Socket Programming Syntax

- int socket(int domain, int type, int protocol);
- domain = AF_INET, AF_INET6
- type = SOCK_STREAM, SOCK_DGRAM
- protocol = 0 (preferred), IPPROTO_TCP, IPPROTO_UDP, IPPROTO_ICMP
- int bind(int socket, struct sockaddr *name, int namelen)
- struct sockaddr_in {
  short sin_family; // e.g. AF_INET, AF_INET6
  unsigned short sin_port; // e.g. htons(3490)
  struct in_addr sin_addr; // see struct in_addr, below
  char sin_zero[8]; // zero this
  };
- struct in_addr {
  unsigned long s_addr; // load with inet_addr()
  };
- struct sockaddr {
  unsigned short sa_family; // address family, AF_xxx
  char sa_data[14]; // 14 bytes of protocol address
  };
- int listen(int socket, int backlog)
- int accept(int socket, struct sockaddr *addr, int *addrlen)
- int connect(int socket, struct sockaddr *addr, int addrlen)
- int send(int socket, const void *buf, int buflen, int flags);
- int recv(int socket, void *buf, int buflen, int flags);
- int sendto(int socket, const void *buf, int buflen, int flags, struct sockaddr* to, int tolen);
- int recvfrom(int socket, void *buf, int buflen, int flags, struct sockaddr* from, int *fromlen);
- int close(int socket)

Multithreading

- int pthread_create(pthread_t *thread, pthread_attr_t *attr, void *(*start_routine)(void *), void *arg);
- void pthread_exit(void *value_ptr);
- int pthread_join(pthread_t thread, void **value_ptr);

Filing Syntax

- ptr = fopen("filepath","mode")
- int fgetc(FILE * fp) //read single char from file
- fclose(fptr)
- type arrayName [arraySize]; // To declare an array
- int strcmp(const char *s1, const char *s2); // string compare

File Headers

- #include <stdio.h>
- #include <string.h>
- #include <sys/socket.h> //socket
- #include <arpa/inet.h> //inet_addr
- #include<pthread.h>
- #include<bool.h>

For RIP Classless Configuration

Router>enable

Router#configure terminal // Global configuration mode

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#router rip

Router(config-router)#version 2

Router(config-router)#no auto-summary

Router(config-router)#network 172.19.0.0

Router(config-router)#network 172.19.32.0

Router(config-router)#network 172.19.160.0

Router(config-router)#exit

Router(config)#exit

NS2 Syntax:

Create Simulation: set ns [new Simulator]

Trace Files for NAM: set nf [open out.nam w]

$ns namtrace-all $nf

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

Routing Algorithm: $ns rtproto <protocol_name>; <protocol_name>: DV

Node creation: set <node_name> [$ns node]

Links Creation: $ns <link_type> <node1> <node2> <Bandwidth> <Delay> <queue_type>

<link_type>: simplex-link, duplex-link; <queue_type>: DropTail, SFQ

Graphical Settings (NAM): $ns <type> <node1> <node2> <option> <args>

<type> : simplex-link-op, duplex-link-op; <option> : orient, queuePos

Transport Layer: set <layer_name> [new Agent/<agent_type>]

<agent_type>: UDP,TCP,Null,TCPSink

Attaching Transport layer: $ns attach-agent <node_name> <layer_name>

Connecting Transport layer: $ns connect <layer_name> <layer_name>

File Transfer Protocol: set <ftp_name> [new Application/FTP]

FTP Attach Agent: <ftp_name> attach-agent <layer_name>

Constant Bit Rate: set <cbr_name> [new Application/Traffic/CBR]

CBR Attach Agent: <cbr_name> attach-agent <layer_name>

CBR Parameters: <cbr_name> set <parameter> <parameter_value>

<parameter>: packetSize_, interval_, rate_

Event Scheduling: $ns at <time_frame_value> "<cbr_name>/<ftp_name> <time_event>"

<time_event>: start, stop

Ending Simulation: $ns at <time_frame_value> "finish"

Run Simulation: $ns run

Link Up/Down: $ns rtmodel-at <time_frame_value> <function> <node1> <node2>

<function>: up,down