SERIE	DOMANDA RISP. CHIUSA 1	a	b	c	d
1	**Qual è la rappresentazione in codice binario del numero 43?**	a. 111010	b. 101010	**c. 101011**	d. 101110
2	**Se un computer non è collegato in rete, è comunque opportuno utilizzare un antivirus?**	a. Sì, perché l'antivirus serve anche a compattare i file sul disco fisso	b. No, è totalmente inutile	c. No, salvo che non vi siano frequenti cadute della tensione di alimentazione	**d. Sì, perché vi sono altre possibili fonti di infezioni**
3	**Cos'è la webmail?**	a. E' il sistema di gestione della Posta Elettronica Certificata (PEC)	**b. E' la posta elettronica gestita direttamente su un sito internet**	c. E' un messaggio di posta che contiene una pagina web	d. E' un dispositivo che consente di inviare messaggi video
4	**Di cosa è acronimo "GUI"?**	a. General Unit Interface	**b. Graphical User Interface**	c. Games Unit Interface	d. Graphical Universal Interface
5	**Cosa è lo SPID**	a.Service Protocol Identifier	**b.Sistema Pubblico di Identità Digitale**	c.E' la Firma Digitale	d.E' la Carta d'identità
6	**Se parlo di IMAP mi riferisco a:**	a: Protocollo di compressione audio	b: Protocollo di compressione video	**c: Protocollo di posta elettronica**	d: Protocollo di gestione mappe
7	**Skype è un software open source?**	a. Si	b. solo nella versione client	c. Solo mediante Facebook	**d. No**
8	**Cosa significa selezione passante:**	a. Si può chiamare all'esterno senza passare per il centralino	b. Per ogni chiamata si deve passare dal centralino	c. Si può passare la linea da un interno ad un altro dello stesso centralino	**d. Si può chiamare un interno direttamente dall'esterno senza passare per il centralino**
9	**Possono esistere 2 indirizzi di posta elettronica uguali?**	**a. No**	b. Si	c. Si ma la password deve essere diversa	d. Si ma deve corrispondere a due diversi proprietari
10	**Ha senso installare un firewall di frontiera tra:**	a. Tra due computer	b. Tra switch di rete locale e device finale dell'utente	c. Tra reti di piani diversi di un edificio	**d. Rete geografica e rete locale**

11	**Come trasformo un documento digitale in un PDF?**	**a. Uso una stampante virtuale PDF**	b. Uso Acrobat reader	c. Stampo e poi scansiono in formato jpg	d. Non è possibile
12	**Cos'è EPUB?**	a. E' una componente Hardware dei dispositivi mobili di ultima generazione	b. E' un documento pieno di immagini	**c. E' il formato standard di pubblicazione degli ebook**	d. E' un vecchio formato di file europeo
13	**Cosa è MOBI?**	a. MObility Base International	**b. Il formato standard degli ebook Kobo**	b. Il formato standard del progetto Bibliotecario Mobi	d. Microsoft Office Balancing Intent
14	**In excel il valore ###### in una cella significa che:**	a: la cella contiene una funzione sbagliata	b: il valore della cella si riferisce ad un'operazione sbagliata	**c: Il valore della cella è più lungo della larghezza della colonna**	d: la cella contiene dei numeri e delle lettere
15	**Un apparecchio telefonico VOIP è alimentato:**	a. Dalla centrale telefonica	b. Dal cavo USB	**c. Dallo switch**	d. Dal pc a cui è collegato
16	**Come creo una macro in word?**	**a. registrando le azioni oppure scrivendo del codice nell'editor**	b. posso solo registrare le azioni che compio	c. devo scriverla in excel e poi esportarla in word	d.devo avere un editor esterno
17	**Cosa è AZW?**	a. Amazon World	b. Archive Zip Word	**c. Un file per Kindle**	d. Un Processore alternativo a Intel
18	**Zoom è un software open source?**	a. solo nella versione su browser	b. solo nella versione client	**c. No**	d. Si
19	**“Shareware” significa:**	a. Software gratuito	b. Software che non prevede diritti d’autore	**c. Software concesso temporaneamente in uso gratuito.**	d. Software non commerciale
20	**Cosa è il Freeware?**	a. un componente del sistema operativo	**b. Un programma gratuito**	c. un componente hardware	d. un applicativo per la firma digitale

21	**se carico un foglio excel in google drive e lo trasformo in un documento google sheet**	a. potrò fare esattamente tutto quello che facevo in excel	b. non funzionerà,	**c. avrò a disposizione un sottoinsieme delle funzioni excel**	d. posso solo leggere i file, non modificarli
22	**Esiste un equivalente di microsoft powerpoint nella google suite?**	a. NO	b. sì, è google powerpoint	**c. sì, è google presentazioni**	d. google doc
23	**Cos'è una macro:**	a. Serve solo per impostare le dimensioni dei caratteri	b. Un pulsante più grande degli altri	c. La lente d'ingrandimento del programma Accessibility	**d. Contiene una serie di comandi registrati**
24	**Cosa è un PDF/A?**	**a. PDF per la conservazione a lungo termine**	b. PDF da mettere in cima alla lista	c. PDF Anonimo	d. PDF criptato
25	**Cosa è una Firma Digitale?**	E' la Carta d'Identità Elettronica	E' la scansione della Firma di qualcuno.	E' lo SPID	**E' un Certificato d'Identità emesso da una CA**
26	**Quale di questi metodi non produce una Firma Digitale?**	a. Utilizzare una smartCard e un lettore	b. Utilizzare un cellulare	c. Utilizzare una tavoletta grafica	**d. Scansionare la mia firma e incollarla sul documento**
27	**Cosa significa la sigla PDF?**	a. Personal Document File	b. Post Department Format	**c. Portable Document Format**	d. Protocol Density Firmware
28	**Se parlo di POP mi riferisco a:**	a: Protocollo di trasmissione video	b: Protocollo di comunicazione deli web	c: Protocollo di compressione dei file	**d: Protocollo di posta elettronica**
29	**Cosa è un file con estensione p7m?**	Si tratta di un file compattato con 7zip	**E' un file firmato digitalmente**	è un PDF a cui viene aggiunto un 7zip	E' un file prodotto da Photoshop
30	**Come gestire il doppio schermo da tastiera:**	a. Ctrl + S	b. Ctrl + P	c. Tasto "Windows " + S	**d. Tasto "Windows " + P**

SERIE	DOMANDA RISP. CHIUSA 2	a	b	c	d
1	**Cosa significa DNS?**	a. Database Network System	**b. Domain Name System**	c. Digital Network Server	d. Discreet National System
2	**Cos'è il firmware**	a. un componente del sistema operativo	**b. un insieme di istruzioni per funzioni base**	c. un componente hardware	d. un applicativo per la firma digitale
3	**La sigla M.2 si riferisce a:**	a: Il connettore dei mouse	b: un connettore per l'audio	c: un connettore per i monitor full HD	**d: un'interfaccia di espansione**
4	**Una copia di backup del disco è:**	a: una copia dei driver del pc	**b: una copia di sicurezza dei dati**	c: una copia solo del sistema operativo	d: una copia solo dei dati personali
5	**Per trasmettere l’audio dal pc al monitor dotato di casse utilizzo :**	**a. Cavo HDMI**	b. Cavo DVI-A	c. Cavo VGA	d. Cavo FTP
6	**Cosa significa VDI?**	a. Venice Digital Infrastructure	b. Video Digital Infrastructure	**c. Virtual Desktop Infrastructure**	d. Visual Digital Interface
7	**Cos’è il tempo di latenza?**	a. l'inverso della frequenza di lavoro di un microprocessore	b. l'inverso della frequenza di aggiornamento di un monitor	c. il tempo che intercorre tra l'accensione di un PC e il caricamento in memoria del sistema operativo	**d. il ritardo tra l'istante in cui si invia una richiesta al microprocessore e l'istante in cui si ottiene il risultato in uscita**
8	**Cosa significa VHD?**	a. Video High Definition	b. Venice Historical Data	c. Visual Head-up Display	**d. Virtual Hard Drive**
9	**Cosa identifica la risoluzione di un monitor?**	a. Hertz	**b. Pixel**	c. Byte	d. Baud
10	**Ho una webcam integrata nel monitor di un pc desktop che non è rilevata dal pc. Controllo:**	a. Cavo VGA	b. Cavo DVI A	**c. Cavo USB**	d. Cavo audio

11	**Che capacità totale presenta un sistema di N dischi di capacità singola C organizzati in RAID 1?**	**a. C**	b. C x N	c. C x 5	d. C x (N-1)
12	**Quale componente hardware permette di scrivere dati sui cd-r (diversi dai cd-rw)?**	a. nessuno, non si possono scrivere dal computer	**b. masterizzatore**	c. Lettore/Cd/DVD	d. registratore
13	**Il pc windows non conserva data e ora, controllo:**	**a. Tensione batteria interna**	b. Tensione alimentatore 5V	c. Tensione alimentatore 12V	d. Tensione scheda grafica
14	**Cosa è il case?**	a. E' l'insieme dei cavi che collegano il computer alla rete	b. Sono dei pacchetti di protocolli	c. Un virus	**d. L'involucro che contiene i componenti di un computer**
15	**il connettore VGA**	a. Ha una levetta di blocco	**b. Ha 15 pin**	c. Ha 25 pin	d. generalmente è di colore bianco
16	**Un pc funziona senza il sistema operativo?**	a. si, i programmi sono più veloci	**b. No**	c. Si, ma non visualizzano le immagini	d. Dipende dalla rete
17	**Quanti dischi al minimo deve avere un RAID 5?**	a. 1	b. 2	**c. 3**	d. 4
18	**Quanti dischi difettosi sopporta un RAID 1 costituito da N dischi?**	a. 0	b. N - 2	c. N/2	**d. N - 1**
19	**Quanti dischi difettosi sopporta al massimo un RAID 5?**	a. 0	**b. 1**	c. nessuno	d. N - 1
20	**Quanti dischi al minimo deve avere un RAID 1?**	a. 1	**b. 2**	c. 3	d. 4

21	**Qual è il compito della ALU?**	**a. Eseguire le operazioni di calcolo.**	b. Controllare eventuali errori dell'hardware.	c. Controllare il funzionamento della memoria.	d. Eseguire le operazioni di input e output.
22	**Quali tra le seguenti è una soluzione adottabile per migliorare le prestazioni di un computer?**	a. Aumentare la ROM;	**b. Aumentare la capacità del disco rigido;**	c. Installare un firewall;	d. Cambiare la batteria del computer;
23	**Quale tra i seguenti componenti di un PC permette di definire il supporto da utilizzare per effettuare il boot del sistema?**	a. RAM	b. Sistema operativo	**c. BIOS**	d. Controller del disco rigido
24	**Che cosa è una virtual private network?**	a. È un tipo di rete ad anello che utilizza esclusivamente collegamenti wireless	b. È una sottorete nell'ambito di una LAN	c. È una rete tra computer che utilizzano il sistema operativo Linux	**d. È una rete privata instaurata tra sistemi che utilizzano un mezzo di trasmissione pubblico**
25	**Quale di queste interfacce non gestisce un disco rigido?**	a: ATA	b. SATA	c. M.2	**d. M.3**
26	**Che capacità totale presenta un sistema di N dischi di capacità singola C organizzati in RAID 5?**	a. C	b. C x N	c. C x 5	**d. C x (N-1)**
27	**Il connettore DisplayPort**	**a. Ha una levetta di blocco**	b. Ha 15 pin	c. Ha le viti di fissaggio	d. Ha il cavo audio separato
28	**il connettore HDMI**	a. Ha una levetta di blocco	**b. Ha 19 pin**	c. Ha le viti di fissaggio	d. Ha il cavo audio separato
29	**E' possibile utilizzare un computer senza il disco rigido?**	a. no	**b. si, avviandolo da una memoria esterna**	c. si, funziona senza problemi ma non salva i dati	d. Non è possibile
30	**La misura dello schermo di un PC in pollici si riferisce:**	a. Larghezza dello schermo	b. Numero di icone disposte sul desktop	**c. Diagonale dello schermo**	d. Altezza dello schermo

SERIE	DOMANDA RISP. CHIUSA 3	a	b	c	d
1	**Come viene chiamata la metodologia di trasmissione che invia un messaggio a tutti gli utenti di una rete?**	a. Radiocast	b. Multicast	c. Omnicast	**d. Broadcast**
2	**Una connessione è di tipo half duplex se:**	**a. La trasmissione è possibile in una sola direzione per volta**	b. I dati possono essere inviati in una sola direzione	c. I dati possono essere inviati simultaneamente nelle due direzioni	d. La trasmissione è monodirezionale
3	**Per permettere a 3 PC di comunicare fra di loro in una stessa lan devo utilizzare:**	a. Router	b. Bridge	**c. Switch**	d. Modem
4	**Qual e' il core (in µm) in una fibra ottica monomodale?**	a. 45-50	b. 25-28	c. 2-4	**d. 8-9**
5	**Active Directory è un insieme di protocolli per:**	a. Gestire le cartelle di una chiavetta USB	b. Streaming video	c. Mantenere attivo e reattivo il filesystem	**d. Autenticazione utente e condivisione risorse**
6	**Un connettore RJ45 ha:**	**a: 8 poli**	b: 6 poli	c:12 poli	d: 2 poli
7	**Qual e' la combinazione rapida da tastiera per I comandi "copia", "incolla" e "taglia" ?**	a. CTRL+C, CTRL+V, CTRL+T	b. CTRL+C, CTRL+X, CTRL+V	**c. CTRL+C, CTRL+V, CTRL+X**	d. CTRL+C, CTRL+I, CTRL+T
8	**La massima lunghezza di un segmento in una rete 100baseT in rame è:**	a. 10 metri	b. 50 metri	**c. 100 metri**	d. 1000 metri
9	**Premendo il tasto Fine in un documento Word che cosa succede?**	a. il cursore si sposta a fine documento	b. Cancello il documento	**c. Il cursore si sposta a fine riga**	d. Viene scritta la parola FINE sul documento
10	**Cosa identifica il termine LAN?**	a. Un programma per la gestione delle di immagini	**b. Una rete locale (Local Area Network)**	c. Un componente della CPU che memorizza l'ultima istruzione	d. Una rete geografica (Large Area Network)

11	**Quale tool viene usato per preparare un disco fisso per l'uso?**	a. Cleandisk	**b. Fdisk**	c. Chldsk	d. Fisdisk
12	**Come si misura il fascio di luce emesso dal videoproiettore**	a: si misura in candele	b: si misura in lux	c: si misura in candele per mq	**d: si misura in ANSI lumen**
13	**Per utilizzare la webcam integrata nel monitor:**	a. Collego il monitor al PC con un cavo FTP	b. Collego il monitor al PC con un cavo DVI	c. Collego il monitor al PC con un cavo VGA	**d. Collego il monitor al PC con un cavo HDMI**
14	**Quale dei seguenti può essere un MAC address Ipv4 corretto?**	a. 01:02:01:2C:4B	b. 07:01:02:01:2C:4B:2C	**c. 07:01:02:01:2C:4B**	d. 4B:5F:2C:ZZ:02:2B
15	**Il compito di un DNS è?**	a. Reindirizzare l'indirizzo IP in un altro indirizzo IP	b. Tradurre l'Indirizzo IP in un indirizzo fisico	c. Entrambi	**d. Tradurre l'indirizzo testuale in un indirizzo IP**
16	**Cos'è un server DNS?**	a. Un host di siti web	b. Un fornitore di servizi internet	**c. Un database che fornisce gli indirizzi IP in base all'URL**	d. Un firewall che controlla gli indirizzi ip in arrivo
17	**Cosa è lo scandisk?**	a. Un antivirus	b. Utilità di sistema che corregge un errata connessione a Internet	**c. Utilità di sistema che controlla l'hard disk o il floppy disk cercando eventuali errori e provvede a correggerli**	d. Utilità di sistema che formatta il disco rigido.
18	**Cos'è il MAC Address?**	a. L'indirizzo TCP/IP del computer	b. L'indirizzo assegnato dal server DHCP	c. Il protocollo di indirizzamento dei computer Mac	**d. l'indirizzo fisico di una scheda di rete**
19	**Un sistema operativo monotasking è in grado di:**	a. Far accedere un solo utente alla volta	b. Gestire audio monocanale	c. Far accedere un solo utente per volta ma gestire più utenti	**d. Eseguire un solo processo alla volta**
20	**Quale comando consente di vedere e modificare I file di registro di Windows?**	a. Sysedit	**b. Regedit**	c. Autoedit	d. Cfgedit

21	**Per terminare il processo 4567 in linux devo digitare:**	**a. kill 4567**	b. shutdown -h 4567	c. ls-l 4567	d. term 4567
22	**I file che vengono eliminati anche dal "cestino" possono essere recuperati?**	a. Si sempre	b. Si, utilizzando il comando File Regedit	**c. Si, ci sono buone possibilità, ma usando programmi specifici**	d. Assolutamente no
23	**A cosa serve il comando ctrl + f in una pagina web?**	a. a copiare il testo selezionato	b. ad aumentare la dimensione dei caratteri	c. a incollare l'immagine copiata	**d. a cercare un testo nel testo**
24	**Cosa utilizzo per trasmettere i segnale video HDMI alla distanza di 50 mt con problemi di interferenza elettromagnetica**	a: utilizzo un cavo HDMI di ottima qualità	b: utilizzo un extender HDMI -cavo UTP	**c: utilizzo un extender HDMI - fibra ottica**	c: utilizzo un trasmettitore HDMI-DVI
25	**Come è possibile visualizzare le proprietà di una cartella in Windows?**	a. Si clicca con il tasto sinistro del mouse sulla cartella e poi con il tasto destro su Proprietà	**b. Si clicca con il tasto destro del mouse sulla cartella e poi con il tasto sinistro su proprietà**	c. Si clicca con il tasto sinistro del mouse su Visualizza	d. Si clicca con il tasto sinistro del mouse sulla cartella
26	**Cosa significa che una versione di Windows è fuori supporto?**	a. Il PC si accende ma non si avvia il sistema operativo	**b. la versione non viene più aggiornata e quindi di comporta rischi di funzionalità e sicurezza**	c: la versione non viene più aggiornata, ma il computer continuerà a funzionare regolarmente senza nessun	d. versione non viene più aggiornata e il pc si bloccherà al prossimo aggiornamento
27	**L'hub è un dispositivo che mette in collegamento I PC di una rete di tipo?**	a. Wi-fi	b. WAN	**c. LAN**	d. Nessuna
28	**Con quale impedenza deve essere terminato un cavo 100baseT?**	a. Con una terminazione RJ11 da 0 Ohm	b. Con una terminazione BNC da 50 Ohm	**c. Non serve nessuna impedenza**	d. Con una terminazione RJ45 da 75 Ohm
29	**Cos'è l'effetto Larsen?**	a: E' un effetto legato alla variazione della luminosità del videoproeittore	**b: E' un effetto legato al rientro nel microfono dei suoni che escono dai diffusori acustici**	c: E' un effetto legato alla variazione della lunghezza d'onda emessa da una fonte in movimento	d: E' un effetto legato alla variazione di lunghezza della luminosità dei videoproiettori
30	**Lo streaming è una trasmissione:**	a. Alternata	b. Bidirezionale	**c. Monodirezionale**	d. Asincrona

SERIE	DOMANDA RISP. CHIUSA 4 legislazione	a	b	c
1	**Domanda: il rappresentante degli studenti nel Nucleo di Valutazione è designato:**	dal Consiglio di Amministrazione	da nessuno, nel Nucleo non è previsto un rappresentante degli studenti	**dall'Assemblea dei Rappresentanti degli Studenti**
2	**Domanda: il Rettore è rieleggibile?**	Si, senza alcuna limitazione	Si, ma una sola volta.	**No**
3	**Domanda: l'Assemblea dei rappresentanti degli studenti è un organo di:**	gestione	**Consuntivo e di garanzia**	controllo
4	**Domanda: il Senato Accademico dura in carica:**	3 anni accademici	2 anni accademici	**3 anni accademici, a eccezione dei rappresentanti degli studenti che durano in carica 2 anni accademici**
5	**DOMANDA: IL CONSIGLIO DI AMMINISTRAZIONE È UN ORGANO:**	di controllo	consultivo	**di governo**
6	**Domanda: Il Collegio dei Revisori dei Conti è un organo di:**	gestione	**controllo**	di garanzia
7	**Domanda: nelle votazioni per l'elezione del Rettore la Commissione Elettorale viene costituita dal:**	Direttore Generale	Dirigente Ufficio Affari Legali	**Decano**
8	**DOMANDA: IL RETTORE È:**	**il rappresentante legale dell'Ateneo**	il rappresentante del personale docente di Ateneo	il rappresentante del personale non docente di Ateneo
9	**Domanda: il Rettore dura in carica:**	4 anni	**6 anni**	5 anni
10	**Domanda: Il Direttore Generale ha diritto di voto nelle sedute del Senato Accademico:**	Si	**No**	Solo negli argomenti che riguardano l'amministrazione dell'Ateneo

11	**Domanda: nelle votazioni per l'elezione del Rettore, in seconda e/o terza votazione il Rettore è eletto:**	a maggioranza assoluta degli aventi diritto	**a maggioranza assoluta dei votanti**	a maggioranza, purché venga raggiunto il quorum del 50% degli aventi diritto al voto.
12	**Domanda: Le votazioni per l'elezione del Rettore vengono indette da:**	**Decano**	Direttore Generale	Rettore uscente
13	**Domanda: Il difensore degli studenti è un organo di:**	gestione	controllo	**Consuntivo e di garanzia**
14	**Domanda: Il Senato Accademico è presieduto dal:**	**Rettore**	Docente ordinario più anziano in ruolo	Direttore Generale
15	**DOMANDA: IL SEGRETARIO DEL CONSIGLIO DI AMMINISTRAZIONE È:**	il Rettore	**il Direttore Generale (o suo delegato)**	il Revisore dei Conti
16	**Domanda: nelle votazioni per l'elezione del Rettore, nella prima votazione il Rettore è eletto**	a maggioranza dei votanti	**a maggioranza assoluta degli aventi diritto**	a maggioranza, purché venga raggiunto il quorum del 75% degli aventi diritto al voto
17	**Domanda: i membri del Nucleo di Valutazione sono designati dal:**	Rettore;	Senato Accademico;	**Consiglio di Amministrazione**
18	**DOMANDA: IL CONSIGLIO DI AMMINISTRAZIONE È PRESIEDUTO DAL:**	**Rettore**	Direttore Generale	Decano di Ateneo
19	**Domanda: Il Direttore di Dipartimento è nominato con decreto del:**	**Rettore**	decano	consiglio di dipartimento
20	**Domanda: fanno parte del Consiglio di Amministrazione anche rappresentanti degli studenti?**	**Si**	No	Si, ma solo per discutere l'importo delle tasse universitarie

21	**Domanda: l’incarico di Direttore Generale viene conferito dal:**	Rettore;	Senato Accademico	**Consiglio di Amministrazione, sentito il Rettore**
22	**Domanda: Il difensore degli studenti è designato:**	dal Rettore	dal Senato Accademico	**dall’Assemblea dei Rappresentanti degli studenti**
23	**DOMANDA: IL NUCLEO DI VALUTAZIONE È UN ORGANO:**	**di controllo**	consultivo	di governo
24	**Domanda: i componenti del Senato Accademico sono immediatamente rinnovabili?**	Si, senza alcuna limitazione	**Si, una sola volta**	no
25	**DOMANDA: IL SENATO ACCADEMICO È UN ORGANO:**	di controllo	consultivo	**di governo**
26	**Domanda: Il Consiglio di Dipartimento viene convocato dal:**	**Direttore di Dipartimento**	Professore ordinario più anziano in ruolo	Segretario di Dipartimento
27	**Domanda: Fanno parte del Senato Accademico 3 rappresentanti degli studenti eletti dagli studenti:**	di tutti gli studenti;	dei soli corsi corsi di dottorato;	**dei corsi di laurea triennali, magistrali e dei corsi di dottorato**
28	**Domanda: l’incarico di Direttore Generale può essere rinnovato?**	**Si**	Si, ma solo una volta	No
29	**Domanda: fanno parte del Senato Accademico anche rappresentanti del personale tecnico amministrativo e CEL?**	**Si**	No	Si, ma solo per discutere gli argomenti riguardanti il personale tecnico amministrativo e i CEL.
30	**Domanda: il Comitato unico di garanzia per le pari opportunità, la valorizzazione del benessere di chi lavora e contro le discriminazioni dura in carica:**	2 anni	**3 anni**	4 anni

SERIE	DOMANDA APERTA 1		DOMANDA APERTA 2
1	Parla dei tipi di hard disk attualmente presenti sul mercato.		Devo preparare 20 pc per un'aula informatica come procedo
2	Cos'è una Macchina Virtuale? elencare vantaggi e svantaggi e descrivere una procedura per creare Macchine Virtuali		Lezione iniziata, docente chiama nervoso e dice che il proiettore non funziona. Che fai?
3	Cosa valuteresti per acquisto di un pc client da usare per applicazioni di office automation e internet?		Un docente segnala genericamente che un pc in dotazione all'aula non si collega alla rete. Come procedi per dare correttamente una risposta?
4	Ipotizzare le diverse configurazioni in base alle varie tipologie di utenti presenti nel contesto universitario: Docente Scientifico, Docente Umanistico, Personale Amministrativo, Tecnico Informatico		Installeresti un solo browser in un PC client dell'università?
5	Programmi per videoconferenze		Un docente chiama nervoso perchè gli studenti collegati da casa non lo sentono: come procedi?
6	Descrivere la procedure per configurare una stampante di rete in ambiente Windows direttamente disponibile tramite IP		Come collegare un pc a un proiettore: tipi di cavo e di condivisione schermo
7	RAM: come installarle nel PC, caratteristiche, tipologie e parametri prestazionali		Un docente vuole effettuare una videoconferenza con un gruppo in una università estera e chiede di avere sicurezza che non ci siano problemi. Come procedi?
8	Strumenti google - Calendario		Modalità provvisoria: cos'è e come funziona?
9	Descrivere una stampante: a cosa serve, come funziona, quali sono i punti deboli, come si collega ad un computer, quale scegliere in base all'utilizzatore (personale docente, personale tecnico amministrativo, studente)		Quali problemi possono manifestarsi durante una video call?
10	didattica a distanza - strumenti del docente - esempi		Connessione ad un dominio

11	Cos’è un driver, a cosa serve, come si usa, problemi legati ai drivers		Descrivere la parti fondamentali di un cablaggio strutturato.
12	Cosa si intende per VDI: pro e contro		Se un portatile con batteria integrata non si avvia cosa fai?
13	Differenze tra un computer e un terminale: descrivi pregi e difetti in particolare negli ambiti universitari		VPN: descrizione e ambito
14	Elencare almeno tre periferiche di input: come funzionano e per quali compiti sono utili.		Connessione pc-proiettore Tipi di cavo
15	Descrivere la procedure per configurare una stampante di rete in ambiente Windows:collegata ad un server Windows e condivisa		Posso rispondere a un docente che mi chiede: quanti studenti possono connettersi in un’aula con wifi?
16	Vantaggi e svantaggi di una scheda video discreta rispetto alla scheda video integrata.		Tipi di HD: come installarli nel PC, caratteristiche e parametri prestazionali
17	Cita alcune differenze rilevanti tra google sheet(fogli) e ms excel		Un pc Windows collegato in rete via cavo non naviga. Elencare e descrivere brevemente come si pensa di procedere per diagnosticare il problema
18	Bios legacy e UEFI: di cosa si parla?		Connessione pc-proiettore - gestione Audio
19	PEC e webmail		Connessione pc-proiettore Tipi di input
20	PDF e PDF/A		Dovendo consegnare ad un utente un PC già configurato come procederesti per l’installazione del nuovo Computer in sostituzione del precedente?

21	**Firma Digitale: cos’è, a cosa serve, come si usa**		**Una stampante della biblioteca non emette stampe: come procedi?**
22	**Descrivere le varie componenti di un PC.**		**Supponiamo di dover installare un server windows. Quali accorgimenti si devono adottare per garantire la continuità operativa dello stesso sia a livello fisico che logico?**
23	**Esiste un equivalente di powerpoint in google suite?**		**Eduroam: cos'è?**
24	**POP e IMAP: differenze e punti di forza**		**VPN: come connettere un PC a una VPN**
25	**Quali sono i parametri da considerare nell’acquisto di un pc**		**Un browser apre finestre di pubblicità: cosa faccio?**
26	**Microsoft Active Directory**		**Un pc non si accende ed emette dei beep: cosa faccio?**
27	**Cita alcune differenze rilevanti tra google doc e ms word**		**Cosa significa che una versione di Windows è fuori supporto?**
28	**Cosa valuteresti per la scelta di un monitor per pc?**		**Quali accortezze useresti nel caso dovessi cambiare una scheda o un banco di memoria?**
29	**Cosa valuteresti per acquisto di un pc per un laboratorio? Ipotizza uno scenario d’uso e motiva la scelta della configurazione.**		**Come cancellare una connessione wifi in Windows 10, in Mac, in Android?**
30	**Come procederesti se dovessi configurare una cinquantina di PC client (Windows) da utilizzare come postazioni di lavoro?**		**C'è stata un'ondata di acqua alta: cosa verificheresti**

and much more. We will look at these issues in great detail later, but for the moment, Fig. 1-18 briefly summarizes how client-server communication might work over a connection-oriented network.

Given that six packets are required to complete this protocol, one might wonder why a connectionless protocol is not used instead. The answer is that in a perfect world it could be, in which case only two packets would be needed: one for the request and one for the reply. However, in the face of large messages in either direction (e.g., a megabyte file), transmission errors, and lost packets, the situation changes. If the reply consisted of hundreds of packets, some of which could be lost during transmission, how would the client know if some pieces were missing? How would the client know whether the last packet actually received was really the last packet sent? Suppose that the client wanted a second file. How could it tell packet 1 from the second file from a lost packet 1 from the first file that suddenly found its way to the client? In short, in the real world, a simple request-reply protocol over an unreliable network is often inadequate. In Chap. 3 we will study a variety of protocols in detail that overcome these and other problems. For the moment, suffice it to say that having a reliable, ordered byte stream between processes is sometimes very convenient.

1.3.5 The Relationship of Services to Protocols

Services and protocols are distinct concepts, although they are frequently confused. This distinction is so important, however, that we emphasize it again here. A *service* is a set of primitives (operations) that a layer provides to the layer above it. The service defines what operations the layer is prepared to perform on behalf of its users, but it says nothing at all about how these operations are implemented. A service relates to an interface between two layers, with the lower layer being the service provider and the upper layer being the service user.

A *protocol*, in contrast, is a set of rules governing the format and meaning of the packets, or messages that are exchanged by the peer entities within a layer. Entities use protocols to implement their service definitions. They are free to change their protocols at will, provided they do not change the service visible to their users. In this way, the service and the protocol are completely decoupled.

In other words, services relate to the interfaces between layers, as illustrated in Fig. 1-19. In contrast, protocols relate to the packets sent between peer entities on different machines. It is important not to confuse the two concepts.

An analogy with programming languages is worth making. A service is like an abstract data type or an object in an object-oriented language. It defines operations that can be performed on an object but does not specify how these operations are implemented. A protocol relates to the *implementation* of the service and as such is not visible to the user of the service.

Many older protocols did not distinguish the service from the protocol. In effect, a typical layer might have had a service primitive SEND PACKET with the

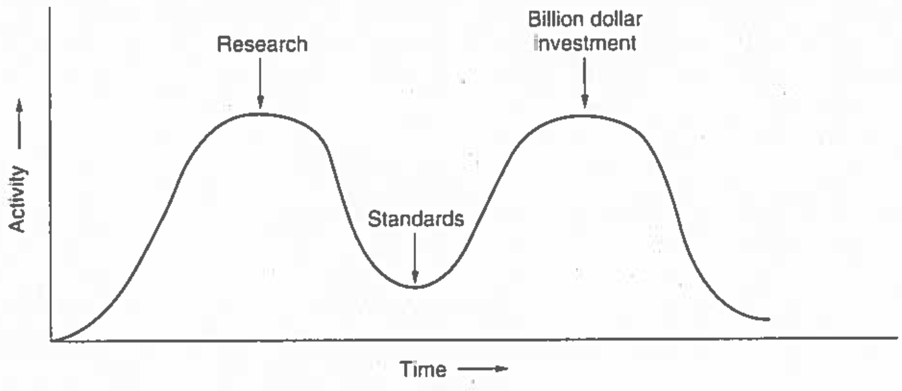


Figure 1-23. The apocalypse of the two elephants.

different ways of doing things that the standards are effectively ignored. If the interval between the two elephants is very short (because everyone is in a hurry to get started), the people developing the standards may get crushed.

It now appears that the standard OSI protocols got crushed. The competing TCP/IP protocols were already in widespread use by research universities by the time the OSI protocols appeared. While the billion-dollar wave of investment had not yet hit, the academic market was large enough that many vendors had begun cautiously offering TCP/IP products. When OSI came around, they did not want to support a second protocol stack until they were forced to, so there were no initial offerings. With every company waiting for every other company to go first, no company went first and OSI never happened.

Bad Technology

The second reason that OSI never caught on is that both the model and the protocols are flawed. The choice of seven layers was more political than technical, and two of the layers (session and presentation) are nearly empty, whereas two other ones (data link and network) are overfull.

The OSI model, along with the associated service definitions and protocols, is extraordinarily complex. When piled up, the printed standards occupy a significant fraction of a meter of paper. They are also difficult to implement and inefficient in operation. In this context, a riddle posed by Paul Mockapetris and cited in (Rose, 1993) comes to mind:

Q: What do you get when you cross a mobster with an international standard?
A: Someone who makes you an offer you can't understand.

In addition to being incomprehensible, another problem with OSI is that some functions, such as addressing, flow control, and error control, reappear again and

That 802.11 is going to cause a revolution in computing and Internet access is now beyond any doubt. Airports, train stations, hotels, shopping malls, and universities are rapidly installing it. Even upscale coffee shops are installing 802.11 so that the assembled yuppies can surf the Web while drinking their lattes. It is likely that 802.11 will do to the Internet what notebook computers did to computing: make it mobile.

1.6 NETWORK STANDARDIZATION

Many network vendors and suppliers exist, each with its own ideas of how things should be done. Without coordination, there would be complete chaos, and users would get nothing done. The only way out is to agree on some network standards.

Not only do standards allow different computers to communicate, but they also increase the market for products adhering to the standard. A larger market leads to mass production, economies of scale in manufacturing, VLSI implementations, and other benefits that decrease price and further increase acceptance. In the following sections we will take a quick look at the important, but little-known, world of international standardization.

Standards fall into two categories: de facto and de jure. **De facto** (Latin for "from the fact") standards are those that have just happened, without any formal plan. The IBM PC and its successors are de facto standards for small-office and home computers because dozens of manufacturers chose to copy IBM's machines very closely. Similarly, UNIX is the de facto standard for operating systems in university computer science departments.

De jure (Latin for "by law") standards, in contrast, are formal, legal standards adopted by some authorized standardization body. International standardization authorities are generally divided into two classes: those established by treaty among national governments, and those comprising voluntary, nontreaty organizations. In the area of computer network standards, there are several organizations of each type, which are discussed below.

1.6.1 Who's Who in the Telecommunications World

The legal status of the world's telephone companies varies considerably from country to country. At one extreme is the United States, which has 1500 separate, privately owned telephone companies. Before it was broken up in 1984, AT&T, at that time the world's largest corporation, completely dominated the scene. It provided telephone service to about 80 percent of America's telephones, spread throughout half of its geographical area, with all the other companies combined

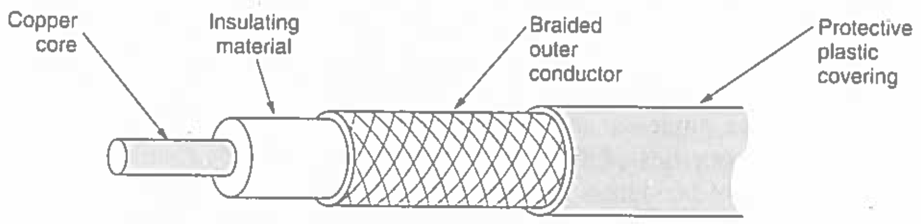


Figure 2-4. A coaxial cable.

2.2.4 Fiber Optics

Many people in the computer industry take enormous pride in how fast computer technology is improving. The original (1981) IBM PC ran at a clock speed of 4.77 MHz. Twenty years later, PCs could run at 2 GHz, a gain of a factor of 20 per decade. Not too bad.

In the same period, wide area data communication went from 56 kbps (the ARPANET) to 1 Gbps (modern optical communication), a gain of more than a factor of 125 per decade, while at the same time the error rate went from 10^{-5} per bit to almost zero.

Furthermore, single CPUs are beginning to approach physical limits, such as speed of light and heat dissipation problems. In contrast, with *current* fiber technology, the achievable bandwidth is certainly in excess of 50,000 Gbps (50 Tbps) and many people are looking very hard for better technologies and materials. The current practical signaling limit of about 10 Gbps is due to our inability to convert between electrical and optical signals any faster, although in the laboratory, 100 Gbps has been achieved on a single fiber.

In the race between computing and communication, communication won. The full implications of essentially infinite bandwidth (although not at zero cost) have not yet sunk in to a generation of computer scientists and engineers taught to think in terms of the low Nyquist and Shannon limits imposed by copper wire. The new conventional wisdom should be that all computers are hopelessly slow and that networks should try to avoid computation at all costs, no matter how much bandwidth that wastes. In this section we will study fiber optics to see how that transmission technology works.

An optical transmission system has three key components: the light source, the transmission medium, and the detector. Conventionally, a pulse of light indicates a 1 bit and the absence of light indicates a 0 bit. The transmission medium is an ultra-thin fiber of glass. The detector generates an electrical pulse when light falls on it. By attaching a light source to one end of an optical fiber and a detector to the other, we have a unidirectional data transmission system that accepts an electrical signal, converts and transmits it by light pulses, and then reconverts the output to an electrical signal at the receiving end.

solar, lunar, and planetary gravity tend to move them away from their assigned orbit slots and orientations, an effect countered by on-board rocket motors. This fine-tuning activity is called **station keeping**. However, when the fuel for the motors has been exhausted, typically in about 10 years, the satellite drifts and tumbles helplessly, so it has to be turned off. Eventually, the orbit decays and the satellite reenters the atmosphere and burns up or occasionally crashes to earth.

Orbit slots are not the only bone of contention. Frequencies are, too, because the downlink transmissions interfere with existing microwave users. Consequently, ITU has allocated certain frequency bands to satellite users. The main ones are listed in Fig. 2-16. The C band was the first to be designated for commercial satellite traffic. Two frequency ranges are assigned in it, the lower one for downlink traffic (from the satellite) and the upper one for uplink traffic (to the satellite). To allow traffic to go both ways at the same time, two channels are required, one going each way. These bands are already overcrowded because they are also used by the common carriers for terrestrial microwave links. The L and S bands were added by international agreement in 2000. However, they are narrow and crowded.

Band	Downlink	Uplink	Bandwidth	Problems
L	1.5 GHz	1.6 GHz	15 MHz	Low bandwidth; crowded
S	1.9 GHz	2.2 GHz	70 MHz	Low bandwidth; crowded
C	4.0 GHz	6.0 GHz	500 MHz	Terrestrial interference
Ku	11 GHz	14 GHz	500 MHz	Rain
Ka	20 GHz	30 GHz	3500 MHz	Rain, equipment cost

Figure 2-16. The principal satellite bands.

The next highest band available to commercial telecommunication carriers is the Ku (K under) band. This band is not (yet) congested, and at these frequencies, satellites can be spaced as close as 1 degree. However, another problem exists: rain. Water is an excellent absorber of these short microwaves. Fortunately, heavy storms are usually localized, so using several widely separated ground stations instead of just one circumvents the problem but at the price of extra antennas, extra cables, and extra electronics to enable rapid switching between stations. Bandwidth has also been allocated in the Ka (K above) band for commercial satellite traffic, but the equipment needed to use it is still expensive. In addition to these commercial bands, many government and military bands also exist.

A modern satellite has around 40 transponders, each with an 80-MHz bandwidth. Usually, each transponder operates as a bent pipe, but recent satellites have some on-board processing capacity, allowing more sophisticated operation. In the earliest satellites, the division of the transponders into channels was static: the bandwidth was simply split up into fixed frequency bands. Nowadays, each

so on.
isist of
STS-1
STS-1
byte by
tes and
ipes for

3

THE DATA LINK LAYER

In this chapter we will study the design principles for layer 2, the data link layer. This study deals with the algorithms for achieving reliable, efficient communication between two adjacent machines at the data link layer. By adjacent, we mean that the two machines are connected by a communication channel that acts conceptually like a wire (e.g., a coaxial cable, telephone line, or point-to-point wireless channel). The essential property of a channel that makes it "wire-like" is that the bits are delivered in exactly the same order in which they are sent.

At first you might think this problem is so trivial that there is no software to study—machine *A* just puts the bits on the wire, and machine *B* just takes them off. Unfortunately, communication circuits make errors occasionally. Furthermore, they have only a finite data rate, and there is a nonzero propagation delay between the time a bit is sent and the time it is received. These limitations have important implications for the efficiency of the data transfer. The protocols used for communications must take all these factors into consideration. These protocols are the subject of this chapter.

After an introduction to the key design issues present in the data link layer, we will start our study of its protocols by looking at the nature of errors, their causes, and how they can be detected and corrected. Then we will study a series of increasingly complex protocols, each one solving more and more of the problems present in this layer. Finally, we will conclude with an examination of protocol modeling and correctness and give some examples of data link protocols.

trailer), so that the datalink layer software need not worry about it. The polynomial algorithm discussed earlier in this chapter might be used, for example.

Initially, the receiver has nothing to do. It just sits around waiting for something to happen. In the example protocols of this chapter we will indicate that the data link layer is waiting for something to happen by the procedure call *wait_for_event(&event)*. This procedure only returns when something has happened (e.g., a frame has arrived). Upon return, the variable *event* tells what happened. The set of possible events differs for the various protocols to be described and will be defined separately for each protocol. Note that in a more realistic situation, the data link layer will not sit in a tight loop waiting for an event, as we have suggested, but will receive an interrupt, which will cause it to stop whatever it was doing and go handle the incoming frame. Nevertheless, for simplicity we will ignore all the details of parallel activity within the data link layer and assume that it is dedicated full time to handling just our one channel.

When a frame arrives at the receiver, the hardware computes the checksum. If the checksum is incorrect (i.e., there was a transmission error), the data link layer is so informed (*event* = *cksum_err*). If the inbound frame arrived undamaged, the data link layer is also informed (*event* = *frame_arrival*) so that it can acquire the frame for inspection using *from_physical_layer*. As soon as the receiving data link layer has acquired an undamaged frame, it checks the control information in the header, and if everything is all right, passes the packet portion to the network layer. Under no circumstances is a frame header ever given to a network layer.

There is a good reason why the network layer must never be given any part of the frame header: to keep the network and data link protocols completely separate. As long as the network layer knows nothing at all about the data link protocol or the frame format, these things can be changed without requiring changes to the network layer's software. Providing a rigid interface between network layer and data link layer greatly simplifies the software design because communication protocols in different layers can evolve independently.

Figure 3-9 shows some declarations (in C) common to many of the protocols to be discussed later. Five data structures are defined there: *boolean*, *seq_nr*, *packet*, *frame_kind*, and *frame*. A *boolean* is an enumerated type and can take on the values *true* and *false*. A *seq_nr* is a small integer used to number the frames so that we can tell them apart. These sequence numbers run from 0 up to and including *MAX_SEQ*, which is defined in each protocol needing it. A *packet* is the unit of information exchanged between the network layer and the data link layer on the same machine, or between network layer peers. In our model it always contains *MAX_PKT* bytes, but more realistically it would be of variable length.

A *frame* is composed of four fields: *kind*, *seq*, *ack*, and *info*, the first three of which contain control information and the last of which may contain actual data to be transferred. These control fields are collectively called the **frame header**.

After transmitting a frame, the sender starts the timer running. If it was already running, it will be reset to allow another full timer interval. The time interval should be chosen to allow enough time for the frame to get to the receiver, for the receiver to process it in the worst case, and for the acknowledgement frame to propagate back to the sender. Only when that time interval has elapsed is it safe to assume that either the transmitted frame or its acknowledgement has been lost, and to send a duplicate. If the timeout interval is set too short, the sender will transmit unnecessary frames. While these extra frames will not affect the correctness of the protocol, they will hurt performance.

(After transmitting a frame and starting the timer, the sender waits for something exciting to happen. Only three possibilities exist: an acknowledgement frame arrives undamaged, a damaged acknowledgement frame staggers in, or the timer expires. If a valid acknowledgement comes in, the sender fetches the next packet from its network layer and puts it in the buffer, overwriting the previous packet. It also advances the sequence number. If a damaged frame arrives or no frame at all arrives, neither the buffer nor the sequence number is changed so that a duplicate can be sent.)

When a valid frame arrives at the receiver, its sequence number is checked to see if it is a duplicate. If not, it is accepted, passed to the network layer, and an acknowledgement is generated. Duplicates and damaged frames are not passed to the network layer.

3.4 SLIDING WINDOW PROTOCOLS

In the previous protocols, data frames were transmitted in one direction only. In most practical situations, there is a need for transmitting data in both directions. One way of achieving full-duplex data transmission is to have two separate communication channels and use each one for simplex data traffic (in different directions). If this is done, we have two separate physical circuits, each with a "forward" channel (for data) and a "reverse" channel (for acknowledgements). In both cases the bandwidth of the reverse channel is almost entirely wasted. In effect, the user is paying for two circuits but using only the capacity of one.

A better idea is to use the same circuit for data in both directions. After all, in protocols 2 and 3 it was already being used to transmit frames both ways, and the reverse channel has the same capacity as the forward channel. In this model the data frames from A to B are intermixed with the acknowledgement frames from A to B. By looking at the *kind* field in the header of an incoming frame, the receiver can tell whether the frame is data or acknowledgement.

Although interleaving data and control frames on the same circuit is an improvement over having two separate physical circuits, yet another improvement is possible. When a data frame arrives, instead of immediately sending a separate

Technically, the MAC sublayer is the bottom part of the data link layer, so logically we should have studied it before examining all the point-to-point protocols in Chap. 3. Nevertheless, for most people, understanding protocols involving multiple parties is easier after two-party protocols are well understood. For that reason we have deviated slightly from a strict bottom-up order of presentation.

4.1 THE CHANNEL ALLOCATION PROBLEM

The central theme of this chapter is how to allocate a single broadcast channel among competing users. We will first look at static and dynamic schemes in general. Then we will examine a number of specific algorithms.

4.1.1 Static Channel Allocation in LANs and MANs

The traditional way of allocating a single channel, such as a telephone trunk, among multiple competing users is Frequency Division Multiplexing (FDM). If there are N users, the bandwidth is divided into N equal-sized portions (see Fig. 2-31), each user being assigned one portion. Since each user has a private frequency band, there is no interference between users. When there is only a small and constant number of users, each of which has a heavy (buffered) load of traffic (e.g., carriers' switching offices), FDM is a simple and efficient allocation mechanism.

However, when the number of senders is large and continuously varying or the traffic is bursty, FDM presents some problems. If the spectrum is cut up into N regions and fewer than N users are currently interested in communicating, a large piece of valuable spectrum will be wasted. If more than N users want to communicate, some of them will be denied permission for lack of bandwidth, even if some of the users who have been assigned a frequency band hardly ever transmit or receive anything.

However, even assuming that the number of users could somehow be held constant at N, dividing the single available channel into static subchannels is inherently inefficient. The basic problem is that when some users are quiescent, their bandwidth is simply lost. They are not using it, and no one else is allowed to use it either. Furthermore, in most computer systems, data traffic is extremely bursty (peak traffic to mean traffic ratios of 1000:1 are common). Consequently, most of the channels will be idle most of the time.

The poor performance of static FDM can easily be seen from a simple queueing theory calculation. Let us start with the mean time delay, T, for a channel of capacity C bps, with an arrival rate of λ frames/sec, each frame having a length drawn from an exponential probability density function with mean $1/\mu$ bits/frame.

4.2.3 Collision-Free Protocols

Although collisions do not occur with CSMA/CD once a station has unambiguously captured the channel, they can still occur during the contention period. These collisions adversely affect the system performance, especially when the cable is long (i.e., large τ) and the frames are short. And CSMA/CD is not universally applicable. In this section, we will examine some protocols that resolve the contention for the channel without any collisions at all, not even during the contention period. Most of these are not currently used in major systems, but in a rapidly changing field, having some protocols with excellent properties available for future systems is often a good thing.

In the protocols to be described, we assume that there are exactly N stations, each with a unique address from 0 to $N-1$ "wired" into it. It does not matter that some stations may be inactive part of the time. We also assume that propagation delay is negligible. The basic question remains: Which station gets the channel after a successful transmission? We continue using the model of Fig. 4-5 with its discrete contention slots.

A Bit-Map Protocol

In our first collision-free protocol, the **basic bit-map method**, each contention period consists of exactly N slots. If station 0 has a frame to send, it transmits a 1 bit during the zeroth slot. No other station is allowed to transmit during this slot. Regardless of what station 0 does, station 1 gets the opportunity to transmit a 1 during slot 1, but only if it has a frame queued. In general, station j may announce that it has a frame to send by inserting a 1 bit into slot j. After all N slots have passed by, each station has complete knowledge of which stations wish to transmit. At that point, they begin transmitting in numerical order (see Fig. 4-6).

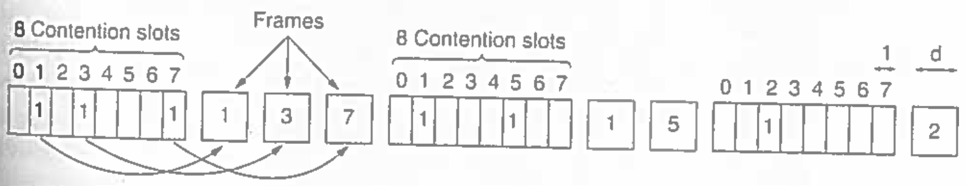


Figure 4-6. The basic bit-map protocol.

Since everyone agrees on who goes next, there will never be any collisions. After the last ready station has transmitted its frame, an event all stations can easily monitor, another N bit contention period is begun. If a station becomes ready just after its bit slot has passed by, it is out of luck and must remain silent until every station has had a chance and the bit map has come around again. Protocols

Now consider the case where Z is still tentatively labeled. Either the label at Z is greater than or equal to that at E, in which case $AXYZE$ cannot be a shorter path than ABE, or it is less than that of E, in which case Z and not E will become permanent first, allowing E to be probed from Z.

This algorithm is given in Fig. 5-8. The global variables *n* and *dist* describe the graph and are initialized before *shortest_path* is called. The only difference between the program and the algorithm described above is that in Fig. 5-8, we compute the shortest path starting at the terminal node, *t*, rather than at the source node, *s*. Since the shortest path from *t* to *s* in an undirected graph is the same as the shortest path from *s* to *t*, it does not matter at which end we begin (unless there are several shortest paths, in which case reversing the search might discover a different one). The reason for searching backward is that each node is labeled with its predecessor rather than its successor. When the final path is copied into the output variable, *path*, the path is thus reversed. By reversing the search, the two effects cancel, and the answer is produced in the correct order.

5.2.3 Flooding

Another static algorithm is **flooding**, in which every incoming packet is sent out on every outgoing line except the one it arrived on. Flooding obviously generates vast numbers of duplicate packets, in fact, an infinite number unless some measures are taken to damp the process. One such measure is to have a hop counter contained in the header of each packet, which is decremented at each hop, with the packet being discarded when the counter reaches zero. Ideally, the hop counter should be initialized to the length of the path from source to destination. If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the subnet.

An alternative technique for damming the flood is to keep track of which packets have been flooded, to avoid sending them out a second time.
achieve this goal is to have the source router put a sequence number in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.

To prevent the list from growing without bound, each list should be augmented by a counter, *k*, meaning that all sequence numbers through *k* have been seen. When a packet comes in, it is easy to check if the packet is a duplicate; if so, it is discarded. Furthermore, the full list below *k* is not needed, since *k* effectively summarizes it.

A variation of flooding that is slightly more practical is **selective flooding**. In this algorithm the routers do not send every incoming packet out on every line, only on those lines that are going approximately in the right direction. There is usually little point in sending a westbound packet on an eastbound line unless the topology is extremely peculiar and the router is sure of this fact.

Fourth, the schemes differ in how they actually manage to arrange for packets that are addressed to one destination to be delivered to a different one. One choice is changing the destination address and just retransmitting the modified packet. Alternatively, the whole packet, home address and all, can be encapsulated inside the payload of another packet sent to the temporary address. Finally, the schemes differ in their security aspects. In general, when a host or router gets a message of the form "Starting right now, please send all of Stephany's mail to me," it might have a couple of questions about whom it was talking to and whether this is a good idea. Several mobile host protocols are discussed and compared in (Hac and Guo, 2000; Perkins, 1998a; Snoeren and Balakrishnan, 2000; Solomon, 1998; and Wang and Chen, 2001).

5.2.10 Routing in Ad Hoc Networks

We have now seen how to do routing when the hosts are mobile but the routers are fixed. An even more extreme case is one in which the routers themselves are mobile. Among the possibilities are:

1. Military vehicles on a battlefield with no existing infrastructure.
2. A fleet of ships at sea.
3. Emergency workers at an earthquake that destroyed the infrastructure.
4. A gathering of people with notebook computers in an area lacking 802.11.

In all these cases, and others, each node consists of a router and a host, usually on the same computer. Networks of nodes that just happen to be near each other are called **ad hoc networks** or **MANETs (Mobile Ad hoc NETworks)**. Let us now examine them briefly. More information can be found in (Perkins, 2001).

What makes ad hoc networks different from wired networks is that all the usual rules about fixed topologies, fixed and known neighbors, fixed relationship between IP address and location, and more are suddenly tossed out the window. Routers can come and go or appear in new places at the drop of a bit. With a wired network, if a router has a valid path to some destination, that path continues to be valid indefinitely (barring a failure somewhere in the system). With an ad hoc network, the topology may be changing all the time, so desirability and even validity of paths can change spontaneously, without warning. Needless to say, these circumstances make routing in ad hoc networks quite different from routing in their fixed counterparts.

A variety of routing algorithms for ad hoc networks have been proposed. One of the more interesting ones is the **AODV (Ad hoc On-demand Distance Vector)** routing algorithm (Perkins and Royer, 1999). It is a distant relative of the Bellman-Ford distance vector algorithm but adapted to work in a mobile environment and takes into account the limited bandwidth and low battery life found in

Lookup can now proceed as follows. The requesting node sends a packet to its successor containing its IP address and the key it is looking for. The packet is propagated around the ring until it locates the successor to the node identifier being sought. That node checks to see if it has any information matching the key, and if so, returns it directly to the requesting node, whose IP address it has.

As a first optimization, each node could hold the IP addresses of both its successor and its predecessor, so that queries could be sent either clockwise or counterclockwise, depending on which path is thought to be shorter. For example, node 7 in Fig. 5-24 could go clockwise to find node identifier 10 but counterclockwise to find node identifier 3.

Even with two choices of direction, linearly searching all the nodes is very inefficient in a large peer-to-peer system since the mean number of nodes required per search is $n/2$. To greatly speed up the search, each node also maintains what Chord calls a **finger table**. The finger table has m entries, indexed by 0 through $m-1$, each one pointing to a different actual node. Each of the entries has two fields: *start* and the IP address of *successor*(*start*), as shown for three example nodes in Fig. 5-24(b). The values of the fields for entry i at node k are:

$$start = k + 2^i \ \ (modulo\, 2^m)$$
$$\text{IP address of } \ successor(start[i])$$

Note that each node stores the IP addresses of a relatively small number of nodes and that most of these are fairly close by in terms of node identifier.

Using the finger table, the lookup of *key* at node k proceeds as follows. If *key* falls between k and *successor*(k), then the node holding information about *key* is *successor*(k) and the search terminates. Otherwise, the finger table is searched to find the entry whose *start* field is the closest predecessor of *key*. A request is then sent directly to the IP address in that finger table entry to ask it to continue the search. Since it is closer to *key* but still below it, chances are good that it will be able to return the answer with only a small number of additional queries. In fact, since every lookup halves the remaining distance to the target, it can be shown that the average number of lookups is $\log_2 n$.

As a first example, consider looking up *key* = 3 at node 1. Since node 1 knows that 3 lies between it and its successor, 4, the desired node is 4 and the search terminates, returning node 4's IP address.

As a second example, consider looking up *key* = 14 at node 1. Since 14 does not lie between 1 and 4, the finger table is consulted. The closest predecessor to 14 is 9, so the request is forwarded to the IP address of 9's entry, namely, that of node 12. Node 12 sees that 14 falls between it and its successor (15), so it returns the IP address of node 15.

As a third example, consider looking up *key* = 16 at node 1. Again a query is sent to node 12, but this time node 12 does not know the answer itself. It looks for the node most closely preceding 16 and finds 14, which yields the IP address of node 15. A query is then sent there. Node 15 observes that 16 lies between it upon timeout. A jumpy sender that times out quickly and retransmits all outstanding packets using go back n will put a heavier load on the system than will a leisurely sender that uses selective repeat. Closely related to this is the buffering policy. If receivers routinely discard all out-of-order packets, these packets will have to be transmitted again later, creating extra load. With respect to congestion control, selective repeat is clearly better than go back n.

Acknowledgement policy also affects congestion. If each packet is acknowledged immediately, the acknowledgement packets generate extra traffic. However, if acknowledgements are saved up to piggyback onto reverse traffic, extra timeouts and retransmissions may result. A tight flow control scheme (e.g., a small window) reduces the data rate and thus helps fight congestion.

At the network layer, the choice between using virtual circuits and using datagrams affects congestion since many congestion control algorithms work only with virtual-circuit subnets. Packet queueing and service policy relates to whether routers have one queue per input line, one queue per output line, or both. It also relates to the order in which packets are processed (e.g., round robin or priority based). Discard policy is the rule telling which packet is dropped when there is no space. A good policy can help alleviate congestion and a bad one can make it worse.

A good routing algorithm can help avoid congestion by spreading the traffic over all the lines, whereas a bad one can send too much traffic over already congested lines. Finally, packet lifetime management deals with how long a packet may live before being discarded. If it is too long, lost packets may clog up the works for a long time, but if it is too short, packets may sometimes time out before reaching their destination, thus inducing retransmissions.

In the transport layer, the same issues occur as in the data link layer, but in addition, determining the timeout interval is harder because the transit time across the network is less predictable than the transit time over a wire between two routers. If the timeout interval is too short, extra packets will be sent unnecessarily. If it is too long, congestion will be reduced but the response time will suffer whenever a packet is lost.

5.3.3 Congestion Control in Virtual-Circuit Subnets

The congestion control methods described above are basically open loop: they try to prevent congestion from occurring in the first place, rather than dealing with it after the fact. In this section we will describe some approaches to dynamically controlling congestion in virtual-circuit subnets. In the next two, we will look at techniques that can be used in any subnet.

One technique that is widely used to keep congestion that has already started from getting worse is **admission control**. The idea is simple: once congestion has been signaled, no more virtual circuits are set up until the problem has gone away. Thus, attempts to set up new transport layer connections fail. Letting more people

5.4 QUALITY OF SERVICE

The techniques we looked at in the previous sections are designed to reduce congestion and improve network performance. However, with the growth of multimedia networking, often these ad hoc measures are not enough. Serious attempts at guaranteeing quality of service through network and protocol design are needed. In the following sections we will continue our study of network performance, but now with a sharper focus on ways to provide a quality of service matched to application needs. It should be stated at the start, however, that many of these ideas are in flux and are subject to change.

5.4.1 Requirements

A stream of packets from a source to a destination is called a **flow**. In a connection-oriented network, all the packets belonging to a flow follow the same route; in a connectionless network, they may follow different routes. The needs of each flow can be characterized by four primary parameters: reliability, delay, jitter, and bandwidth. Together these determine the **QoS (Quality of Service)** the flow requires. Several common applications and the stringency of their requirements are listed in Fig. 5-30.

Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

Figure 5-30. How stringent the quality-of-service requirements are.

The first four applications have stringent requirements on reliability. No bits may be delivered incorrectly. This goal is usually achieved by checksumming each packet and verifying the checksum at the destination. If a packet is damaged in transit, it is not acknowledged and will be retransmitted eventually. This strategy gives high reliability. The four final (audio/video) applications can tolerate errors, so no checksums are computed or verified.

File transfer applications, including e-mail and video, are not delay sensitive. If all packets are delayed uniformly by a few seconds, no harm is done. Interactive applications, such as Web surfing and remote login, are more delay sensitive.

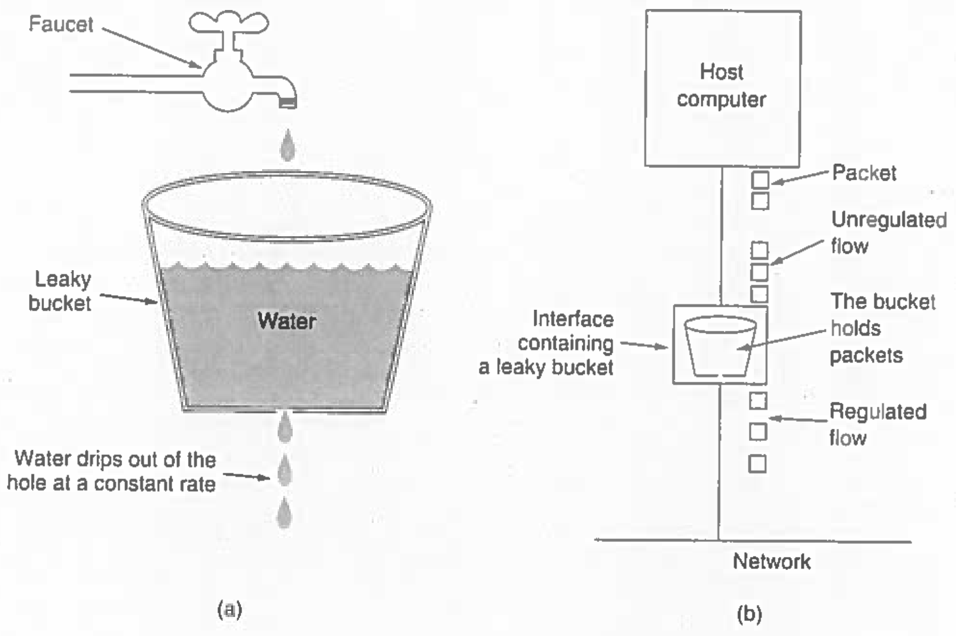


Figure 5-32. (a) A leaky bucket with water. (b) A leaky bucket with packets.

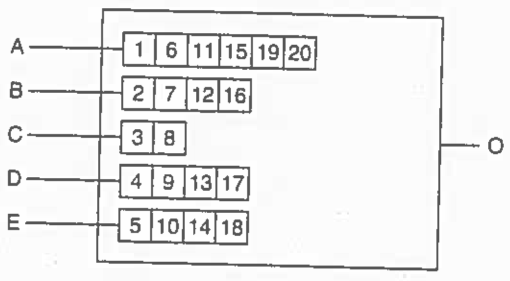
anism turns an uneven flow of packets from the user processes inside the host into an even flow of packets onto the network, smoothing out bursts and greatly reducing the chances of congestion.

When the packets are all the same size (e.g., ATM cells), this algorithm can be used as described. However, when variable-sized packets are being used, it is often better to allow a fixed number of bytes per tick, rather than just one packet. Thus, if the rule is 1024 bytes per tick, a single 1024-byte packet can be admitted on a tick, two 512-byte packets, four 256-byte packets, and so on. If the residual byte count is too low, the next packet must wait until the next tick.

Implementing the original leaky bucket algorithm is easy. The leaky bucket consists of a finite queue. When a packet arrives, if there is room on the queue it is appended to the queue; otherwise, it is discarded. At every clock tick, one packet is transmitted (unless the queue is empty).

The byte-counting leaky bucket is implemented almost the same way. At each tick, a counter is initialized to n. If the first packet on the queue has fewer bytes than the current value of the counter, it is transmitted, and the counter is decremented by that number of bytes. Additional packets may also be sent, as long as the counter is high enough. When the counter drops below the length of the next packet on the queue, transmission stops until the next tick, at which time the residual byte count is reset and the flow can continue.

(1990) suggested an improvement in which the round robin is done in such a way as to simulate a byte-by-byte round robin, instead of a packet-by-packet round robin. In effect, it scans the queues repeatedly, byte-for-byte, until it finds the tick on which each packet will be finished. The packets are then sorted in order of their finishing and sent in that order. The algorithm is illustrated in Fig. 5-36.



(a)

Packet	Finishing time
C	8
B	16
D	17
E	18
A	20

(b)

Figure 5-36. (a) A router with five packets queued for line *O*. (b) Finishing times for the five packets.

In Fig. 5-36(a) we see packets of length 2 to 6 bytes. At (virtual) clock tick 1, the first byte of the packet on line *A* is sent. Then goes the first byte of the packet on line *B*, and so on. The first packet to finish is *C*, after eight ticks. The sorted order is given in Fig. 5-36(b). In the absence of new arrivals, the packets will be sent in the order listed, from *C* to *A*.

One problem with this algorithm is that it gives all hosts the same priority. In many situations, it is desirable to give video servers more bandwidth than regular file servers so that they can be given two or more bytes per tick. This modified algorithm is called **weighted fair queueing** and is widely used. Sometimes the weight is equal to the number of flows coming out of a machine, so each process gets equal bandwidth. An efficient implementation of the algorithm is discussed in (Shreedhar and Varghese, 1995). Increasingly, the actual forwarding of packets through a router or switch is being done in hardware (Elhanany et al., 2001).

5.4.3 Integrated Services

Between 1995 and 1997, IETF put a lot of effort into devising an architecture for streaming multimedia. This work resulted in over two dozen RFCs, starting with RFCs 2205–2210. The generic name for this work is **flow-based algorithms** or **integrated services**. It was aimed at both unicast and multicast applications. An example of the former is a single user streaming a video clip from a news site. An example of the latter is a collection of digital television stations broadcasting their programs as streams of IP packets to many receivers at various locations. Below we will concentrate on multicast, since unicast is a special case of multicast.

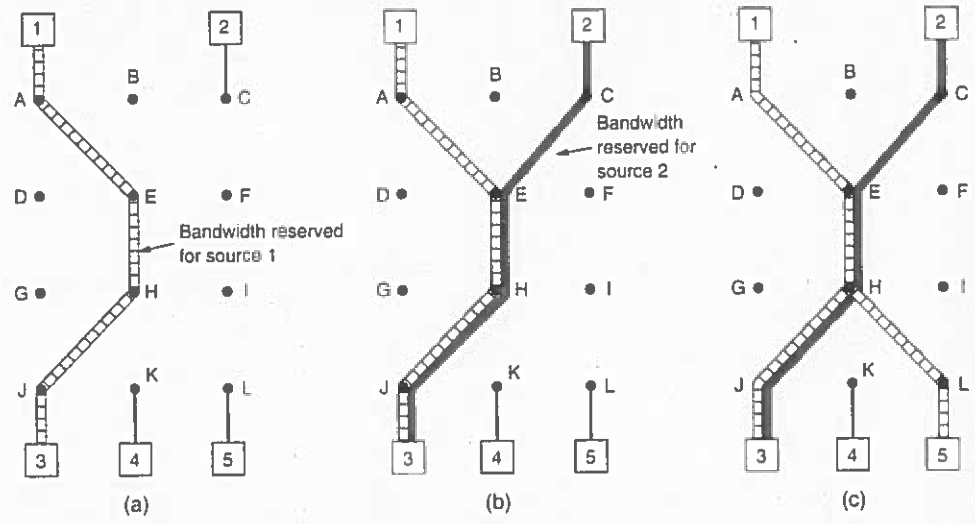

1
2
A
B
C
D
E
F
G
H
I
J
K
L
3
4
5
Bandwidth reserved for source 1
Bandwidth reserved for source 2
(a)
(b)
(c)

Figure 5-38. (a) Host 3 requests a channel to host 1. (b) Host 3 then requests a second channel, to host 2. (c) Host 5 requests a channel to host 1.

5.4.4 Differentiated Services

Flow-based algorithms have the potential to offer good quality of service to one or more flows because they reserve whatever resources are needed along the route. However, they also have a downside. They require an advance setup to establish each flow, something that does not scale well when there are thousands or millions of flows. Also, they maintain internal per-flow state in the routers, making them vulnerable to router crashes. Finally, the changes required to the router code are substantial and involve complex router-to-router exchanges for setting up the flows. As a consequence, few implementations of RSVP or anything like it exist yet.

For these reasons, IETF has also devised a simpler approach to quality of service, one that can be largely implemented locally in each router without advance setup and without having the whole path involved. This approach is known as **class-based** (as opposed to flow-based) quality of service. IETF has standardized an architecture for it, called **differentiated services**, which is described in RFCs 2474, 2475, and numerous others. We will now describe it.

Differentiated services (DS) can be offered by a set of routers forming an administrative domain (e.g., an ISP or a telco). The administration defines a set of service classes with corresponding forwarding rules. If a customer signs up for DS, customer packets entering the domain may carry a *Type of Service* field in them, with better service provided to some classes (e.g., premium service) than to others. Traffic within a class may be required to conform to some specific shape, The result of all these factors is that the network designers are not free to choose any maximum packet size they wish. Maximum payloads range from 48 bytes (ATM cells) to 65,515 bytes (IP packets), although the payload size in higher layers is often larger.

An obvious problem appears when a large packet wants to travel through a network whose maximum packet size is too small. One solution is to make sure the problem does not occur in the first place. In other words, the internet should use a routing algorithm that avoids sending packets through networks that cannot handle them. However, this solution is no solution at all. What happens if the original source packet is too large to be handled by the destination network? The routing algorithm can hardly bypass the destination.

Basically, the only solution to the problem is to allow gateways to break up packets into **fragments**, sending each fragment as a separate internet packet. However, as every parent of a small child knows, converting a large object into small fragments is considerably easier than the reverse process. (Physicists have even given this effect a name: the second law of thermodynamics.) Packet-switching networks, too, have trouble putting the fragments back together again.

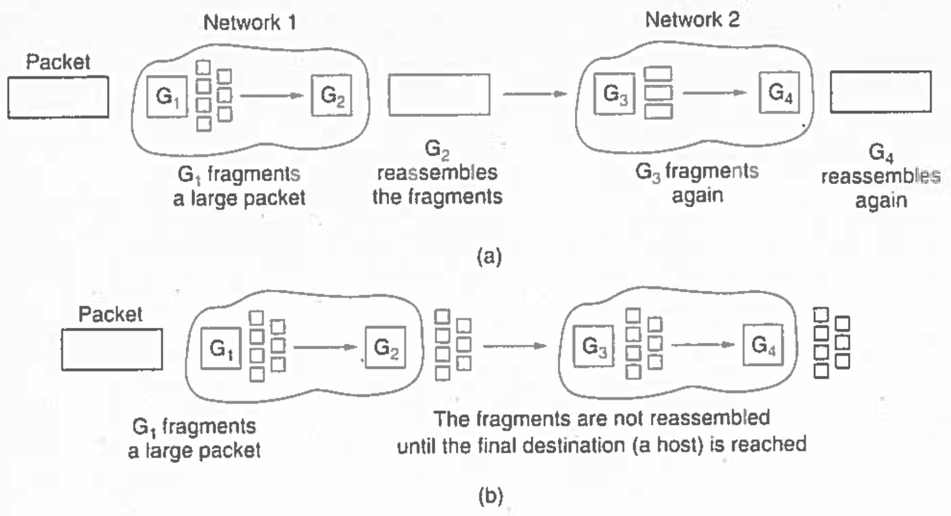

Network 1
Network 2
Packet
G1
G2
G3
G4
G1 fragments a large packet
G2 reassembles the fragments
G3 fragments again
G4 reassembles again
(a)
Packet
G1 fragments a large packet
The fragments are not reassembled until the final destination (a host) is reached
(b)

Figure 5-50. (a) Transparent fragmentation. (b) Nontransparent fragmentation.

Two opposing strategies exist for recombining the fragments back into the original packet. The first strategy is to make fragmentation caused by a "small-packet" network transparent to any subsequent networks through which the packet must pass on its way to the ultimate destination. This option is shown in Fig. 5-50(a). In this approach, the small-packet network has gateways (most likely, specialized routers) that interface to other networks. When an oversized packet arrives at a gateway, the gateway breaks it up into fragments. Each frag-

What actually happened is as follows. The NAT designers observed that most IP packets carry either TCP or UDP payloads. When we study TCP and UDP in Chap. 6, we will see that both of these have headers containing a source port and a destination port. Below we will just discuss TCP ports, but exactly the same story holds for UDP ports. The ports are 16-bit integers that indicate where the TCP connection begins and ends. These ports provide the field needed to make NAT work.

When a process wants to establish a TCP connection with a remote process, it attaches itself to an unused TCP port on its own machine. This is called the **source port** and tells the TCP code where to send incoming packets belonging to this connection. The process also supplies a **destination port** to tell who to give the packets to on the remote side. Ports 0–1023 are reserved for well-known services. For example, port 80 is the port used by Web servers, so remote clients can locate them. Each outgoing TCP message contains both a source port and a destination port. Together, these ports serve to identify the processes using the connection on both ends.

An analogy may make the use of ports clearer. Imagine a company with a single main telephone number. When people call the main number, they reach an operator who asks which extension they want and then puts them through to that extension. The main number is analogous to the company's IP address and the extensions on both ends are analogous to the ports. Ports are an extra 16-bits of addressing that identify which process gets which incoming packet.

Using the *Source port* field, we can solve our mapping problem. Whenever an outgoing packet enters the NAT box, the 10.*x*.*y*.*z* source address is replaced by the company's true IP address. In addition, the TCP *Source port* field is replaced by an index into the NAT box's 65,536-entry translation table. This table entry contains the original IP address and the original source port. Finally, both the IP and TCP header checksums are recomputed and inserted into the packet. It is necessary to replace the *Source port* because connections from machines 10.0.0.1 and 10.0.0.2 may both happen to use port 5000, for example, so the *Source port* alone is not enough to identify the sending process.

When a packet arrives at the NAT box from the ISP, the *Source port* in the TCP header is extracted and used as an index into the NAT box's mapping table. From the entry located, the internal IP address and original TCP *Source port* are extracted and inserted into the packet. Then both the IP and TCP checksums are recomputed and inserted into the packet. The packet is then passed to the company router for normal delivery using the 10.*x*.*y*.*z* address.

NAT can also be used to alleviate the IP shortage for ADSL and cable users. When the ISP assigns each user an address, it uses 10.*x*.*y*.*z* addresses. When packets from user machines exit the ISP and enter the main Internet, they pass through a NAT box that translates them to the ISP's true Internet address. On the way back, packets undergo the reverse mapping. In this respect, to the rest of the Internet, the ISP and its home ADSL/cable users just looks like a big company.

one bad

) to 255

the path
exceed-
ısed, the

ıs).

'hat kind
hop-by-
ory. All

ode 194)
he data-
the first
nessage.
: use of
fer giga-

be inter-
options
s, so ini-
and host
e day.

The routing header lists one or more routers that must be visited on the way to the destination. It is very similar to the IPv4 loose source routing in that all addresses listed must be visited in order, but other routers not listed may be visited in between. The format of the routing header is shown in Fig. 5-71.

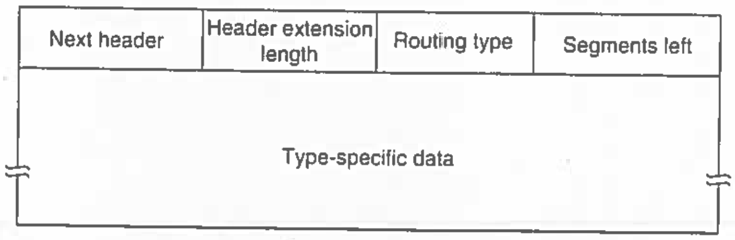


Figure 5-71. The extension header for routing.

The first 4 bytes of the routing extension header contain four 1-byte integers. The *Next header* and *Header entension length* fields were described above. The *Routing type* field gives the format of the rest of the header. Type 0 says that a reserved 32-bit word follows the first word, followed by some number of IPv6 addresses. Other types may be invented in the future as needed. Finally, the *Segments left* field keeps track of how many of the addresses in the list have not yet been visited. It is decremented every time one is visited. When it hits 0, the packet is on its own with no more guidance about what route to follow. Usually at this point it is so close to the destination that the best route is obvious.

[The fragment header deals with fragmentation similarly to the way IPv4 does. The header holds the datagram identifier, fragment number, and a bit telling whether more fragments will follow. In IPv6, unlike in IPv4, only the source host can fragment a packet. Routers along the way may not do this. Although this change is a major philosophical break with the past, it simplifies the routers' work and makes routing go faster. As mentioned above, if a router is confronted with a packet that is too big, it discards the packet and sends an ICMP packet back to the source. This information allows the source host to fragment the packet into smaller pieces using this header and try again.]

The authentication header provides a mechanism by which the receiver of a packet can be sure of who sent it. The encrypted security payload makes it possible to encrypt the contents of a packet so that only the intended recipient can read it. These headers use cryptographic techniques to accomplish their missions.

Controversies

Given the open design process and the strongly-held opinions of many of the people involved, it should come as no surprise that many choices made for IPv6 were highly controversial, to say the least. We will summarize a few of these briefly below. For all the gory details, see the RFCs.

ɔits in and
is the rea-

e up table
ıts: asym-
:an issue a
sent to the

ndently of
more data
lel, a con-

se simple
ıt, either a
ɔr simplic-
e also as-
oing first.
ore realis-

NT

e.
w

6.1.3 Berkeley Sockets

Let us now briefly inspect another set of transport primitives, the socket primitives used in Berkeley UNIX for TCP. These primitives are widely used for Internet programming. They are listed in Fig. 6-5. Roughly speaking, they follow the model of our first example but offer more features and flexibility. We will not look at the corresponding TPDUs here. That discussion will have to wait until we study TCP later in this chapter.

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Figure 6-5. The socket primitives for TCP.

The first four primitives in the list are executed in that order by servers. The SOCKET primitive creates a new end point and allocates table space for it within the transport entity. The parameters of the call specify the addressing format to be used, the type of service desired (e.g., reliable byte stream), and the protocol. A successful SOCKET call returns an ordinary file descriptor for use in succeeding calls, the same way an OPEN call does.

Newly-created sockets do not have network addresses. These are assigned using the BIND primitive. Once a server has bound an address to a socket, remote clients can connect to it. The reason for not having the SOCKET call create an address directly is that some processes care about their address (e.g., they have been using the same address for years and everyone knows this address), whereas others do not care.

Next comes the LISTEN call, which allocates space to queue incoming calls for the case that several clients try to connect at the same time. In contrast to LISTEN in our first example, in the socket model LISTEN is not a blocking call.

To block waiting for an incoming connection, the server executes an ACCEPT primitive. When a TPDU asking for a connection arrives, the transport entity creates a new socket with the same properties as the original one and returns a file descriptor for it. The server can then fork off a process or thread to handle the connection on the new socket and go back to waiting for the next connection on the original socket. ACCEPT returns a normal file descriptor, which can be used for reading and writing in the standard way, the same as for files.

Now let us look at the client side. Here, too, a socket must first be created using the SOCKET primitive, but BIND is not required since the address used does not matter to the server. The CONNECT primitive blocks the caller and actively starts the connection process. When it completes (i.e., when the appropriate TPDU is received from the server), the client process is unblocked and the connection is established. Both sides can now use SEND and RECV to transmit and receive data over the full-duplex connection. The standard UNIX READ and WRITE system calls can also be used if none of the special options of SEND and RECV are required.

Connection release with sockets is symmetric. When both sides have executed a CLOSE primitive, the connection is released.

6.1.4 An Example of Socket Programming: An Internet File Server

As an example of how the socket calls are used, consider the client and server code of Fig. 6-6. Here we have a very primitive Internet file server along with an example client that uses it. The code has many limitations (discussed below), but in principle the server code can be compiled and run on any UNIX system connected to the Internet. The client code can then be compiled and run on any other UNIX machine on the Internet, anywhere in the world. The client code can be executed with appropriate parameters to fetch any file to which the server has access on its machine. The file is written to standard output, which, of course, can be redirected to a file or pipe.

Let us look at the server code first. It starts out by including some standard headers, the last three of which contain the main Internet-related definitions and data structures. Next comes a definition of *SERVER_PORT* as 12345. This number was chosen arbitrarily. Any number between 1024 and 65535 will work just as well as long as it is not in use by some other process. Of course, the client and server have to use the same port. If this server ever becomes a worldwide hit (unlikely, given how primitive it is), it will be assigned a permanent port below 1024 and appear on *www.iana.org*.

The next two lines in the server define two constants needed. The first one determines the chunk size used for the file transfer. The second one determines how many pending connections can be held before additional ones are discarded upon arrival.

After the declarations of local variables, the server code begins. It starts out by initializing a data structure that will hold the server's IP address. This data structure will soon be bound to the server's socket. The call to *memset* sets the data structure to all 0s. The three assignments following it fill in three of its fields. The last of these contains the server's port. The functions *htonl* and *htons* have to do with converting values to a standard format so the code runs correctly on both big-endian machines (e.g., the SPARC) and little-endian machines (e.g., the Pentium). Their exact semantics are not relevant here.

he socket

needs the
Since the
; comput-

x) can be

ey can be
x) by

on is also

Its error
ver heard
platform
egal, such
. Since it
nread), its
ete, work-
ve it. For

l between
the data
error con-

se differ-
h the two
ters com-
this phys-
important

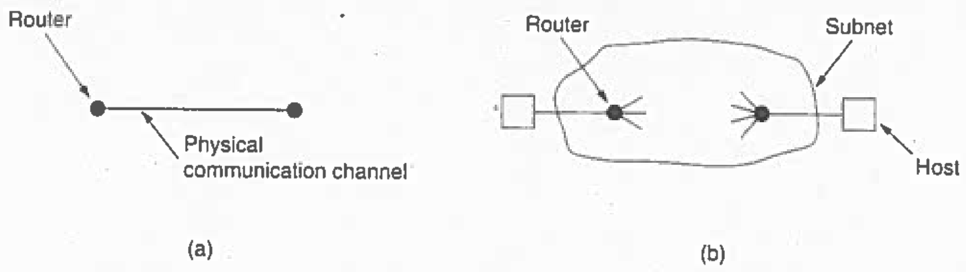


Figure 6-7. (a) Environment of the data link layer. (b) Environment of the transport layer.

For one thing, in the data link layer, it is not necessary for a router to specify which router it wants to talk to—each outgoing line uniquely specifies a particular router. In the transport layer, explicit addressing of destinations is required.

For another thing, the process of establishing a connection over the wire of Fig. 6-7(a) is simple: the other end is always there (unless it has crashed, in which case it is not there). Either way, there is not much to do. In the transport layer, initial connection establishment is more complicated, as we will see.

Another, exceedingly annoying, difference between the data link layer and the transport layer is the potential existence of storage capacity in the subnet. When a router sends a frame, it may arrive or be lost, but it cannot bounce around for a while, go into hiding in a far corner of the world, and then suddenly emerge at an inopportune moment 30 sec later. If the subnet uses datagrams and adaptive routing inside, there is a nonnegligible probability that a packet may be stored for a number of seconds and then delivered later. The consequences of the subnet's ability to store packets can sometimes be disastrous and can require the use of special protocols.

A final difference between the data link and transport layers is one of amount rather than of kind. Buffering and flow control are needed in both layers, but the presence of a large and dynamically varying number of connections in the transport layer may require a different approach than we used in the data link layer. In Chap. 3, some of the protocols allocate a fixed number of buffers to each line, so that when a frame arrives a buffer is always available. In the transport layer, the larger number of connections that must be managed make the idea of dedicating many buffers to each one less attractive. In the following sections, we will examine all of these important issues and others.

6.2.1 Addressing

When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. (Connectionless transport has the same problem: To whom should each message be sent?) The method normally used is to define transport addresses to which processes can

alled **ports.**
eneric term
; in the net-
P addresses

ıd transport
ttach them-
connections
;APs is that
s needed to

P 1522
lf to a
on the
ed, for

of-day,
source
lts in a

transport connection being established between the application process on host 1 and server 1 on host 2.

3. The application process then sends over a request for the time.
4. The time server process responds with the current time.
5. The transport connection is then released.

Note that there may well be other servers on host 2 that are attached to other TSAPs and waiting for incoming connections that arrive over the same NSAP.

The picture painted above is fine, except we have swept one little problem under the rug: How does the user process on host 1 know that the time-of-day server is attached to TSAP 1522? One possibility is that the time-of-day server has been attaching itself to TSAP 1522 for years and gradually all the network users have learned this. In this model, services have stable TSAP addresses that are listed in files in well-known places, such as the */etc/services* file on UNIX systems, which lists which servers are permanently attached to which ports.

While stable TSAP addresses work for a small number of key services that never change (e.g. the Web server), user processes, in general, often want to talk to other user processes that only exist for a short time and do not have a TSAP address that is known in advance. Furthermore, if there are potentially many server processes, most of which are rarely used, it is wasteful to have each of them active and listening to a stable TSAP address all day long. In short, a better scheme is needed.

One such scheme is shown in Fig. 6-9 in a simplified form. It is known as the **initial connection protocol**. Instead of every conceivable server listening at a well-known TSAP, each machine that wishes to offer services to remote users has a special **process server** that acts as a proxy for less heavily used servers. It listens to a set of ports at the same time, waiting for a connection request. Potential users of a service begin by doing a CONNECT request, specifying the TSAP address of the service they want. If no server is waiting for them, they get a connection to the process server, as shown in Fig. 6-9(a).

After it gets the incoming request, the process server spawns the requested server, allowing it to inherit the existing connection with the user. The new server then does the requested work, while the process server goes back to listening for new requests, as shown in Fig. 6-9(b).

While the initial connection protocol works fine for those servers that can be created as they are needed, there are many situations in which services do exist independently of the process server. A file server, for example, needs to run on special hardware (a machine with a disk) and cannot just be created on-the-fly when someone wants to talk to it.

To handle this situation, an alternative scheme is often used. In this model, there exists a special process called a **name server** or sometimes a **directory server**. To find the TSAP address corresponding to a given service name, such as

oping, com-
)wn) longest
nitialized to
varded. The
comes zero.
ed, with the
time. This
elf is a non-
, for exam-
me periodi-

ad, but also
ce T, which
tiple is pro-
ait a time T
w gone and
the blue to

oof way to
Tomlinson
s own. The
ts of it are

here it was
-day clock.
assumed to
rvals. Fur-
number of
assumed to

s are never
der k bits of
, unlike our
a different
by the time
number are
numbers is

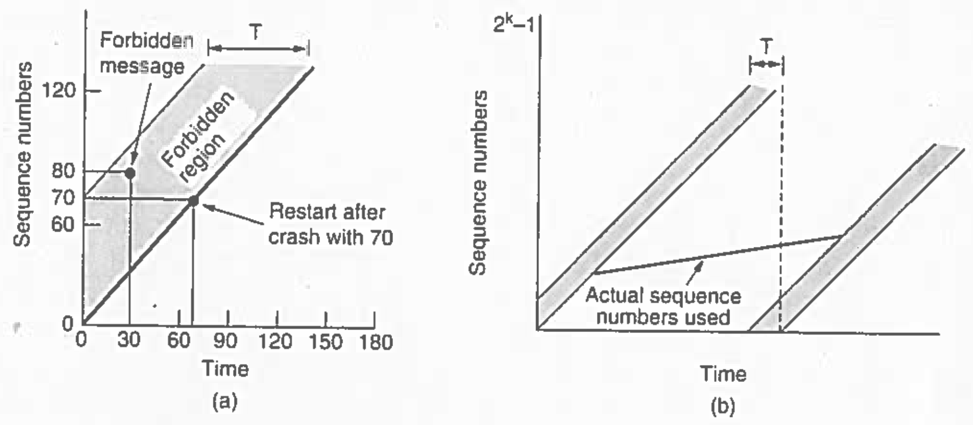


Figure 6-10. (a) TPDUs may not enter the forbidden region. (b) The resynchronization problem.

Once both transport entities have agreed on the initial sequence number, any sliding window protocol can be used for data flow control. In reality, the initial sequence number curve (shown by the heavy line) is not linear, but a staircase, since the clock advances in discrete steps. For simplicity we will ignore this detail.

A problem occurs when a host crashes. When it comes up again, its transport entity does not know where it was in the sequence space. One solution is to require transport entities to be idle for T sec after a recovery to let all old TPDUs die off. However, in a complex internetwork, T may be large, so this strategy is unattractive.

To avoid requiring T sec of dead time after a crash, it is necessary to introduce a new restriction on the use of sequence numbers. We can best see the need for this restriction by means of an example. Let T, the maximum packet lifetime, be 60 sec and let the clock tick once per second. As shown by the heavy line in Fig. 6-10(a), the initial sequence number for a connection opened at time x will be x. Imagine that at $t = 30$ sec, an ordinary data TPDU being sent on (a previously opened) connection 5 is given sequence number 80. Call this TPDU X. Immediately after sending TPDU X, the host crashes and then quickly restarts. At $t = 60$, it begins reopening connections 0 through 4. At $t = 70$, it reopens connection 5, using initial sequence number 70 as required. Within the next 15 sec it sends data TPDUs 70 through 80. Thus, at $t = 85$ a new TPDU with sequence number 80 and connection 5 has been injected into the subnet. Unfortunately, TPDU X still exists. If it should arrive at the receiver before the new TPDU 80, TPDU X will be accepted and the correct TPDU 80 will be rejected as a duplicate.

To prevent such problems, we must prevent sequence numbers from being used (i.e., assigned to new TPDUs) for a time T before their potential use as initial all the work has been done and the connection should be terminated is not so obvious. One can envision a protocol in which host 1 says: I am done. Are you done too? If host 2 responds: I am done too. Goodbye, the connection can be safely released.

Unfortunately, this protocol does not always work. There is a famous problem that illustrates this issue. It is called the **two-army problem**. Imagine that a white army is encamped in a valley, as shown in Fig. 6-13. On both of the surrounding hillsides are blue armies. The white army is larger than either of the blue armies alone, but together the blue armies are larger than the white army. If either blue army attacks by itself, it will be defeated, but if the two blue armies attack simultaneously, they will be victorious.

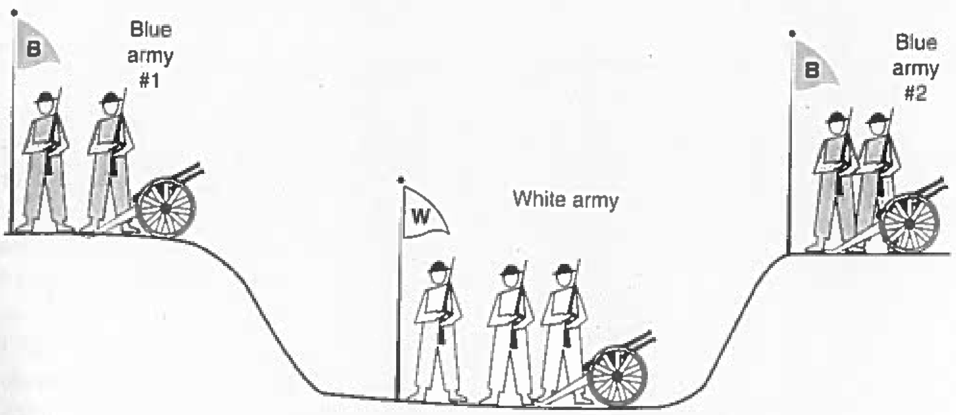


Figure 6-13. The two-army problem.

The blue armies want to synchronize their attacks. However, their only communication medium is to send messengers on foot down into the valley, where they might be captured and the message lost (i.e., they have to use an unreliable communication channel). The question is: Does a protocol exist that allows the blue armies to win?

Suppose that the commander of blue army #1 sends a message reading: "I propose we attack at dawn on March 29. How about it?" Now suppose that the message arrives, the commander of blue army #2 agrees, and his reply gets safely back to blue army #1. Will the attack happen? Probably not, because commander #2 does not know if his reply got through. If it did not, blue army #1 will not attack, so it would be foolish for him to charge into battle.

Now let us improve the protocol by making it a three-way handshake. The initiator of the original proposal must acknowledge the response. Assuming no messages are lost, blue army #2 will get the acknowledgement, but the commander of blue army #1 will now hesitate. After all, he does not know if his acdedicate any buffers, but rather to acquire them dynamically at both ends. Since the sender cannot be sure the receiver will be able to acquire a buffer, the sender must retain a copy of the TPDU until it is acknowledged. On the other hand, for file transfer and other high-bandwidth traffic, it is better if the receiver does dedicate a full window of buffers, to allow the data to flow at maximum speed. Thus, for low-bandwidth bursty traffic, it is better to buffer at the sender, and for high-bandwidth smooth traffic, it is better to buffer at the receiver.

As connections are opened and closed and as the traffic pattern changes, the sender and receiver need to dynamically adjust their buffer allocations. Consequently, the transport protocol should allow a sending host to request buffer space at the other end. Buffers could be allocated per connection, or collectively, for all the connections running between the two hosts. Alternatively, the receiver, knowing its buffer situation (but not knowing the offered traffic) could tell the sender "I have reserved *X* buffers for you." If the number of open connections should increase, it may be necessary for an allocation to be reduced, so the protocol should provide for this possibility.

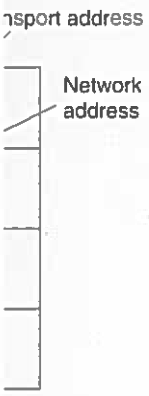
A reasonably general way to manage dynamic buffer allocation is to decouple the buffering from the acknowledgements, in contrast to the sliding window protocols of Chap. 3. Dynamic buffer management means, in effect, a variable-sized window. Initially, the sender requests a certain number of buffers, based on its perceived needs. The receiver then grants as many of these as it can afford. Every time the sender transmits a TPDU, it must decrement its allocation, stopping altogether when the allocation reaches zero. The receiver then separately piggybacks both acknowledgements and buffer allocations onto the reverse traffic.

Figure 6-16 shows an example of how dynamic window management might work in a datagram subnet with 4-bit sequence numbers. Assume that buffer allocation information travels in separate TPDUs, as shown, and is not piggybacked onto reverse traffic. Initially, *A* wants eight buffers, but is granted only four of these. It then sends three TPDUs, of which the third is lost. TPDU 6 acknowledges receipt of all TPDUs up to and including sequence number 1, thus allowing *A* to release those buffers, and furthermore informs *A* that it has permission to send three more TPDUs starting beyond 1 (i.e., TPDUs 2, 3, and 4). *A* knows that it has already sent number 2, so it thinks that it may send TPDUs 3 and 4, which it proceeds to do. At this point it is blocked and must wait for more buffer allocation. Timeout-induced retransmissions (line 9), however, may occur while blocked, since they use buffers that have already been allocated. In line 10, *B* acknowledges receipt of all TPDUs up to and including 4 but refuses to let *A* continue. Such a situation is impossible with the fixed window protocols of Chap. 3. The next TPDU from *B* to *A* allocates another buffer and allows *A* to continue.

Potential problems with buffer allocation schemes of this kind can arise in datagram networks if control TPDUs can get lost. Look at line 16. *B* has now allocated more buffers to *A*, but the allocation TPDU was lost. Since control TPDUs are not sequenced or timed out, *A* is now deadlocked. To prevent this sit-

r's window
es with the
to block.
ıonitor both
apacity can
lged during
ːasurement,
:'s carrying
vledgement
ın be meas-
acity avail-
djusted fre-
r, the Inter-

rcuits, and
ure. In the
ways. For
ort connec-
ıme way is
'ard multi-
ort connec-
te host.



Multiplexing can also be useful in the transport layer for another reason. Suppose, for example, that a subnet uses virtual circuits internally and imposes a maximum data rate on each one. If a user needs more bandwidth than one virtual circuit can provide, a way out is to open multiple network connections and distribute the traffic among them on a round-robin basis, as indicated in Fig. 6-17(b). This modus operandi is called **downward multiplexing**. With *k* network connections open, the effective bandwidth is increased by a factor of *k*. A common example of downward multiplexing occurs with home users who have an ISDN line. This line provides for two separate connections of 64 kbps each. Using both of them to call an Internet provider and dividing the traffic over both lines makes it possible to achieve an effective bandwidth of 128 kbps.

6.2.6 Crash Recovery

If hosts and routers are subject to crashes, recovery from these crashes becomes an issue. If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward. If the network layer provides datagram service, the transport entities expect lost TPDUs all the time and know how to cope with them. If the network layer provides connection-oriented service, then loss of a virtual circuit is handled by establishing a new one and then probing the remote transport entity to ask it which TPDUs it has received and which ones it has not received. The latter ones can be retransmitted.

A more troublesome problem is how to recover from host crashes. In particular, it may be desirable for clients to be able to continue working when servers crash and then quickly reboot. To illustrate the difficulty, let us assume that one host, the client, is sending a long file to another host, the file server, using a simple stop-and-wait protocol. The transport layer on the server simply passes the incoming TPDUs to the transport user, one by one. Partway through the transmission, the server crashes. When it comes back up, its tables are reinitialized, so it no longer knows precisely where it was.

In an attempt to recover its previous status, the server might send a broadcast TPDU to all other hosts, announcing that it had just crashed and requesting that its clients inform it of the status of all open connections. Each client can be in one of two states: one TPDU outstanding, *S1*, or no TPDUs outstanding, *S0*. Based on only this state information, the client must decide whether to retransmit the most recent TPDU.

At first glance it would seem obvious: the client should retransmit only if and only if it has an unacknowledged TPDU outstanding (i.e., is in state *S1*) when it learns of the crash. However, a closer inspection reveals difficulties with this naive approach. Consider, for example, the situation in which the server's transport entity first sends an acknowledgement, and then, when the acknowledgement has been sent, writes to the application process. Writing a TPDU onto the output stream and sending an acknowledgement are two distinct events that cannot be

ıre certainly
ets are lost,
vering from
ıce is better.
little at the
the future.

ıaringly and
is generally
repeating it

little bit on
nall amount
eing lost. A
ime, wastes
ɔd reason.

ɛtworking is
up this soft-
2001).
TPDU and
cessing is to
it specially.
ABLISHED
arts to close

) state when
ıere that the
a user-space
ling process
does is test
ide is trying
DU is being
nditions are
ransport en-

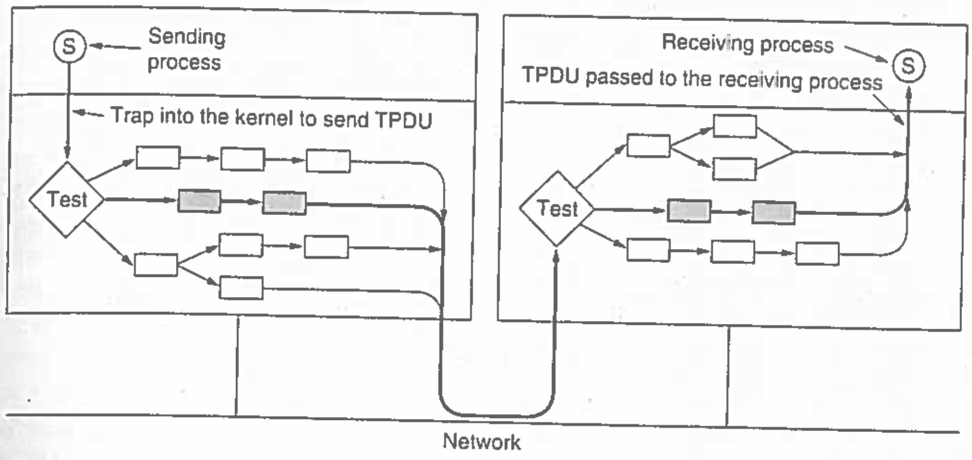

S
Sending process
Trap into the kernel to send TPDU
Test
Receiving process
S
TPDU passed to the receiving process
Test
Network

Figure 6-44. The fast path from sender to receiver is shown with a heavy line. The processing steps on this path are shaded.

In the usual case, the headers of consecutive data TPDUs are almost the same. To take advantage of this fact, a prototype header is stored within the transport entity. At the start of the fast path, it is copied as fast as possible to a scratch buffer, word by word. Those fields that change from TPDU to TPDU are then overwritten in the buffer. Frequently, these fields are easily derived from state variables, such as the next sequence number. A pointer to the full TPDU header plus a pointer to the user data are then passed to the network layer. Here the same strategy can be followed (not shown in Fig. 6-44). Finally, the network layer gives the resulting packet to the data link layer for transmission.

As an example of how this principle works in practice, let us consider TCP/IP. Fig. 6-45(a) shows the TCP header. The fields that are the same between consecutive TPDUs on a one-way flow are shaded. All the sending transport entity has to do is copy the five words from the prototype header into the output buffer, fill in the next sequence number (by copying it from a word in memory), compute the checksum, and increment the sequence number in memory. It can then hand the header and data to a special IP procedure for sending a regular, maximum TPDU. IP then copies its five-word prototype header [see Fig. 6-45(b)] into the buffer, fills in the *Identification* field, and computes its checksum. The packet is now ready for transmission.

Now let us look at fast path processing on the receiving side of Fig. 6-44. Step 1 is locating the connection record for the incoming TPDU. For TCP, the connection record can be stored in a hash table for which some simple function of the two IP addresses and two ports is the key. Once the connection record has been located, both addresses and both ports must be compared to verify that the correct record has been found.