.net/nsnam/ns-allinone-2.30.tar.gz?download

You are requesting file: /nsnam/ns-allinone-2.30.tar.gz
Please select a mirror

Host	Location	Continent	Download
mesh solutions	Duesseldorf, Germany	Europe	Download
BELNET	Brussels, Belgium	Europe	Download
puzzle itc	Bern, Switzerland	Europe	Download
OPTUSnet	Sydney, Australia	Australia	Download
HEAnet	Dublin, Ireland	Europe	Download
NCHC 國網中心 TWAREN	Tainan, Taiwan	Asia	Download
SWITCH	Lausanne, Switzerland	Europe	Download
Superb Internet	McLean, Virginia	North America	Download
UFPR C3SL	Curitiba, Brazil	South America	Download
easynews	Phoenix, AZ	North America	Download
OVH.com	Paris, France	Europe	Download

SF.net
Projects
My Page
Help
Create, Participate, Evaluate
invent
Simplifying the Integration of Open Source and Linux
Find out more

Setup for Linux-based system

- ns-allinone-2.29.tar ←約65mb
- 假設檔案放在/root
  - Step 1: cd /root
  - Step 2: tar zxvf ns-allinone-2.29.tar.gz
  - Step 3: cd /root/ns-allinone-2.29
  - Step 4: ./install

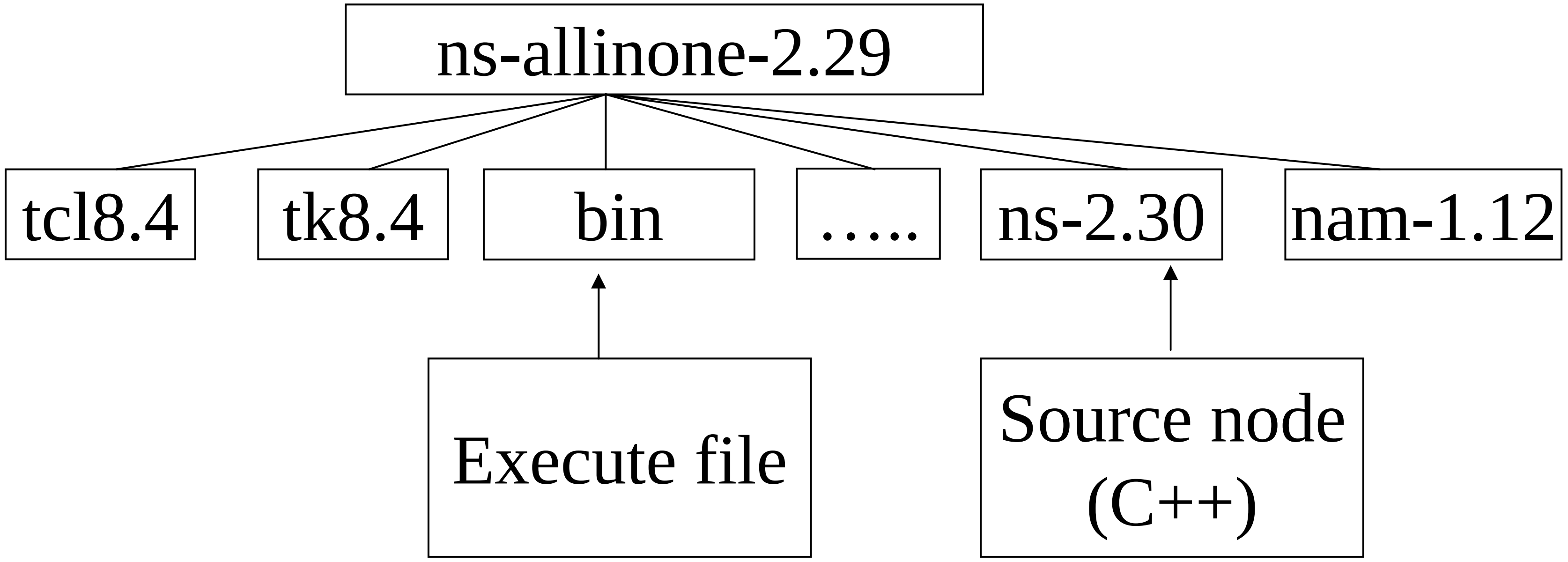
How to run a scenario in NS2

- Step 1: cd /root/ns-allinone-2.29
- Step 2: cd bin
- Step 3: ./ns my_scenario.tcl

在win9x / 2000/ xp 環境下安裝ns2模擬軟體

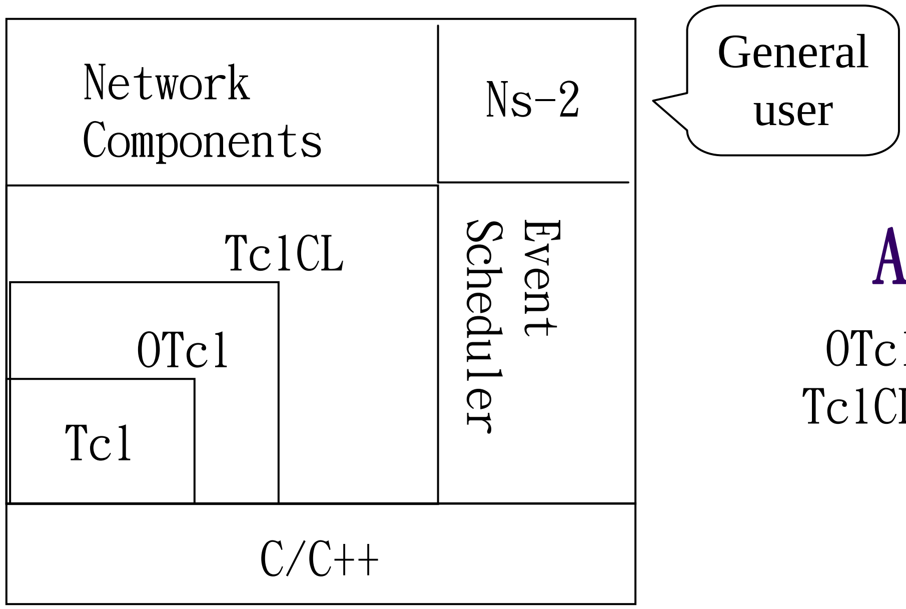
- 中文教學網站-柯志亨(Chih-Heng, Ke)
  - http://140.116.72.80/~smallko/ns2/ns2.htm
- 安裝流程
  - http://140.116.72.80/~smallko/ns2/setup.htm
- **Cygwin** is a collection of free software tools originally developed by Cygnus Solutions to allow various versions of Microsoft Windows to act somewhat like a Unix system.

NS-2 Directory Structure



NS view

- NS2 is an *object-oriented, discrete event driven* network simulator developed at UC Berkely written in C++ and OTcl.
  - C++ for "data"
    - Per packet action
  - OTcl for "control"
    - Periodic or triggered action



Architecture

OTcl: object-oriented Tcl
TclCL: C++ and OTcl linkage

Why using two programming language to implement?

- 模擬器有兩方面的事情需要做

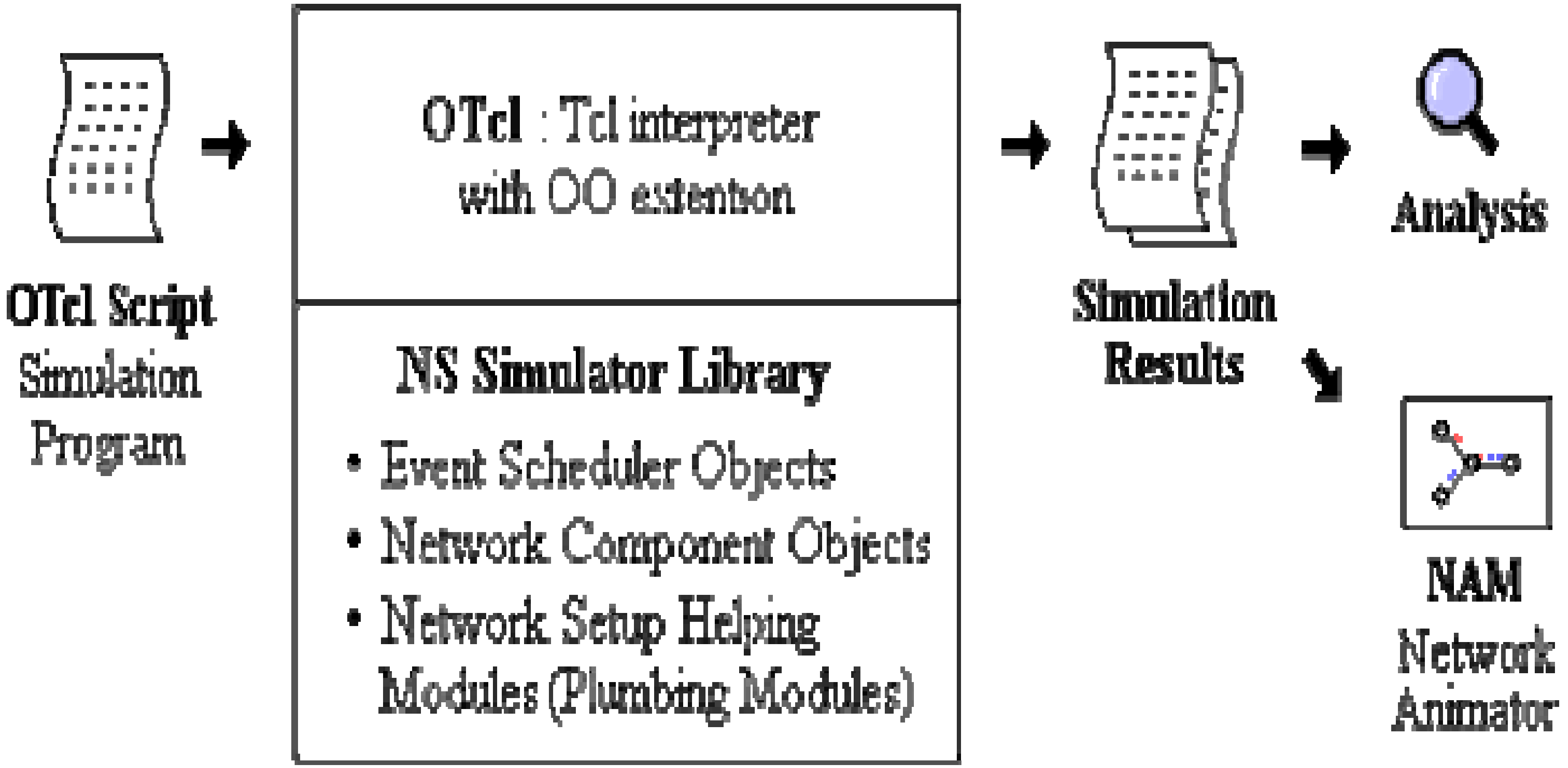
●第一:需要一種程式設計語言，能夠有效率的處理位元組（Byte），封包標頭（Packet Header）等資訊，需要應用合適的演算法在大量的資料上。因此，程式內部的運行速度（run-time speed）就非常重要。(c++)

●第二: 許多網路中的研究工作都圍繞著網路元件和環境參數的設置和改變而進行的，需要在短時間內快速的開發和模擬出所需要的網路環境（scenarios），並且方便修改和發現、修復。在這種情形下，(run-around time)就顯得很重要了，因為模擬環境的建立和參數資訊的配置只需要運行一次。(Otcl)

Protocols

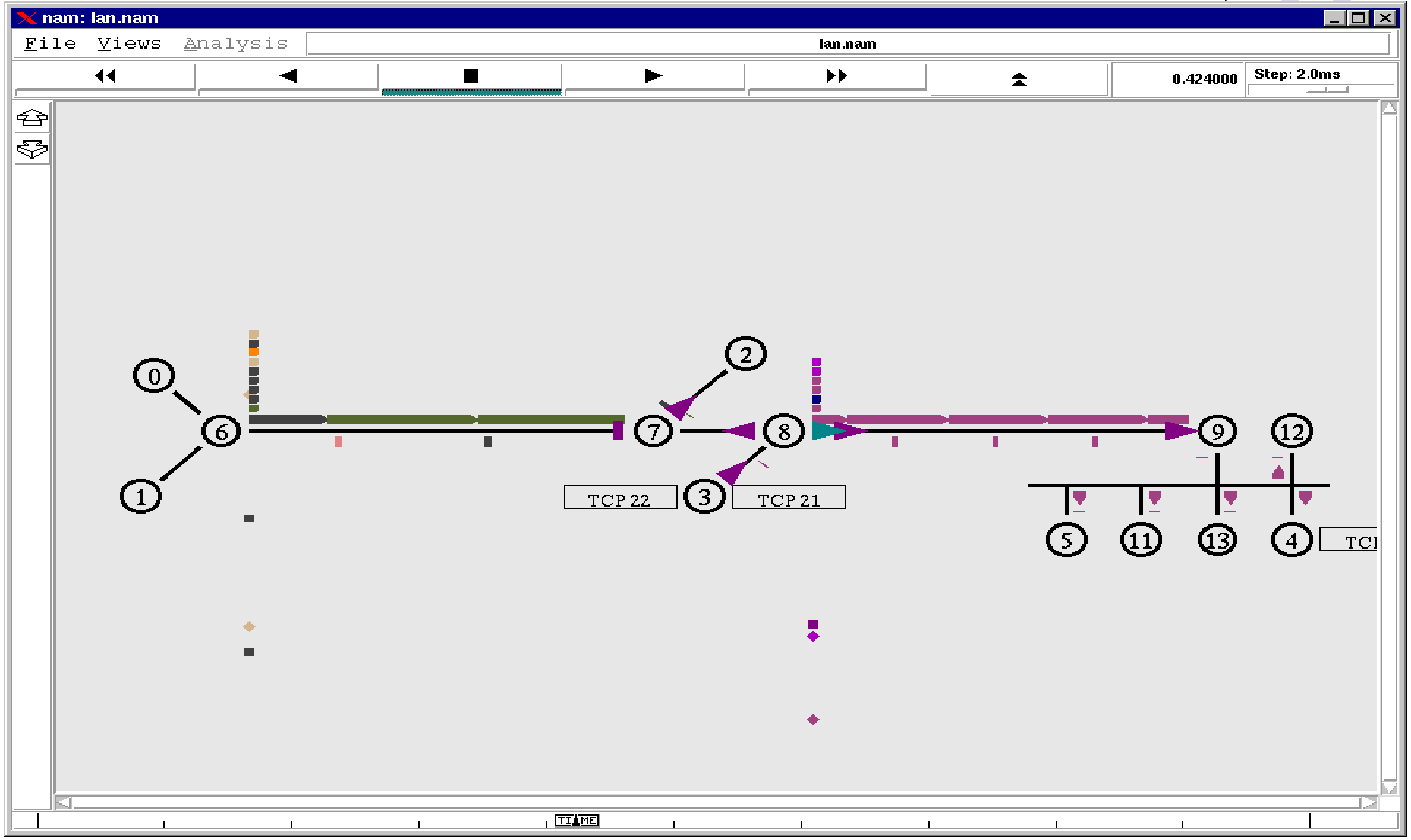
- **(Application) Traffic models and applications**
  - Web, FTP, telnet, constant-bit rate, real audio
- **(Translation) Transport protocols**
  - Unicast: TCP(Reno,Vegas,etc.), UDP
  - Multicast: SRM
- **(Network) Routing and queueing**
  - Wired routing: Dijkstra,DV
  - Wireless routing: mobile IP
  - Ad hoc routing :DSR,AODV,TORA,DSDV
  - Queueing protocols: RED, drop-tail, SFQ etc
- **(Physical) Physical media**
  - Wired(point-to-point, LANs), wireless (multiple propagation models)…

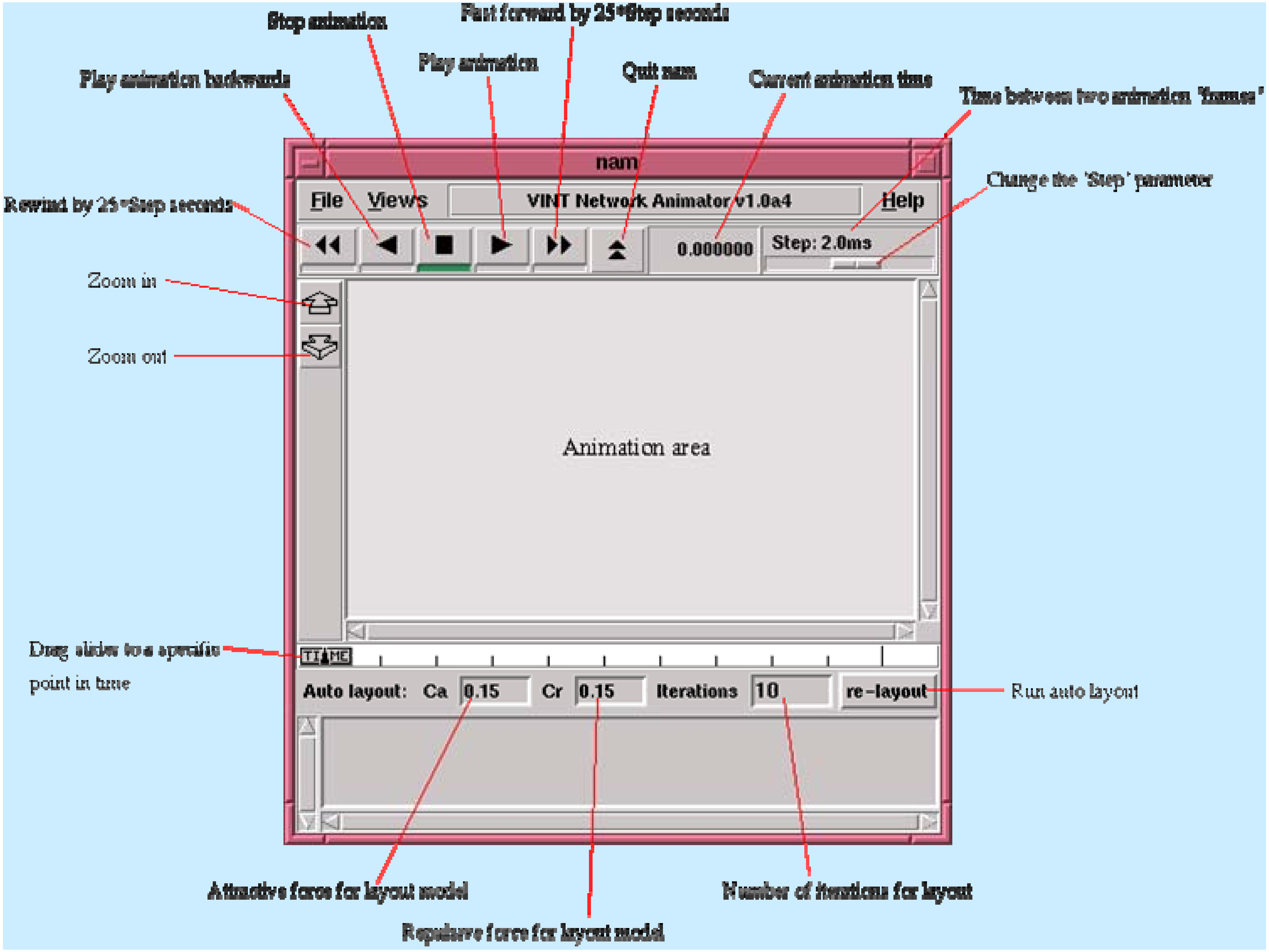
User view for NS-2



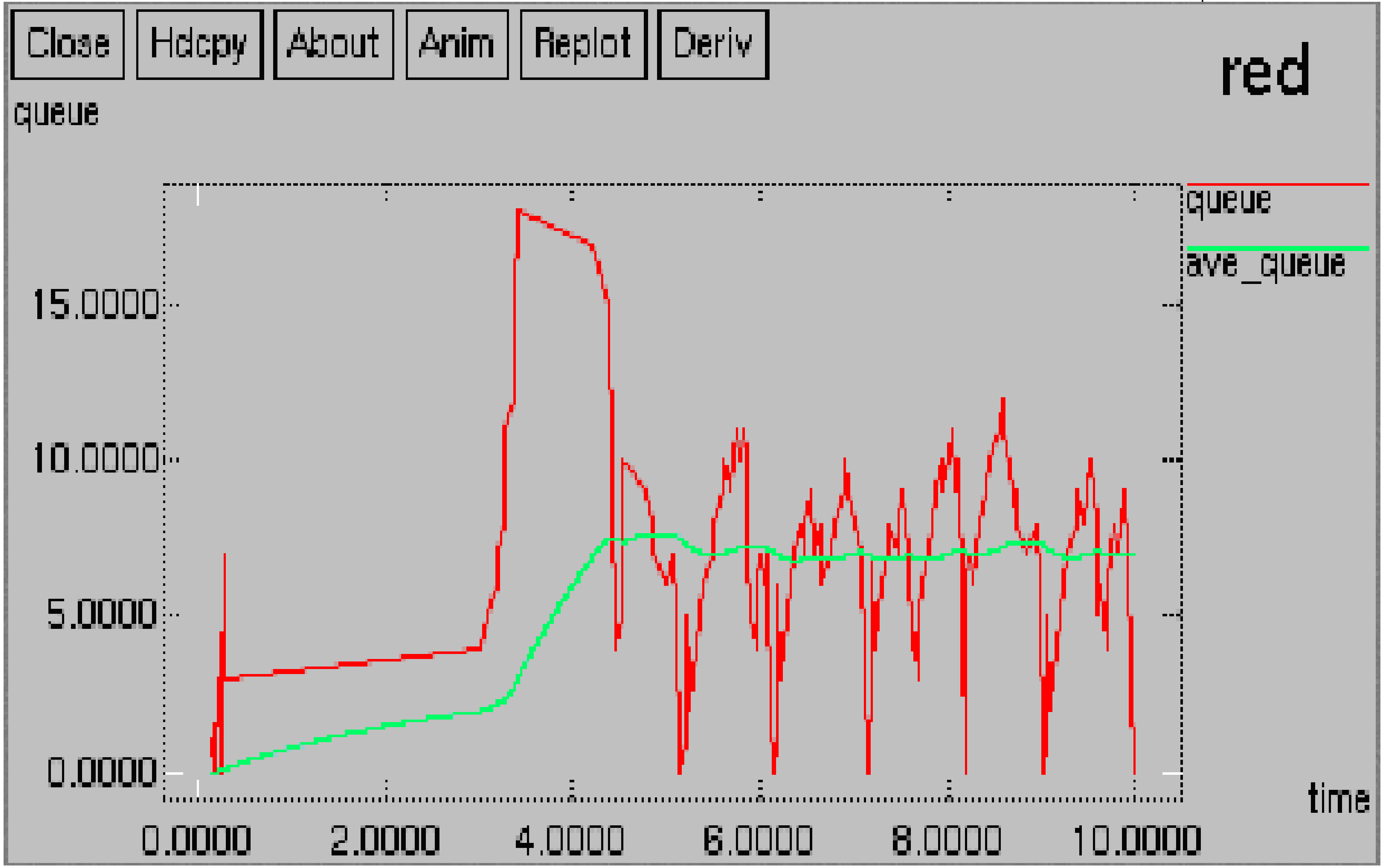
Visualization (nam)

- NAM: Viewing network simulation traces and real world packet traces.



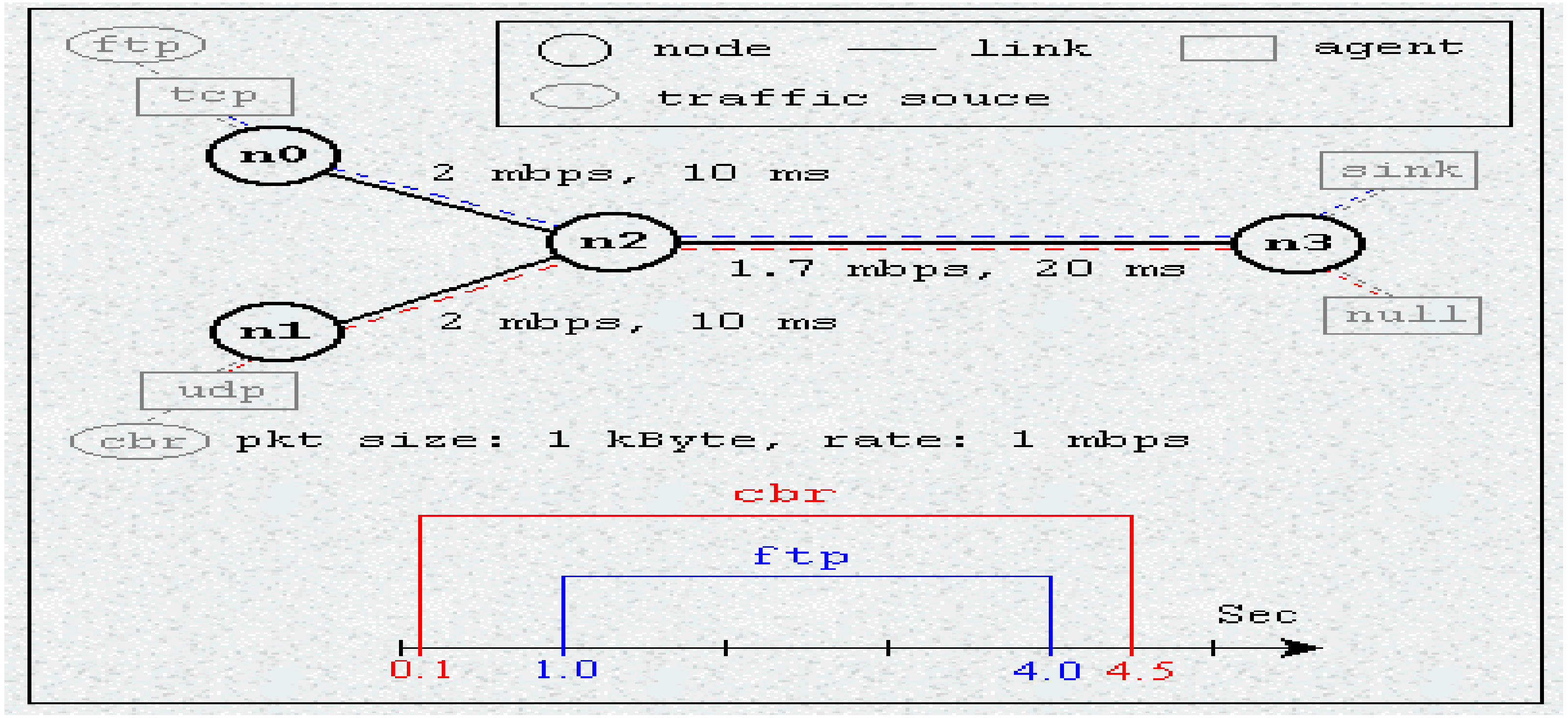

Stop animation
Fast forward by 25*Step seconds
Play animation
Play animation backwards
Quit nam
Current animation time
Time between two animation 'frames'
Change the 'Step' parameter
Rewind by 25*Step seconds
Zoom in
Zoom out
nam
File
Views
VINT Network Animator v1.0a4
Help
0.000000
Step: 2.0ms
Animation area
Drag slider to a specific point in time
TIME
Auto layout:
Ca
0.15
Cr
0.15
Iterations
10
re-layout
Run auto layout
Attractive force for layout model
Repulsive force for layout model
Number of iterations for layout

Visualization (xgraph)



Getting Start

- The main structure of TCL program (scenario)
  - **First part: 前置作業**
  (trace file open, create simulator)
  - **Second part:主程式碼**
  (simulation environment)
  - **Third part: 時間排程**
  (time schedular, event occur)



The first part:前置作業 (pre-definition)

- 1.建立一個模擬器物件 (Must)
```
set ns [new Simulator]
```

- 2.開啟一個nam trace 檔案 (Option)
```
set nf [open out.nam w]
$ns namtrace-all $nf
```

- 3.宣告一個finish程序(Must)
```
proc finish {} {
    global ns nf
    $ns flush-trace
    # Close the trace file
    close $nf
    # Execute nam on the trace file
   exec nam out.nam &
   exit 0
}
```

The second part:主程式碼 (simulation environment)

- 1.建立節點及其屬性
- 2.建立節點之間的實體連線及其性質
- 3.建立傳輸的應用程式及其傳輸速率
- 4.觀察節點的接收流量
- 5.預設節點為wird node
  - 5.1建立 wireless mobile IP連線
  - 5.2建立 ad-hoc網路

1、建立有線的節點 (Created wired nodes)

- set my_node0 [$ns node]
- set my_node1 [$ns node]

2、建立實體連線 (Created physical link)

- $ns duplex-link $n0 $n2 2Mb 10ms SFQ
- $ns simple-link $n1 $n2 3Mb 2ms RED
- 雙向、單向，頻寬，延遲時間，佇列的種類 (DropTail, FQ, SFQ, RED, CBQ)

3、設定佇列的大小 (set queue length)

- $ns queue-limit $my_node0 $my_node1 50

4、設定節點及連線的label、顏色(color) (Option)

```
$ns duplex-link-op $n0 $n1 color "green”
$ns duplex-link-op $n0 $n1 label “line1”
```

5、設定連線的成本 (link cost)

- $ns cost $my_node1 $my_node2 10
- $ns cost $my_node0 $my_node2 5

6、設定繞路(routing)屬性

- 預設 : static routing
- 靜態 routing : $ns rtproto Static
- 動態 routing : $ns rtproto DV
- $ns rtproto DV $n1 $n2 $n3

7、建立UTP連線

```
set udp0 [new Agent/UDP]      #建立Agent
$ns attach-agent $n0 $udp0   #此Agent連於n0節點上

set cbr0 [new Application/Traffic/CBR] #應用程式
$cbr0 set packetSize_ 400              #封包大小(bytes)
$cbr0 set interval_ 0.25               #設定傳送的間隔
$cbr0 attach-agent $udp0

set sink [new Agent/LossMonitor]    #建立接受端
$ns attach-agent $n3 $sink          #接受端節點
$ns connect $udp0 $sink        #將傳送端及接受端連在一起
```

8、建立FTP連線

```
set tcp [new Agent/TCP]              #建立TCP連線
$ns attach-agent $n0 $tcp            #將此TCP連至n0節點

set ftp [new Application/FTP]        #建立一個FTP應用程式
$ftp attach-agent $tcp               #將此FTP連至TCP物件

set sink [new Agent/TCPSink]         #建立TCP的接受端
$ns attach-agent $n3 $sink           #將接受端連至n3節點
$ns connect $tcp $sink               #將傳送端及接受端連
                                       在一起
```

第三部份　時間排程
(time schedular, event occur)

- 此部份要安排事件發生的前後順序　例如：
- 第2秒，　Ａ節點開始傳輸ＵＴＰ封包到Ｂ節點
  第3秒，　Ｃ節點開始傳輸ＴＣＰ封包到Ｄ節點
  第17秒，Ａ節點結束傳送
  第16秒，C節點結束傳送

```
$ns at 2   "$cbr0 start"
$ns at 17  “$cbr0 stop”
$ns at 3   "$ftp start"
$ns at 16  “$ftp stop”
$ns at 20  "finish"
```

- 設定某連線中斷及修復

```
$ns rtmodel-at 1.0 down $my_node1 $my_node
$ns rtmodel-at 2.0 up $my_node1 $my_node2
```

(Can use any Editor, file_name.tcl)

```
set ns [new Simulator]

set nf [open out.nam w]
$ns namtrace-all $nf

proc finish {} {
      global ns nf
      $ns flush-trace
      close $nf
      exec nam out.nam &
      exit 0
}

 set n0 [$ns node]
 set n1 [$ns node]
 set n2 [$ns node]
 set n3 [$ns node]

  $ns duplex-link $n0 $n2 2Mb 10ms DropTail
  $ns duplex-link $n1 $n2 2Mb 10ms DropTail
  $ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

  $ns queue-limit $n2 $n3 10
```

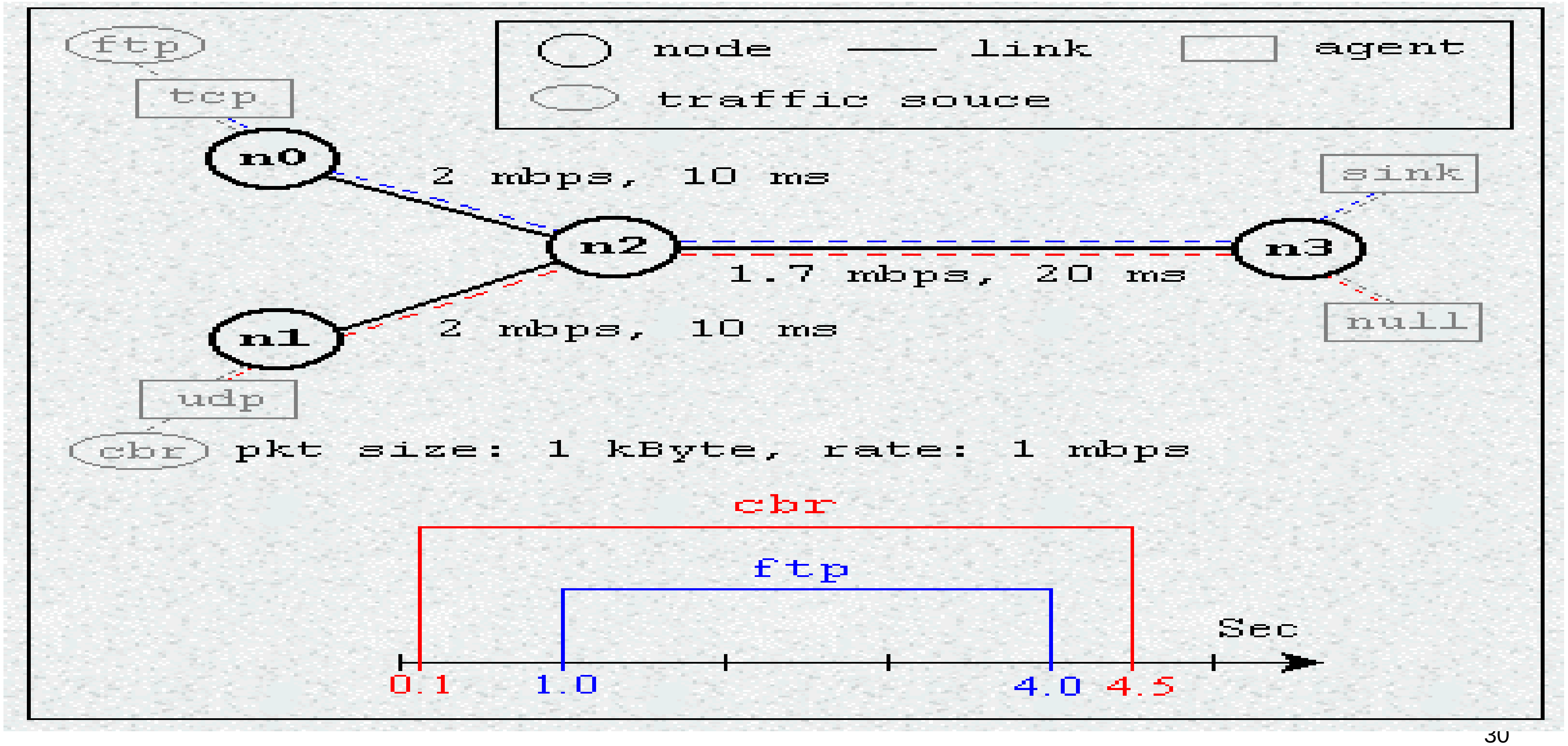
```
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp

set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set interval_ 0.25
```

```
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
$ns at 5.0 "finish"
$ns run
```



A part of trace file

1	2	3	4	5	6	7	8	9	10	11	12
r	1.340958	3	2	ack	40	-------	2	3.2	0.1	11	181
+	1.340958	2	0	ack	40	-------	2	3.2	0.1	11	181
-	1.340958	2	0	ack	40	-------	2	3.2	0.1	11	181
+	1.341225	2	3	cbr	210	-------	0	0.0	3.1	84	166
-	1.341225	2	3	cbr	210	-------	0	0.0	3.1	84	166
r	1.453111	1	0	tcp	60	-------	2	0.1	3.2	97	81
+	1.453111	0	2	tcp	60	-------	2	0.1	3.2	97	81
d	1.453111	0	2	tcp	60	-------	2	0.1	3.2	97	81

每個欄位所代表的意義如下：

1.代表事件的類別

    r：代表目的端收到packet
    +：代表 packet 放入 queue 中
    -：代表 packet 從 queue 中取出
    d：代表 queue 已經滿了，這個 packet 被 drop 掉

2.代表事件發生的時間

3.代表 packet 的 source node

4.代表 packet 的 destination node

5.代表 packet 的類別

6.代表 packet 的大小 (encoded in IP header)

7.代表 packet 的 flags

8.代表 connection(flow) 的 id

9.代表 source address (node.port)

10.代表 destinations address (node.port)

11.代表 packet 的 sequence number (network layer protocol's)

12.代表 packet 的 id (unique)

Http simulation

- **Chapter 38 in ns_doc.pdf**
  - Web cache as an application
- **Source node**
  - /root/ns-allinone-2.30/ns-2.30/webcache

```
class Page {
public:
        Page(int size) : size_(size) {}
        int size() const { return size_; }
        int& id() { return id_; }
        virtual WebPageType type() const = 0;

protected:
        int size_;
        int id_;
};
```

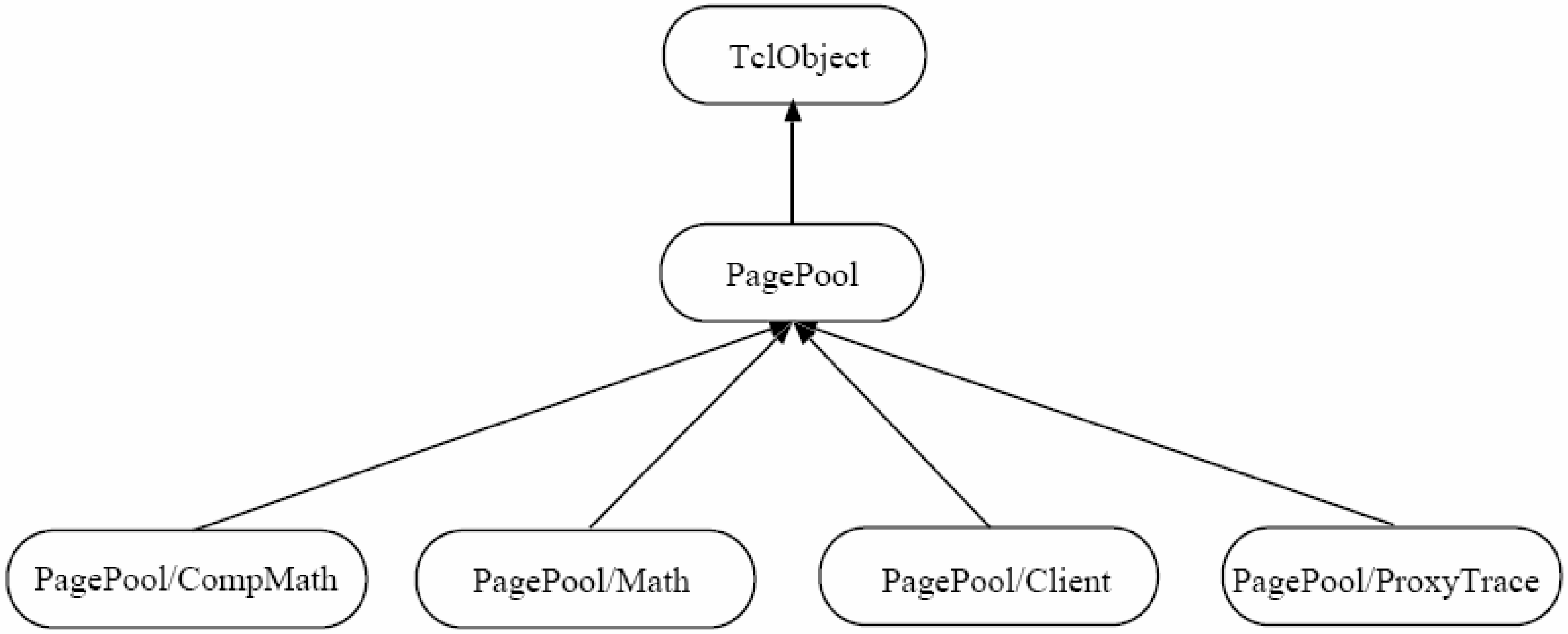


Figure 38.3: Class hierarchy of page pools

Web cache example

```
set ns [new Simulator]
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

set log [open “http.log” w]
set pgp [new PagePool/Math]
set tmp [new RandomVariable/Constant]
$tmp set val_ 1024
$pgp ranvar-size $tmp

set tmp [new RandomVariable/Exponential]
$tmp set avg_ 5
$pgp ranvar-age $tmp
```

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

set cache [new Http/Cache $ns $node(e)]
$cache log $log

set client [new Http/Client $ns $node(c)]
set tmp [new RandomVariable/Exponential]
$tmp set avg_ 5
$client set-interval-generator $tmp
$client set-page-generator $pgp
$client log $log
```

```
set startTime 1
set finishTime 50
$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
    $ns flush-trace
    flush $log
    close $log
    exit 0
}    $ns run
```

- 8.249329 i 0 C GET p _o47:0 s 1 z 43
- 8.255491 i 1 E MISS p _o47:0 c 0 s 2 z 43
- 8.406574 i 2 S MOD p _o47:0 m 8.406574066666666 n 17.44158715846218
- 8.562675 i 1 E ENT p _o47:0 m 8.406574066666666 z 1024 s 2
- 8.562675 i 1 E SND c 0 p _o47:0 z 1024
- 8.569591 i 0 C RCV p _o47:0 s 1 l 0.32026160000000026 z 1024
- 14.12012 i 0 C GET p _o47:0 s 1 z 43
- 14.122186 i 1 E HIT p _o47:0 c 0 s 2
- 14.122186 i 1 E SND c 0 p _o47:0 z 1024
- 14.129102 i 0 C RCV p _o47:0 s 1 l 0.008981600000000256 z 1024
- 17.441587 i 2 S MOD p _o47:0 m 17.44158715846218 n 19.672494112692458
- 19.672494 i 2 S MOD p _o47:0 m 19.672494112692458 n 30.538695496573247
- 29.87684 i 0 C GET p _o47:0 s 1 z 43
- 29.878906 i 1 E HIT p _o47:0 c 0 s 2
- 29.878906 i 1 E SND c 0 p _o47:0 z 1024
- 29.885822 i 0 C STA p _o47:0 s 2 l 12.444234441537823
- 29.885822 i 0 C RCV p _o47:0 s 1 l 0.008981600000020324 z 1024
- 30.538695 i 2 S MOD p _o47:0 m 30.538695496573247 n 34.832854175815498
- 30.632538 i 0 C GET p _o47:0 s 1 z 43
- 30.634604 i 1 E HIT p _o47:0 c 0 s 2
- 30.634604 i 1 E SND c 0 p _o47:0 z 1024
- 30.64152 i 0 C STA p _o47:0 s 2 l 13.199932341537824
- 30.64152 i 0 C RCV p _o47:0 s 1 l 0.008981600000020324 z 1024

Object Type	*Event Type*	*Values*
E	HIT	⟨Prefix⟩
E	MISS	⟨Prefix⟩ z ⟨RequestSize⟩
E	IMS	⟨Prefix⟩ z ⟨Size⟩ t ⟨CacheEntryTime⟩
E	REF	p ⟨PageID⟩ s ⟨ServerID⟩ z ⟨Size⟩
E	UPD	p ⟨PageID⟩ m ⟨LastModifiedTime⟩ z ⟨PageSize⟩ s ⟨ServerID⟩
E	GUPD	z ⟨PageSize⟩
E	SINV	p ⟨PageID⟩ m ⟨LastModTime⟩ z ⟨PageSize⟩
E	GINV	p ⟨PageID⟩ m ⟨LastModTime⟩
E	SPF	p ⟨PageID⟩ c ⟨DestCache⟩
E	RPF	p ⟨PageID⟩ c ⟨SrcCache⟩
E	ENT	p ⟨PageID⟩ m ⟨LastModifiedTime⟩ z ⟨PageSize⟩ s ⟨ServerID⟩
C	GET	p ⟨PageID⟩ s ⟨PageServerID⟩ z ⟨RequestSize⟩
C	STA	p ⟨PageID⟩ s ⟨OrigServerID⟩ l ⟨StaleTime⟩
C	RCV	p ⟨PageID⟩ s ⟨PageServerID⟩ l ⟨ResponseTime⟩ z ⟨PageSize⟩
S	INV	p ⟨PageID⟩ m ⟨LastModifiedTime⟩ z ⟨Size⟩
S	UPD	p ⟨PageID⟩ m ⟨LastModifiedTime⟩ z ⟨Size⟩
S	SND	p ⟨PageID⟩ m ⟨LastModifiedTime⟩ z ⟨PageSize⟩ t ⟨Requesttype⟩
S	MOD	p ⟨PageID⟩ n ⟨NextModifyTime⟩

⟨Prefix⟩ is the information common to all trace entries. It includes:

p ⟨PageID⟩ | c ⟨RequestClientID⟩ | s ⟨PageServerID⟩

Object Type	*Event Type*	*Explaination*
E	HIT	Cache hit. PageSererID is the id of the "owner" of the page.
E	MISS	Cache miss. In this case the cache will send a request to the server to fetch the page.
E	IMS	If-Modified-Since. Used by TTL procotols to validate an expired page.
E	REF	Page refetch. Used by invalidation protocols to refetch an invalidated page.
E	UPD	Page update. Used by invalidation protocols to "push" updates from parent cache to children caches.
E	SINV	Send invalidation.
E	GINV	Get invalidation.
E	SPF	Send a pro forma
E	RPF	Receive a pro forma
E	ENT	Enter a page into local page cache.
C	GET	Client sends a request for a page.
C	STA	Client gets a stale hit. OrigModTime is the modification time in the web server, CurrModTime is the local page's modification time.
C	RCV	Client receives a page.
S	SND	Server send a response.
S	UPD	Server pushes a page update to its "primary cache". Used by invalidation protocol only.
S	INV	Server sends an invalidation message. Used by invalidation protocol only.
S	MOD	Server modified a page. The page will be modified next at ⟨NextModifyTime⟩.

PagePool/Math

- This is the simplest type of page pool. It has only one page, whose size can be generated by a given random variable.
- Page modification sequence and request sequence are generated using two given random variables.

PagePool/CompMath

- By a compound page we mean a page which consists of a main text page and a number of embedded objects, e.g., GIFs. We model a compound page as a main page and several component objects.

- The main page is always assigned with ID 0. All component pages have the same size; both the main page size and component object size is fixed, but adjustable through OTcl-bound variables main_size_ and comp_size_, respectively. The number of component objects can be set using the OTcl-bound variable num_pages_.

PagePool/ProxyTrace

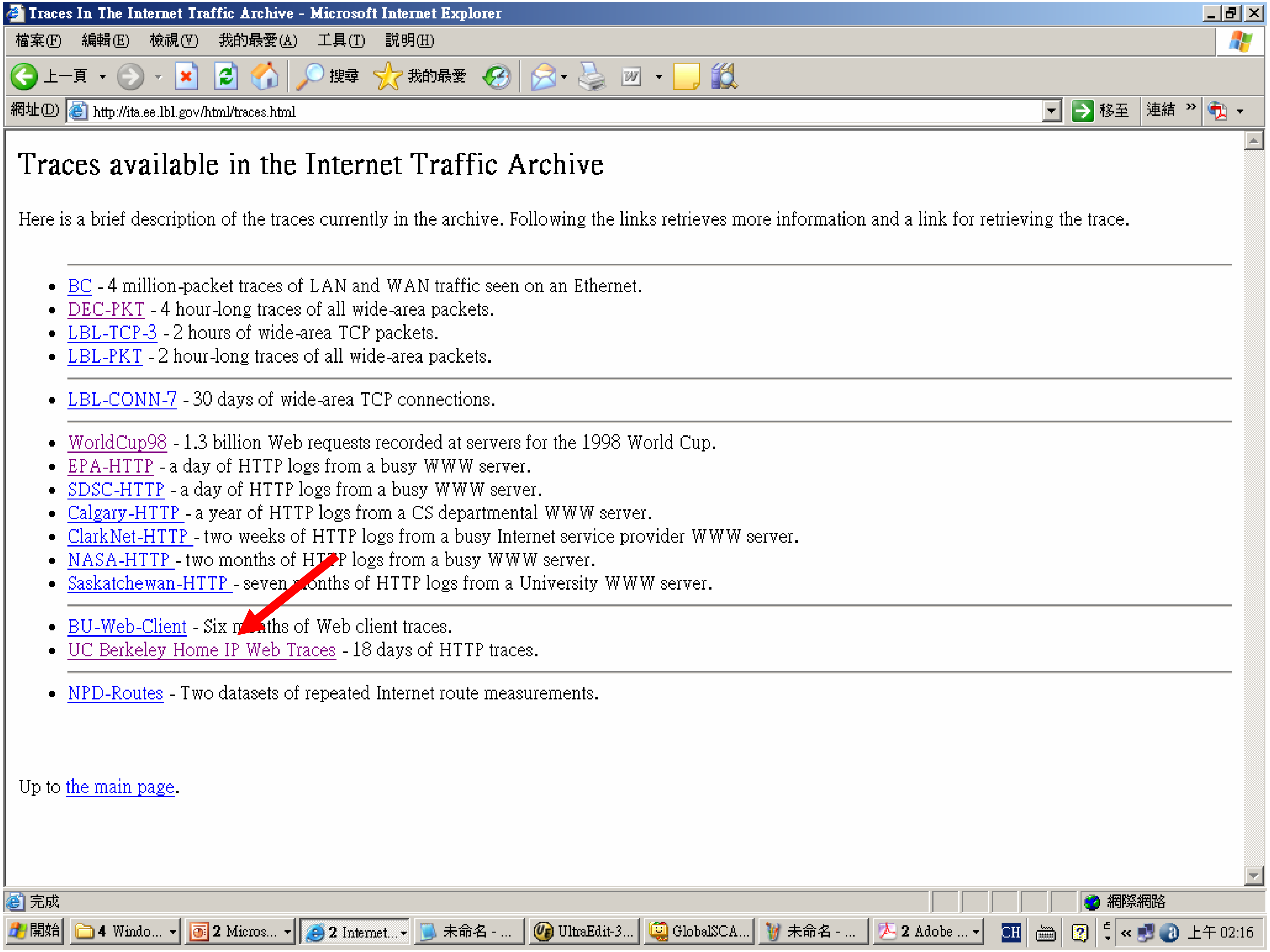
- The above two page pool synthesize request stream to a single web page by two random variables: one for request interval, another for requested page ID. Sometimes users may want more complicated request stream, which consists of multiple pages and exhibits spatial locality and temporal locality.

- There exists one proposal (SURGE [3]) which generates such request streams, we choose to provide an alternative solution: use real web proxy cache trace (or server trace).

- The class PagePool/ProxyTrace uses real traces to drive simulation. Because there exist many web traces with different formats, they should be converted into a intermediate format before fed into this page pool. The converter is available in NS-2.

Real Traces in the Internet Traffic Archive

- http://ita.ee.lbl.gov/html/traces.html


Traces In The Internet Traffic Archive - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
網址(D) http://ita.ee.lbl.gov/html/traces.html
移至
連結
完成
網際網路
開始
4 Windo...
2 Micros...
2 Internet...
未命名 - ...
UltraEdit-3...
GlobalSCA...
未命名 - ...
2 Adobe ...
CH
上午 02:16

Traces available in the Internet Traffic Archive

Here is a brief description of the traces currently in the archive. Following the links retrieves more information and a link for retrieving the trace.

- BC - 4 million-packet traces of LAN and WAN traffic seen on an Ethernet.
- DEC-PKT - 4 hour-long traces of all wide-area packets.
- LBL-TCP-3 - 2 hours of wide-area TCP packets.
- LBL-PKT - 2 hour-long traces of all wide-area packets.

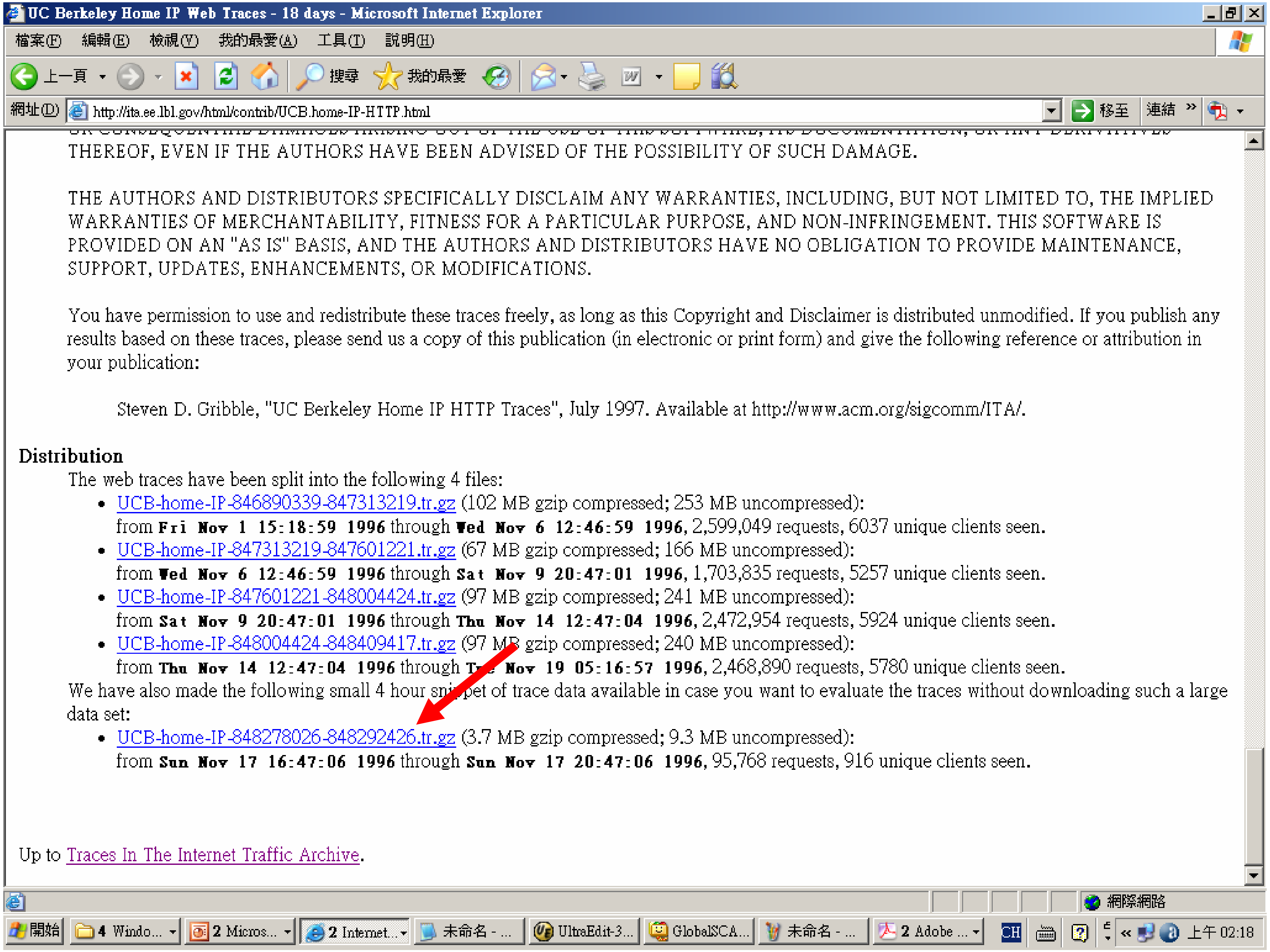
- LBL-CONN-7 - 30 days of wide-area TCP connections.

- WorldCup98 - 1.3 billion Web requests recorded at servers for the 1998 World Cup.
- EPA-HTTP - a day of HTTP logs from a busy WWW server.
- SDSC-HTTP - a day of HTTP logs from a busy WWW server.
- Calgary-HTTP - a year of HTTP logs from a CS departmental WWW server.
- ClarkNet-HTTP - two weeks of HTTP logs from a busy Internet service provider WWW server.
- NASA-HTTP - two months of HTTP logs from a busy WWW server.
- Saskatchewan-HTTP - seven months of HTTP logs from a University WWW server.

- BU-Web-Client - Six months of Web client traces.
- UC Berkeley Home IP Web Traces - 18 days of HTTP traces.

- NPD-Routes - Two datasets of repeated Internet route measurements.

Up to the main page.


UC Berkeley Home IP Web Traces - 18 days - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛
網址(D) http://ita.ee.lbl.gov/html/contrib/UCB.home-IP-HTTP.html
移至 連結
網際網路
開始 4 Windo... 2 Micros... 2 Internet... 未命名 - ... UltraEdit-3... GlobalSCA... 未命名 - ... 2 Adobe ... CH 上午 02:18

THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

You have permission to use and redistribute these traces freely, as long as this Copyright and Disclaimer is distributed unmodified. If you publish any results based on these traces, please send us a copy of this publication (in electronic or print form) and give the following reference or attribution in your publication:

Steven D. Gribble, "UC Berkeley Home IP HTTP Traces", July 1997. Available at http://www.acm.org/sigcomm/ITA/.

Distribution

The web traces have been split into the following 4 files:

- UCB-home-IP-846890339-847313219.tr.gz (102 MB gzip compressed; 253 MB uncompressed):
  from **Fri Nov 1 15:18:59 1996** through **Wed Nov 6 12:46:59 1996**, 2,599,049 requests, 6037 unique clients seen.
- UCB-home-IP-847313219-847601221.tr.gz (67 MB gzip compressed; 166 MB uncompressed):
  from **Wed Nov 6 12:46:59 1996** through **Sat Nov 9 20:47:01 1996**, 1,703,835 requests, 5257 unique clients seen.
- UCB-home-IP-847601221-848004424.tr.gz (97 MB gzip compressed; 241 MB uncompressed):
  from **Sat Nov 9 20:47:01 1996** through **Thu Nov 14 12:47:04 1996**, 2,472,954 requests, 5924 unique clients seen.
- UCB-home-IP-848004424-848409417.tr.gz (97 MB gzip compressed; 240 MB uncompressed):
  from **Thu Nov 14 12:47:04 1996** through **Tue Nov 19 05:16:57 1996**, 2,468,890 requests, 5780 unique clients seen.

We have also made the following small 4 hour snippet of trace data available in case you want to evaluate the traces without downloading such a large data set:

- UCB-home-IP-848278026-848292426.tr.gz (3.7 MB gzip compressed; 9.3 MB uncompressed):
  from **Sun Nov 17 16:47:06 1996** through **Sun Nov 17 20:47:06 1996**, 95,768 requests, 916 unique clients seen.

Up to Traces In The Internet Traffic Archive.

Download and Extract

- If the file is put in /root
- Step 1: cd /root
- Step 2: tar zxvf UCB-home-IP-848278026-848292426.tr.gz
- Step 3: cd /root/UCB-home-IP-848278026-848292426.tr.gz
- Step 4: cp X /root/ns-allinone-2.29/ns-2.29/indep-utils/webtrace-conv

How to convert real trace for NS2 ?

- cd /root/ns-allinone-2.29/ns-2.29/indep-utils/webtrace-conv
- Four directory
  - dec
    - DEC proxy trace (1996),
  - epa
    - EPA web server trace
  - nlanr
    - NLANR proxy trace
  - ucb
    - UCB Home-IP trace

- cd /root/ns-allinone-2.29/ns-2.29/indep-utils/webtrace-conv/ucb
- ./ucb-tr-stat 1000 50 0 < ucb_trace_file.tr
- reqlog
  - {<time> <clientID> <serverID> <URL_ID>}
  - i <Duration> <Number_of_unique_URLs>
- pglog
  - <serverID> <URL_ID> <PageSize> <AccessCount>

- PagePool/ProxyTrace takes these two file as input, and use them to drive simulation. Because most existing web proxy traces do not contain complete page modification information, we choose to use a bimodal page modification model.
- We allow user to select x% of the pages to have one random page modification interval generator, and the rest of the pages to have another generator. In this way, it’s possible to let x% pages to be dynamic, i.e., modified frequently, and the rest static. Hot pages are evenly distributed among all pages.
- For example, assume 10% pages are dynamic, then if we sort pages into a list according to their popularity, then pages 0, 10, 20, . . . are dynamic, rest are static. Because of this selection mechanism, we only allow bimodal ratio to change in the unit of 10%.

simple-webcache-trace.tcl

```
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.log" w]

# 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/Exponential]
$tmp set avg_ 5                                   ;# average page age
$pgp ranvar-dp $tmp

# Static page age generator
set tmp [new RandomVariable/Constant]
$tmp set val_ 10000
$pgp ranvar-sp $tmp
```

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

set cache [new Http/Cache $ns $node(e)]
$cache log $log

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

set startTime 1
set finishTime 50
$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
    $ns flush-trace
    flush $log
    close $log
    exit 0
}

$ns run
```

```
1 i 0 C GET p _o47:30 s 1 z 43
1.006162 i 1 E MISS p _o47:30 c 0 s 2 z 43
1.157245 i 2 S MOD p _o47:30 m 1.1572450666666669 n 7.0280359876835199
1.374656 i 0 C GET p _o47:34 s 1 z 43
1.376722 i 1 E MISS p _o47:34 c 0 s 2 z 43
1.422442 i 0 C GET p _o47:52 s 1 z 43
1.424508 i 1 E MISS p _o47:52 c 0 s 2 z 43
1.427165 i 2 S MOD p _o47:34 m 1.4271650666666664 n 10001.427165066667
1.480237 i 1 E ENT p _o47:30 m 1.1572450666666669 z 5331 s 2
1.480237 i 1 E SND c 0 p _o47:30 z 5331
1.497075 i 0 C RCV p _o47:30 s 1 l 0.49707466666666611 z 5331
1.53068 i 2 S MOD p _o47:52 m 1.5306797333333331 n 10001.530679733334
1.583752 i 1 E ENT p _o47:34 m 0 z 340 s 2
1.583752 i 1 E SND c 0 p _o47:34 z 340
1.586056 i 0 C RCV p _o47:34 s 1 l 0.21139973333333328 z 340
1.844387 i 0 C GET p _o47:45 s 1 z 43
1.846453 i 1 E MISS p _o47:45 c 0 s 2 z 43
1.896896 i 2 S MOD p _o47:45 m 1.8968960666666665 n 10001.896896066666
1.949968 i 1 E ENT p _o47:52 m 0 z 28543 s 2
1.949968 i 1 E SND c 0 p _o47:52 z 28543
1.98759 i 0 C RCV p _o47:52 s 1 l 0.56514766666666372 z 28543
2.088032 i 1 E ENT p _o47:45 m 0 z 6667 s 2
2.088032 i 1 E SND c 0 p _o47:45 z 6667
2.106034 i 0 C RCV p _o47:45 s 1 l 0.26164746666666655 z 6667
2.348237 i 0 C GET p _o47:5 s 1 z 43
2.350303 i 1 E MISS p _o47:5 c 0 s 2 z 43
2.400746 i 2 S MOD p _o47:5 m 2.4007460666666667 n 10002.400746066667
2.451402 i 1 E ENT p _o47:5 m 0 z 83 s 2
```

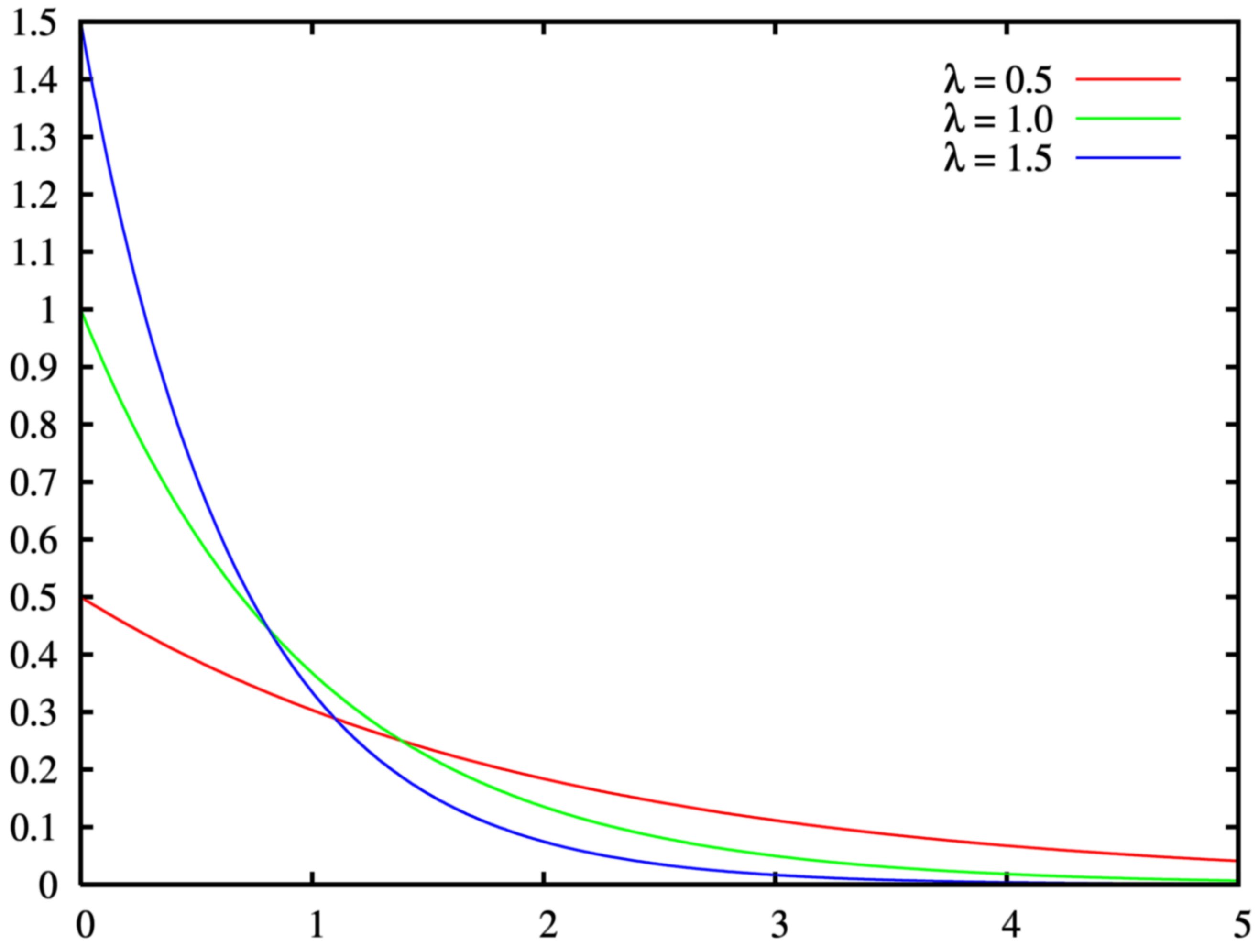
Object Type	*Event Type*	*Values*
E	HIT	⟨Prefix⟩
E	MISS	⟨Prefix⟩ z ⟨RequestSize⟩
E	IMS	⟨Prefix⟩ z ⟨Size⟩ t ⟨CacheEntryTime⟩
E	REF	p ⟨PageID⟩ s ⟨ServerID⟩ z ⟨Size⟩
E	UPD	p ⟨PageID⟩ m ⟨LastModifiedTime⟩ z ⟨PageSize⟩ s ⟨ServerID⟩
E	GUPD	z ⟨PageSize⟩
E	SINV	p ⟨PageID⟩ m ⟨LastModTime⟩ z ⟨PageSize⟩
E	GINV	p ⟨PageID⟩ m ⟨LastModTime⟩
E	SPF	p ⟨PageID⟩ c ⟨DestCache⟩
E	RPF	p ⟨PageID⟩ c ⟨SrcCache⟩
E	ENT	p ⟨PageID⟩ m ⟨LastModifiedTime⟩ z ⟨PageSize⟩ s ⟨ServerID⟩
C	GET	p ⟨PageID⟩ s ⟨PageServerID⟩ z ⟨RequestSize⟩
C	STA	p ⟨PageID⟩ s ⟨OrigServerID⟩ l ⟨StaleTime⟩
C	RCV	p ⟨PageID⟩ s ⟨PageServerID⟩ l ⟨ResponseTime⟩ z ⟨PageSize⟩
S	INV	p ⟨PageID⟩ m ⟨LastModifiedTime⟩ z ⟨Size⟩
S	UPD	p ⟨PageID⟩ m ⟨LastModifiedTime⟩ z ⟨Size⟩
S	SND	p ⟨PageID⟩ m ⟨LastModifiedTime⟩ z ⟨PageSize⟩ t ⟨Requesttype⟩
S	MOD	p ⟨PageID⟩ n ⟨NextModifyTime⟩

⟨Prefix⟩ is the information common to all trace entries. It includes:

p ⟨PageID⟩ | c ⟨RequestClientID⟩ | s ⟨PageServerID⟩

Object Type	*Event Type*	*Explaination*
E	HIT	Cache hit. PageSererID is the id of the "owner" of the page.
E	MISS	Cache miss. In this case the cache will send a request to the server to fetch the page.
E	IMS	If-Modified-Since. Used by TTL procotols to validate an expired page.
E	REF	Page refetch. Used by invalidation protocols to refetch an invalidated page.
E	UPD	Page update. Used by invalidation protocols to "push" updates from parent cache to children caches.
E	SINV	Send invalidation.
E	GINV	Get invalidation.
E	SPF	Send a pro forma
E	RPF	Receive a pro forma
E	ENT	Enter a page into local page cache.
C	GET	Client sends a request for a page.
C	STA	Client gets a stale hit. OrigModTime is the modification time in the web server, CurrModTime is the local page's modification time.
C	RCV	Client receives a page.
S	SND	Server send a response.
S	UPD	Server pushes a page update to its "primary cache". Used by invalidation protocol only.
S	INV	Server sends an invalidation message. Used by invalidation protocol only.
S	MOD	Server modified a page. The page will be modified next at ⟨NextModifyTime⟩.

Exponential distribution pdf



9、觀察接收封包的情況

```
set f0 [open out1.tr w]
proc record {} {
        global sink  f0
        set ns [Simulator instance]
        set time 0.1
        set bw0 [$sink0 set bytes_]
        set now [$ns now]
        puts $f0 "$now [expr $bw0/$time]"
        $sink0 set bytes_ 0
        $ns at [expr $now+$time] "record"
}
```

- **LossMonitor**
  - **nlost_** :Number of packets lost
  - **npkts_** :Number of packets received
  - **bytes_** :Number of bytes received
  - **lastPktTime_** :Time at which the last packet was received
  - **expected_** :The expected sequence number of the next packet