Trabajo Final Integrador

Sistemas Operativos y Redes de Computadoras
Departamento de Ingeniería de Computadoras
Facultad de Informática - Universidad Nacional del Comahue

Integración de TCP en Xinu: Segunda parte

El presente trabajo es una propuesta de trabajo final integrador para acreditar las materias Sistemas Operativos I y Redes de Computadoras I y se propone a tres estudiantes que trabajarán en grupo.

Consideraciones

La forma de trabajo propuesta incluye reuniones para acordar el enunciado, lectura del material y delineación de las tareas asignadas a cada persona por parte del grupo, planificación de reuniones de avance, entregas parciales y entrega final. Se ha acordado con les estudiantes que la culminación de este trabajo debería ser durante el primer cuatrimestre del año 2025. El material bibliográfico principal propuesto son los libros utilizados en las cursadas de las materias (SO y RDC). Se entrega el código de Xinu a utilizar (que tiene el protocolo TCP integrado) y una plantilla de latex para el informe.

Xinu

Xinu es un sistema operativo pequeño y elegante que admite la creación dinámica de procesos, asignación dinámica de memoria, comunicación de red, sistemas de archivos locales y remotos, un shell y funciones de E/S independientes del dispositivo.

Xinu actualmente cuenta con la implementación de tres protocolos de red: arp, ip, udp. A esta lista se agrega el protocolo TCP, integrado por el grupo de trabajo anterior a la versión xinu-x86-gui, a partir del código provisto por el profesor Douglas Comer (que era el trabajo doctoral de un estudiante). También cuenta con un driver para la placa de red Intel 82545EM Ethernet NIC. Utilizando qemu o virt-manager, es posible conectar un sistema XINU a una red local.

La implementación actual de TCP presenta un error, el cual es reproducible de la siguiente manera: *Luego de tres conexiones TCP, el código falla.*

Enunciados

Se proponen diferentes enunciados, todos relacionados con la implementación mencionada anteriormente de TCP en *xinu*:

1. Lograr reproducir el error mencionado anteriormente e intentar repararlo. El establecimiento de la conexión TCP es mediante el conocido protocolo de tres vías. Las conexiones

podrían ser desde un cliente Xinu que se conecta a un servidor GNU/Linux que está escuchando.

2. Identificar si la implementación de TCP provee control de congestión y control de flujo, proponer una forma para verificar su funcionamiento y describir la forma en que lo hace.

3. Migrar el código fuente de un cliente TELNET open source a XINU. Verificar su funcionamiento contra un servidor TELNET en una máquina GNU/LINUX o MAC.

Entregas

Se deberá entregar el código (nuevo o modificado) vinculado con la realización de los enunciados, y un informe (usando la plantilla de latex provista).

Detalle del código:

- El código usado para verificar la existencia o no del error.
- El código usado para verificar el funcionamiento del control de flujo y de congestión.
- El código del cliente Telnet funcionando en *Xinu*.

Detalle del informe:

- Introducción: Objetivos y metodología.
- Marco teórico: El marco teórico debería ir desde lo general (mencionado por arriba) hasta lo particular (agregando mas detalle a medida que profundizan).
  - Conceptos teóricos de Sistemas Operativos: describir desde lo teórico los conceptos utilizados y mencionados a lo largo del trabajo.
  - Conceptos teóricos de Redes de Computadoras: Protocolos, capas, capa de transporte. TCP como protocolo de capa de trasporte. Funciones de TCP, etc.
  - Preguntas guía: Cómo se vinculan estos conceptos?, cómo se relaciona TCP y el sistema operativo?, y un programa como Telnet y un sistema operativo?
- Desarrollo de los enunciados.
  - Descripción de las tareas realizadas.
  - Limitaciones y problemas encontrados.
  - Soluciones propuestas cuando corresponda.
- Conclusiones y trabajos futuros.

Anexo

- Cliente TELNTE: https://gist.github.com/legnaleurc/7638738
- Extraído del trabajo del grupo anterior: La mayoría del protocolo puede encontrarse en el directorio /net/tcp. Este incluye 39 archivos de código C que implementan la gran parte de TCP. En estos se trabaja con la parte conceptual de la maquina de estados del protocolo TCP vista en la sección 2.2. A continuación, una breve descripción de los archivos:

  1. mq.c implementa colas de mensajes. Permite que los procesos se comuniquen entre sí enviando y recibiendo mensajes de manera sincronizada.
  2. tcbclear.c define una función que toma un puntero a un TCB como argumento y pone en cero todos los campos dentro de esa estructura.
  3. tcbref.c maneja el conteo de referencias para manejar la memoria de las estructuras TCB. Si un TCB llega a 0, se libera su memoria.
  4. tcp hton.c convierte los campos TCP del orden de bytes del host al orden de bytes de la red.
  5. tcp in.c maneja los segmentos TCP recibidos por el sistema. Su trabajo es encontrar el segmento TCP que debe ser procesado posteriormente por la la función tcpdisp que depende del estado del TCB.
  6. tcp init.c inicializa la tabla TCB con semáforos y colas de mensajes para cada entrada. También inicializa una estructura de control y crea un proceso separado.(tcp out) para administrar la comunicación saliente de TCP.
  7. tcp out.c lee continuamente comandos de una cola de mensajes y los ejecuta en función del TCB asociado. Se encarga de enviar información, reconocimientos para confirmar la recepción de datos, manejar el tiempo que espera antes de volver a intentar enviar datos si hay un problema, y también maneja eventos que indican que la conexión se cerró.
  8. tcp recv.c define una función llamada tcp recv utilizada para leer datos de una conexión TCP en el sistema operativo Xinu. 9. tcp register.c define dos funciones principales encargadas de abrir conexiones TCP en el sistema operativo Xinu:
     - checktuple: verifica si una combinación específica de direcciones IP, puertos locales y remotos ya está en uso para una conexión TCP.
     - tcp register: llama a un TCP master device y obtiene el identificador de un TCP slave device para la conexión.
  9. tcp send.c se define función para enviar datos a través de una conexión TCP establecida colocándolos en el búfer de envío del TCB.
  10. tcpabort.c se encarga de cerrar un TCB de TCP. Cambia el estado del TCB a cerrado, libera cualquier proceso que esté esperando leer o escribir datos en la conexión, y opcionalmente decrementa la cuenta de referencias del TCB si la conexión estaba en progreso.
  11. tcpack.c se define función para enviar o programar el envío de un paquete de reconocimiento TCP (ACK) en una conexión TCP establecida.
  12. tcpalloc.c se define función para asignar memoria y configurar un paquete de red para transportar datos en una conexión TCP establecida.

13. tcpchecksum.c define dos funciones relacionadas con el cálculo de sumas de comprobación.
    - localcksum: se utiliza para sumar bloques de memoria generales.
    - tcpcksum: calcula la suma de comprobación del encabezado TCP de un paquete de red teniendo en cuenta información adicional para garantizar la integridad de los datos durante la transmisión.
14. tcpclose.c se encarga de cerrar una conexión TCP existente. Realiza diferentes acciones dependiendo del estado de la conexión (escuchando, estableciéndose o establecida), liberando recursos y notificando a los procesos en espera.
15. tcpclosing.c verifica si se ha enviado el último byte de una conexión TCP en proceso de cierre (indicado por el FIN) y programa la expiración del TCB en un tiempo determinado para liberar los recursos asociados a la conexión.
16. tcpcwait.c define que no se debe hacer nada cuando el TCB se encuentra en estado WAIT.
17. tcpdata.c se encarga de procesar un segmento TCP entrante que contiene datos para una conexión TCP existente.
18. tcpdisp.c define qué hacer cuando se recibe un paquete TCP en función del estado actual de la conexión.
19. tcpestd.c se encarga de gestionar un segmento TCP entrante para una conexión TCP que tiene el estado ESTABLISHED (establecida) y el cierre de la conexión.
20. tcpfin1.c se encarga de procesar un segmento TCP entrante para una conexión TCP que se encuentra en estado FIN WAIT 1. Gestiona la recepción del FIN del remitente, el envío de ACKs y el cambio de estado en el proceso de cierre de una conexión.
21. tcpfin2.c se encarga de procesar un segmento TCP entrante para una conexión TCP que se encuentra en estado FIN WAIT 2. Espera la confirmación (ACK) del FIN enviado por el receptor y, una vez recibido, marca la conexión para que expire en un tiempo determinado.
22. tcplastack.c confirma el ACK final para el FIN enviado por el receptor, elimina el temporizador de retransmisión asociado, marca la conexión como cerrada y reduce la referencia del TCB para su eventual liberación de memoria.
23. tcplisten.c maneja los segmentos SYN entrantes para iniciar nuevas conexiones TCP. Busca un dispositivo libre en la tabla de TCBs, inicializa un nuevo TCB para la conexión, notifica a cualquier proceso en espera y envía una respuesta SYN-ACK para establecer la comunicación.
24. tcpminit.c.orig es una copia del archivo original tcpminit.c
25. tcpmopen.c.orig es una copia del archivo original tcpmopen.c
26. tcpnextseg.c calcula el desplazamiento del siguiente segmento que se debe enviar en una conexión TCP.
27. tcpparse.c se encarga de analizar una cadena de texto con formato específico y extraer la dirección IP, el puerto y un indicador de estado activo/pasivo para una conexión TCP.
28. tcpreset.c se encarga de enviar un segmento TCP con la bandera RST (reinicio) porque ocurrió un evento que requiere cerrar una conexión de manera repentina.

29. tcprto.c calcula el tiempo de espera de retransmisión (RTO) para una conexión TCP específica.
30. tcpsendseg.c crea y envía un segmento TCP por una conexión específica. También administra los timers de RTO y ACK, y actualiza las variables de RTT de la conexión si se envían datos en el segmento.
31. tcpsynrcv.c procesa un segmento TCP entrante en el estado SYN-RECEIVED verificando el ACK y el sequence number esperado. Si la verificación es correcta, la función cambia el estado a ESTABLISHED, procesa cualquier dato recibido y permite que los procesos en espera lean datos de la conexión.
32. tcpsynsent.c maneja un segmento TCP entrante cuando la conexión se encuentra en estado SYN-SENT. Este estado se refiere que el emisor envió un segmento SYN al receptor y esta esperando respuesta.
33. tcptimer.c gestiona eventos TCP que tienen timer.
34. tcpwait.c gestiona los segmentos TCP entrantes que llegan cuando la conexión tiene estado TIME-WAIT.
35. tcpupdate.c actualiza la ventana de recepción, el RTT y el temporizador RTO en función del ACK que se recibió.
36. tcpwake.c despierta procesos en espera (lectores o escritores) de una conexión TCP que pueden continuar con su ejecución y retorna la cantidad que despertó.
37. tcpxmit.c transmite segmentos TCP si es necesario. Verifica las condiciones de envió en base al estado de la conexión.
38. timer.c implementa un sistema de temporizadores para el envió de mensajes.

Además, en el directorio /include, se encuentran los archivos header que definen las estructuras básicas usadas durante TCP, tales como tcb.h, net.h y mq.h.