Quesito 1. Sei processi, idendificati dalle lettere A-F, arrivano in successione all'elaboratore (prima A, poi B, etc.) registrandosi come pronti pressoché simultaneamente (con differenze trascurabili ai fini dei calcoli successivi). I processi hanno un tempo stimato di esecuzione di 7, 6, 10, 2, 7 e 3 unità di tempo rispettivamente, mentre i loro valori di priorità statica (per le politiche che ne facciano uso) valgono rispettivamente 5, 1, 4, 2, 6 e 3 (dove 6 è il valore della priorità massima). Per ognuno dei seguenti algoritmi di ordinamento eseguiti dal *dispatcher*, determinare il tempo medio di completamento (*turnaround*) ed il tempo medio di attesa, trascurando i tempi dovuti allo scambio di contesto:

- Priorità senza prerilascio: un processo per volta, fino al completamento
- *Round-robin* senza priorità con quanto fissato in 3 unità di tempo
- *First-Come-First-Served* (FCFS) senza prerilascio: un processo per volta, fino al completamento
- *Shortest-Job-First* (SJF) senza prerilascio: un processo per volta, fino al completamento

Quesito 2 Si raffrontino le politiche di ordinamento (*scheduling*) di Linux e Windows NT, illustrando i benefici attesi dalle rispettive caratteristiche e valutandone criticamente le differenze.

Quesito 3 Un progettista di sistema operativo ha deciso di usare nodi indice (*i-node*) per la realizzazione del proprio *file system*, stabilendo che essi abbiano la stessa dimensione di un blocco del disco, fissata a 256 *byte*. Il nodo indice primario è organizzato in modo da contenere 13 campi di indirizzo di blocchi (indirizzamento diretto), e tre campi puntatori a nodi indice di primo, secondo e terzo livello di indirezione rispettivamente. Utilizzando questa organizzazione e sapendo che un indirizzo di blocco occupa 32 *bit*, si vuole allocare un *file* di 9.800 *record* da 60 *byte* ciascuno, imponendo che un *record* non possa essere suddiviso su due blocchi. Calcolare quanti blocchi verranno utilizzati per allocare il *file* dati e quanti per gestire la sua allocazione tramite *i-node*. Determinare inoltre l'occupazione totale in memoria secondaria.

Quesito 4. SMTP (*Simple Mail Transfer Protocol*), come fedelmente indica il suo nome, è un protocollo di grande semplicità, e per questo di enorme diffusione, che assicura il trasferimento di posta elettronica tra entità denominate *Mail Transfer Agent* (MTA), presso le quali si trovano le caselle postali degli utenti. Uno specifico standard (denominato RFC 822) definisce il formato e la codifica dei messaggi di posta elettronica. Il primo richiede la presenza di determinate intestazioni (*header*), mentre la seconda ammette soltanto caratteri ASCII codificati su 7 *bit*. In questo contesto:

1. si indichi il livello della gerarchia di protocolli TCP/IP sul quale si poggia SMTP
2. si spieghi in che modo gli MTA attuali consentano ai loro utenti il trasferimento di messaggi contenenti parti in codifica diversa da ASCII 7-*bit*, pur continuando ad utilizzare lo standard SMTP.

Quesito 5. Lo schema logico riportato in figura 1 rappresenta la rete dati di una piccola azienda composta da due reparti operativi ed una stanza per i gestori della rete.
Il reparto operativo 1 è composto da 6 postazioni di lavoro (H1-H6) con traffico prevalente di tipo utente-*server*, e facenti capo al SERVER_1, mentre le 3 postazioni H7-H9 del reparto operativo 2 sono caratterizzate da un traffico prevalente utente-*server*, e fanno capo al SERVER_2. Gli ulteriori due *server* (3 e 4) posizionati nella stanza gestori rete offrono servizi *Web* accessibili rispettivamente ad utenti interni all'azienda (*Intranet Server*) ed ad utenti esterni all'azienda (*Internet Server*). A tutti gli utenti interni, con esclusione dei *server*, deve essere anche garantito un traffico di accesso all'*Intranet Server* di 250 Kbps e alla rete *Internet* di 150 Kbps. Sapendo che tutti i dispositivi di rete dell'azienda sono di standard *Fast-Ethernet*, quindi operanti a 100 Mbps, si calcolino i flussi di traffico massimo determinati dalla configurazione *hardware* della rete nel caso peggiore di attività simultanea di tutti gli utenti.

L'azienda ha acquistato dal proprio ISP solo il seguente indirizzo statico di accesso ad *Internet*:

- *host address* = 118.67.131.246
- *subnet mask* = 255.255.255.128
- *default gateway* = 118.67.131.254

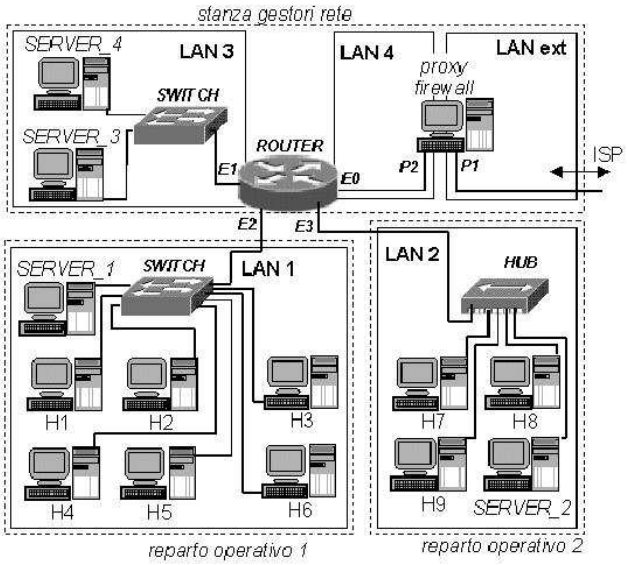

Figura 1: Schema della rete dati dell'azienda.

ed al suo interno vuole sfruttare la tecnologia NAT (*Network Address Trans lation*) presente nel *proxy/firewall* e distribuire gli indirizzi della rete privata 192.168.255.0 tra le sottoreti dell'azienda, così da assicurare ad ogni sottorete esattamente 14 indirizzi IP utilizzabili, anticipando eventuali sviluppi futuri dell'azienda.
Si proponga un possibile piano di assegnazione degli indirizzi IP a tutti i dispositivi della rete, individuando per ognuno la coppia *IP-address/subnet-mask*.

Quesito 6 L'amministratore delegato di una azienda distributrice di *software open source*, di fresca costituzione, è stato incaricato di compiere i passi necessari per registrare l'azienda col nome "*Osrise*", nel dominio di primo livello più appropriato alle attività aziendali, e per dotarla degli indirizzi necessari per ricevere posta elettronica, senza però dover acquisire una connessione *Internet* propria e permanente.
Si dettaglino le azioni che dovranno essere intraprese a tal fine, sia dall'amministratore delegato che da ogni sua eventuale controparte.

Soluzione 1 (punti 5).
Priorità senza prerilascio (un processo per volta, fino al completamento):
◇ T = 0 (arrivo di tutti i processi, inizio esecuzione di E)
a questo istante, da cui si inizia a valutare lo stato dello schedulatore di basso livello, si possono considerare gia arrivati tutti i sei processi in esame. Il S.O. li inserisce nella coda dei processi pronti in base al valore delle rispettive priorità, che sarà così ordinata:

processo	*posizione*	*priorità*
E	1	6
A	2	5
C	3	4
F	4	3
D	5	2
B	6	1

lo schedulatore quindi manderà in esecuzione E.
◇ T = 7 (termine del processo E, inizio esecuzione di A): il processo E termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti il primo processo, a priorità maggiore tra quelli attualmente presenti, cioè il processo A.
◇ T = 14 (termine del processo A, inizio esecuzione di C): il processo A termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti il primo processo, a priorità maggiore tra i presenti, cioè il processo C.
◇ T = 24 (termine del processo C, inizio esecuzione di F): il processo C termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti il primo processo, a priorità maggiore tra i presenti, cioè il processo F. ◇ T = 27 (termine del processo F, inizio esecuzione di D): il processo F termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti il primo processo, a priorità maggiore tra tutti i presenti, cioè il processo D.
◇ T = 29 (termine del processo D, inizio esecuzione di B): il processo D termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti il primo ed unico processo presente, cioè il processo B.
◇ T = 35 (termine del processo B): il processo B termina l'esecuzione e passa allo stato di terminato. Il S.O non trova più processi utente in attesa nella coda dei processi pronti, e quindi manda in esecuzione i processi di sistema in attesa.
Riepilogando otteniamo:

dispatcher con politica a priorità senza prerilascio

processo	arrivo	esecuzione		tempo di turnaround	tempo di attesa
		inizio	termine		
A	0	7	14	14	7
B	0	29	35	35	29
C	0	14	24	24	14
D	0	27	29	29	27
E	0	0	7	7	0
F	0	24	27	27	24
			valori medi:	**22,66**	**16,83**

***Round-robin* senza priorità, con quanto fissato in 3 unità di tempo**:
◇ T = 0 (arrivo di tutti i processi, inizio esecuzione di A): a questo istante, da cui si inizia a valutare lo stato dello schedulatore di basso livello, si possono considerare gia arrivati tutti i sei processi in esame. Il S.O. li inserisce nella coda dei processi pronti in base al rispettivo ordine di arrivo, che viene successivamente mantenuto e che indicheremo come $[A,B,C,D,E,F]$. Lo schedulatore manderà quindi in esecuzione il processo A.
◇ T = 3 (fine quanto per A, inizio esecuzione di B): il processo A sospende l'esecuzione per fine quanto e viene posto all'ultimo posto della coda dei processi pronti, che diventa: $[B,C,D,E,F,A]$. Il S.O preleva da essa il primo processo, cioè il processo B.
◇ T = 6 (fine quanto per B, inizio esecuzione di C): il processo B sospende l'esecuzione per fine quanto e viene posto all'ultimo posto della coda dei processi pronti, che diventa: $[(C,D,E,F,A,B]$. Il S.O preleva da essa il primo processo, cioè il processo C.
◇ T = 9 (fine quanto per C, inizio esecuzione di D): il processo C sospende l'esecuzione per fine quanto e viene posto all'ultimo posto della coda dei processi pronti, che diventa: $[D,E,F,A,B,C]$. Il S.O preleva da essa il primo processo, cioè il processo D.

◇ T = 11 (termine del processo D, inizio esecuzione di E): il processo D termina regolarmente l'esecuzione . Il S.O preleva dalla coda dei processi pronti $[E,F,A,B,C]$ il primo processo, cioè il processo E.
◇ T = 14 (fine quanto per E, inizio esecuzione di F): il processo E sospende l'esecuzione per fine quanto e viene posto all'ultimo posto della coda dei processi pronti, che diventa: $[F,A,B,C,E]$. Il S.O preleva da essa il primo processo, cioè il processo F.
◇ T = 17 (termine del processo F riprende l'esecuzione di A): il processo F termina regolarmente l'esecuzione. Il S.O preleva dalla coda dei processi pronti $[A,B,C,E]$ il primo processo, cioè il processo A.
◇ T = 20 (fine quanto per A riprende l'esecuzione di B): il processo A sospende l'esecuzione per fine quanto e viene posto all'ultimo posto della coda dei processi pronti, che diventa: $[B,C,E,A]$. Il S.O preleva da essa il primo processo, cioè il processo B.
◇ T = 23 (termine del processo B riprende l'esecuzione di C): il processo B termina regolarmente l'esecuzione. Il S.O preleva dalla coda dei processi pronti $[C,E,A]$ il primo processo, cioè il processo C.
◇ T = 26 (fine quanto per C riprende l'esecuzione di E): il processo C sospende l'esecuzione per fine quanto e viene posto all'ultimo posto della coda dei processi pronti, che diventa: $[E,A,C]$. Il S.O preleva da essa il primo processo, cioè il processo E.
◇ T = 29 (fine quanto per E riprende l'esecuzione di A): il processo E sospende l'esecuzione per fine quanto e viene posto all'ultimo posto della coda dei processi pronti, che diventa: $[A,C,E]$. Il S.O preleva da essa il primo processo, cioè il processo A.
◇ T = 30 (termine del processo A riprende l'esecuzione di C): il processo A termina regolarmente l'esecuzione. Il S.O preleva dalla coda dei processi pronti $[C,E]$ il primo processo, cioè il processo C.
◇ T = 33 (fine quanto per C riprende l'esecuzione di E): il processo C sospende l'esecuzione per fine quanto e viene posto all'ultimo posto della coda dei processi pronti, che diventa: $[E,C]$. Il S.O preleva da essa il primo processo, cioè il processo E.
◇ T = 34 (termine del processo E riprende l'esecuzione di C): il processo E termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti $[C]$ il primo ed unico processo presente, cioè il processo C.
◇ T = 35 (termine del processo C): il processo C termina l'esecuzione e passa allo stato di terminato. Il S.O non trova più processi utente in attesa nella coda dei processi pronti, e quindi manda in esecuzione i processi di sistema in attesa.
Riepilogando otteniamo il seguente ordinamento complessivo:
AAA.BBB.CCC.DD.EEE.FFF.AAA.BBB.CCC.EEE.A.CCC.E.C,
con l'andamento specifico dei vari processi riportato in tabella (in maiuscolo l'esecuzione, in minuscolo l'attesa) e riassunto in figura :

proc.	*andamento*
A	AAA.aaa.aaa.aa.aaa.aaa.AAA.aaa.aaa.aaa.A
B	bbb.BBB.bbb.bb.bbb.bbb.bbb.BBB
C	ccc.ccc.CCC.cc.ccc.ccc.ccc.ccc.CCC.ccc.c.CCC.c.C
D	ddd.ddd.ddd.DD
E	eee.eee.eee.ee.EEE.eee.eee.eee.eee.EEE.e.eee.E
F	fff.fff.fff.ff.fff.FFF

dispatcher con politica Round Robin senza priorità

processo	tempo di turnaround	tempo di attesa
A	30	23
B	23	17
C	35	25
D	11	9
E	34	27
F	17	14
valori medi:	**25,00**	**19,16**

First Come First Served (FCFS) senza prerilascio:
◇ T = 0 (arrivo di tutti i processi, inizio esecuzione di A): a questo istante si possono considerare già arrivati tutti i sei processi in esame. Il S.O. li inserisce nella coda dei processi pronti in base al rispettivo ordine di arrivo, che viene successivamente mantenuto e che indicheremo come $[A,B,C,D,E,F]$. Lo schedulatore manderà quindi in esecuzione il processo A.
◇ T = 7 (termine del processo A, inizio esecuzione di B): il processo A termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti $[B,C,D,E,F]$ il primo processo, cioè il processo B.
◇ T = 13 (termine del processo B, inizio esecuzione di C) il processo B termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti $[C,D,E,F]$ il primo processo, cioè il processo C.
◇ T = 23 (termine del processo C, inizio esecuzione di D) il processo C termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti $[D,E,F]$ il primo processo, cioè il processo D.
◇ T = 25 (termine del processo D, inizio esecuzione di E) il processo D termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti $[E,F]$ il primo processo, cioè il processo E.
◇ T = 32 (termine del processo E, inizio esecuzione di F) il processo E termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda dei processi pronti $[F]$ il primo ed unico processo presente, cioè il processo

F.
◇ T = 35 (termine del processo F) il processo F termina l'esecuzione e passa allo stato di terminato. Il S.O non trova più processi utente in attesa nella coda dei processi pronti, e quindi manda in esecuzione i processi di sistema in attesa.
Riepilogando otteniamo il seguente ordinamento complessivo:
AAAAAAA.BBBBBB.CCCCCCCCCC.DD.EEEEEEE.FFF,
con l'andamento specifico dei vari processi riportato in tabella (in maiuscolo l'esecuzione, in minuscolo l'attesa) e riassunto in figura :

processo	*andamento*
A	AAAAAAA
B	bbbbbbb.BBBBBB
C	ccccccc.cccccc.CCCCCCCCCC
D	ddddddd.dddddd.ddddddddddd.DD
E	eeeeeee.eeeeee.eeeeeeeeee.ee.EEEEEEE
F	fffffff.ffffff.fffffffffff.ff.fffffff.FF

dispatcher con politica First Come First Served

processo	tempo di turnaround	tempo di attesa
A	7	0
B	13	7
C	23	13
D	25	23
E	32	25
F	35	32
valori medi:	**22,50**	**16,66**

Shortest Job First (SJF) senza prerilascio:
◇ T = 0 (arrivo di tutti i processi, inizio esecuzione di D): a questo istante si possono considerare gia arrivati tutti i sei processi in esame. Il S.O. li inserisce nella coda dei processi pronti in base alla loro durata di esecuzione, ad iniziare dai più brevi: $[D,F,B,A,E,C]$. (Si noti che abbiamo due processi con la stessa durata: per i calcoli dei tempi medi è assolutamente indifferente come essi vengano posizionati nella coda dei processi pronti.) Lo schedulatore manderà quindi in esecuzione D.
◇ T = 2 (termine del processo D, inizio esecuzione di F): il processo A termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda $[F,B,A,E,C]$ il primo processo, cioè F.
◇ T = 5 (termine del processo F, inizio esecuzione di B): il processo F termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda $[B,A,E,C]$ il primo processo, cioè B.
◇ T = 11 (termine del processo B, inizio esecuzione di A): il processo B termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda $[A,E,C]$ il primo processo, cioè A.
◇ T = 18 (termine del processo A, inizio esecuzione di E): il processo A termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda $[E,C]$ il primo processo, cioè E.
◇ T = 25 (termine del processo E, inizio esecuzione di C): il processo E termina l'esecuzione e passa allo stato di terminato. Il S.O preleva dalla coda $[C]$ l'unico processo rimasto.
◇ T = 35 (termine del processo C): il processo C termina l'esecuzione e passa allo stato di terminato. Il S.O non trova più processi utente in attesa nella coda dei processi pronti, e quindi manda in esecuzione i processi di sistema in attesa.
Riepilogando otteniamo il seguente ordinamento complessivo:
DD.FFF.BBBBBB.AAAAAAA.EEEEEEE.CCCCCCCCCC,
con l'andamento specifico dei vari processi riportato in tabella (in maiuscolo l'esecuzione, in minuscolo l'attesa) e riassunto in figura :

processo	*andamento*
A	aa.aaa.aaaaaa.AAAAAAA
B	bb.bbb.BBBBBB
C	cc.ccc.cccccc.ccccccc.ccccccc.CCCCCCCCCC
D	DD
E	ee.eee.eeeeee.eeeeeee.EEEEEEE
F	ff.FFF

dispatcher con politica Shortest Job First

processo	tempo di turnaround	tempo di attesa
A	18	11
B	11	5
C	35	25
D	2	0
E	25	18
F	5	2
valori medi:	**16,00**	**10.16**

Soluzione 2 (punti 6). La politica di ordinamento adottata da Linux è illustrata a pagina 142 delle dispense di lezione e più diffusamente discussa nelle sezioni 10.1-3 del testo di riferimento "Modern Operating Systems". Le caratteristiche salienti di tale politica sono le seguenti:

1. esegue ordinamento per *thread*, ignorando i processi di appartenenza
2. opera su base di prerilascio e quanti di tempo, dove il prerilascio avviene: all'arrivo di una *thread* a priorità maggiore; a fine quanto; qualora la *thread* in esecuzione si ponga in attesa di un evento

3. seleziona per miglior fattore di merito (*goodness*) nell'ambito di 3 classi di appartenenza staticamente determinata (ossia senza promozione o retrocessione tra classi):
   (A) a tempo reale, con quanti di durata infinita, prerilascio e merito valutato come la priorità effettiva della *thread* elevata di un fattore 1000
   (B) a tempo reale, con quanti di durata finita e variabile, prerilascio e merito valutato come somma della priorità e dell'ultimo quanto non utilizzato
   (C) a divisione di tempo pura, con quanti di durata finita e variabile, prerilascio e merito nullo
4. ad ogni completamento di quanto da parte di una qualunque *thread* j, riassegna un quanto Q a ciascuna *thread* delle ultime due classi, secondo l'equazione: $\forall i, Q_i = Q_i/2 +$ priorità effettiva

a seguito della quale le *thread* di classe (A) hanno assoluta precedenza ed arrivano presto al completamento, quelle di classe (B) si alternano all'esecuzione con quanti di durata descrescente e merito variabile, e quelle di classe (C) hanno limitate opportunità di esecuzione.

La politica di ordinamento adottata da Windows NT è illustrata alle pagine 210-212 delle dispense di lezione e più diffusamente discussa nelle sezioni 11.1-4 del testo di riferimento "Modern Operating Systems". Le caratteristiche salienti di tale politica sono le seguenti:

1. esegue ordinamento per *thread*, ignorando i processi di appartenenza, i quali infatti non hanno stato
2. opera su base di prerilascio e quanti di tempo, dove il prerilascio avviene con modalità simili a quelle di Linux
3. seleziona per migliore valore di priorità corrente, la quale può elevarsi o decrescere rispetto alla priorità base assegnata inizialmente:
   - elevandosi al completamento di un'operazione di I/O, così da favorire un più elevato utilizzo dei relativi dispositivi
   - elevandosi all'ottenimento di una risorsa protetta da semaforo ed alla ricezione di un evento atteso, così da consentire una più rapida risposta agli eventi
   - decadendo ad ogni completamento di quanto
   - elevandosi temporaneamente, per la durata di 2 quanti, nel caso in cui la *thread* sia stata pronta e tuttavia non selezionata per una durata prefissata (dove quest'ultima euristica tende ad alleviare, in modo palliativo, l'evenienza di casi di inversione di priorità)

a seguito della quale lo stile di programmazione e la frequenza di accesso all'I/O (senza alcuna discriminazione di utilità d'uso) possono significativamente promuovere la priorità corrente di *thread* di bassa priorità iniziale, a discapito di *thread* di maggiore utilità di sistema.

Soluzione 3 (punti 4). Richiamiamo i dati del problema:
N_R = numero di *record* che compongono il *file*= 9.800
D_R = dimensione di un *record* = 60 *byte*
D_I = dimensione di un indirizzo = 32 *bit* = 4 *byte*
D_B = dimensione di un blocco = 256 *byte*
N_{RB} = numero di *record* per blocco = $int(D_B/D_R) = int(256/60) = Iint(4,26) = 4$
N_{BF} = numero di blocchi occupati dal *file*= $\lceil N_R/N_{RB} \rceil = \lceil (9.8/4) \rceil = 2.450$
N_{IB} = numero di indirizzi in un blocco = $D_B/D_I = 256/4 = 64$.
I blocchi da indirizzare sono $N_{BF} = 2.450$.

- Di questi, 13 possono essere indirizzati direttamente dall'*i-node* principale.
- Dei rimanenti $2.450 - 13 = 2.437$, N_{IB} (cioè 64) sono indirizzabili indirettamente tramite l'indirizzo di indirezione di primo livello dell'*i-node* principale.
- Dei rimanenti $2.437 - 64 = 2.373$, si possono utilizzare $1 + int(2374/64) = 38$ blocchi da 64 indirizzi indiretti ciascuno tramite il meccanismo di indirezione doppia, di cui i primi 37 saranno utilizzati completamente (per $64 \times 37 = 2.368$ indirizzi) mentre i rimanenti $2.373 - 2.368 = 5$ indirizzi andranno posizionati nel 38^o blocco, come mostrato in figura 2.

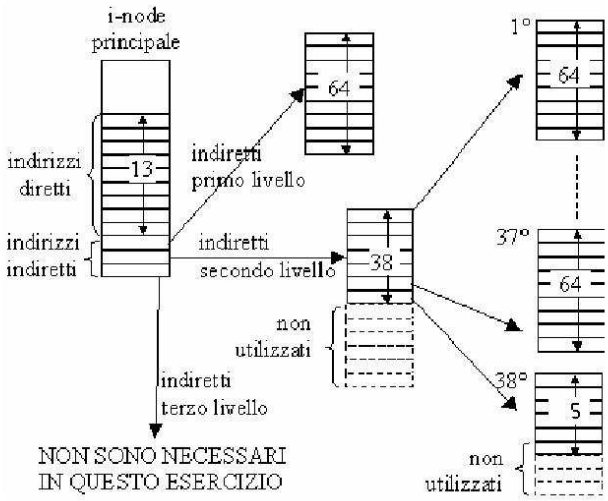

Figura 2: Allocazione del *file*.

Con le indicazioni riportate in figura possiamo concludere che:

- per allocare il *file* dati sono necessari N_{BF} blocchi, ossia 2.450 blocchi
- per gestire l'allocazione del *file* sono necessari:
  1 blocco per l'*i-node* principale
  1 blocco di indirizzi ad indirezione di primo livello
  $1 + 38$ blocchi per l'indirezione di secondo livello
  per un totale di 1+1+1+38 = 41 blocchi
- l'occupazione totale in memoria secondaria vale $2.450 + 41 = 2.491$ blocchi da 256 *byte*, per un totale di $2.491 \times 256 = 637.696$ *byte* equivalenti a circa 622 K*byte*.

Soluzione 4 (punti 4).

1. Come indicato a pagina 429 delle dispense di lezione e come ovvio riflesso della necessità di stabilire una connessione diretta (*end-to-end*) tra gli MTA dei nodi mittente e destinatario, SMTP si poggia necessariamente sul livello trasporto.
2. A tal fine, gli MTA attuali usano un insieme di estensioni di SMTP collettivamente denominate MIME, da *Multipurpose Internet Mail Extensions*, le quali, senza alterare SMTP, semplicemente estendono l'ambito di ciò che possa legalmente essere definito "posta elettronica", fornendo anche delle tecniche di trasformazione dei dati in codifiche compatibili con ASCII 7-*bit*. Un esempio di tale codifica di trasformazione è quella detta `base64`, mediante la quale ogni gruppo di 3 *byte* in qualunque formato viene rappresentato come una sequenza di 4 caratteri in codifica ASCII su 6 *bit*.

Soluzione 5 (punti 7). Il *router* separa le reti locali, isolandone i domini di diffusione distinti. Per il calcolo dei flussi nel caso peggiore possiamo pertanto analizzare il traffico separatamente per ogni LAN.
Per prima cosa si individuano i flussi utili, cioè quelli indicati dal testo. Detti:

X	flusso di dati gestito da un generico utente della rete LAN 1 (da H1 a H6)
Y	flusso di dati gestito da un generico utente della rete LAN 2 (da H7 a H9)
I	flusso di informazioni web per ogni utente relative ad *Intranet* (250 Kbps)
J	flusso di informazioni web per ogni utente relative ad *Internet* (150 Kbps)
ext	flusso di informazioni web generato da utenti esterni per visitare il sito aziendale (ininfluente ai fini del calcolo dei flussi interni)

si ottiene facilmente la distribuzione rappresentata in figura 3.
Passiamo ora ad analizzare le singole reti.

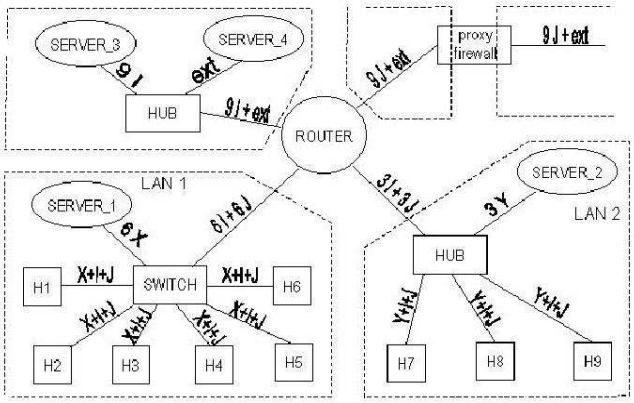

Figura 3: Distribuzione del flussi utili sulla rete.

LAN1
La rete LAN1 è gestita da uno *switch* che segmenta totalmente la rete. Poiché lo *switch* opera in modo da garantire per ogni segmento di rete la banda massima potremmo scrivere le seguenti condizioni, una per ogni ramo dello *switch*:

$$\begin{array}{rcl} X+I+J & \leq & 100 \text{ Mbps} \\ 6X & \leq & 100 \text{ Mbps} \\ 6I+6J & \leq & 100 \text{ Mbps} \end{array}$$

Sostituendo i valori di I e J fissati dal testo otteniamo:

$$\begin{array}{rcl} X & \leq & 100-0,25-0,15 = 99,6 \text{ Mbps} \\ X & \leq & 100/6 = 16,67 \text{ Mbps} \\ 6\times 0,25 + 6\times 0,15 & \leq & 100 \text{ Mbps (sempre soddisfatta)} \end{array}$$

Scegliendo la condizione più ristrettiva otteniamo che il valore di X vale al massimo $16,67$ Mbps.

LAN2
La rete LAN2 è gestita da un *hub* che come sappiamo condivide la banda tra tutti i rami della stessa rete. Abbiamo quindi che in ogni porta dell'*hub* possiamo supporre la presenza di un traffico determinato dalla somma di tutti i traffici utili che fanno capo all'*hub*. Possiamo pertanto scrivere una sola condizione, valida in tutti i rami dell'*hub*:

$$3(Y+I+J)+3Y+(3I+3J) \leq 100 \text{ Mbps}.$$

Raccogliendo e sostituendo i valori di I e J fissati dal testo otteniamo:

$$6Y \leq 100-6I-6J \rightarrow Y = (100-1,5-0,9)/6 = 16,26 \text{ Mbps}.$$

Altre LAN
Il traffico nelle altre reti è limitato dagli utenti o da fenomeni esterni, quali il traffico proveniente dall'esterno verso il sito aziendale, e quindi non produce altri vincoli significativi.

Attribuzione indirizzi IP
L'indirizzo acquistato dall'azienda non è influente nella pianificazione degli indirizzi IP della rete aziendale, se si esclude la configurazione della porta P1 del *proxy/firewall*. Per gli altri dispositivi abbiamo a disposizione un'intera classe C di indirizzi riservati 192.168.255.0. Il dato di progetto da cui partire per l'individuazione del *subnetting* è il numero massimo di *host* per ogni sottorete, che viene fissato in 14 unità utili. Ricordando la suddivisione degli indirizzi e tenendo presente il fatto che in ogni rete o sottorete dobbiamo riservare 2 indirizzi (che quindi non sono utilizzabili per gli *host*), ricaviamo che il *subnetting* dovrà riservare uno spazio di indirizzamento di 16 unità per il campo *host*, enumerabili con 4 *bit* ($2^4 = 16$). Di conseguenza, il campo *subnet* deve essere definito da 4 *bit* (dato che la rete di partenza è di classe C, e quindi mette a disposizione per le sottoreti un totale di 8 *bit*). Abbiamo quindi la condizione riportata in figura 4.
Il numero di reti enumerabili con questa suddivisione dei campi d'indirizzo (4 *bit* per il campo *subnet*) vale $2^4 - 2 = 14$. Tale quantità è sufficiente a soddisfare le nostre esigenze, che sono limitate all'identificazione di sole 4 sottoreti interne.
Queste le caratteristiche degli indirizzi IP delle prime quattro sottoreti utili:

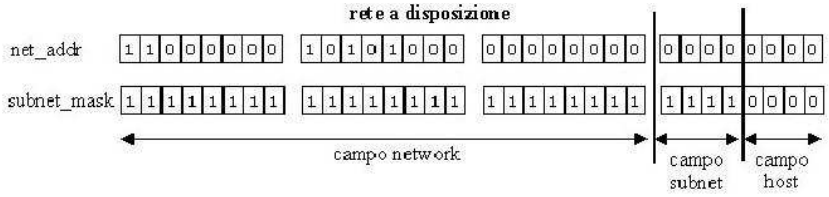

Figura 4: Struttura degli indirizzi utilizzabili dall'azienda.

11000000 . 10101000 . 11111111. 0001 0000	192.168.255.16	inutilizzabile (indirizzo della I sottorete)
11000000 . 10101000 . 11111111. 0001 0001	192.168.255.17	1^o indirizzo di *host* utilizzabile nella Ia sottorete
...	...	...
11000000 . 10101000 . 11111111. 0001 1110	192.168.255.30	ultimo indirizzo di *host* utilizzabile nella Ia sottorete
11000000 . 10101000 . 11111111. 0001 1111	192.168.255.31	inutilizzabile (*broadcast* nella Ia sottorete)
11000000 . 10101000 . 11111111. 0010 0000	192.168.255.32	inutilizzabile (indirizzo della II sottorete)
11000000 . 10101000 . 11111111. 0010 0001	192.168.255.33	1^o indirizzo di *host* utilizzabile nella II sottorete
...	...	...
11000000 . 10101000 . 11111111. 0010 1110	192.168.255.46	ultimo indirizzo di *host* utilizzabile nella II sottorete
11000000 . 10101000 . 11111111. 0010 1111	192.168.255.47	inutilizzabile (*broadcast* nella II sottorete)
11000000 . 10101000 . 11111111. 0011 0000	192.168.255.48	inutilizzabile (indirizzo della III sottorete)
11000000 . 10101000 . 11111111. 0011 0001	192.168.255.49	1^o indirizzo di *host* utilizzabile nella III sottorete
...	...	...
11000000 . 10101000 . 11111111. 0011 1110	192.168.255.62	ultimo indirizzo di *host* utilizzabile nella III sottorete
11000000 . 10101000 . 11111111. 0011 1111	192.168.255.63	inutilizzabile (*broadcast* nella III sottorete)
11000000 . 10101000 . 11111111. 0100 0000	192.168.255.64	inutilizzabile (indirizzo della IV sottorete)
11000000 . 10101000 . 11111111. 0100 0001	192.168.255.65	1^o indirizzo di *host* utilizzabile nella IV sottorete
...	...	...
11000000 . 10101000 . 11111111. 0100 1110	192.168.255.78	ultimo indirizzo di *host* utilizzabile nella IV sottorete
11000000 . 10101000 . 11111111. 0100 1111	192.168.255.79	inutilizzabile (*broadcast* nella IV sottorete)

Di seguito, infine, si riporta invece una possibile attribuzione di indirizzi IP per i dispositivi aziendali interni:

rete	dispositivo	indirizzo IP	*subnet-mask*	
LAN 1	H1	192.168.255.17	255.255.255.240	
	H2	192.168.255.18	255.255.255.240	
	H3	192.168.255.19	255.255.255.240	
	H4	192.168.255.20	255.255.255.240	
	H5	192.168.255.21	255.255.255.240	
	H6	192.168.255.22	255.255.255.240	
	SERVER1	192.168.255.29	255.255.255.240	
	Router E2	192.168.255.30	255.255.255.240	porta *router*-LAN1
LAN 2	H7	192.168.255.33	255.255.255.240	
	H8	192.168.255.34	255.255.255.240	
	H9	192.168.255.35	255.255.255.240	
	SERVER2	192.168.255.45	255.255.255.240	
	Router E3	192.168.255.46	255.255.255.240	porta *router*-LAN2
LAN 3	SERVER3	192.168.255.60	255.255.255.240	
	SERVER4	192.168.255.61	255.255.255.240	
	Router E1	192.168.255.62	255.255.255.240	porta *router*-LAN3
LAN 4	P2	192.168.255.77	255.255.255.240	porta *proxy-router*
	Router E0	192.168.255.78	255.255.255.240	porta *router*-LAN4
LAN ext	P1	118.67.131.246	255.255.255.128	porta *proxy*-ISP

Soluzione 6 (punti 6). Come noto, vi sono due classi di domini di primo livello: i 7 domini "storici" (`com`, `edu`, `gov`, `mil`, `org`, `int`, `net`), istituiti precedentemente all'internalizzazione di *Internet*; e quelli corrispondenti alle aree geografiche nazionali, designate dall'abbreviazione ufficiale a 2 lettere sancita da ISO 3166.
In tale contesto, il nostro amministratore ha 2 scelte: o chiedere la registrazione di "*Osrise*" nel dominio `org`, chiaramente il più appropriato alla connotazione aziendale, oppure registrarsi presso il dominio nazionale competente (p.es. `it`).
Assumiamo che la prima richiesta abbia avuto successo, ciò comportando che non nel dominio non risultino già presenti nomi uguali o similari. L'azienda avrà così il nome di dominio `Osrise.org`, al quale però non verrà automaticamente associato alcun servizio *Internet*.
Affinché la denominazione ottenuta sia operativa, occorrerà allora identificare e comunicare al gestore del dominio `org` l'indirizzo dei *server* (uno o più) presso i quali inoltrare le richieste di connessione SMTP per la posta elettronica.

Il nostro amministratore delegato dovrà pertanto stringere accordi con un ISP che accetti di incorporare nel proprio dominio un indirizzo esterno, dato che non esiste alcun ISP al livello del dominio `org`.
Assumendo che il nostro personaggio abbia ottimi legami con il Dipartimento di Matematica dell'Università di Padova, possiamo immaginare che i gestori del corrispondente dominio (di fatto, un ISP non commerciale) si dicano disponibili all'accordo, fornendo pertanto l'indirizzo del server SMTP del dominio `math.unipd.it`, che, come sappiamo da pagina 419 delle dispense di lezione, è: `mailsrv`.
Per completare l'opera, al nostro basterà passare questa informazione al gestore del dominio `org`, il quale inserirà nella propria base dati DNS un *resource record* di tipo:

```
Osrise.org ... IN MX 1 mailsrv.math.unipd.it
```

con ciò specificando che il dominio *Internet* (IN) denominato `Osrise.org` possiede un *server* di posta (MX, Mail eXchange) primario (1) localizzato all'indirizzo `mailsrv.math.unipd.it`. A questo punto, all'azienda basterà connettersi, secondo accordi ed esigenze, mediante protocollo SMTP, con il nodo `mailsrv.math.unipd.it` per effettuare i propri trasferimenti di posta elettronica.