Resumen

Lautaro Bachmann

Contents

Contents

Introducción a las Redes de Computadoras – Parte 1

¿Qué tipos de máquinas queremos poder interconectar por medio de redes?

Hosts o sistemas finales:

dispositivos de cómputo

Incluye: distintos tipos de computadoras

y dispotitivos IoT

"Fun" Internet-connected devices

Web-enabled toaster + weather forecaster

IP picture frame

Tweet-a-watt: monitor energy use

Internet refrigerator

Pet tracking Smart lighting

Dispositivos IoT

Dispositivos IoT pueden:

Intercambiar datos con otros dispositivos y aplicaciones interconectados.

Recolectar datos de otros dispositivos y procesar los datos localmente o enviarlos a servidores centralizados para procesar los datos.

Realizar algunas tareas localmente y otras tareas dentro de la infraestructura de la red

Redes de Computadoras

¿Qué es una red de computadoras?

Una red de computadoras es un conjunto de sistemas finales interconectados.

¿Qué significa que dos computadoras están interconectadas?

Dos computadoras

están interconectadas si pueden intercambiar información.

¿De qué manera puede hacerse la interconexión?

La conexión puede hacerse por **medios de transmisión:**

cable de cobre, fibra óptica, microondas, etc.

El intercambio de información entre hosts se hace por medio de **señales** que viajan en los medios de transmisión.

¿Qué servicios o usos proporcionan las redes de computadoras?

Compartir recursos:

medio de comunicación entre personas:

Socializar:

Trabajo colaborativo

Comercio electrónico Entretenimiento:

¿Qué hacer para que los hosts de varias redes de distinto tipo se puedan comunicar entre sí?

Varias redes de computadoras pueden ser **interconectadas**

entre sí para formar redes más grandes.

La **internet** es el ejemplo de red de redes más grande.

Sistemas Operativos de Red

Para poder aprovechar y gestionar los distintos tipos de redes se definen **sistemas operativos de red.**

Aplicaciones de Red

Las redes de computadora se usan para proveer distintos **servicios:**

Para proveer servicios se crean

aplicaciones de red. Para programarlas se usan

APIs y middlewares.

Y estos últimos se basan en el sistema operativo de red.

Interredes

¿Cómo comunicar personas pertenecientes a redes diferentes?

Solución:
usar interredes

Interred
conjunto de redes interconectadas

puertas de enlace:
conectan redes de distintas tecnologías.

Internet
es una interred.

La Internet

La **internet**

está formada por billones de dispositivos de computación conectados entre sí.

En la internet se ejecutan **aplicaciones de red.**

La internet es una red de redes que interconecta varias redes entre sí.

Para envío y recepción de mensajes entre sistemas finales se usan **protocolos.**

Estructura de la Internet

Hosts acceden a la internet a través de

proveedores de servicios de internet de acceso

¿Qué tipos de ISP de acceso existen?
ISP residenciales (p.ej. compañías de cable, telefónicas,

ISP empresarial (da acceso a sus empleados).

ISPs universitaria (da acceso a docentes, estudiantes y personal).

Celulares.

ISPs que proveen acceso a WiFI en aeropuertos, hoteles, restaurantes,

ISPs de capa superior

redes proveedoras de contenido

¿Qué redes tenemos en cada nivel de la jerarquía?

Internet de las Cosas (IoT)

¿Qué es el IoT?

paradigmas de redes anteriores

Redes de área amplia (WANs)

red de área amplia (WAN)

cómo está organizada una WAN?

¿Cómo se hace para enviar mensajes en una WAN?

Encolado y pérdida de paquetes

Algoritmos de enrutamiento

¿Cuánto demora el almacenamiento y reenvío?

Sistema telefónico fijo (p.ej. DSL):

Arquitectura de red celular

Sistema de fibra a la casa:

Redes de Área Metropolitana (MAN)

tipos:

MAN basada en TV por cable

Access net: cable network

Redes de Área Local

¿Dónde puede usarse una LAN?

¿Qué tipos de hosts se comunican a una LAN?

tipos de LAN:

Difusión:

¿A quién puede estar destinado un mensaje cuando se usa difusión?¿Qué

Red hogareña

Internet

En la internet se ejecutan **aplicaciones de red.**

La internet es una red de redes que interconecta varias redes entre sí.

Para envío y recepción de mensajes entre sistemas finales se usan **protocolos.**

Estructura de la Internet

Hosts acceden a la internet a través de

proveedores de servicios de internet de acceso (ISPs de acceso).

¿Qué tipos de ISP de acceso existen?

ISP residenciales compañías de cable, telefónicas, fibra a la casa (FTTH), etc.).

ISP empresarial (da acceso a sus empleados).

ISPs universitaria (da acceso a docentes, estudiantes y personal).

Celulares.

ISPs que proveen acceso a WiFI (p.ej. en aeropuertos, hoteles, restaurantes, etc.

Las ISP de acceso son interconectadas a través de redes ISP nacionales e internacionales de más alto nivel llamados

ISPs de capa superior

o globales de tránsito. son ISP que proveen **servicios de tránsito.** Una ISP de capa superior consiste de **enrutadores de alta velocidad**

interconectados con **enlaces de fibra óptica** de alta velocidad.

Las ISP globales de tránsito deben estar interconectadas entre sí.

Cada red ISP, ya sea de acceso o de capa superior, es manejada independientemente.

redes proveedoras de contenido

(por ejemplo, Google, Facebook, Microsoft, Apple, etc.).

¿Por qué se usan

Para reducir pagos a redes de tránsito global.

Para tener control sobre cómo sus servicios son entregados a los usuarios finales.

¿A qué redes se conectan

A ISP regionales e ISP de acceso.

¿Qué redes tenemos en cada nivel de la jerarquía?

"tier-1" ISPs comerciales

(p.ej. redes globales de tránsito) cobertura nacional e internacional.

Redes proveedoras de contenido En el medio **ISP regionales. Finalmente ISPs de acceso**

Internet de las Cosas (IoT)

¿Qué es el IoT?

es extender Internet desde "computadoras" a "objetos", sin necesidad de un "humano" en el medio.

IOT nace de paradigmas de redes anteriores

Machine-to-Machine (M2M):
redes para conectar máquinas entre sí.

Radio-Frequency ID (RFID):
para chips embebidos en productos que hacen saltar alarmas en locales.

Wireless Sensor Networks (WSN):
sensores distribuidos conectados a una red.

Mobile Ad-Hoc Networks (MANET):
redes de autos que se comunican entre ellos.

Domótica (Smart home):

dispositivos hogareños conectados en red Ciudades, rural (Smart cities)

Vehículos
(Vehicle to everything)

Industria (Industria 4.0):

se conectan dispositivos en sistema productivo, en una fábrica.

Cyber-physical systems (CPS)

Redes de área amplia (WANs)

red de área amplia (WAN)

cubre un área geográfica grande, típicamente un país o hasta un continente. Una red de área amplia va a permitir interconectar varias redes hogareñas e institucionales

¿Cómo se hace para enviar mensajes en una WAN?

Algoritmo de almacenamiento y reenvío. Un paquete sigue una ruta de enrutadores.

El paquete se almacena enteramente en cada enrutador de la ruta.

El paquete almacenado en un enrutador espera allí hasta que la línea requerida de salida esté libre y luego se reenvía al siguiente enrutador.

Encolado y pérdida de paquetes

Si la tasa de llegada al enlace (en bits) excede la tasa de transmisión del enlace por un período de tiempo.

¿Qué va a suceder?

Los paquetes se van a encolar, y esperarán a ser transmitidos en el enlace.

Los paquetes pueden ser descartados (perdidos) si la memoria (el búfer) se llena.

¿Cuánto demora el almacenamiento y reenvío?

dnodal = dproc + dqueue + dtrans + dprop

dproc: procesamiento del nodo

dqueue: demora por encolado

Chequeo de errores

Tiempo de espera en el enlace de salida para transmisión.

Determinar la línea de salida

typically $<$ msec

Depende de cuán congestionado está el enrutador

Sistema telefónico fijo (p.ej. DSL):

Cada domicilio está conectado por un cable de cobre a una **End office**

Toda oficina central está conectada a una **Toll office.**

Toll offices son usadas para reenvío de mensajes.

Toll offices unidas por cables (de fibra óptica).

Redes de Área Metropolitana (MAN)

Una **red de área metropolitana (MAN)**

cubre una ciudad.

tipos:

Redes de cable:

se basan en la red de TV por cable.

Redes móbiles:

son redes inalámbricas de alta velocidad.

Redes de Área Local

Una **red de área local**

(LAN) es una red operada privadamente dentro de un edificio o casa.

¿Dónde puede usarse una LAN?

Una LAN puede usarse en un hogar o en una organización (pública o privada).

Las LAN usadas por compañías se llaman

redes empresariales.

¿Qué tipos de hosts se comunican a una LAN?

Las LAN se usan para comunicar PCs, notebooks, celulares, impresoras, electrónicos del hogar, etc.

La idea es que los hosts puedan compartir recursos e intercambiar información.

tipos de LAN:

LAN inalámbricas:
en su forma más simple las máquinas se comunican entre sí por medio de una estación base (access point).

La Ethernet:
las máquinas se conectan por medio de cables a un commutador (switch).

Difusión:

Si una máquina envía un mensaje, todas las demás lo reciben.

¿A quién puede estar destinado un mensaje cuando se usa difusión?¿Qué

Estar destinado a una única máquina Ser enviado a todas las máquinas (broadcasting)

Ser enviado a un grupo de máquinas en particular (multicasting)

Colisión:

más de una máquina manda simultáneamente un mensaje. Los mensajes colisionan y se dañan.

¿Qué hay que hacer en relación a las colisiones?

Evitar o minimizar colisiones. Detectar las colisiones Tratar las colisiones

Redes de acceso empresarial

Internet

Red dorsales (backbone) están conectadas a varias WAN

Redes metropolitanas pueden conectarse a WANs

LANs están conectadas a WANs o a redes metropolitanas

Protocolos

Protocolos de comunicación definen:

formato,

orden de **mensajes enviados y recibidos** entre máquinas de la red, y **acciones tomadas** en la transmisión y recepción de mensajes # Sistemas Operativos de Redes

Los sistemas operativos de redes (SOR) están organizadas como una pila de capas o niveles, cada una construida arriba de la que está debajo de ella.

La cantidad de capas, los nombres de las capas, sus contenidos y su función, difieren de un tipo de red a otro.

Jerarquías de Protocolos

¿Cuál es el propósito de una capa en arquitecturas multicapa?

ofrecer ciertos servicios a las capas superiores ocultar la implementación a las capas superiores

operaciones y servicios primitivos ofrecidos por una capa a capa superior.

¿Cómo afecta esto al propósito de las capas?

Una capa n se piensa como una conversación entre la capa n de una máquina con la capa n de otra máquina,

sin tener que preocuparnos de ciertos problemas que resuelven las capas inferiores a la capa n.

¿Cuándo ocurren comunicaciones entre capas consecutivas?

Durante el envío de mensaje: cada capa pasa los datos y la información de control a la capa inmediatamente inferior, hasta que se alcanza la capa más baja.

Durante la recepción de mensaje: cada capa pasa cierta información conteniendo los datos a la capa inmediatamente superior hasta que alcanza la capa más alta.

Procesos de aplicación (capa 5 o capa de aplicación)

Produce un mensaje y lo pasa a la capa 4 para su transmisión.

La capa 4 (capa de transporte)

pone un encabezado en el mensaje para identificarlo y pasa el resultado a la capa 3.

El encabezado contiene números de secuencia para que la capa 4 en la máquina de destino entregue los mensajes en el orden correcto.

Capa 3 (capa de red):

Hay limitaciones en el tamaño de los mensajes de capa 3.

Divide en paquetes los mensajes que llegan.

A cada paquete se le coloca un encabezado.

Decide cuál de las líneas que salen usar

Pasa los paquetes a la capa 2.

La capa 2 (capa de enlace de datos)

agrega un encabezado y un terminador, a cada pieza

pasa la unidad resultante a la capa 1 para su transmisión.

Aspectos de Diseño de las Capas

Problema: Hace falta un mecanismo para identificar a las máquinas de una red.

Solución: Se usan direcciones para las máquinas.

Situación indeseable: mensajes que llegan al receptor se pierden. – ¿Por qué se imaginan que puede pasar esto?

Causa: un emisor rápido satura de datos al receptor hasta que este ya no puede almacenar más datos que le llegan y comienza a perder datos.

Problema: ¿Cómo evitar que un emisor rápido sature de datos a un receptor lento?

Idea de Solución: Uso de retroalimentación al emisor. • O sea, indicarle cuándo y cuánto puede enviar.

Fragmentación de mensajes

Es común que las capas imponen un tamaño máximo a los mensajes.

Situación indeseable: mensajes que llegan no pueden ser aceptados en una capa.

Causa: los procesos son incapaces de aceptar mensajes que superan una cierta longitud • por ejemplo en la capa de enlace de datos y en la capa de red • Por ejemplo en una inter-red (redes aceptan tamaños max)

Fragmentación de mensajes

Problema: ¿Cómo tratar un mensaje demasiado largo?
Idea de solución: fragmentar mensajes, transmitir fragmentos y re-ensamblar mensajes.

Congestión

Debido a las limitaciones de los enrutadores y las líneas de salida una red tiene una determinada capacidad de conducción de mensajes..

Situación indeseable: los mensajes enviados de host de origen a destino se pierden antes de llegar o demoran demasiado en llegar

congestión

La red no puede manejar la carga de paquetes que recibe de manera aceptable (esperas inaceptables o pérdida de paquetes)

Problema: ¿Cómo controlar la congestión?
Idea de solución: que máquinas emisoras se enteren de la congestión y reduzcan el tráfico de salida.

Capa de aplicación

En la capa de aplicación tenemos las aplicaciones de red.

Cada aplicación de red ofrece un servicio específico con su propia forma de interfaz con el usuario.

Hay dos opciones para desarrollar aplicaciones de red:

El programador para especificar la comunicación usa

una (API). Una API es conjunto básico de funciones a ser usadas.

La socket API es el estándar de facto para el software que se comunica sobre la internet.

El programador se apoya en middlewares para construir la aplicación de red.

Una

middleware provee servicios al software de la aplicación que • hacen más fácil a los desarrolladores implementar la comunicación y la entrada/salida de modo que • se pueden enfocar en el propósito específico de la aplicación.

La capa de aplicación: TCP/IP

La capa de aplicación en TCP/IP contiene varios protocolos de nivel mas alto: transferencia de archivos (FTP), correo electrónico (SMTP), para resolución de nombres de host en sus direcciones de red (DNS), para páginas web (HTTP), etc.

Capa de transporte

La capa de red provee comunicación entre hosts

los paquetes de la comunicación entre los hosts siguen rutas elegidas por la capa de red.

La capa de transporte (CT) provee comunicación entre procesos.

La CT mejora los servicios de la CR como veremos. La CT se ejecuta por completo en los hosts.

Entidad de transporte (ET) = software/hardware de la CT.

¿Qué cosas se debería solucionar la CT?

– Uso de temporizadores y las retransmisiones de paquetes. – Uso de búferes y control de flujo. – Evitar congestionar la red poniendo demasiados paquetes en ella.

• Cuando la CR pierde paquetes, la CT puede solucionarlo.

La capa de transporte: TCP/IP

Capa de transporte. Tiene dos protocolos: TCP

• TCP divide el flujo de bytes entrantes en mensajes discretos y pasa cada uno de ellos a la capa de interred. • TCP proporciona entrega confiable y en orden de los mensajes.

• Permite que un flujo de bytes que se origina en una máquina se entregue sin errores en otra máquina en la interred. • Reensamblaje de los mensajes recibidos en el receptor. • Mensajes recibidos son confirmados. • TCP también maneja el control de flujo y el control de congestión.

UDP

UDP proporciona entrega de mensajes no confiable y desordenada.

Un mensaje puede entregarse con errores, o no entregarse, o varios mensajes pueden entregarse en forma desordenada.

¿Entonces qué cosas de TCP puedo sacar de UDP?

Mensajes recibidos no son confirmados. • Control de flujo, control de congestión, retransmisiones cuando se recibe mensaje erróneo.

¿Entonces para que tipo de aplicaciones se puede usar UDP?

Se

usa para aplicaciones que no usan el control de flujo ni la secuenciación de mensajes. – Uso en consultas de solicitud-respuesta y en aplicaciones de transmisión de voz y video.

Capa de Red

Objetivos de la (capa 3).

– Algoritmos de almacenamiento y reenvío

– Control de congestión.

– Resolver problemas que surgen cuando un mensaje tiene que viajar por redes de distinta tecnología para llegar a destino.

Enrutamiento

Situación indeseable: un mensaje demora demasiado en llegar

Causa: en determinadas redes (p.ej. WAN, internet, etc.) hay múltiples rutas entre el origen y el destino – Y justo se toma una ruta demasiado lenta/larga entre origen y destino

Problema: ¿Cuando hay múltiples rutas entre el origen y el destino cómo elegir la mejor o las mejores?

De esto se encargan los algoritmos de enrutamiento

Capa de interred:

permite que los hosts inyecten paquetes dentro de cualquier red,

¿Cómo viajan paquetes diferentes entre dos hosts? Estos viajan a su destino de manera independiente. **¿Qué consecuencias tiene esto?**

Paquetes pueden llegar en un orden distinto al cual fueron enviados.

¿Qué se hace con los paquetes que llegaron fuera de orden?

las capas mas altas deberán ordenarlos, si se desea una entrega ordenada.

La capa de red: TCP/IP

¿Cómo se distingue entre diferentes máquinas (que tienen una conexión a internet)?

Direcciones IP

¿Cómo son los paquetes que se envían? Paquetes IP.

¿Cómo se hace el enrutamiento?

Hay protocolos de enrutamiento: se usan OSPF y BGP para enrutamiento de paquetes.

Procesos en comunicación

procesos: programas ejecutándose dentro de un host

Proceso cliente: proceso que inicia la communicación

Proceso servidor: proceso que espera ser contactado

Dentro del mismo host, dos procesos se comunican usando comunicación inter-procesos

Los procesos en diferentes hosts se comunican intercambiando mensajes

Direccionando Procesos

¿Cómo se identifican los procesos?

Para recibir mensajes, los procesos deben tener un identificador

Para recibir mensajes, los procesos deben tener un identificador Un host tiene una dirección IP única de 32-bits

Identificadores de proceso incluyen tanto direcciones IP y número de puerto.

¿Es la dirección IP del host en el cual ejecuta un proceso suficiente para identificar el proceso?

no, porque muchos procesos pueden estar ejecutándose en el mismo host.

Capa de Enlace de Datos

Objetivo de la (capa 2)

transformar un medio de transmisión puro en una línea de comunicación que aparezca libre de errores de transmisión.

¿Qué problemas de diseño hay que considerar?

Fragmentación de paquetes en tramas, cuando un paquete es demasiado grande

– Tramas

Tramas de confirmación de recepción son usadas cuando el servicio es confiable.

– Control de flujo. • Para evitar que un emisor rápido sature a un receptor lento.

– Control de acceso a un canal compartido: se busca manejar y minimizar o evitar colisiones.

– Control de errores

- **Situación indeseable: mensajes llegan con errores**

- Causa: medio físico de comunicaciones es imperfecto y ocasiona errores

Capa Física

¿Cuál es el propósito de la capa física (CF)?

Transportar un stream de datos de una máquina otra usando medios físicos.

¿La CF consiste solo de medios físicos?

– No. Los medios físicos se conectan entre sí usando dispositivos como codecs, modems, multiplexores, demultiplexores, etc.

Capa Física: medios físicos

• • • • bit: se propaga entre pares de transmisor/receptor Enlace físico: lo que yace entre el transmisor & receptor Medios guiados: –Las señales se propagan en medios sólidos: copper, fiber, coax Medios no guiados: –Las señales se propagan libremente, e.g., radio Par trenzado (TP) 2 cables de cobre aislados • Category 5: 100 Mbps, 1 Gbps Ethernet • Category 6: 10Gbps

coaxial cable: 2 conductores concéntricos de cobre bidireccionales broadband: • Múltiples canales en el cable

Cable de fibra óptica: Fibra de vidrio que transporta pulsos de luz, cada pulso es un bit Operan a alta velocidad: • high-speed point-to-point transmission (e.g., 10's-100's Gbps transmission rate) Baja tasa de errores: • repeaters spaced far apart • immune to electromagnetic noise

radio link types: terrestrial microwave

LAN (e.g., WiFi)

wide-area (e.g., cellular)

satellite

Protocolos IoT

• • • • • • 802.15.4 – LR WPAN: es una colección de estándares para redes de área personal de tasa de transferencia baja (LR-WPANs). 6LoWPAN: (IPv6 over Low Power Wireless Personal Area Networks) trae el protocolo IP a los dispositivos de baja potencia que tienen capacidad de procesamiento limitada. CoAP: CoAP es un protocolo para usarse en dispositivos de internet restringidos en recursos (p.ej: nodos de redes de sensores inalámbricas). Websocket: WebSocket se basa en TCP y permite streams de mensages a ser enviados en ambos sentidos entre cliente y servidor, mientras se mantiene la conexión TCP abierta. DDS (Data distribution service): es un middleware centrado en datos para la comunicación de dispositivo-a-dispositivo or máquina-a-máquina. XMPP (Extensible Messaging and Presence Protocol) es un protocolo para comunicación de tiempo real y streaming de datos XML entre entidades de red. XMPP soporta caminos de comunicación client-to-server y server-to-server.

Críticas al modelo de referencia TCP/IP

• • **Problemas:**

– No se distingue entre servicio e interfaz. • Se quiere comunicar con aplicaciones por medio de direcciones físicas de interfaces. No hay dirección de nodo o aplicación. – No es un modelo general: no esta ajustado para describir ninguna pila de protocolos mas que TCP/IP. – No se mencionan las capas físicas y de enlace de datos – Protocolos altamente entrincherados y difíciles de remplazar.

Modelo Híbrido

Cómputo en la Nube (Cloud)

• Nube se refiere a una red pública, privada o híbrida que proporciona servicios remotos

– Permite la manipulación, configuración y acceso a recursos de hardware y software de forma remota.

Hay 3 familias de recursos:

– Recursos de procesamiento:

• Aplicaciones

• Máquinas virtuales

• Contenedores: (p.ej: Dockers)

– Recursos de almacenamiento: Recursos de almacenamiento:

• Almacenamiento de archivos: (p.ej. dropbox) dropbox)

Almacenamiento de bloques:

– Recursos de infraestructura: combinación de elementos de procesamiento y almacenamiento conectados en una red interna virtual

• **Cómo se relaciona esto con redes?**

– 1) El cliente se conecta a los data centers por medio de redes TCP/IP (Internet), bajo el paradigma cliente-servidor (Se muestra en figura anterior). – 2) Dentro de los data-centers, el hardware de los servidores se conecta en red ("blades", "patcheras", "racks"). – 3) Dentro de los data-centers, los recursos de los servidores se abstraen en máquinas virtuales o contenedores que se conectan por medio de redes virtuales entre ellos.

Hay una red virtual adentro de una red física.

• **Servicios** • • **Infrastructure-as-a-service (IaaS):**

Platform-as-a-service (PaaS):

• **Software-as-a-service:** • **Virtualización:**

– Virtualización es el proceso de convertir un recurso IT físico en un recurso IT virtual. – Servidores virtuales usan guest OS. – La funcionalidad del software de virtualización incluye servicios de sistema relacionados a gestión de máquina virtual; este gestor de máquina virtual se llama hipervisor. – Tanto guest OS y software de aplicación ejecutando en servidor virtual no son conscientes del proceso de virtualización.

• **Containerización:**

– Se empaqueta el código de la aplicación junto con los archivos de configuración relacionados, librerías y dependencias requeridas para que pueda ejecutar. – Este paquete de software o contenedor se abstrae del SO y es portable. – Las aplicaciones son desplegadas en contenedores. Cada contenedor ejecuta en un proceso.

– Usar contenedores permite a varios servicios de la nube ejecutarse como un servidor (físico o virtual) único mientras se accede al mismo SO.

Convenciones a respetar

• • • • • • • • • • • • B mayúscula = 1 byte = 8 bits (= 2^3 bits) 1KB = 2^10 B = 1024 B (=2^13 bits = 8192 bits) kibibyte 1MB = 2^20 B = 1.048.576 B 1GB = 2^30 B En resumen, para los casos anteriores se usan potencias de 2 junto con bytes. En cambio, b minúscula = 1 bit 1Kb = 10^3 b = 1000 b – Kilo bit 1Mb = 10^6 b = 1.000.000 b – Mega bit 1Gb = 10^9 b =

Capa de aplicación

En la capa de aplicación tenemos las aplicaciones de red.

Cada aplicación de red ofrece un servicio específico – con su propia forma de interfaz con el usuario.

dos enfoques para desarrollar aplicaciones de red:

1. El programador para especificar la comunicación usa una interfaz para programas de aplicación (API).

– Una API es conjunto básico de funciones. – La socket API se usa para el software que se comunica sobre la internet.

2. La Web: El programador se apoya en la tecnología de la web para construir una aplicación de red..

– La Web provee servicios al software de la aplicación que • hacen más fácil a los desarrolladores implementar la comunicación y la entrada/salida de modo que • se pueden enfocar en el propósito específico de la aplicación.

Arquitecturas de aplicaciones

Las aplicaciones red suelen usar uno de los siguientes estilos de arquitectura:
• cliente-servidor • peer-to-peer (P2P)

Arquitecturas Cliente-Servidor

• • En el modelo cliente-servidor hay dos procesos que se comunican: uno en la máquina cliente y otro en la máquina servidor. Forma de la comunicación: 1. El proceso cliente manda solicitud al proceso servidor, 2. el proceso cliente espera un mensaje de respuesta; 3. luego el proceso servidor recibe y procesa la solicitud; 4. el proceso servidor manda mensaje de respuesta al proceso cliente.

Características de los servidores:

• Siempre están en un host; • con dirección IP permanente; • se pueden usar centros de datos para escalabilidad.

Características de los clientes:

• Pueden estar conectados intermitentemente; • usando direcciones IP dinámicas; • Los clientes no se comunican directamente entre sí.

Aplicaciones Cliente Servidor en internet usando UDP

Pasos de una aplicación cliente-servidor usando UDP.

1. Cliente crea datagrama con IP y puerto del servidor y envía datagrama • Datagrama puede perderse. 2. Si llega, servidor lee datagrama 3. Servidor envía respuesta especificando dirección y puerto de cliente • Datagrama puede perderse. 4. Si llega, cliente lee datagrama. 5. Cliente finaliza • Evaluación: – No se dice qué se hace si respuesta no llega al cliente. – Es responsabilidad de la aplicación red manejar esto.

Aplicaciones Cliente Servidor en internet usando TCP

Pasos de una aplicación cliente-servidor usando TCP:

1. Se ejecuta proceso del servidor 2. Servidor espera por pedido de conexión entrante. 3. El cliente requiere pedido de conexión al servidor 4. El servidor acepta la conexión con el cliente 5. El cliente envía pedido al servidor 6. El servidor lee el pedido 7. El servidor envía la respuesta 8. El cliente lee la respuesta – TCP provee transferencia de stream de bytes ordenada 9. Si hay más pedidos al servidor: Goto 5 10. El cliente cierra la conexión 11. El servidor cierra la conexión

Arquitectura P2P

Características de una arquitectura P2P:

• Mínimo o ningún apoyo en servidores. • Hosts arbitrarios (llamados compañeros) se comunican directamente entre sí. • Compañeros piden servicio de otros compañeros, y proveen servicio en retorno a otros compañeros • Nuevos compañeros traen nueva capacidad de servicio, así como nuevas demandas de servicios. • Los compañeros se conectan intermitentemente y cambian las direcciones IP.

Ejemplos

Distribución de archivos: la aplicación distribuye distribuye un archivo

de una única fuente a un gran a un gran número de compañeros.

Un ejemplo es BitTorrent.

Bases de datos distribuidas

Streaming: VoIP:

P2P vs cliente-servidor

Problema: ¿Cuánto tiempo se requiere para distribuir un archivo (de tamaño F) de un servidor a N compañeros?

La capacidad de subida y de bajada de compañeros es un recurso limitado.

Distribución de Archivos: Cliente-Servidor

¿Qué Parámetros hay que considerar? – Tasa de subida del enlace de acceso al compañero i: ui – Tasa de subida del enlace de acceso al servidor: us – Tasa de descarga del enlace de acceso al compañero i: di – Tamaño del archivo a ser distribuido: F – Número de compañeros que quieren adquirir una copia del archivo: N

El tiempo de distribución es el tiempo que toma obtener una copia del archivo por los N compañeros. – Asumimos

Asumimos que la internet tiene abundante ancho de banda y todos los cuellos de botella suceden en ISP de acceso.

– Asumimos que los servidores y clientes no participan de otras aplicaciones de red.

Transmisión del servidor:

Transmisión del servidor: debe enviar secuencialmente (subida) N copias de archivo a cada peer (manda NF bits). – Tiempo para enviar 1 copia: F/us

– Tiempo para enviar N copias: NF/us

– Demasiado trabajo del servidor

Descarga del Cliente:

cada cliente debe descargar una copia de archivo. dmin = min{d1,dp,...,dN}. Tiempo de descarga del cliente con dmin: F/dmin segs Este es el tiempo de descarga peor.

** Dc-s > max{NF/us„F/dmin}**

Tiempo para distribuir F a N clientes usando enfoque cliente-servidor

Distribución de Archivos: P2P

Al comienzo de la distribución solo el servidor tiene el archivo.

Para que la comunidad de compañeros reciba este archivo,

el servidor debe enviar cada bit del archivo al menos una vez en su enlace de acceso.

En P2P cada compañero puede redistribuir cualquier porción del archivo que ha recibido a cualesquiera otros compañeros.

Así los compañeros asisten al servidor en el proceso de distribución.

Cuando un compañero recibe algo de datos de un archivo, puede usar su capacidad de subida para redistribuir los datos a los otros compañeros.

La capacidad total de subida del sistema es:

utotal = us + Sui utotal

Por lo tanto el tiempo mínimo de distribución es: NF/utotal

Transmisión de servidor: debe subir al menos una copia.

Tiempo para enviar una copia : F/us

cliente:

cada cliente debe descargar la copia de un archivo

ui Tiempo mínimo de descarga de cliente: F/dmin

clientes: como agregado deben subir NF bits

Tasa de subida máxima

es us + Sui

DP2P > max{F/us„F/dmin„NF/(us + Sui)}

tiempo para distribuir F a N clientes usando enfoque P2P

Distribución de archivos P2P: BitTorrent

El archivo se divide en trozos de 256Kb.

Los compañeros en torrent envían/reciben trozos.

tracker: lleva la pista de compañeros torrent: grupo de compañeros Participando en Torrent intercambiando trozos de un archivo

Cuando compañero se une a Torrent:
No tiene trozos, pero va a acumularlos a lo largo del tiempo de otros compañeros

Se registra con tracker para obtener lista de compañeros.

Se conecta con un subconjunto de compañeros ("vecinos.")

Un compañero avisa periódicamente a tracker que está en BitTorrent.

Mientras descarga, compañero sube trozos a otros compañeros. Un compañero puede cambiar de compañeros con los cuales intercambia trozos. Los compañeros pueden ir y venir.

Una vez que un compañero tiene un archivo completo: Puede (egoístamente) irse o (altruísticamente) permanecer en Torrent subiendo trozos.

BitTorrent: pedir y enviar trozos de archivos

Enviar trozos: tit-for-tat

Pedir trozos:
En

Alicia envía trozos a aquellos 4 compañeros actualmente enviándole a Alicia trozos a la velocidad mayor. re-evalua los 4 mejores cada 10 seg • Cada 30 seg: elegir aleatoriamente otro compañero, y comenzar enviándole trozos.

En un momento dado, diferentes compañeros tienen diferentes subconjuntos de trozos de archivos • Periódicamente, Alicia pide a cada compañero por una lista de trozos que ellos tienen.

Alicia puede hacer esto porque
Un compañero nuevo elegido puede unirse a los 4 de más arriba. Pero para esto tiene que ser uno de los 4 mejores subidores para Alicia.

sabe qué trozos tienen sus compañeros.

Conviene pedir primero los trozos
menos comunes (i.e. con menos copias en compañeros).

Protocolos de capa de aplicación

Reglas

Cosas a definer en un protocolo de capa de aplicación:
de cuándo y cómo los procesos envían y responden a mensajes.

Tipos de mensajes intercambiados

Estado

de la aplicación. En qué consiste y cómo se lo mantiene.

Sintaxis del mensaje:

Tipos de protocolos
Protocolos abiertos: Son definidos en RFCs

Semántica del mensaje

Permiten interoperabilidad

P.ej: HTTP,

Protocolos propietarios: P.ej: Skype

FTP: Protocolo de Transferencia de Archivos

Algunas características de FTP:
Usado para transferir archivo hacia/desde host remoto

Cada archivo tiene restricciones de acceso y poseción.

FTP permite inspeccionar carpetas.

FTP permite mensajes de control textuales.

3 tipos de mensajes son intercambiados:
Uso de comandos enviados al servidor ftp

Mensajes de respuesta a comandos del servidor ftp Mensajes con datos enviados.

Sintaxis de Mensajes de respuesta
Código de estatus y frase **Ejemplos**

Sintaxis de comandos

USER username

331 Username OK, password required

PASS password

LIST return list of file in current directory

RETR filename retrieves (gets) file

STOR filename stores (puts) file onto remote host

452 Error writing file

Reglas:

1. Cliente FTP contacta servidor FTP en puerto 21, usando TCP.
2. El cliente es autorizado en la conexión de control.
3. El cliente inspecciona directorio remoto, envía comandos sobre la conexión de control.

Se comienza con identificación de usuario y password.

Estado

4. Cuando el servidor recibe un comando de transferencia de archivo, el servidor abre una 2da conexión de datos TCP

El servidor FTP mantiene el "estado": directorio corriente, autenticación previa.

(para el archivo) con el cliente. con el cliente.

La Web: Parte 2

HTML

es el lenguaje estándar para crear páginas web describe la estructura de una página web indica al navegador como mostrar el contenido de la página Un documento HTML es una serie de elementos:

Un elemento es contenido encerrado entre etiquetas.

Una etiqueta tiene un nombre.

Una etiqueta está demarcada entre ‘<‘ y ‘>’.

Etiquetas pueden tener o no atributos.

Un atributo tiene un nombre y un valor (que es un string) separados por ’=’.

Páginas dinámicas

Páginas dinámicas: Son páginas web generadas por programas que se ejecutan en el servidor (posiblemente con una base de datos).

Tareas que suelen hacer Procesar parámetros de formularios Procesar encabezados de pedido HTTP Pedir datos a fuentes de datos Generar página web con los datos recibidos. Generar encabezados de respuesta HTTP

PHP

Enfoque PHP (Preprocesador de Hipertexto).

Se definen páginas dinámicas mediante la inserción de comandos especiales dentro de páginas HTML

El código PHP es interpretado por un servidor web.

PHP se diseñó para trabajar con el servidor web Apache.

PHP puede ser usado en la mayoría de los servidores web.

Algunas cosas que puede hacer PHP:

puede generar contenido de página dinámica

puede operar con archivos en el servidor.

puede recolectar datos de formulario

puede enviar y recibir cookies

puede acceder a encabezados de pedido HTTP

permite definir encabezados de respuesta HTTP

Acceso a campos de formularios:

$_POST: usado para recolectar datos de formulario luego de someter un formulario con método POST.

$_GET: usado para recolectar datos de formulario luego de someter un formulario con método GET. Se usa como en el ítem anterior.

Tipos de datos de PHP

String:

Float: Boolean: Array: Object:

Acceso a información de encabezados HTTP:

$_SERVER: contiene información de encabezados, caminos y localización de scripts.

Para acceder a encabezados poner como argumento alguna de las siguientes:

HTTP_USER_AGENT, SERVER_ADDR, SERVER_NAME, SERVER_SOFTWARE, SERVER_PROTOCOL, REQUEST_METHOD, REQUEST_TIME, QUERY_STRING, HTTP_ACCEPT, HTTP_ACCEPT_CHARSET, HTTP_HOST, etc.

P.ej.: Para acceder al encabezado User-Agent: $_SERVER['HTTP_USER_AGENT']

Definición de cookies:

Setcookie() define cookie para ser enviada junto con el resto de los encabezados HTTP.

Esta función debe usarse antes de generar cualquier salida, o sea antes que la etiqueta

Acceso al valor de una cookie:

Capa de Transporte Transferencia de datos confiable y control de flujo

Entrega de datos confiable

La capa de transporte debe soportar al menos un protocolo para entrega de datos confiable.

Estos protocolos asumen que el canal puede: Corromper paquetes

Perder paquetes

La transferencia de datos es en un sentido, o sea hay un emisor y un receptor.

Preliminares

CT se ocupa de uso de temporizadores y retransmisiones de paquetes. Paquetes perdidos deben retransmitirse.

Sabemos que un paquete no se perdió

porque fue confirmado con un paquete de confirmación de recepción.

Podemos asumir que si pasa un cierto tiempo y no fue confirmado entonces se perdió y hay que retransmitirlo.

Para medir el tiempo: – Usar temporizadores (timers)

El mismo paquete llega dos o más veces al receptor y la capa de transporte la pasa a la capa de aplicación más de una vez.

asignar números de secuencia a los paquetes que salen. La idea es que dado un número de secuencia de un segmento que acaba de llegar,

el receptor puede usar ese número de secuencia para decidir si el segmento es un duplicado y en ese caso descartarlo.

Protocolo de Parada y Espera

Comportamiento del emisor:

Suposición: el canal de comunicaciones subyacente puede perder paquetes

El emisor envía paquete P y para de enviar.

Espera: El emisor espera una cantidad "razonable" de tiempo para el ACK

Los paquetes tienen N°de secuencias

Si llega el ACK a tiempo, se envía siguiente paquete. Goto 2.

Se trabaja con Acks

Sino se retransmite paquete P. Goto 2.

El receptor debe especificar N° de secuencia del paquete siendo confirmado.

Si hay paquete o ACK demorado pero no perdido:

Se usan retransmisiones de paquetes.

La retransmisión va a ser un duplicado con igual N°de secuencia ; luego se descarta en el receptor.

Para esto se requiere de uso de temporizadores.

Desempeño de Parada y Espera

desempeño pobre.

** Denvío es la demora en enviar un paquete. U sender : utilización– fracción del tiempo en que el emisor está ocupado enviando. RTT es tiempo de ida y vuelta de un bit: RTT = 30 msec.**

El protocolo de red limita el uso de recursos físicos.

Operación de Parada y Espera

Protocolos de tubería

Tubería: el emisor puede enviar múltiples paquetes al vuelo a ser confirmados

El rango de números de sequencia debe ser incrementado usando palabras de más de un bit.

Hay que usar búferes en el emisor.

dos formas genéricas de protocolos de tubería:

retroceso N y repetición selectiva

Tubería: utilización

incrementada

Protocolos de tubería: visión general

Retroceso-N:

Repetición selectiva:

Receptor envía ack acumulativo

El receptor envía confirmaciones individuales para cada paquete

No confirma paquetes si hay un agujero.

El emisor tiene un timer para el paquete más viejo no confirmado

El emisor mantiene un timer para cada paquete no confirmado

Cuando expira el timer retransmite todos los paquetes no confirmados.

Cuando el timer expira, retransmite solo ese paquete no confirmado.

Uso de búferes en el emisor

La ET emisora debe manejar búferes para los mensajes de salida.

porque:

puede hacer falta retransmitirlos

¿Cómo se usan búferes en el emisor? • **El emisor almacena en búfer todas los segmentos**

hasta que se confirma su recepción.

Retroceso N

Si un paquete T a la mitad de una serie larga se daña o pierde:

La CT receptora debe entregar paquetes a la capa de aplicación en secuencia.

Por lo que no se pueden entregar a la capa de aplicación los paquetes que llegaron bien después de T.

¿qué debe hacerse con los paquetes correctos que le siguen a un paquete que se perdió?

Con retroceso N el receptor descarta todos los paquetes subsecuentes al paquete perdido, sin enviar ack para los paquetes descartados. Suposición:

Hay un límite en la cantidad de paquetes enviados y no confirmados + paquetes por enviar que puede almacenar el emisor en búferes.

¿Cómo representar

eficientemente?

Usar intervalos de números de secuencia dentro del espacio de números de secuencia.

Un intervalo de esos recibe el nombre de ventana corrediza.

Retroceso-N: en el emisor

La"ventana" permite hasta N paquetes consecutivos sin confirmar

ventana emisora = tramas enviadas sin ack positivo o tramas listas para ser enviadas.

timeout(n): retransmite paquete n y todos los paquetes de mayor N°de secuencia en la ventana.

un paquete de confirmación solamente,

número de secuencia

Enviar ACK con N° de secuencia más alto tal que los N° de secuencia anteriores fueron recibidos.

A esto se le llama ACK acumulativo.

Si se pierde un segmento llegan bien varios de los siguientes, para estos se generan ACKs duplicados.

Para los números de secuencia, el receptor maneja variable expectedSeqnum que es el número de secuencia más chico que no llegó aun.

Retroceso-N en acción

¿Si el espacio de secuencia es de MAX_SEQ + 1 números de secuencia (estos comienzan desde 0), se puede hacer la ventana emisora de tamaño MAX_SEQ + 1?

La respuesta es no

El tamaño de la ventana emisora no puede superar MAX_SEQ cuando hay MAX_SEQ + 1 números de secuencia.

¿Cómo evitar que haya más de MAX_SEQ paquetes sin ack pendientes?

prohíbir a la CR que moleste con más trabajo.

Usar enable_network_layer y disablenetwork_layer.

¿Cuál es el problema principal de retroceso N?

El uso ineficiente del canal frente a segmentos perdidos o demorados.

Repetición Selectiva

¿qué debe hacerse con los paquetes correctos que le siguen a un paquete que se perdió?

Los paquetes en buen estado recibidos después de un paquete dañado E se almacenan en búfer.

Cuando el paquete E llega correctamente, el receptor entrega a la capa de aplicación, en secuencia, todos los paquetes posibles que ha almacenado en el búfer.

Mecanismo común de retransmisiones: o El temporizador de E termina y el emisor lo manda de nuevo.

Una solución mejor:

Uso de una ack negativa (NAK) por el receptor.

Así se estimula la retransmisión de paquetes antes que temporizadores terminen y así se mejora el rendimiento.

¿Y si ¿Y si la NAK se pierde?

El receptor confirma individualmente todos los paquetes recibidos correctamente.

Hay búferes para paquetes según se necesiten para su entrega eventual en orden a la capa de aplicación.

El emisor solo reenvía paquetes para los cuales el ACK no fue recibido o se recibió un NAK.

Hay un temporizador del emisor para cada paquete no confirmado.

Ventana del emisor

Contiene N N°de secuencias consecutivos Limita N°de sequencias a enviar a paquetes no confirmados.

¿Qué tipos de paquetes puede haber en la ventana del emisor?

Paquetes enviados y confirmados porque antes hay paquetes no confirmados

Paquetes enviados y no confirmados Paquetes listos para enviarse en búfer **¿Cómo representar el conjunto de paquetes que puede almacenar en búfer el receptor?**

Usar intervalos de números de secuencia dentro del espacio de números de secuencia. Un intervalo de esos recibe el nombre de ventana corrediza.

Tipos de paquetes que puede haber en la ventana del receptor: Paquetes esperados y no recibidos Paquetes recibidos fuera de orden Paquetes aceptables en la ventana que no han llegado aun

Se mantiene en búfer un paquete aceptado por la ventana receptora

hasta que todos los que le preceden hayan sido pasados a la capa de aplicación.

Repetición Selectiva: ventanas del emisor y del receptor

Repetición Selectiva

Algunos detalles

tamaño de ventana emisora comienza en 0 y crece hasta MAX_SEQ.

El receptor tiene un búfer para cada N° de secuencia en su ventana.

¿Qué se hace cuando llega un paquete?

Cuando llega un paquete, su número de secuencia es revisado para ver si cae dentro de la ventana.

De ser así, y no ha sido recibido aun, se acepta y almacena.

Repetición selectiva en acción

Dilema de repetición

selectiva

Repetición Selectiva

Regla para el tamaño de la ventana receptora:

Tamaño de ventana receptora = (MAX_SEQ + 1)/2. Con tamaños mayores de ventana receptora no funciona.

En encabezado de paquete hay N° de secuencia de k bits.

¿Cómo transmitir datos entre dos máquinas y en ambas direcciones eficientemente?

llevar a caballito (piggybacking).

cuando llega un segmento S con datos, el receptor se aguanta y espera hasta que la capa de aplicación le pasa el siguiente paquete P.

La confirmación de recepción de S se anexa a P en un segmento de salida (usando el campo ack en el encabezado del segmento de salida).

extender repetición selectiva para tener flujos de datos entre 2 máquinas en las dos direcciones?

Se usa llevar a caballito.

La capa de transporte para mandar un ack, debe esperar por un paquete al cual superponer un ack.

¿Cómo evitar retrasar demasiado envío de confirmaciones de recepción por no tener tráfico de regreso?

método que usa temporizador auxiliar

tras llegar un paquete de datos en secuencia, se arranca un temporizador auxiliar mediante start_ack_timer.

Si no se ha presentado tráfico de regreso antes de que termine este temporizador, se envía un paquete de ack independiente.

tiempo de temporizador auxiliar « tiempo de temporizador de retransmisiones.

« significa mucho menor. «

para asegurarse que la ack de un paquete correctamente recibido llegue antes que el emisor termine su temporización y retransmita el paquete.

Control de flujo

Control de Flujo: Hay que evitar que un host emisor rápido desborde a un host receptor lento.

Tipo de control de flujo del que se ocupa la capa de enlace de datos:

Control de flujo entre dos máquinas directamente conectadas entre sí

¿Por qué puede necesitarse control de flujo en la capa de transporte si la capa de enlace de datos lo hace?

El receptor puede demorarse en procesar mensajes debido a los problemas de la red:

pérdida de segmentos,

no se pueden procesar segmentos porque faltan anteriores.

Uso de búferes

¿Porqué esto es necesario?

Si la llegada de segmentos del emisor es mucho más rápido que el receptor para procesar los segmentos recibidos,

entonces el receptor necesitará poder almacenar segmentos antes de procesarlos.

El receptor puede acumular una cantidad de segmentos suficientes antes de pasarlos a la capa de aplicación para que los procese.

Los segmentos pueden llegar desordenados;

por lo tanto si llegan un grupo de segmentos y faltan segmentos previos a ellos, habrá que almacenarlos segmentos de ese grupo en buffer.

¿Qué hace el receptor con los búferes si tiene varias conexiones?

Solución 1: se usan los búferes a medida que llegan segmentos.

Solución 2: se dedican conjuntos de búferes específicos a conexiones específicas.

¿Qué hace el receptor cuando entra un segmento?

Cuando entra un segmento el receptor intenta adquirir un búfer nuevo;

si hay uno disponible, se acepta el segmento; de otro modo se lo descarta.

Suposición: cambia el patrón de tráfico de la red; se abren y cierran varias conexiones en el receptor.

Consecuencias: El receptor y el emisor deben ajustar dinámicamente sus alojamientos de búferes.

Esto significa ventanas de tamaños variables.

Ahora el emisor no sabe cuántos datos puede mandar en un momento dado, pero sí sabe cuántos datos le gustaría mandar.

¿Qué reglas cumpliría un protocolo entonces?

El host emisor solicita espacio en búfer en el otro extremo.

Para estar seguro de no enviar de más y sobrecargar al receptor. o Porque sabe cuánto necesita.

¿Qué pasa con el receptor al recibir ese pedido?

Sabe cuál es su situación y cuánto espacio puede otorgar. Aquí el receptor reserva una cierta cantidad de búferes al emisor.

Los búferes podrían repartirse por conexión, o no.

¿Qué pasa si los búferes se reparten por conexión y aumenta la cantidad de conexiones abiertas?

El receptor necesita ajustar dinámicamente sus reservas de búferes.

¿Cómo funciona la comunicación entre host emisor y host receptor usando la solución?

Inicialmente el emisor solicita una cierta cantidad de búferes, con base en sus necesidades percibidas.

El receptor otorga entonces tantos búferes como puede.

El receptor, sabiendo su capacidad de manejo de búferes podría indicar al emisor “te he reservado X búferes”.

¿Cómo hace el receptor con las confirmaciones de recepción?

El receptor puede incorporar tanto las ack como las reservas de búfer al en el mismo segmento.

El emisor lleva la cuenta de su asignación de búferes con el receptor.

¿qué pasa con la asignación de búferes (disponibles) en el emisor cada vez que el emisor envía un segmento?

Debe disminuir su asignación

¿qué pasa si la asignación de búferes (disponibles) en el emisor llega a 0?

El emisor debe detenerse por completo

Situación: Información de reserva de búferes viaja en segmento que no contiene datos y ese segmento se pierde. Esto termina ocasionando deadlock.

Solución: Cada host puede enviar periódicamente un segmento de control con el ack y estado de búferes de cada conexión. Así el estancamiento se romperá tarde o temprano.

Control de flujo en TCP

No se requiere:

que los emisores envíen datos tan pronto como llegan de la aplicación.

que los receptores envíen confirmaciones de recepción tan pronto como sea posible.

que los receptores entreguen datos a la aplicación apenas los reciben.

Esta libertad puede explotarse para mejorar el desempeño.

Campo Tamaño de ventana en el encabezado TCP:

N° de bytes que pueden enviarse comenzando por el byte cuya recepción se ha confirmado.

0: indica que se han recibido los bytes hasta n° de confirmación de recepción – 1, inclusive, pero el receptor quisiera no recibir más datos por el momento.

El permiso para enviar puede otorgarse enviando un segmento con el mismo n° de confirmación de recepción y un campo tamaño de ventana distinto de 0.

¿Qué se hace si la ventana anunciada por el receptor es de 0?

El emisor debe detenerse hasta que el proceso de aplicación del host receptor retire algunos datos del búfer

en cuyo momento el TCP puede anunciar una ventana más grande.

En TCP los hosts en cada lado de una conexión tienen un buffer de recepción circular para la conexión. **un buffer de recepción circular**

Cuando la conexión TCP recibe bytes en el orden correcto y en secuencia, coloca los datos en el buffer de recepción.

Cálculo de tamaño de ventana:

Tamaño de ventana =RcvBuffer − [LastByteRcvd − LastByteRead]

Inicialmente se puede mandar:

Tamaño de ventana = tamaño RcvBuffer

Propiedad a respetar en el emisor:

LastByteSent − LastByteAcked ≤ tamaño de ventana

¿Cómo manejar pérdidas de segmentos en TCP?

Solución 1:

el receptor solicita segmento/s específico/s mediante segmento especial llamado NAK.

Tras recibir segmento/s faltante/s, el receptor puede enviar una confirmación de recepción de todos los datos que tiene en búfer.

Cuando el receptor nota una brecha entre el número de secuencia esperado y el número de secuencia del paquete recibido, el receptor envía un NAK en un campo de opciones.

Solución 2:

(acks selectivos) el receptor le dice al emisor que piezas recibió.

El emisor puede así reenviar los datos no confirmados que ya envió.

Se usan dos campos de opciones:

Sack permited option: se envía en segmento SYN para indicar que se usarán acks selectivos.

Capa de Transporte Administración del temporizador de retransmisiones en TCP

Administración del temporizador del TCP

¿qué tan grande debe ser el intervalo de expiración del temporizador de retransmisión?

Si se hace demasiado corto

Ocurrirán retransmisiones innecesarias.

Si se hace demasiado largo?

Sufrirá el desempeño por el gran retardo de retransmisión de cada paquete perdido

La varianza y la media de la distribución de llegada de las ack pueden variar a medida que se generan y se resuelven congestionamientos.

Idea: Ajustar constantemente el intervalo de expiración del temporizador, con base en mediciones continuas del desempeño de la red.

Solución: Algoritmo de Jacobson (1988) usado por TCP

Por cada conexión el TCP mantiene una variable, RTT (round time),

significa estimación actual del tiempo de ida y vuelta al destino.

Al enviarse un segmento se inicia un temporizador,

para saber el tiempo que tarda el ack,

y para habilitar una retransmisión si se tarda demasiado.

Si llega el ack antes de expirar el temporizador:

TCP mide el tiempo que tardó el ack , digamos M,

entonces actualiza el RTT así:

RTT = RTT + (1-) M,

es el peso que se le da al valor anterior. Por lo común = 7/8.

Un RTT inicial de 1 sec se aconseja en RFC 6298.

Dado RTT, hay que elegir una expiración adecuada del temporizador de retransmisión.

Solución hacer que el valor de timeout sea sensible tanto a la variación de RTT como a la varianza de la función de densidad de probabilidad del tiempo de llegada de los ack.

Se Se mantiene una variable amortiguada D (la desviación media).

Al llegar un ack, se calcula |RTT M|.

Se mantiene en D mediante:

D = D + (1)|RTT M|,

donde típicamente es ¾

D es una aproximación bastante cercana a la desviación estándar.

¿Cómo estimar la expiración del temporizador? ¿De qué parámetros depende?

La mayoría de las implementaciones TCP usan ahora este algoritmo y establecen:

Expiración del temporizador = RTT + 4 x D.

Con esto menos del 1% de los ack vienen en más de 4 desviaciones estándares tarde.

¿qué se hace al recolectar muestras M cuando expira el temporizador de un segmento y se envía de nuevo?

Cuando llega el ack no es claro si éste se refiere a la primera transmisión o a una posterior.

Si se adivina mal, se puede contaminar seriamente la estimación del RTT.

¿Cómo se puede estimar el temporizador de retransmisiones en ese caso?

Solución: (algoritmo de Karn)

No actualizar el RTT (cuando llega ack) de ninguno de los segmentos retransmitidos.

Cuando ocurre un timeout se duplica la expiración del temporizador.

Tan pronto se recibe un ack de segmento no retransmitido, el RTT estimado es actualizado y la expiración del temporizador se computa nuevamente usando la fórmula anterior.

El algoritmo de Karn lo usan la mayoría de las implementaciones TCP.

UDP (protocolo de datagramas de usuario)

Es no orientado a la conexión.

segmentos = encabezado de 8 B + carga útil.

2 puertos de 16b.

El campo longitud UDP incluye el encabezado de 8 bytes y los datos.

UDP no realiza:

control de flujo, control de congestión, o retransmisión cuando se recibe un segmento erróneo.

Todo lo anterior le corresponde a los procesos de usuario.

UDP es especialmente útil en las situaciones clienteservidor.

El cliente envía una solicitud corta al servidor y espera una respuesta corta.

¿Qué pasa si se pierde la solicitud o la respuesta?

Capa de Transporte Complementos de control de flujo

Uso de búferes

Aun si el receptor está de acuerdo en usar búferes, todavía queda la cuestión del tamaño de estos.

Solución 1: Si la mayoría de los segmentos tiene aproximadamente el mismo tamaño organizar los búferes como un grupo de búferes de tamaño idéntico,

Si el tamaño de búfer se escoge igual al tamaño del segmento más grande, se desperdiciará espacio cada vez que llegue un segmento corto.

Si el tamaño se escoge menor que el tamaño máximo de segmento, se requerirán varios búferes para los segmentos grandes, con la complejidad inherente.

Solución 2: Uso de búferes de tamaño variable

la ventaja aquí es un mejor uso de la memoria, al costo de una administración de búferes más complicada.

Solución 3:

dedicar un solo búfer circular grande por conexión,

se

se hace buen uso de la memoria cuando todas las conexiones tienen una carga alta,

pero es deficiente si algunas conexiones cuentan con poca carga.

Control de flujo en TCP

Cuando la ventana es de 0, el emisor no puede enviar segmentos, salvo en dos situaciones:

pueden enviarse datos urgentes

el emisor puede enviar un segmento de 1 B para hacer que el receptor re-anuncie el siguiente byte esperado y el tamaño de la ventana.

TCP proporciona esta opción para evitar un bloqueo irreversible si llega a perderse un anuncio de ventana.

Un tamaño de ventana más grande permitirá al emisor continuar enviando datos, pero como el campo de tamaño de ventana es de 16 bits, es imposible expresar tal tamaño.

Solución (opción de escala de ventana):

permitir al emisor y al receptor negociar un factor de escala de ventana.

Ambos lados pueden desplazar el tamaño del campo de ventana hasta 14 bits a la izquierda,

permitiendo por lo tanto ventanas de hasta 2^30 bytes.

Capa de Transporte Control de Congestión

Control de Congestión

Si un emisor manda a un receptor más información que la capacidad de carga de la subred:

la subred se congestionará pues será incapaz de entregar los segmentos a la velocidad con que llegan.

Se necesita un mecanismo de control de congestión basado en la capacidad de carga de la subred.

El mismo debe aplicarse al emisor.

Para controlar la congestion:

En TCP algunos hosts disminuirán la tasa de datos.

Para llevar la cuenta de cuántos datos un host puede enviar por la red:

TCP maneja una ventana para la congestión (VC) cuyo tamaño es el número de bytes que el emisor puede tener en la red en todo momento.

En TCP el host tiene una forma de detectar congestión.

En TCP cuando un host detecta congestion:

El host ajusta el tamaño de la VC.

La expiración de un temporizador causada por un paquete perdido se puede deber a: ruido en la línea de transmisión o

el descarte de paquetes en el enrutador congestionado.

Hoy la pérdida de paquetes por errores de transmisión es rara debido a que las troncales de larga distancia son de fibra óptica.

Luego, la mayoría de las expiraciones de tiempo en Internet se deben a la congestión.

Solucíón de TCP:

Todos los algoritmos de congestión de TCP suponen que las expiraciones de tiempo son causados por congestión.

¿Cómo calcular un tamaño para la ventana de congestión (VC)?

probar con un mínimo de datos e ir duplicando gradualmente hasta que no se pueda más.

Un algoritmo basado en esta idea se llama arranque lento.

Algoritmo de arranque lento

El emisor asigna a la VC el segmento de tamaño máximo (STM) usado por la conexión; entonces envía 1 STM.

Emisor y receptor se ponen de acuerdo en el tamaño del STM.

Si se recibe el ack de este segmento antes que expire el temporizador, el emisor agrega el equivalente en bytes de un segmento a la VC para hacerla de 2 STM y envía dos segmentos.

Cuando la VC es de n segmentos, si de todos los n se reciben acks a tiempo, se aumenta la VC en la cuenta de bytes correspondiente a n segmentos.

La VC sigue creciendo exponencialmente hasta expiración temporizador (timeout) o alcanzar el tamaño de la ventana receptora.

Si ocurre timeout se recorta la VC a tamaño VC/2, o sea no se enviarán ráfagas de segmentos mayores a VC/2.

¿Cómo puede reconocer rápidamente el emisor que uno de sus paquetes se perdió?

Asumimos: cada paquete que llega al receptor dispara un paquete ack.

Cuando se pierde un segmento y otros segmentos luego del segmento perdido llegan al receptor:

El receptor genera acks que confirman lo mismo

Se llaman acks duplicados.

¿Qué significa que el emisor recibió un ack duplicado?

Es probable que llegó otro segmento al receptor y el segmento perdido no dio señales de vida.

Significado de recibir acks duplicados:

Como segmentos pueden tomar distintos caminos, pueden llegar fuera de orden y esto va a disparar acks duplicados incluso cuando no se ha perdido ningún segmento.

Si se pierde un segmento, habrá probablemente varios ack duplicados.

Solución: TCP asume que 3 acks duplicados implican que el paquete se perdió.

Luego ese paquete puede retransmitirse inmediatamente y antes de que expire el temporizador.

Esta heurística se llama retransmisión rápida.

Solución 2: Algoritmo de control de congestión de Internet (o TCP Tahoe):

Usa un umbral además de las ventanas de recepción y congestión.

Al ocurrir una expiración del temporizador o detectarse 3 acks duplicados, se fija el umbral en la mitad de la ventana de congestión actual, y la ventana de congestión se restablece a un segmento máximo.

Luego se usa el arranque lento para determinar lo que puede manejar la red, excepto que el crecimiento exponencial termina al alcanzar el umbral.

A partir del punto en el que se alcanza el umbral las transmisiones exitosas aumentan linealmente la ventana de congestión (en un segmento máximo por ráfaga).

Recomenzar con una ventana de congestión de un paquete toma un RTT (para todos los datos previamente transmitidos que dejen la red y para ser confirmados, incluyendo el paquete retransmitido).

Si no ocurren más expiraciones de temporizador/3 acks duplicados, la ventana de congestión continuará creciendo hasta el tamaño de la ventana del receptor.

En ese punto dejará de crecer y permanecerá constante mientras no ocurran más expiraciones de temporizador y la ventana del receptor no cambie de tamaño.

Comenzar con arranque lento cada vez que se pierde un paquete puede ser demasiado.

¿Qué se puede hacer para resolver este problema?

TCP Tahoe:

Invariante: tamaño ventana congestión $\leq$ tamaño ventana receptor Se usa arranque lento hasta alcanzar el umbral

Luego vienen incrementos aditivos hasta alcanzar timeout o 3 acks duplicados

Luego el umbral se fija a la mitad del tamaño de la ventana de congestión

Goto 1

Solución: Algoritmo de TCP Reno

Evitar arranque lento (excepto cuando la conexión es comenzada) cuando expira el temporizador de re-envíos.

Funcionamiento:

1. Luego de iniciada la conexión se comienza con arranque lento. 2. A continuación la ventana de congestión crece linealmente hasta que se detecta una pérdida de paquete. – Se cuentan acks duplicados – Se considera pérdida de paquete 3 acks duplicados
2. El paquete perdido es retransmitido (usando retransmisión rápida).
3. Recuperación rápida: – Se manda un paquete por cada ack duplicado recibido. – Un RTT luego de la retransmisión rápida el paquete perdido es confirmado. – La recuperación rápida termina con esa confirmación de recepción.
4. Luego de recibir el nuevo ack: • la ventana de congestión de una conexión se achica a la mitad de lo que era cuando se encontraron 3 duplicados (decrecimiento multiplicativo). • El conteo de ack duplicados se pone en 0.
5. Luego la ventana de congestión va incrementando de a un segmento por cada RTT (crecimiento aditivo). 7. Este comportamiento continua indefinidamente.

Capa de Transporte Direccionamiento

Direccionamiento

Existe un proceso especial llamado servidor de directorio que para cada tipo de servicio sabe cuáles son los puertos de los servidores que prestan ese tipo de servicio. Pasos usuario establece una conexión con el servidor de directorio

usuario envía un mensaje especificando el nombre del servicio.

servidor de directorio le devuelve la dirección puerto.

usuario libera la conexión con el servidor de directorio y establece una nueva con el servicio deseado.

cuando se crea un servicio nuevo?

servicio nuevo debe registrarse en el servidor de directorio, dando su nombre de servicio como la dirección de su puerto.

servidor de directorio registra esta información en su base de datos.

¿Cómo hacer para que un proceso servidor atienda a las necesidades de una máquina cliente?

Usar servidor que ejecuta los servidores inactivos

Servidor de procesos = intermediario de los servidores de menor uso.

pasos de la solución.

Escucha en un grupo de puertos al mismo tiempo esperando una solicitud de conexión

Un usuario emite una solicitud CONNECT, especificando el puerto del servicio que desea.

Si no hay ningún servidor esperándolo, consigue una conexión al servidor de procesos.

El servidor de procesos genera el servidor solicitado, permitiéndole heredar la conexión con el usuario existente.

El nuevo servidor hace el trabajo requerido y el servidor de procesos retorna a escuchar solicitudes nuevas.

Puertos bien conocidos

N° puertos bien conocidos $<$ 1024 Tabla de puertos bien conocidos. Demonios = procesos servidores que atienden en un puerto

P. ej. que el demonio FTP se conecte a sí mismo al puerto 21 en el tiempo de arranque.

Problema: Se podría llenar la memoria con demonios que están inactivos la mayor parte del tiempo.

Solución: Un solo demonio llamado inetd (demonio de internet), escucha un conjunto de puertos al mismo tiempo y espera por un pedido de conexión.

Usuarios potenciales de un servicio comienzan a hacer pedido CONNECT especificando el puerto del servicio que quieren. – Si no hay ningún servidor esperando por ellos, inetd bifurca un nuevo proceso y ejecuta el demonio apropiado en él, y ese demonio maneja la solicitud.

Inetd aprende qué puertos va a usar de un archivo de configuración.

Capa de Transporte Establecimiento y liberación de conexiones

Comparación de segmentos

T sec es el tiempo de vida de paquete

Se eliminan paquetes viejos que andan dando vueltas por ahí.

El origen etiqueta los segmentos con n° de secuencia que no van a reutilizarse dentro de T sec.

El T debería ser lo suficientemente grande como para incluir retransmisión confirmada de un paquete.

Establecimiento de Conexión

Como al establecer una conexión se usan segmentos, una conexión debería tener un N° inicial de secuencia con el que comienza a operar.

vincular N° inicial de secuencia de algún modo al tiempo y para medir el tiempo usar un reloj.

Implementación de la idea (de Tomlinson): Cada host tiene un reloj de hora del día.

Los relojes de los hosts no necesitan ser sincronizados;

se supone que cada reloj es un contador binario que se incrementa a si mismo en intervalos uniformes.

El reloj continua operando aun ante la caída del host

Cuando se establece una conexión los k bits de orden mayor del reloj = número inicial de secuencia.

Para lograr que al regresar al principio de los n° de secuencia, los segmentos viejos con el mismo n° de secuencia hayan desaparecido hace mucho tiempo

el espacio de secuencia debe ser lo suficientemente grande.

Problema: Cuando un host se cae, al reactivarse sus ET no saben dónde estaban en el espacio de secuencia.

Solución:

requerir que las ET estén inactivas durante T segundos tras una recuperación para permitir que todos los segmentos viejos expiren

¿Cómo hacer para establecer una conexión entre dos hosts?

Para establecer conexión el host de origen envía un segmento CONNECTION REQUEST al destino y espera una respuesta CONNECTION ACCEPTED. **Situación: No se recuerda en el destino n° de secuencias para conexiones.**

Solución: Acuerdo de tres vías de Tomlinson de 1975.

Caso de operación Normal • Fijarse en el número de secuencia del segmento de datos enviado.

Solución: Acuerdo de tres vías de Tomlinson de 1975. Caso de segmento CR duplicado con retraso:

Al rechazar el host 1 el intento establecimiento de conexión del host 2,

el host 2 se da cuenta de que fue engañado por un duplicado con retardo y abandona la conexión;

así, un duplicado con retardo no causa daño.

Solución: Acuerdo de tres vías de Tomlinson de 1975. Caso de tanto segmento CR como de datos con retraso.

cuando llega el segundo segmento retrasado al host 2,

el hecho de que se confirmó la recepción de z en lugar de y indica al host 2 que este también es un duplicado viejo.

Establecimiento de una conexión TCP

El n° de secuencia inicial de una conexión no es 0.

Se usa un esquema basado en reloj con un pulso de reloj cada 4 sec.

Al caerse un host, no podrá reiniciarse durante el tiempo máximo de paquete

para asegurar que no haya paquetes de conexiones previas vagando por Internet.

Campos del encabezado TCP para el establecimiento de conexiones

SYN se usa para establecer conexiones.

Solicitud de conexión: SYN = 1 y ACK = 0.

La respuesta de conexión sí lleva una confirmación de recepción, por lo que tiene SYN = 1 y ACK = 1.

En TCP las conexiones usan el acuerdo de 3 vías

Para establecer una conexión, el servidor, espera pasivamente una conexión entrante ejecutando LISTEN y ACCEPT y

especificando cierto origen o bien nadie en particular.

En el lado del cliente ejecuta CONNECT

la cual envía un segmento TCP con el bit SYN encendido y el bit ACK apagado, y espera una respuesta.

Al llegar el segmento al destino, la ETCP allí revisa si hay un proceso que haya ejecutado un LISTEN en el puerto indicado en el campo puerto de destino.

Si no lo hay envía una respuesta con el bit RST encendido para rechazar la conexión.

Si algún proceso está escuchando en el puerto ese proceso recibe el segmento TCP entrante y puede entonces aceptar o rechazar rechazar la conexión; si la acepta se envía un segmento de ack.

La secuencia de segmentos TCP enviados en el caso normal se muestra en la Figura siguiente.

Liberación de Conexiones

¿Cómo hacer un protocolo para liberación de conexiones?

Podríamos pensar en un protocolo en el que:

el host 1 dice "ya terminé ¿terminaste también?"; – Si el host 2 responde "Ya terminé también. Adiós", la

"Ya terminé también. Adiós", la conexión puede liberarse con seguridad. liberarse con seguridad.

un protocolo así no siempre funciona.

problema de los dos ejércitos

No existe un protocolo que resuelva el problema de los dos ejércitos:

¿Cómo hacer para que no ocurran las conexiones abiertas a medias?

Solución 1: Pudimos haber evitado este problema no permitiendo que el emisor se diera por vencido tras N reintentos, sino obligándolo a seguir insistiendo hasta recibir una respuesta.

Problema

si se permite que expire el temporizador en el otro lado, entonces el emisor continuará eternamente, pues nunca aparecerá una respuesta.

¿Cómo acabar con las conexiones abiertas a medias de los dos lados?

Solución 2: Una manera de matar conexiones abiertas a medias es:

si no ha llegado ningún segmento durante una cierta cantidad de segundos al host 2, se libera automáticamente la conexión en el host 2.

Luego el host 1 detectará la falta de actividad y también se desconectará.

Esta solución también resuelve el caso que la red “se rompió” y los hosts ya no pueden conectarse.

¿Cómo se puede implementar la solución anterior?

Es necesario que cada ET tenga un temporizador que se detenga y se reinicie con cada envío de un segmento.

no se puede garantizar absolutamente que cuando se libera una conexión no ocurre pérdida de datos. Pero si se puede limitar mucho que esto suceda.

La liberación simétrica:

Cada parte se cierra por separado, independientemente de la otra

Una de las partes emite un DISCONNECT porque ya no tiene más datos por enviar y aun está dispuesta a recibir datos de la otra parte.

Una

Una conexión se libera cuando ambas partes han emitido una primitiva DISCONNECT.

Es ideal cuando cada proceso tiene una cantidad fija de datos por enviar y sabe con certidumbre cuándo los ha enviado.

En otras situaciones la determinación de si se ha efectuado o no todo el trabajo o si debe terminarse o no la conexión no es tan obvia.

TCP trabaja con liberación simétrica.

Liberación de una conexión TCP

Campo usado por TCP para liberación de conexiones:

FIN: especifica que el emisor no tiene más datos que transmitir.

Tras cerrar una conexión, un proceso puede continuar recibiendo datos indefinidamente.

Ambos segmentos, SYN y FIN, tienen n° de secuencia, y por tanto, tienen la garantía de procesarse en el orden correcto.

Resumen

Para liberar una conexión cualquiera de las partes puede enviar un segmento TCP con el bit FIN establecido, lo que significa que no tiene más datos por transmitir, pero todavía puede recibir datos del otro lado.

Al confirmarse la recepción del FIN, ese sentido se apaga (el receptor del ack no va a enviar más). • Sin embargo puede continuar un flujo de datos indefinido en el otro sentido.

Cuando ambos sentidos se han apagado, se libera la conexión.

Capa de Transporte Generalidades e introducción a TCP

Propósito de la capa de transporte

provee comunicación lógica entre procesos de aplicación que ejecutan en diferentes sistemas finales.

La CT se implementa (salvo alguna excepción que veremos más adelante) solo en los sistemas finales.

Comunicación lógica:

como si los hosts ejecutando los procesos estuvieran directamente conectados.

Para mejorar la calidad los servicios de la CR.

P.ej: retransmisiones de paquetes perdidos en redes no orientadas a la conexión.

P.ej: cuando hay congestión en la red, regulando de manera fina la variación de la tasa de transmisión de paquetes de los hosts.

La CT se ejecuta por completo en los hosts/sistemas finales.

Entidad de transporte (ET) = software/hardware de la CT.

Problemas que soluciona la capa de transporte

Uso de temporizadores y las retransmisiones de paquetes. El direccionamiento explícito de los destinos. destinos.

Uso de búferes y control de flujo. Evitar congestionar la red poniendo demasiados paquetes en ella.

Cuando la CR pierde paquetes, la CT puede solucionarlo.

Segmento = unidad de datos del protocolo de transporte

Confirmaciones de recepción de paquetes enviados.

Tipos de paquetes que deben ser confirmados.

paquete de datos

paquetes con información de control.

Para la entrega ordenada de segmentos al host de destino se puede: Numerar los segmentos a enviar

(usando números de secuencia) – respetando el orden del flujo de datos recibido de la capa de aplicación.

Usar para cada número de segmento enviado un temporizador de retransmisiones.

Mandar confirmaciones de recepción (ACK) para segmentos recibidos correctamente.

Si expira el temporizador de un segmento sin recibir el ACK, retransmitir el segmento correspondiente.

Los segmentos recibidos son re-ensamblados en orden y entregados a la capa de aplicación del receptor.

TCP

TCP (protocolo de control de transmisión)

Meta: proporcionar

proporcionar un flujo de bytes confiable de extremo a extremo a través de una interred no confiable.

se adapta dinámicamente a las propiedades de la interred y se sobrepone a muchos tipos de fallas.

ET TCP (ETCP).

Usaremos la palabra TCP para referirnos: a veces a la ETCP y a veces al protocolo TCP.

Problemas que resuelve TCP:

Retransmisión de paquetes:

uso de números de secuencia, confirmaciones de recepción y temporizadores.

Fijar la duración de temporizadores de retransmisiones

Manejo de conexiones entre pares de procesos Direccionamiento Control de congestión Control de flujo

Una ETCP acepta flujos de datos a transmitir de procesos locales, – Cada flujo de datos se divide en fragmentos que no excedan los 64 KB llamados segmentos, – y se envía cada segmento dentro de un datagrama IP.

El servicio TCP se obtiene al hacer que tanto el servidor como el cliente creen sockets. Dirección de un socket = IP + Puerto

Para obtener el servicio TCP se debe establecer una conexión explícitamente entre el socket en la máquina emisora y uno en la máquina receptora.

Un socket puede usarse para múltiples conexiones al mismo tiempo: dos o más conexiones pueden terminar en el mismo socket.

Las conexiones se identifican mediante los identificadores de sockets de los dos extremos: (socket1, socket2).

Cada byte de un flujo de datos a enviar en una conexión TCP tiene su propio número de secuencia de 32 bits. Esto impone un límite en el tamaño de un flujo de datos.

números de secuencia?

para confirmaciones de recepción y para otros asuntos según veremos.

La ETCP emisora y la receptora intercambian datos en forma de segmentos. **Segmento = encabezado TCP ++ (0 o más bytes) de datos.**

Límites que restringen el tamaño de un segmento

Cada segmento, debe caber en la carga útil de 65.515 bytes del IP.

Cada red tiene una unidad máxima de transferencia (MTU) y cada segmento debe caber en la MTU.

usualmente de 1500 1500 bytes

La capa de red

no proporciona ninguna garantía de que los datagramas se entregarán de manera apropiada,

tampoco garantiza que se entregarán.

TCP:

Si un datagrama se recibe correctamente se confirma su recepción.

Si no se confirma la recepción de un datagrama luego de un intervalo de tiempo entonces se debe retransmitir.

Corresponde a TCP terminar los temporizadores y retransmitir los datagramas conforme sea necesario.

Problema: Los datagramas que llegan podrán hacerlo en el orden incorrecto.

Esto sucede cuando se trabaja con redes de datagramas.

la capa de aplicación receptor procesar mensajes

en orden

Solución: Corresponde a TCP reensamblar los mensajes en la secuencia apropiada.

Cuando un transmisor envía un segmento, también inicia un temporizador.

Cuando llega el segmento a destino, la ETCP receptora devuelve un segmento contiene un número de confirmación de recepción de recepción igual al siguiente número de secuencia que espera recibir.

Si el temporizador expira antes de llegar el ack, el emisor envía de nuevo el segmento.

Problemas por TCP

Pueden llegar segmentos fuera de orden,

los bytes 3072-4095 podrían llegar pero no enviarse el ack, porque los bytes 2048-3071 no han aparecido aun.

Habrá que esperar a veces antes de entregar segmentos a la capa de aplicación.

Situación: retransmisiones

incluir rangos de bytes diferentes a los de la transmisión original.

porque:

Hay nuevos datos pare enviar y se los puede mandar.

Se requiere una administración cuidadosa para llevar el control de los bytes que se han recibido correctamente en un momento determinado

Segmentos

TCP

Encabezado fijo de 20 bytes

Opciones de encabezado en palabras de 32 bits

Datos opcionales

Los segmentos sin datos se usan para acks y mensajes de control.

Puerto de origen y puerto de destino:

16 b cada uno.

puerto más la dirección IP host

forman un punto terminal único de 48 b.

Los puntos terminales de origen y de destino en conjunto identifican la conexión.

El encabezado del segmento TCP

número de secuencia

es un número de byte en el flujo de bytes transmitido y

corresponde al primer byte en el segmento.

Tiene 32 b de longitud.

número de confirmación de recepción

indica el siguiente byte esperado del flujo de bytes a transmitir.

Tiene 32 b de longitud.

ACK se establece en 1 para indicar que el n° de confirmación de recepción es válido.

ACK = 0

el segmento no contiene una confirmación de recepción.

longitud del encabezado TCP: N° de palabras de 32 bits en el encabezado TCP.

Problemas de tener segmentos duplicados retrasados y su resolución

Comparación de segmentos

Sí, por ejemplo, cuando se pierde un ack y un segmento se retransmite.

También cuando por congestión un segmento se demora, expira su temporizador y el segmento se retransmite.

No se pueden entregar segmentos duplicados a la capa de aplicación.

¿Cómo hacer para saber eficientemente si dos segmentos son diferentes o no? Ineficiente es comparar los segmentos bit a bit, **Idea: Numerar los segmentos con números de secuencia.**

Entonces paquetes con n° de secuencia diferentes son distintos.

Esta idea funcionaría bien si tenemos n° de secuencia de tamaño arbitrario.

¿Los números de secuencia pueden ser de tamaño arbitrario?

No, porque queremos que los segmentos tengan longitud máxima.

Por lo tanto el espacio de números de secuencia es finito;

¿La idea de solo usar espacio de secuencia finito y numerar segmentos con n° de secuencia funciona bien?

no

Porque números de secuencia alcanzan el máximo y vuelven a reiniciarse y aumentan hasta alcanzar el x de nuevo.

¿por qué sucede eso?

El espacio de secuencia no es suficientemente grande.

El duplicado retrasado permanece demasiado en la red.

Duplicados retrasados

Sucede la siguiente situación: un segmento S con n° de secuencia x queda demorado debido a que la red está congestionada.

El temporizador de retransmisiones asociado a S expira y se retransmite S.

El protocolo de enrutamiento cambia las rutas y la retransmisión de S llega rápido a destino.

Pero aun quedó en la red un duplicado retrasado de S (de n° de secuencia x).

Ese duplicado retrasado de S más adelante llega a destino generando problemas.

- Este tipo de problemas es tan serio que debe ser evitado.

¿Cómo encarar problemas de duplicados retrasados?

Idea: Asegurar que ningún paquete viva más allá de T sec. (tiempo de vida de paquete) – Esto se refiere a paquetes de datos, retransmisiones de ellos y a confirmaciones de recepción. – Eliminar paquetes viejos que andan dando vueltas por ahí.

¿Cómo resolver el problema de duplicados retrasados dentro de una conexión?

Asumiendo que T es el tiempo de vida de paquete, el origen etiqueta los segmentos con n° de secuencia que no van a reutilizarse dentro de T sec.

¿Cómo evitar que duplicado retrasado que pasa de una conexión a otra genere problema?

Como al establecer una conexión se usan segmentos, una conexión debería tener un N° inicial de secuencia con el que comienza a operar.

En los dos ejemplos últimos sucede lo siguiente:

Hay conexión nueva que rápidamente llega a n° de secuencia x para la cual hay duplicado retrasado que genera problemas.

Idea de solución: hay que escoger como número inicial de secuencia de la conexión nueva un n° de secuencia que haga imposible o improbable que el duplicado retrasado de n° de secuencia x genere problemas.

Además se mantiene dentro de una conexión que el origen etiqueta los segmentos con n° de secuencia que no van a reutilizarse dentro de T sec (tiempo de vida del paquete).

Implementación 1

al crear una nueva conexión cada extremo genera un n° de secuencia de 32 bits aleatorio que pasa a ser el número inicial de secuencia para los datos enviados. – Alguna implementación de TCP usa esta solución.

¿Por qué tiende a funcionar?

La probabilidad de que un paquete duplicado retrasado genere problemas en una conexión siguiente es baja debido a la elección aleatoria del número inicial de secuencia de la conexión siguiente.

vincular n° de secuencia de algún modo al tiempo y para medir el tiempo usar un reloj. Cada host tiene un reloj de hora del día. Los relojes de los hosts no necesitan ser sincronizados;

se supone que cada reloj es un contador binario que se incrementa a si mismo en intervalos uniformes.

El reloj continua operando aun ante la caída del host