บทที่ 3

เอ็นเอสทู

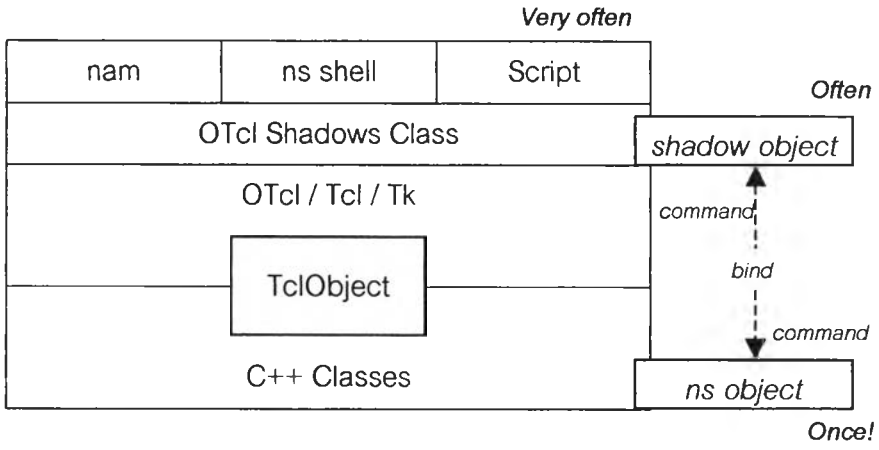
เอ็นเอสทูเป็นซอฟต์แวร์ที่ถูกพัฒนาขึ้นมาเพื่อช่วยจำลองการทำงานของระบบเครือข่ายต่างๆ เช่น แลน เครือข่ายไร้สาย เครือข่ายดาวเทียม ฯลฯ โดยเอ็นเอสทูถูกนำมาใช้กันอย่างแพร่หลาย เนื่องจากเอ็นเอสทูเป็นซอฟต์แวร์เปิด ดังนั้นทุกคนสามารถนำเอ็นเอสทูไปใช้จำลองการทำงานด้านต่างๆ ได้ และยังสามารถทำการดัดแปลงแก้ไขโค้ดเดิม เพื่อเพิ่มการจำลองการทำงานแบบใหม่ และส่วนประกอบใหม่ๆ เข้าไปได้ ซึ่งในปัจจุบันมีเว็บไซต์ของเอ็นเอสทูอย่างเป็นทางการ [3] โดยภายในเว็บไซต์จะมีรายละเอียดต่างๆ ของเอ็นเอสทู เช่น วิธีการติดตั้ง วิธีการใช้งานและปัญหาต่างๆ ที่พบจากเอ็นเอสทู เป็นต้น

สำหรับโปรแกรมจำลองการทำงานอื่นๆ ส่วนใหญ่จะเป็นโปรแกรมที่เขียนขึ้นมาใช้ทดสอบเฉพาะส่วนที่สนใจ และไม่เผยแพร่ให้ผู้อื่น ทำให้เมื่อมีการจำลองการทำงานในลักษณะเดิมๆ จะต้องเขียนโปรแกรมใหม่ทั้งหมด ซึ่งทำให้ไม่สะดวก และเสียเวลาอย่างมาก อย่างเช่น WebCASE [2] เป็นโปรแกรมจำลองการทำงานของเว็บแคชที่เน้นขั้นตอนวิธีการแทนที่เอกสารภายในแคชเพียงตัวเดียวเท่านั้น ไม่มีความต้องกันของเว็บแคช และไม่สามารถทำงานในระบบที่มีแคชมากกว่าหนึ่งตัวได้ ดังนั้นในงานวิทยานิพนธ์นี้จึงใช้เอ็นเอสทูซึ่งใช้กันอย่างแพร่หลายเป็นโปรแกรมที่ใช้จำลองการทำงานของเว็บแคชโดยพัฒนาส่วนประกอบเพิ่มเติมเข้าไปเพื่อให้สามารถจำลองการทำงานได้หลายรูปแบบ โดยใช้เอ็นเอสทูเป็นเครื่องมือในการจำลองเครือข่าย ซึ่งสามารถกำหนดโทโพโลจีแบบต่างๆ ได้ จากโหนด และลิงค์ที่สร้างขึ้นมา แต่ละโหนดสามารถกำหนดให้ทำหน้าที่เป็นเว็บไคลแอนท์ เว็บแคช หรือเว็บเซิร์ฟเวอร์ก็ได้ โดยสามารถกำหนดโหนดที่ทำหน้าที่ต่างๆ ได้มากกว่า 1 โหนด และทำการเชื่อมต่อโหนดต่างๆ ด้วยลิงค์เพื่อให้แต่ละโหนดสามารถสื่อสารแลกเปลี่ยนข้อมูลร่วมกันได้ ในส่วนของเว็บแคชเมื่อได้พัฒนาส่วนประกอบเพิ่มเติมเข้าไปจะทำให้สามารถจำลองการทำงานโดยใช้วิธีการต่างๆ ได้หลายวิธี ทั้งในส่วนความต้องกันของเว็บแคช ส่วนขั้นตอนวิธีการแทนที่ และส่วนสถาปัตกรรมของการแคช ซึ่งจะทำให้สามารถจำลองการทำงานของเว็บแคชได้หลายรูปแบบ

เนื่องจากเอ็นเอสทูพัฒนามาจากภาษาซีพลัสพลัส และภาษาโอทีซีแอล ดังนั้นการพัฒนาส่วนประกอบต่างๆ สำหรับเอ็นเอสทูจึงพัฒนาจากสองภาษานี้เป็นหลัก ซึ่งการที่จะพัฒนาส่วนประกอบต่างๆ ให้สามารถทำงานร่วมกับเอ็นเอสทูได้จะต้องรู้ และเข้าใจถึงโครงสร้าง และการทำงานของเอ็นเอสทู ดังรายละเอียดที่จะกล่าวถึงในบทนี้

3.1 โครงสร้างของเอ็นเอสทู

เอ็นเอสทูเป็นซอฟต์แวร์สำหรับจำลองการทำงานที่พัฒนาขึ้นมาจากภาษาซีพลัสพลัส และภาษาโอทีซีแอล สาเหตุที่เอ็นเอสทูพัฒนามาจาก 2 ภาษา เนื่องจากซอฟต์แวร์สำหรับจำลองการทำงานประกอบด้วยส่วนสำคัญ 2 ส่วน ส่วนแรกคือ รายละเอียดของการจำลองการทำงานของโพรโทคอลต่างๆ ซึ่งจำเป็นต้องใช้ภาษาสำหรับสร้างโปรแกรมระบบ (System Programming Language) ที่มีประสิทธิภาพในการจัดการขนาดของข้อมูล (Byte) ส่วนหัวของกลุ่มข้อมูล (Packet header) และการทำให้ขั้นตอนวิธีต่างๆ สามารถทำงานได้บนชุดของข้อมูลขนาดใหญ่ ดังนั้นระยะเวลาที่ใช้ในการทำงาน (Run-time) จึงเป็นสิ่งที่สำคัญสำหรับงานเหล่านี้ ส่วนเวลาที่ใช้ในการค้นหาบั๊ก การแก้ไขบั๊ก การสั่งให้ทำงานอีกครั้ง หรือการแปลโปรแกรมใหม่อีกครั้ง (Recompile) นั้นสำคัญน้อยกว่า ส่วนที่สอง คือ ส่วนอื่นๆ ของการวิจัยระบบเครือข่าย ซึ่งรวมถึงการปรับค่าพารามิเตอร์ หรือโครงแบบ (Configuration) ที่หลากหลาย ความรวดเร็วในการสำรวจแผนการ (Scenario) ต่างๆ ซึ่งในกรณีเหล่านี้เวลาที่ใช้ในการปรับเปลี่ยนแบบจำลอง และการสั่งให้ทำงานอีกครั้งเป็นสิ่งที่สำคัญมากกว่า ดังนั้นเอ็นเอสทูจึงประกอบด้วย 2 ภาษา ซึ่งภาษาซีพลัสพลัสนั้นทำงานได้รวดเร็วแต่การปรับเปลี่ยนทำได้ช้า ทำให้เหมาะสำหรับการใช้ในส่วนรายละเอียดของโพรโทคอล สำหรับภาษาโอทีซีแอลทำงานได้ช้ากว่า แต่สามารถเปลี่ยนแปลงได้อย่างรวดเร็ว ทำให้เหมาะกับการทำโครงแบบสำหรับการจำลองการทำงานแบบต่างๆ ดังรูปที่ 3.1



รูปที่ 3.1 แสดงโครงสร้างของเอ็นเอสทู

การที่จะทำให้สามารถใช้ภาษาซีพลัสพลัส และภาษาโอทีซีแอลร่วมกันได้ จะต้องใช้ TclCL (Tcl with classes) เป็นตัวต่อประสาน ซึ่งใน TclCL จะประกอบด้วยคลาสเป็นจำนวนมาก แต่เอ็นเอสทูจะใช้คลาสเพียง 6 คลาสเท่านั้น ซึ่งคลาสทั้ง 6 ประกอบด้วย

1. คลาส Tcl เป็นคลาสที่หุ้มห่ออินสแตนส์จริงๆ ของตัวแปลคำสั่งภาษาโอทีซีแอล (OTcl interpreter) และจัดเตรียมเมท็อดเพื่อเข้าถึงและสื่อสารกับตัวแปลคำสั่งภาษาโอทีซีแอล ซึ่งเมท็อดที่อธิบายอยู่ในหัวข้อนี้จะเกี่ยวข้องกับการพัฒนาเอ็นเอสทูด้วยภาษาซีพลัสพลัส โดยเมท็อดที่คลาส Tcl จัดเตรียมไว้ให้มีจุดประสงค์เพื่อการดำเนินการดังต่อไปนี้

- เพื่อให้ได้การอ้างอึงถึงอินสแตนส์ของคลาส Tcl
- เพื่อเรียกกระบวนงาน (Procedure) ภาษาโอทีซีแอลผ่านทางตัวแปลคำสั่ง
- เพื่อดึง หรือส่งผลกลับไปยังตัวแปลคำสั่ง
- เพื่อรายงานถึงข้อผิดพลาดต่างๆ และหยุดการทำงานตามข้อผิดพลาด
- เพื่อเก็บ และค้นดู TclObject
- เพื่อให้เข้าถึงตัวแปลคำสั่งได้โดยตรง

2. คลาส TclObject เป็นคลาสพื้นฐานของคลาสส่วนใหญ่ที่อยู่ในลำดับชั้นของคลาสในโอทีซีแอล และซีพลัสพลัส อ็อบเจกต์ทุกๆ อ็อบเจกต์ในคลาสนี้ถูกสร้างภายในตัวแปลคำสั่งโดยผู้ใช้ และอ็อบเจกต์ที่เหมือนกับอ็อบเจกต์เงาจะถูกสร้างในลำดับชั้นของคลาสในซีพลัสพลัส อ็อบเจกต์ทั้งสองนี้มีความเกี่ยวเนื่องกันอย่างใกล้ชิด โดยเมท็อดในคลาส TclObject ใช้สำหรับการดำเนินการดังต่อไปนี้

- สร้าง และการลบ TclObject
- เชื่อมโยงตัวแปร (Variable Binding)
- ติดตามตัวแปร (Variable Tracing)
- กำหนดและการเรียกใช้เมท็อด command

3. คลาส TclClass คลาสนี้เป็นคลาสเสมือนโดยแท้ ซึ่งสืบทอดมาจากคลาสพื้นฐานโดยจะมีหน้าที่ 2 อย่าง คือ สร้างลำดับชั้นของคลาสในโอทีซีแอลเพื่อที่จะเป็นเหมือนกระจก (Mirror) ของลำดับชั้นของคลาสในซีพลัสพลัส และเพื่อจัดเตรียมเมท็อดสำหรับ สร้างอินสแตนท์ TclObject ตัวใหม่ขึ้นมา ซึ่งคลาสแต่ละคลาสที่สืบทอดมานั้นจะสัมพันธ์กับคลาสในซีพลัสพลัสโดยเฉพาะในลำดับชั้นของคลาสในซีพลัสพลัส และสามารถสร้างอินสแตนท์ของอ็อบเจกต์ใหม่สำหรับคลาสที่สัมพันธ์กันได้

4. คลาส TclCommand คลาสนี้จัดเตรียมเพียงแค่กระบวนการสำหรับเอ็นเอสทูในการส่งคำสั่งธรรมดาๆ ไปยังตัวแปลคำสั่ง ซึ่งจะได้สามารถถูกกระทำได้ภายในโกลบอลคอนเท็กซ์โดยตัวแปลคำสั่ง ในไฟล์ misc.cc ที่อยู่ใน ~ns (NS directory) มีฟังก์ชัน 2 อย่างคือ *ns-random* และ

ns-version โดยทั้ง 2 ฟังก์ชันเริ่มทำงานโดยฟังก์ชัน init_misc() ซึ่งถูกกำหนดใน misc.cc และ init_misc จะถูกเรียกโดย *Tcl_AppInit()* ระหว่างการเริ่มต้นทำงาน

5. คลาส EmbeddedTcl เอ็นเอสทูอนุญาตให้สามารถพัฒนาฟังก์ชันต่างๆ ได้ทั้งโดยใช้ซีพลัสพลัส หรือโอทีซีแอล โดยจะถูกประเมินตอนเริ่มต้นการทำงาน ตัวอย่างเช่น สคริปต์ ~tclcl/tcl-object.tcl หรือสคริปต์ทั้งหลายใน ~ns/tcl/lib ซึ่งการโหลดและการประเมินสคริปต์ทั้งหลายทำผ่านอ็อบเจกต์ในคลาส EmbeddedTcl

วิธีการในการต่อขยายเอ็นเอสทูที่ง่าย คือ เพิ่มโค้ด OTcl ไปใน ~tclcl/tcl-object.tcl หรือผ่านสคริปต์ใน ~ns/tcl/lib สำหรับในแบบหลังเอ็นเอสทูจะทำต้นแบบจากไฟล์ ~ns/tcl/lib/ns-lib.tcl โดยอัตโนมัติ ดังนั้นผู้พัฒนาจะต้องเพิ่มเติมบรรทัดเข้าไปในไฟล์นี้เพื่อให้สคริปต์ของผู้พัฒนาถูกทำต้นแบบโดยเอ็นเอสทูอย่างอัตโนมัติในตอนเริ่มการทำงาน ตัวอย่างเช่น ไฟล์ ~ns/tcl/webcache/fifo.tcl กำหนดอินสแตนท์ของกระบวนงานบางอย่างในการทำขั้นตอนการแทนที่ของเว็บแคช ใน ~ns/tcl/lib/ns-lib.tcl ก็จะต้องเพิ่มบรรทัดเข้าไปดังนี้

source tcl/webcache/fifo.tcl

มีข้อควรจำอยู่ 3 ข้อ คือ ข้อแรก ถ้าโค้ดที่เพิ่มเข้าไปมีข้อผิดพลาดระหว่างการประเมิน เอ็นเอสทูจะไม่สามารถทำงานได้ ข้อที่สอง ผู้ใช้สามารถโอเวอร์ไรด์โค้ดในสคริปต์ได้อย่างเปิดเผย และข้อสุดท้ายทุกครั้งที่มีการเพิ่มเติมสคริปต์หรือแก้ไขสคริปต์ ผู้ใช้จะต้องทำการแปลโปรแกรมใหม่อีกครั้งเพื่อให้การเปลี่ยนแปลงนั้นเกิดผล

6. คลาส InstVar คลาสนี้มีหน้าที่กำหนดเมท็อด และกระบวนการในการเชื่อมโดยตัวแปรซีพลัสพลัสในอ็อบเจกต์เงาที่ผ่านการแปลโปรแกรมแล้ว เพื่อที่จะกำหนดตัวแปรอินสแตนท์ของโอทีซีแอลในอ็อบเจกต์ที่ผ่านการแปลคำสั่งแล้วที่เท่าเทียมกัน การเชื่อมโยงนี้ทำเพื่อให้สามารถกำหนดค่าของตัวแปร หรือเข้าถึงได้จากภายในตัวแปลคำสั่ง หรือจากภายในโค้ดที่ผ่านการแปลโปรแกรมแล้วได้ตลอดเวลา

อินสแตนท์ของคลาสตัวแปรนี้มีด้วยกันอยู่ 5 ชนิด คือ คลาส InstVarReal, คลาส InstVarTime, คลาส InstVarBandwidth, คลาส InstVarInt และคลาส InstVarBool สอดคล้องกับการเชื่อมโยงค่าตัวแปรแบบจำนวนจริง เวลา แบนด์วิดท์ จำนวนเต็ม และค่าตัวแปรทางตรรกศาสตร์ ตามลำดับ

อินสแตนท์ของตัวแปรถูกสร้างโดยการกำหนดชื่อของตัวแปรที่ผ่านการแปลคำสั่งแล้ว และที่อยู่ของสมาชิกของตัวแปรในอ็อบเจกต์ที่ผ่านการแปลโปรแกรมแล้ว ตัวสร้างสำหรับคลาสพื้นฐาน InstVar จะสร้างอินสแตนท์ของตัวแปรในตัวแปลคำสั่ง และจะตั้งรูทีนจุดดักเพื่อดักการเข้าถึงตัวแปรทุกครั้งผ่านทางตัวแปลคำสั่ง

เมื่อไรก็ตามที่ตัวแปรถูกอ่านผ่านทางตัวแปลคำสั่ง รูทีนจุดดักจะถูกเรียกก่อนที่จะมีการอ่านเกิดขึ้น และจะไปเรียกฟังก์ชัน *get* ซึ่งส่งค่าปัจจุบันของตัวแปรกลับมา แล้วค่านี้จะถูกกำหนดในตัวแปรที่ผ่านการแปลคำสั่ง และจะถูกอ่านด้วยตัวแปลคำสั่งในที่สุด

ในทำนองเดียวกันเมื่อไรก็ตามที่ตัวแปรถูกกำหนดผ่านทางตัวแปลคำสั่ง รูทีนจุดดักจะถูกเรียกหลังจากที่มีการเขียนเสร็จเรียบร้อยแล้ว โดยรูทีนจะได้รับค่าปัจจุบันที่ถูกกำหนดโดยตัวแปลคำสั่ง และรูทีนจุดดักจะเรียกฟังก์ชัน *set* เพื่อกำหนดค่าของสมาชิกที่ผ่านการแปลโปรแกรมแล้วให้เป็นค่าปัจจุบันที่กำหนดภายในตัวแปลคำสั่ง

3.2 การทำงานของเอ็นเอสทู

เอ็นเอสทูสามารถทำงานได้ใน 2 ลักษณะ คือ

3.2.1 แบบเชิงโต้ตอบ (Interactive Mode)

เมื่อผู้ใช้กะทำข้อความสั่งใดๆ ผลลัพธ์ก็จะแสดงออกมาในทันที ดังตัวอย่าง

```
$ ns
% set ns [new Simulator]
_o1
% $ns at 1 "puts \"Hello World\""
1
% $ns at 1.5 "exit"
2
% $ns run
Hello World
$
```

3.2.2 แบบกลุ่ม (Batch Mode)

ผู้ใช้จะต้องสร้างไฟล์ขึ้นมา โดยในไฟล์นั้นจะมีข้อความสั่งต่างๆ ที่ใช้ในการทำงานตามที่ผู้ใช้ต้องการ ดังตัวอย่าง

simple.tcl

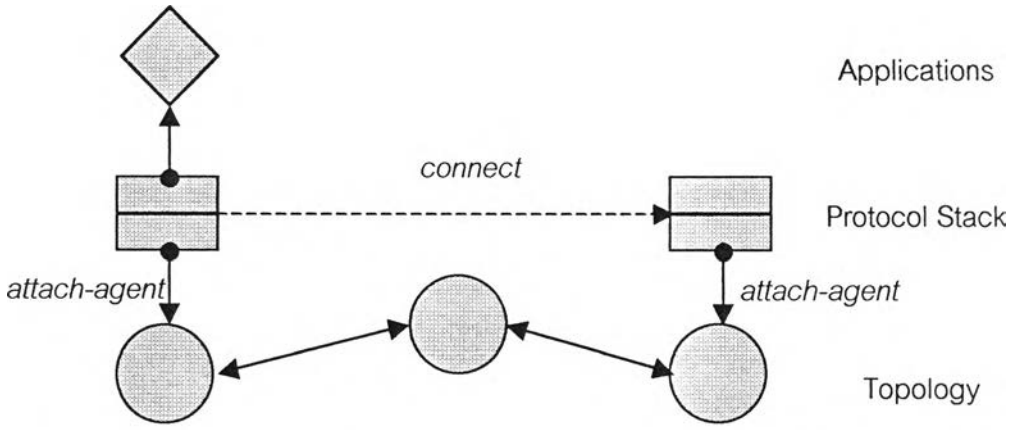
```
set ns [new Simulator]
$ns at 1 "puts \"Hello World\""
$ns at 1.5 "exit"
$ns run
```

$ ns simple.tcl

Hello World

$

โดยส่วนใหญ่การจำลองการทำงานจะทำในแบบที่สอง คือ แบบกลุ่ม เนื่องจากการจำลองการทำงานแต่ละแบบจำเป็นต้องเขียนสคริปต์เพื่อสร้างโครงแบบของการทำงาน การเชื่อมต่อของส่วนประกอบต่างๆ เช่น โหนด ลิงค์ และโพรโทคอล เป็นต้น การกำหนดเวลาตามเหตุการณ์ที่จะจำลองการทำงาน และส่วนอื่นๆ ดังรูปที่ 3.2 เพื่อใช้ในการจำลองการทำงานนั้นๆ ซึ่งจะสะดวกและกระทำได้ง่ายกว่าการทำงานแบบเชิงโต้ตอบ



รูปที่ 3.2 แสดงการเชื่อมต่อส่วนประกอบต่างๆ ของเอ็นเอสทูสำหรับจำลองการทำงาน

สำหรับส่วนประกอบพื้นฐานที่มีอยู่ในเอ็นเอสทู และวิธีการในการเรียกใช้ มีดังนี้

1. การสร้างตัวกำหนดเหตุการณ์ต่างๆ (Creating Event Scheduler)
   - สร้างตัวกำหนดเหตุการณ์
     - *set ns [new Simulator]*
   - กำหนดเหตุการณ์
     - *$ns at <time> <event>*
     - *<event>* เป็นคำสั่งของเอ็นเอสทูหรือทีซีแอลที่ถูกต้อง
   - เริ่มการทำงาน
     - *$ns run*
2. การติดตาม (Tracing)
   - การติดตามกลุ่มข้อมูลบนลิงค์ทั้งหมด
     - *$ns trace-all [open test.out w]*

- การติดตามกลุ่มข้อมูลบนลิงค์ทั้งหมดตามรูปแบบของนามรุ่น 1 (Nam-1)
  - *$ns namtrace-all [open test.nam w]*
- การติดตามกลุ่มข้อมูลบนลิงค์ที่กำหนด
  - *$ns trace-queue $n0 $n1*
  - *$ns namtrace-queue $n0 $n1*

3. การสร้างเครือข่าย (Creating Network)

- โหนด (Node)
  - *set n0 [$ns node]*
  - *set n1 [$ns node]*
- ลิงค์ (Link)
  - *$ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>*
  - *<queue_type>: DropTail, RED, CBQ, FQ, SFQ, DRR*
- แลน (LAN)
  - *$ns make-lan <node_list> <bandwidth> <delay> <ll_type> <ifq_type> <mac_type> <channel_type>*
  - *<ll_type>: LL*
  - *<ifq_type>: Queue/DropTail,*
  - *<mac_type>: MAC/802_3*
  - *<channel_type>: Channel*

4. การใส่ความผิดพลาดต่างๆ (Inserting errors)

- สร้างมอดูลความผิดพลาด
  - *set loss_module [new ErrorModel]*
  - *$loss_module set rate_ 0.01*
  - *$loss_module unit pkt*
  - *$loss_module ranvar [new RandomVariable/Uniform]*
  - *$loss_module drop-target [new Agent/Null]*
- การใส่มอดูลความผิดพลาด
  - *$ns lossmodel $loss_module $n0 $n1*

5. การจัดเตรียมการจัดเส้นทาง (Setup Routing)

- ยูนิคาสท์ (Unicast)
  - *$ns rtproto <type>*

- *<type>: Static, Session, DV, cost, multi-path*

- มัลติคาสท์ (Multicast)
  - *$ns multicast*
  - *$ns mrtproto <type>*
  - *<type>: CtrMcast, DM, ST, BST*

6. การสร้างการเชื่อมต่อ (Creating Connection)

- ยูดีพี (UDP)
  - *set udp [new Agent/UDP]*
  - *set null [new Agent/Null]*
  - *$ns attach-agent $n0 $udp*
  - *$ns attach-agent $n1 $null*
  - *$ns connect $udp $null*
- ทีซีพี (TCP)
  - *set tcp [new Agent/TCP]*
  - *set tcpsink [new Agent/TCPSink]*
  - *$ns attach-agent $n0 $tcp*
  - *$ns attach-agent $n1 $tcpsink*
  - *$ns connect $tcp $tcpsink*

7. การสร้างแทรฟฟิค (Creating Traffic)

1) บนยูดีพี

- ซีบีอาร์ (CBR)
  - *set src [new Application/Traffic/CBR]*
- เอ็กซ์โพเนนเชียล (Exponential) หรือ พาเรโต (Pareto)
  - *set src [new Application/Traffic/Exponential]*
  - *set src [new Application/Traffic/Pareto]*

2) บนทีซีพี

- เอฟทีพี (FTP)
  - *set ftp [new Application/FTP]*
  - *$ftp attach-agent $tcp*
- *เทลเน็ต (Telnet)*
  - *set telnet [new Application/Telnet]*

- *$telnet attach-agent $tcp*

3) **ใช้ข้อมูลจริง** (Trace Driven)

- **ใช้ข้อมูลจริง**
  - *set tfile [new Tracefile]*
  - *$tfile filename <file>* *;# <file> : Binary format*
  - *set src [new Application/Traffic/Trace]*
  - *$src attach-tracefile $tfile*

8. การจำลองการทำงานในระดับงานประยุกต์ (Application-Level Simulation)

- ลักษณะสำคัญของการจำลองการทำงานในระดับงานประยุกต์
  - *ทำงานอยู่บนโพรโทคอลนำส่ง (Transport protocol)*
  - *ใช้ส่งผ่านข้อมูลผู้ใช้ เช่น ส่วนหัวของเอชทีทีพี (HTTP header)*
- วิธีการที่ใช้ส่งผ่านข้อมูล
  - *บนทีซีพี ใช้ Agent/Message*
  - *บนยูดีพี ใช้ Application/TcpApp*
- เว็บแคช (Web Cache)
  - *set cache [new HttpCache $ns $node]*
  - *$cache connect $server*
- เว็บไคลแอนท์ (Web Client)
  - *set client [new Http/Client $ns $node]*
  - *$client connect $server*
  - *$client set-page-generator $pgp*
  - *$client start-session $cache $server*
- เว็บเซิร์ฟเวอร์ (Web Server)
  - *set server [new Http/Server $ns $node]*
  - *$server set-page-generator $pgp*

3.3 การพัฒนาส่วนประกอบสำหรับเอ็นเอสทู

การพัฒนาส่วนประกอบเพิ่มเติมเข้าไปในเอ็นเอสทูสามารถทำได้โดยใช้ 2 ภาษา คือ

3.3.1 โอทีซีแอล

การใช้ภาษาโอทีซีแอลในการเพิ่มเติมส่วนประกอบเข้าไปในเอ็นเอสทูนั้น ทำได้ 2 แบบคือ

1) ไม่ต้องทำการแปลโปรแกรมใหม่อีกครั้งหนึ่ง โดยเพิ่มเติมโค้ดเข้าไปในสคริปต์ที่ใช้จำลองการทำงาน
2) ต้องทำการแปลโปรแกรมใหม่อีกครั้งหนึ่ง เมื่อมีการดัดแปลงโค้ด หรือเมื่อเพิ่มไฟล์ใหม่เข้าไปในเอ็นเอสทู ซึ่งในกรณีหลังจะต้องทำการเปลี่ยนแปลง Makefile (NS_TCL_LIB) และ tcl/lib/ns-lib.tcl ก่อนที่จะแปลโปรแกรม ตัวอย่างเช่น ส่วนประกอบใหม่อยู่ในไฟล์ tcl/webcache/fifo.tcl ใน Makefile จะต้องเพิ่มข้อความเข้าไป 1 บรรทัดในส่วนของ NS_TCL_LIB ดังนี้

```
NS_TCL_LIB = \
        tcl/webcache/fifo.tcl \
        ...
```

และใน tcl/lib/ns-lib.tcl ต้องเพิ่มข้อความเข้าไป 1 บรรทัด ดังนี้

```
Class Simulator
...
source ../webcache/fifo.tcl
```

3.3.2 ซีพลัสพลัส

สำหรับการใช้ภาษาซีพลัสพลัสในการเพิ่มเติมส่วนประกอบเข้าไปในเอ็นเอสทู ทำได้แบบเดียวคือ ต้องทำการแปลโปรแกรมใหม่อีกครั้งหนึ่ง แต่มี 2 วิธีการ ดังนี้

1) ทำการดัดแปลงโค้ดเดิมที่มีอยู่ ซึ่งจะต้องกระทำข้อความสั่ง make depend ก่อนที่จะทำการแปลโปรแกรมอีกครั้งหนึ่ง
2) ทำการเพิ่มไฟล์ใหม่เข้าไปในเอ็นเอสทู ซึ่งในกรณีนี้จะต้องมีการเปลี่ยนแปลง Makefile และกระทำข้อความสั่ง make depend ก่อนที่จะทำการแปลโปรแกรมอีกครั้งหนึ่ง

โดยผู้ใช้งานสามารถพัฒนาส่วนประกอบจากภาษาใดภาษาหนึ่งอย่างเดียว หรือใช้ทั้งสองภาษาก็ได้ โดยอาศัย TclCL เป็นตัวประสานระหว่างซีพลัสพลัส และโอทีซีแอล ดังรายละเอียดในหัวข้อที่ 3.1