1.

a. The utilization of the CPU is $\frac{5}{19}$, the utilization of the disk is $\frac{8}{19}$, and the utilization of the network is $\frac{6}{19}$.

b. The bandwidth of the web server is $\frac{1}{19}$ requests/msec, (i.e., $\frac{1000}{19} \approx \mathbf{52.63}$ requests/sec)

c. For N=2, we can see from the table below that a pattern happens periodically with the time interval = 21msec, the system enters into the steady state. The utilization of the CPU, disk, and network are $\frac{10}{21}, \frac{16}{21}, \frac{12}{21}$, respectively.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
CPU	A	A	B	B							A	A							A	B	B	A	A	
Disk			A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B						A
Net													A	A	A	A	A	A				B	B	B

	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
CPU				B	B	B		A	A							A	B	B	A	A		
Disk	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B						A	A
Net	B	B	B							A	A	A	A	A	A				B	B	B	B

	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
CPU			B	B	B		A	A							A	B	B	A	A			
Disk	A	A	A	A	A	A	B	B	B	B	B	B	B	B						A	A	A
Net	B	B							A	A	A	A	A	A				B	B	B	B	B

d. For N=2, the capacity of the web server is $\frac{2}{21}$ requests/msec. (i.e., $\frac{2000}{21} \approx \mathbf{95.24}$ requests/sec). The **blue** cells are the time instant that a process will be finished.

e. For N=3, we can similarly find a pattern with the time interval = 16 msec. The utilization of the CPU, disk, and network are $\frac{10}{16}, \frac{16}{16}, \frac{12}{16}$, respectively. And the capacity of the web server is **0.125** requests/msec. (i.e., **125** requests/sec).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
CPU	A	A	B	B	C	C					A	A							A	B	B	A	A	
Disk			A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B	C	C	C	C	C	C
Net													A	A	A	A	A	A				B	B	B

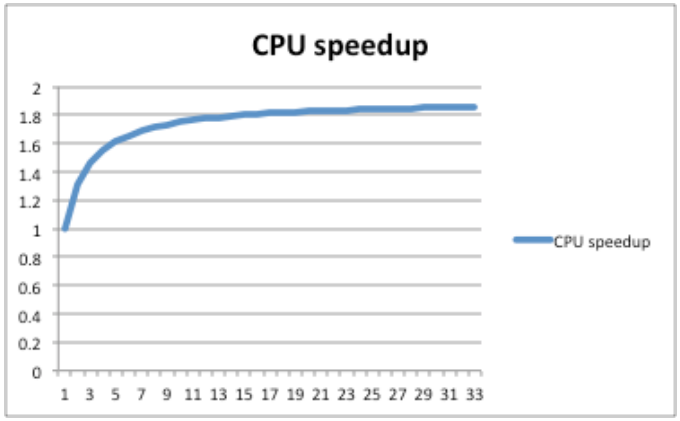
	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
CPU			C	C	B	B	B				C	A	A	C	C			
Disk	C	C	A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B
Net	B	B	B		C	C	C	C	C	C				A	A	A	A	A

f. The maximum capacity of this web server is **125** requests/sec. The reason is that the **utilization of the Disk is 100** when N=3, it becomes the bottleneck resource.

g. The ***Disk*** should be sped up, since it's the bottleneck of the system.

2. Here, take MPL = 2 as an example.

a. The speedup is $\boldsymbol{speedup_{CPU} = \frac{1}{1-\frac{10}{21}*(1-\frac{1}{r})}}$

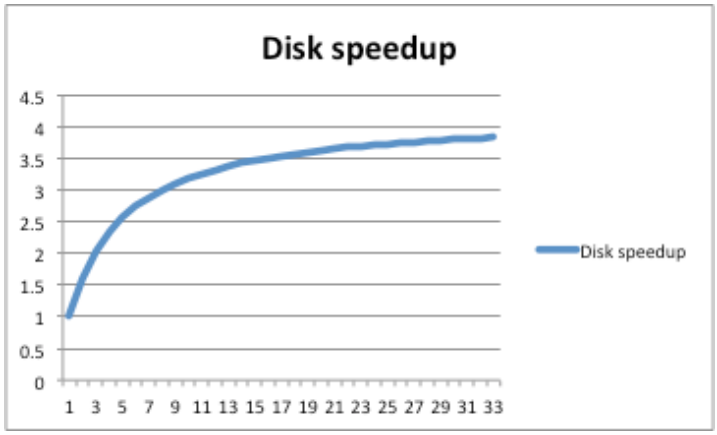


b. When r goes to infinity, the speedup is bounded by $\frac{1}{1-\frac{10}{21}} = \frac{21}{11}$.

c.

Disk:

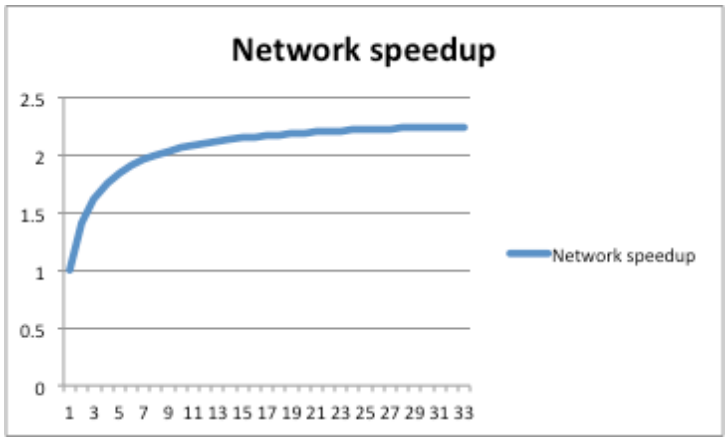
The speedup is $\boldsymbol{speedup_{Disk}} = \frac{1}{1-\frac{16}{21}*(1-\frac{1}{r})}$



When r goes to infinity, the speedup is bounded by $\frac{1}{1-\frac{16}{21}} = \frac{21}{5}$.

Network:

The speedup is $\boldsymbol{speedup_{Network}} = \frac{1}{1-\frac{12}{21}*(1-\frac{1}{r})}$



When r goes to infinity, the speedup is bounded by $\frac{1}{1-\frac{12}{21}} = \frac{21}{9}$.

d. We need to speedup the Disk and Network:

For Disk: $\frac{16}{21*r} = \frac{10}{21}$, so we have r = 1.6.

For Network: $\frac{12}{21*r} = \frac{10}{21}$, so we have r = 1.2.

e. The resulting capacity in part d is $\frac{2}{15}$ requests/msec. (i.e., $\frac{2000}{15} \approx \mathbf{133.33}$ requests/sec)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
CPU	A	A	B	B				A	A				B	B	A	A	A			B	B	B	A	A
Disk			A	A	A	A	A	B	B	B	B	B						A	A	A	A	A	B	B
Net										A	A	A	A	A	B	B	B	B	B					

	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
CPU				B	B	A	A	A			B	B	B	A	A				B	B	A	A
Disk	B	B	B						A	A	A	A	A	B	B	B	B	B				
Net	A	A	A	A	A	B	B	B	B	B						A	A	A	A	A	B	B

3. For this problem, you can calculate the expected access time by $\boldsymbol{r * HitTime + (1 - r) * MissTime}$, where $\boldsymbol{r}$ is the cache hit rate. Assuming that each cache access takes one unit of time T, and each main memory access takes 8 units of time, i.e., 8T. Therefore, the expected access time for each access is $\boldsymbol{r * T + (1 - r) * 8T}$.

a. Under the situation where the cache hit rate is $\boldsymbol{r = 95\%}$: 1. Cutting the latency of the main memory by 50%, we know that $\boldsymbol{speedup_{overall} = \frac{0.95*T+0.05*8T}{0.95*T+\frac{0.05*8T}{2}} \approx 1.174}$. 2. Cutting the latency of the cache by 20%, we know that $\boldsymbol{speedup_{overall} = \frac{0.95*T+0.05*8T}{0.95*T*0.8+0.05*8T} \approx 1.164}$

b. Assuming the cache hit rate is $\boldsymbol{r}$, so the main memory hit rate is $\boldsymbol{1- r}$. Similarly, we can calculate the speedup in these two situations: The $\boldsymbol{speedup_{cache} = \frac{r*T+(1-r)*8T}{r*T*0.8+(1-r)*8T}}$, and the $\boldsymbol{speedup_{main\ memory} = \frac{r*T+(1-r)*8T}{r*T+(1-r)*8T*0.5}}$.

if $\boldsymbol{speedup_{cache} > speedup_{main\ memory}}$, we get $\boldsymbol{r > \frac{20}{21} \approx 0.9524}$. Therefore, if cache hit rate is more than $\mathbf{0.9524}$, then we should cut the latency of the cache by 20%. Otherwise, we should cut the latency of the memory by 50%.

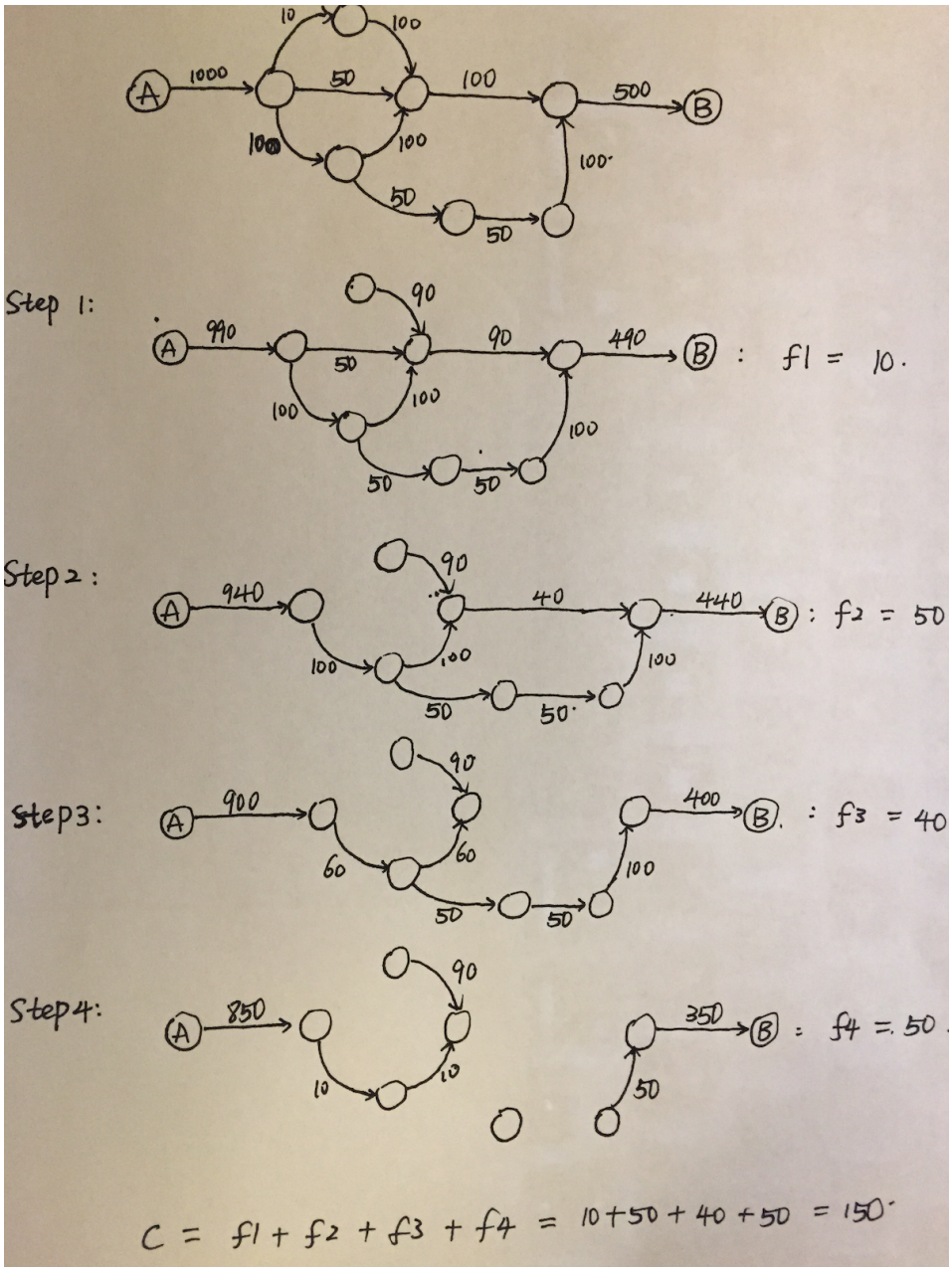
c. Assuming that the hit rate is $\boldsymbol{h}$, after applying these two optimization methods, the total latency will become $\boldsymbol{h * T * 0.8 + (1 - h) * 8T * 0.5}$. Therefore, the overall speedup is $\boldsymbol{speedup_{overall} = \frac{h*T+(1-h)*8T}{h*T*0.8+(1-h)*8T*0.5} = \frac{40-35h}{20-16h}}$.

4. a. The packet loss rate over the H hops is $p + p * (1-p) + \cdots + p * (1-p)^{H-1} = 1 - (1-p)^H$. Therefore, the possibility that a packet reaches the destination is $(1-p)^H$. The effective throughput is $\frac{1500-150}{1500} * 10Mbps * (1-p)^H$.

b. When $H = 1$ and $P = 0.01$, the effective throughput is 8.91Mbps.

c. We need to show that $\frac{1500-150}{1500} * 10Mbps * (1-p)^L \geq 6Mbps$. Therefore, we can obtain $p \leq 1 - (\frac{2}{3})^{\frac{1}{L}}$

5.



a. The capacity is 150. In particular, you can get this result by eliminating links in different ways.

b. Assuming the source and destination of the graph are represented by ***s*** and ***t***, respectively.
1) Apply the algorithm in a) to get the residual graph.
2) For each eliminated edge (*x, y*), if there is a **positive-capacity path** from *s* to *x* and from *y* to *t* in the residual graph, then the edge (*x, y*) is such an edge that can increase the capacity of the network by upgrading the capacity of this single existing link in the network.

In particular, there is **a simple brute force method**: we can try to increase the capacity of each edge, and then check whether the capacity of the network has been increased. If so, it is such an edge; otherwise, it is not such an edge. Similarly, it can be used to c).

In our particular network, upgrade the link of capacity 100 that is between the link with capacity 500 and the link with capacity 50.

c. For this question, we need to find such an edge that will increase the capacity of the network the most. To achieve this goal, we need to modify the algorithm in b) as follows:
1) Apply the algorithm in a) to get the residual graph.
2) For each eliminated edge (*x, y*), it needs to find the maximum path capacity from *s* to *x*, and find the maximum path capacity from *y* to *t*, then minimum path capacity of them is the capacity that this edge can increase in the network. Specifically, you can find the maximum path capacity by using **DFS** algorithm or others.
3) After visiting all the eliminated edges, it report the maximum capacity obtained from 2) as the result.
Finally, you can find that the network capacity is 10 more by upgrading a single network link (i.e., the capacity becomes 160).

d. According to the "**Max-flow min-cut theorem**" (**http://en.wikipedia.org/wiki/Max-flow_min-cut_theorem**), which states that in a flow network, the maximum amount of flow passing from the source to the sink is equal to the minimum capacity that, when removed in a specific way from the network, causes the situation that no flow can pass from the source to the sink. If it finds one or more minimum s-t cut(s), which can be a set of edges (links), then it will report the edge with the maximum capacity. Since it removes the edge with the maximum weight from the minimum cut, the new cut will be the minimum cut, so the capacity of the network (i.e., max-flow) will decrease the most.

According to the above analysis, we should remove the link of capacity 100 that is between the link with capacity 500 and the link with capacity 50.