Quesito 1. La soluzione del problema del produttore-consumatore mostrata in figura 1 è sbagliata. Si individui l'errore, spiegando quale situazione erronea potrebbe verificarsi nell'esecuzione concorrente dei due processi così come formulati, proponendo una versione corretta ottenuta apportando il minor numero possibile di modifiche.

```
void produttore () {
  int prod;
  while (true) {
    prod = produci();
    V(mutex);
    inserisci(prod); /* in struttura comune
    P(mutex);
  }
}
```

```
void consumatore () {
  int prod;
  while (true) {
    P(mutex);
    prod = preleva(); /* da struttura comune
    V(mutex);
    consuma(prod);
  }
}
```

Figura 1: Soluzione errata del problema produttore-consumatore.

Quesito 2. Un progettista vuole confrontare tra loro diverse architetture di *file system* usando come metrica il rapporto inflattivo, prodotto da ciascun architettura, tra l'ampiezza di memoria complessiva necessaria e quella effettivamente richiesta per l'allocazione di un *file* tipo composto da 10.000 strutture indivisibili, ossia non partizionabili su 2 blocchi distinti, ciascuna ampia 100 *byte* (B).
Vogliamo aiutare questo progettista fornendogli il dato riguardante l'architettura NTFS, sotto le seguenti condizioni:

- indici di blocco su disco ampi 64 *bit*
- blocchi su disco ampi 1 kB
- *record* di MFT (*Master File Table*) ampio 1 kB
- 416 B riservati per l'attributo dati in ciascun *record* base
- 800 B riservati per l'attributo dati in ciascuna sua estensione
- sul disco sono presenti 100 sequenze distinte di blocchi contigui, ciascuna ampia 25 blocchi.

Quesito 3. Dato il sistema descritto dalla seguente rappresentazione insiemistica delle assegnazioni di risorsa:

$$\begin{aligned} P &= \{P_1, P_2, P_3, P_4, P_5, P_6\} \\ R &= \{R_1^2, R_2^1, R_3^2, R_4^1\} \\ E &= \{P_2 \to R_2, P_3 \to R_4, P_5 \to R_4, P_6 \to R_3, P_4 \to R_1, \\ &\quad R_1 \to P_1, R_1 \to P_2, R_2 \to P_3, R_3 \to P_4, R_3 \to P_5, R_4 \to P_6\} \end{aligned}$$

si rappresenti il relativo grafo di allocazione delle risorse, determinando se il sistema si trovi attualmente in situazione di stallo oppure no. Successivamente, si verifichi se e come evolva lo stato del sistema a seguito della la richiesta della risorsa R_2 da parte del processo P_1.

Quesito 4. Per far fronte ad un ramo di rete piuttosto congestionato, un nodo M deve frammentare i propri segmenti destinati al nodo D. Tali frammenti sono numerati consecutivamente, usando un campo contatore di ampiezza 4 *bit*, affinché D possa provvedere alla ricostruzione dei segmenti originari. M e D usano entrambi la variante "*Selective Repeat*" dello *Sliding Window Protocol* con finestra di emissione/ricezione ampia 7 (0..6). M fissa inoltre un proprio intervallo di *time out* di ampiezza 5 s. e provvede ad inviare frammenti al ritmo di 1/s. Il tempo impiegato da un frammento per percorrere la distanza M-D, in entrambe le direzioni, è sempre pari ad 1 s. Date queste condizioni, si mostri la sequenza di scambio di frammenti e conferme tra M e D fino alla ricezione di conferma del frammento di indice 6 da parte di M, assumendo che non vada a buon fine il primo invio dei frammenti di indice $2, 4, 5$. Si assuma che, successivamente alla notifica di *time out*, M dia preferenza al re-invio del corrispondente frammento rispetto all'emissione di nuovi frammenti entro l'intervallo corrente di invio.

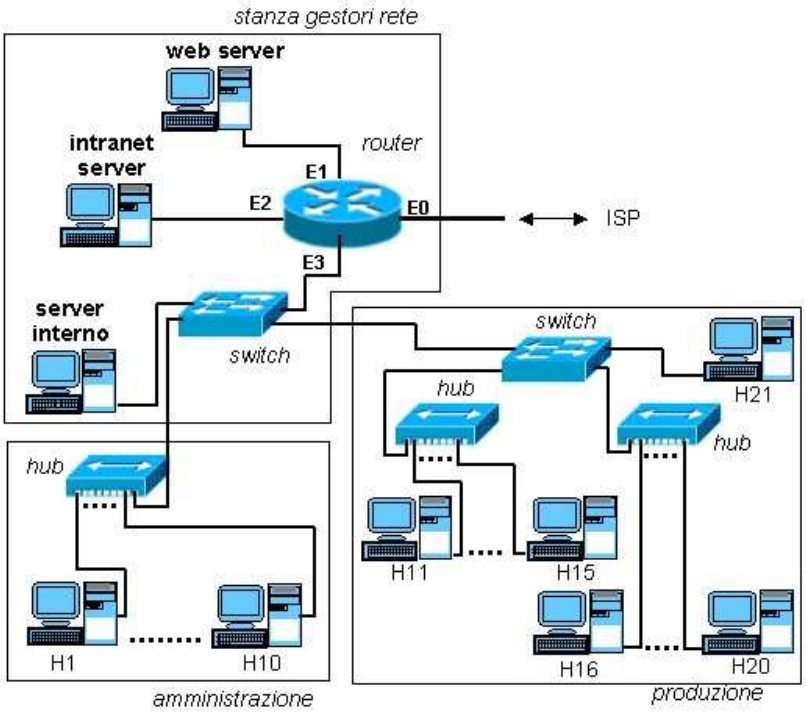

Figura 2: Schema logico della rete dati dell'azienda.

Quesito 5. Lo schema logico riportato in figura 2 rappresenta la rete dati di una piccola azienda composta da due reparti operativi (*amministrazione* e *produzione*) ed una stanza per i gestori della rete.
Il reparto *amministrazione* è composto da 10 postazioni di lavoro con traffico prevalentemente di tipo utente-servente e facenti capo al server interno. Le 11 postazioni del reparto *produzione* sono invece caratterizzate, per il 50%, da traffico di tipo utente-servente che fa capo al server interno e, per il rimanente 50%, da traffico diretto verso l'intranet server. La postazione H21 deve poi avere un traffico utile doppio delle altre 10 del reparto *produzione*. Infine, la postazione web server, collocata nella *stanza gestori rete*, offre servizi Web prevalentemente utilizzati da utenti esterni all'Azienda.
Sapendo che tutti i dispositivi di rete sono di standard *Fast-Ethernet*, pertanto operanti a 100 Mbps, si calcolino i flussi di traffico massimo determinati dalla configurazione fisica della rete nel caso peggiore di traffico simultaneo da parte di tutti gli utenti.

L'Azienda in questione accede ad *Internet* mediante un unico indirizzo IP statico fornito direttamente dal proprio ISP. Al suo interno, invece, l'Azienda decide di condividere gli indirizzi privati di una intera classe C (192.168.9.0), sfruttando la funzione di traduzione degli indirizzi (NAT) resa disponibile dal proprio *router*.
Si proponga una suddivisione degli indirizzi privati utili in sottoreti di uguali dimensioni e con medesima *subnet mask*, istituendo una sottorete per ogni diramazione interna del *router*. Si compili poi una tabella di associazione tra gli indirizzi IP disponibili ed i dispositivi di rete dell'Azienda.

Quesito 6. Un'entità di trasporto TCP ha appena rilevato un *time out* relativo all'invio di un particolare segmento. All'istante t del *time out*, i parametri di controllo di congestione di quell'entità avevano i seguenti valori:

parametro	**simbolo**	**valore corrente (kB)**
finestra di congestione	$CW(t)$	18
finestra di ricezione	$RW(t)$	64
ampiezza massima di segmento	CS	1
soglia	TS	48

Si indichi il valore assunto da CW dopo aver avuto successo con 4 sequenze (*burst*) consecutive di invio di singoli segmenti, ciascuno di ampiezza CS, per un'ampiezza complessiva pari ad una singola finestra di invio, assumendo che il valore RW rimanga costante durante l'intero periodo di interesse.

Soluzione 1 (punti 5). Nel modulo `produttore()` di figura 1 la sezione critica è chiaramente caratterizzata dalla chiamata al sottoprogramma `inserisci()`. Il codice d'ingresso alla sezione critica è però deliminato dall'invocazione della primitiva `V()` sul semaforo `mutex`. Come sappiamo, questa primitiva serve a risvegliare processi eventualmente sospesi sul semaforo, e non a garantire controllo di mutua esclusione nell'accesso alla sezione critica corrispondente. Analogamente, l'uscita dalla sezione critica viene seguita dalla chiamata alla primitiva `P()` sul semaforo `mutex`. Questa primitiva serve invece a sospendere il processo chiamante qualora il corrispondente semaforo risultasse occupato.
Il codice del modulo `consumatore()` usa invece correttamente tali primitive.
È pertanto evidente che l'errore da identificare nel quesito è quello presente nel modulo `produttore()` e consiste nell'inversione dell'ordine di invocazione delle primitive `P()` e `V()`. Tale errore determina, per il produttore, la mancata garanzia di mutua esclusione nel primo accesso alla struttura dati condivisa con il consumatore, oltre causare al modulo `consumatore` un inutile ritardo (pari al tempo d'esecuzione del servizio `produci`) nell'acquisizione del semaforo `mutex`.
Occorre invero notare che l'interazione produttore/consumatore, oltre al problema di mutua esclusione nell'accesso alla struttura dati condivisa, comporta anche un problema di reciproca sincronizzazione rispetto all'effettiva presenza di dati (lato consumatore) ed alla disponibilità di spazio in struttura per nuovi prodotti (lato produttore). Sul modo in cui il codice proposto affronti questa problematica non possiamo in realtà pronunziarci; possiamo però assumere che il codice dei sottoprogrammi `inserisci()` e `preleva()` se ne occupi debitamente, limitandoci pertanto a trattare il problema visibile della mancata mutua esclusione tra i due processi.

Soluzione 2 (punti 5). Osserviamo prima di tutto che NTFS non è in grado di ospitare l'intero *file* campione entro un singolo *record* MFT, poiché lo spazio in esso disponibile (in un *record* base) per il valore dell'attributo dati è largamente inferiore all'ampiezza del *file*. Sarà dunque necessario utilizzare uno schema di memorizzazione alternativo.
Determiniamo allora il numero di blocchi necessari per ospitare il *file* campione con le caratteristiche descritte nel quesito. Essendo ampio 1 kB, ciascun blocco su disco potrà ospitare non più di $N = \lfloor \frac{1.024}{100} \rfloor = 10$ strutture. Per l'intero *file* occorreranno allora $M = \lceil \frac{10.000}{N} \rceil = 1.000$ blocchi.
A questo punto possiamo determinare quante sequenze contigue di blocchi occorreranno per descrivere l'allocazione del *file*. Essendo ciascuna sequenza ampia $A = 25$ blocchi, avremo bisogno di $S = \lceil \frac{M}{A} \rceil = 40$ sequenze distinte, il che è ammissibile essendo $S < 100$. Ciascuna sequenza è descritta in un *record* MFT da una coppia di indici di blocco che occupa $L = 2 \times \frac{64}{8} = 16$ B/sequenza.
Poiché l'attributo dati nel *record* base ha ampiezza massima 416 B, i primi 16 dei quali sono riservati per gli indici {base, limite} del *file*, esso può ospitare fino a $U = \lfloor \frac{416-16}{L} \rfloor = 25$ descrittori di sequenze, con ciò restando $S - U = 15$ sequenze da descrivere su *record* di estensione. Un singolo *record* di estensione sarà più che sufficiente all'uopo, potendo rappresentare fino a $\frac{800}{L} = 50 > S - U = 15$ sequenze.
In conclusione, abbiamo determinato che l'allocazione dell'intero *file*, ampio $C = 10.000 \times 100 = 1.000.000$ B, impiegherà:

- 2 *record* MFT, pari a $2 \times 1 = 2$ kB
- M blocchi su disco, pari a $1.000 \times 1 = 1.000$ kB

per un totale di: $W = 1.002$ kB, a fronte dei C richiesti, con rapporto inflattivo pari a: $\frac{W}{C} - 1 = 2.6\%$.

Soluzione 3 (punti 4). La figura 3 riporta la versione grafica della rappresentazione insiemistica data. In essa si distinguono due percorsi chiusi:

$$\text{percorso 1: } \{P_5 \to R_4 \to P_6 \to R_3 \to P_5\}$$
$$\text{percorso 2: } \{P_2 \to R_2 \to P_3 \to R_4 \to P_6 \to R_3 \to P_4 \to R_1 \to P_2\}$$

in entrambi i quali è presente almeno una risorsa a molteplicità maggiore di 1, per cui non si può affermare nulla a priori, ma va verificato il problema specifico.
Il percorso 1 condivide la risorsa R_3 con il percorso 2, il che comporta la stessa caratteristica per entrambi: o sono entrambi bloccati o si potranno liberare successivamente. Il percorso 1 dipende dall'evoluzione della risorsa R_3, cioè dalla possibilità che essa si liberi indipendentemente dai processi P_5 e P_6. Lo stato della risorsa R_3 dipende anche dal percorso 2, e quindi dall'evoluzione della risorsa R_1 (l'altra risorsa a molteplicità doppia). R_1 è in possesso del processo P_1, che non è incluso nei due percorsi chiusi, e che quindi procede regolarmente e prima o poi dovrebbe rilasciare R_1. Quando ciò avverrà, P_4 potrà accedere alla risorsa R_1, e quindi riprendere ad avanzare.

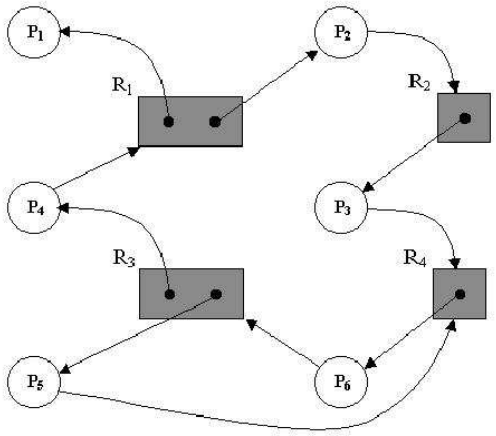

Figura 3: Grafo di allocazione delle risorse per il sistema dato.

Al proprio completamento, P_4 rilascerà un'istanza della risorsa R_3, che potrà essere assegnata al processo P_6, che potrà così avanzare fino al proprio completamento. Infine verrà rilasciata anche R_4, che potrà essere assegnata a P_5 od a P_3. La sequenza di completamento e di rilascio procede poi a catena, consentendo la ripresa di tutti gli altri processi in attesa.
Si può pertanto concludere che la situazione proposta non è di stallo.
Qualora invece il processo P_1 richiedesse la risorsa R_1, partendo dalla situazione iniziale, questo creerebbe un nuovo percorso chiuso che manterrebe sospeso anche P_1, così determinando lo stallo del sistema.

Soluzione 4 (punti 7). La versione base dello *Sliding Window Protocol* prevede che D confermi ad M soltanto l'arrivo dei frammenti di indice immediatamente successivo all'ultima ricezione corretta, confermata ed entro l'intervallo di ricezione corrente, scartando invece tutti quelli arrivati fuori ordine. Per contro, M, dopo aver inviato tutti i frammenti di una finestra, dovrà aspettare le corrispondenti conferme, provvedendo alla riemissione di quelli per i quali si sia verificato *time out*. Ad ogni conferma inviata, D avanza di 1 la propria finestra di ricezione. Ad ogni conferma ricevuta, M avanza di 1 la propria finestra di invio, potendo così procedere con l'invio di nuovi frammenti.
La variante *Selective Repeat* di tale protocollo vuole ridurre il carico di rete derivante dal reinvio di frammenti già correttamente ricevuti. A tal fine, essa consente a D di trattenere nella propria finestra di ricezione, ma senza inviare conferma, i frammenti di indice entro l'intervallo di ricezione corrente, che siano stati ricevuti fuori ordine, inviando poi conferma ad M di intere sottosequenze contigue così prodottesi, ed avanzando conseguentemente il proprio intervallo.
Con queste premesse ed i dati forniti dal quesito possiamo facilmente ricostruire la sequenza di comunicazioni tra M e D di interesse, ossia fino alla conferma del frammento di indice 6. La sequenza è illustrata in figura 4.

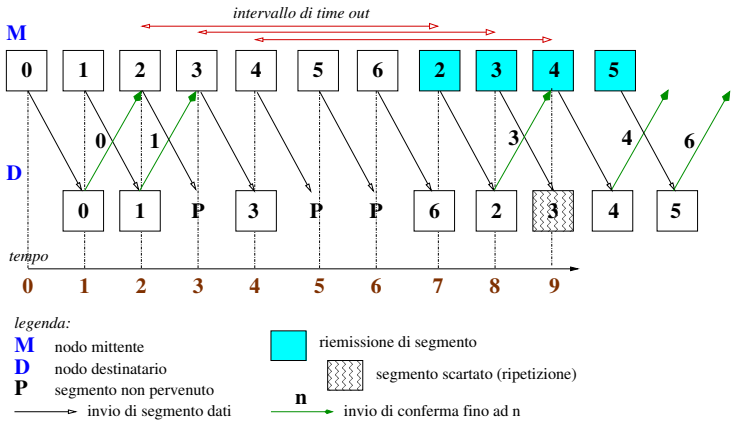

Figura 4: Sequenza di comunicazioni tra M e D fino alla conferma del frammento di indice 6.

Come indicato dal quesito, il primo mancato arrivo concerne il frammento di indice 2, che sappiamo essere stato inviato all'instante 2. Il *time out* corrispondente a questo invio scadrà pertanto all'instante $2+5=7$.
A quell'instante, M avrà inviato frammenti consecutivi sino all'indice 6, senza aver però ricevuto alcuna nuova conferma, poiché D non avrà comunque ricevuto i frammenti di indice 4 e 5 e non avrà confermato i frammenti di indice 3 e 6, giunti fuori sequenza. Come specificato nel quesito, al segnale di *time out* M preferirà ripetere l'emissione dei frammenti non confermati (di indice 2-5).
All'arrivo del frammento 2, D potrà inviare conferma fino all'indice 3, dovendo però scartare il nuovo invio del frammento 3 (già correttamente ricevuto). All'arrivo del frammento 4, D confermerà fino all'indice 4, mentre all'arrivo del frammento 5, potrà confermare fino all'indice 6.

Soluzione 5 (punti 7). Il *router* aziendale separa le sottoreti interne isolandone i domini di diffusione. Ai fini del calcolo di flusso massimo, possiamo pertanto analizzare il traffico separatamente per ogni sottorete. Per prima cosa, occorre individuare i flussi utili che derivano dalle caratteristiche dell'architettura di rete descritta nel quesito:

flusso dati	**proveniente da**	**sottorete**
X	nodi Hi, $i \in \{1..10\}$	LAN 3, reparto *amministrazione*
Y	nodi Hj, $j \in \{11..20\}$	LAN 3, reparto *produzione* sotto *hub*
Z	nodo H21	LAN 3, reparto *produzione* sotto *switch*

Tabella 1: Tipologie di flussi presenti nella rete aziendale.

Utilizzando questa simbologia ed i rispettivi flussi si può impostare la ripartizione mostrata in figura 5, che discuteremo di seguito per ciascuna sottorete singolarmente.

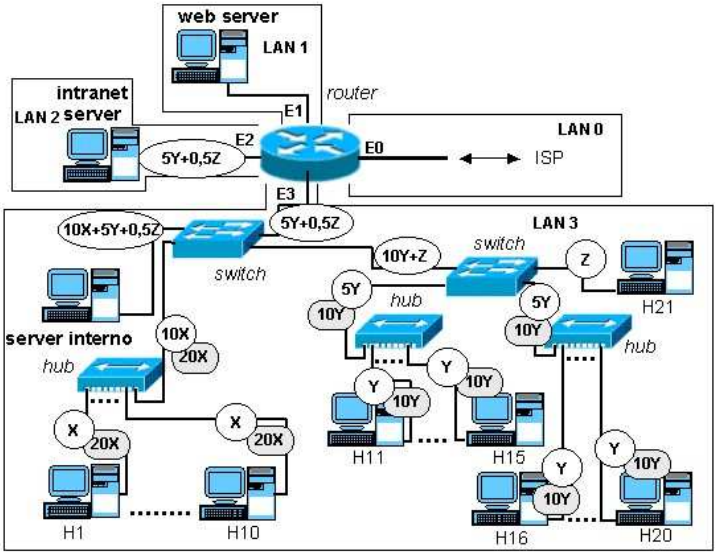

Figura 5: Ripartizione del traffico in ciascun ramo della rete aziendale.

LAN0 e LAN1:
Il loro traffico dipende solo dagli utenti che navigano sul sito *Web*, e quindi sono limitati solo dal traffico esterno.
LAN3:
Questa rete interna, che fa capo alla porta E3 del *router*, racchiude al suo interno sia il reparto *amministrazione* che quello *produzione*, ed è gestita globalmente da uno *switch* che segmenta totalmente le due aree di lavoro.
Il reparto *amministrazione* è deliminato da un *hub* ad 11 porte, 10 delle quali collegate con i nodi Hi, $i \in \{1..10\}$ ciascuno dei quali produce un traffico utile X. Come sappiamo, l'*hub* riflette su ciascuna delle sue 11 porte la somma di tutti i suoi flussi utili. Pertanto, su ogni ramo dell'*hub* vale la seguente condizione: 100 Mbps $\geq X \times 10 + 10X = 20X$, da cui otteniamo: $X \leq \frac{100}{20} = 5$ Mbps.
L'architettura di rete del reparto *produzione* prevede invece uno *switch* collegato a due *hub* ed ad un nodo singolo (H21). Poiché ciascuno dei due *hub* collega un gruppo di 5 nodi, essi sono del tutto uguali ai fini del nostro calcolo, per cui il valore dei flussi determinato per l'uno varrà anche per l'altro.

Con ragionamento analogo a quello appena svolto per il reparto *amministrazione*, fissando l'attenzione sul gruppo di nodi Hi, $i \in \{11..15\}$, abbiamo: 100 Mbps $\geq Y \times 5 + 5Y = 10Y$, da cui: $Y \leq \frac{100}{10} = 10$ Mbps, dove, come detto, lo stesso risultato varrà anche per i nodi Hj, $j \in \{16..20\}$.
Per il calcolo di Z, il traffico utile del nodo H21, direttamente collegato allo *switch* dobbiamo effettuare un discorso a parte piuttosto articolato. Cominciamo col ricordare che lo *switch* opera in modo da garantire ad ogni segmento di rete la banda massima. Pertanto, considerando ogni ramo dello *switch* che delimita il reparto *produzione*, potremmo imporre le seguenti condizioni:

flusso (Mbps)	**ramo diretto a**
$Z \leq 100$	H21
$10Y \leq 100$	Hi, $i \in \{11..20\}$
$10Y + Z \leq 100$	*switch* di *stanza gestori rete*

Tuttavia, a queste condizioni dobbiamo affiancare quelle determinate dai flussi diretti sui rimanenti 3 rami dello *switch* della *stanza gestori rete*. Di tali flussi: uno (pari a $10X$) proviene dal reparto *amministrazione*; un altro, sul ramo E3, porta il 50% del traffico proveniente dal reparto *produzione* (pari a $(10Y + Z)/2$); il terzo, infine, porta il rimanente 50% del traffico del reparto *produzione* insieme all'intero traffico del reparto *amministrazione*. Da ciò otteniamo 3 ulteriori condizioni:

flusso (Mbps)	**ramo diretto a**
$10X \leq 100$	Hj, $i \in \{1..10\}$
$(10Y + Z)/2 \leq 100$	intranet server via E3
$10X + (10Y + Z)/2 \leq 100$	server interno

Infine, l'ultimo dato di progetto da considerare impone che il traffico disponibile alla postazione H21 sia doppio rispetto a quello delle altre postazioni del reparto *produzione*, ossia: $Z = 2Y$.
Possiamo ora imporre il valore di $X = 5$ Mbps precedentemente calcolato, nell'equazione che vogliamo massimizzare, $10X + (10Y + Z)/2 = 100$, ottenendo: $5Y + \frac{Z}{2} = 50 \Rightarrow 5Y + \frac{2Y}{2} = 6Y = 50 \Rightarrow Y = \frac{50}{6} = 8,\bar{3}$ e $Z = 2Y = 16,\bar{6}$.
Come sappiamo dal quesito, ogni postazione Hi, $i \in \{11..20\}$ del reparto *produzione* ripartisce la propria quota Y di traffico al 50% (dunque $4,1\bar{6}$) tra il server interno ed il server intranet. Similmente, farà la postazione H21 con la propria quota Z di traffico utile.
LAN2:
La rete LAN2 è composta solo dal server interno, ed il suo traffico utile nel caso peggiore è dato dalla somma dei traffici interni precedentemente calcolati, e vale: $5Y + \frac{Z}{2} = 50$ Mbps.

Attribuzione di indirizzi IP:
L'indirizzo acquistato dall'Azienda non influisce nella ripartizione degli indirizzi IP interni alla rete aziendale, ma solo nella configurazione della porta E0 del *router*. Per gli altri dispositivi abbiamo a disposizione un'intera classe C di indirizzi riservati 192.168.9.0. Il dato di progetto da cui partire per l'individuazione del *subnetting* è che tutte le sottoreti abbiano le stesse dimensioni e la stessa *subnet mask*.
Il numero S di sottoreti utili rappresentabili con una maschera di *subnet* ampia N *bit* è dato dall'equazione $S = 2^N - 2$. Poiché il quesito ci richiede di creare 3 sottoreti, imporremo $S = 3 = 2^n - 2$, ottenendo $n = log_2(S+2) = log_2 5 = 2,32$, e, conseguentemente $N = \lceil n \rceil = 3$, fissato al più piccolo valore intero maggiore di n.
Come mostrato in figura 6, effettueremo il *subnetting* richiesto partizionando la parte di nodo dell'indirizzo di classe C dato, ampia 8 *bit*, in una parte *subnet* ampia 3 *bit* ed una parte di nodo ridotta ai rimanenti 5 *bit*. Con tali 5 *bit* possiamo rappresentare $2^5 - 2 = 30$ indirizzi IP utili per sottorete, ampiamente sufficienti per le postazioni presenti in ciascuna sottorete dell'azienda.

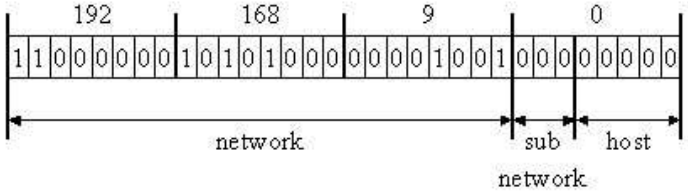

Figura 6: Ripartizione dei campi dell'indirizzamento IP interno all'Azienda.

LAN 1	11000000.10101000.00001001.001	0.0000	192.168.9.32	indirizzo di LAN 1
	11000000.10101000.00001001.001	0.0001	192.168.9.33	1^o indirizzo di nodo in LAN 1
	...		...	...
	11000000.10101000.00001001.001	1.1110	192.168.9.62	30^o indirizzo di nodo in LAN 1
	11000000.10101000.00001001.001	1.1111	192.168.9.63	diffusione in LAN 1
LAN 2	11000000.10101000.00001001.010	0.0000	192.168.9.64	indirizzo di LAN 2
	11000000.10101000.00001001.010	0.0001	192.168.9.65	1^o indirizzo di nodo in LAN 2
	...		...	...
	11000000.10101000.00001001.010	1.1110	192.168.9.94	30^o indirizzo di nodo in LAN 2
	11000000.10101000.00001001.010	1.1111	192.168.9.95	diffusione in LAN 2
LAN 3	11000000.10101000.00001001.011	0.0000	192.168.9.96	indirizzo di LAN 3
	11000000.10101000.00001001.011	0.0001	192.168.9.97	1^o indirizzo di nodo in LAN 3
	...		...	...
	11000000.10101000.00001001.011	1.1110	192.168.9.126	30^o indirizzo di nodo in LAN 3
	11000000.10101000.00001001.011	1.1111	192.168.9.127	diffusione in LAN 3

Tabella 2: Ripartizione degli indirizzi nelle prime 3 sottoreti interne disponibili.

Le prime 3 sottoreti utili rappresentabili il *subnetting* appena descritto avranno pertanto le caratteristiche illustrate in tabella 2.
Da questa ripartizione possiamo determinare una possibile attribuzione di indirizzi IP alle varie postazioni della rete aziendale, come mostrato in tabella 3.

rete	**dispositivo**	***IP address***	***subnet mask***	
LAN 3	H1	192.168.9.97	255.255.255.224 (/27)	
	H2	192.168.9.98	/27	
	...			
	H10	192.168.9.106	/27	
	H11	192.168.9.107	/27	
	H12	192.168.9.108	/27	
	...			
	H20	192.168.9.116	/27	
	H21	192.168.9.117	/27	
	server interno	192.168.9.125	/27	
	Router E3	192.168.9.126	/27	porta *router*-LAN3
LAN 2	intranet server	192.168.9.93	/27	porta server intranet
	Router E2	192.168.9.94	/27	porta *router*-LAN2
LAN 1	server web	192.168.9.61	/27	porta server web
	Router E1	192.168.9.62	/27	porta *router*-LAN1
LAN 0	*Router* E0	dati forniti dall'ISP		

Tabella 3: Una possibile attribuzione di indirizzi IP interni alle postazioni della rete aziendale.

Soluzione 6 (punti 4). Sappiamo che le entità di trasporto TCP usano un'euristica detta "algoritmo *slow start*" per ripristinare un valore ottimale di CW a seguito di un *time out*, usando poi il valore minimo tra CW ed RW per fissare l'ampiezza della propria finestra di invio. Sappiamo anche che all'algoritmo viene applicato un parametro di soglia TS oltre il quale CW debba cambiare il proprio ritmo di crescita. Ad ogni occorrenza di *time out*, il valore di tale parametro, inizializzato alla dimensione massima di segmento, $2^{16} - 1$ *byte*, viene ridotto alla metà del valore corrente di CW. Vediamo allora in dettaglio come varia il valore di CW, espresso in kB, nell'intervallo di tempo di interesse del quesito:

◇ **istante t_0 di trattamento del *time out***
$TS(t_0) = \frac{CW(t)}{2} = \frac{18}{2} = 9$
$CW(t_0) = CS = 1 < TS = 9 < RW = 48$

◇ **istante t_1 successivo al 1^o *burst* andato a buon fine**
$CW(t_1) = CW(t_0) + \frac{CW(t_0)}{CS} \times CS = 1 + 1 = 2 < TS < RW$

◇ **istante t_2 successivo al 2^o *burst* andato a buon fine**
$CW(t_2) = CW(t_1) + \frac{CW(t_1)}{CS} \times CS = 2 + 2 = 4 < TS < RW$

◇ **istante t_3 successivo al 3^o *burst* andato a buon fine**
$CW(t_3) = CW(t_2) + \frac{CW(t_2)}{CS} \times CS = 4 + 4 = 8 < TS < RW$

◇ **istante t_4 di inizio emissione del 4^o *burst*, tutto a buon fine**

- **Segmento** 1 **di** $\frac{CW(t_3)}{CS} = 8$
  $CW(t_4) = CW(t_3) + CS = 8 + 1 = 9 = TS < RW$.
  Avendo raggiunto il valore di soglia, CW dovrà d'ora in poi crescere linearmente, al ritmo di $\frac{CS \times CS}{CW}$ *byte* per invio, ossia al più di 1 CS dopo la conferma di $\frac{CW}{CS}$ segmenti.
- ...
- **Segmento** 8 **di** 8
  $CW(t_{11}) = CW(t_4) + 7 \times \frac{CS \times CS}{CW(t_4)} = 9 + \frac{7}{9}$, con valori espressi in unità CS.

Notiamo dunque che, a regime, ossia dopo altri 2 invii di segmento andati a buon fine, il controllo di congestione (ovvero la modalità del protocollo operativa dall'istante in cui $CW = TS$, quando viene interrotta la progressione *slow start*) incrementerà il valore di CW di 1 CS per ogni emissione con successo dell'intera finestra di invio corrente.