Keyan Vakil
CS350 - HW1

1. System consists of CPU, Disk, and Network Interface

Usage of each resource for Processing a requests:

1. CPU: 2 ms
2. Disk: 6 ms
3. CPU: 1 ms
4. Network Interface: 4 ms
5. CPU: 2 ms

Assume a 1 represents the resource is active and an empty cell is idle and each part of the graph represents 1 ms

```
| CPU               | 1 | 1 |   |   |   |   |   |
 | 1 |   |   |   |   | 1 | 1 |
| Disk              |   |   | 1 | 1 | 1 | 1 | 1 | 1
 |   |   |   |   |   |   |   |   |
| Network Interface |   |   |   |   |   |   |   |
 |   | 1 | 1 | 1 | 1 |   |   |
```

total running time = 2 ms + 6 ms + 1 ms + 4 ms + 2 ms = 15 ms

a. Utilization = 1 - (Idle time / total running time)

CPU Utilization:

CPU Idle time = 6 ms + 4 ms = 10 ms
U = 1 - (10 ms / 15 ms)
U = (5 ms / 15 ms)
U = 0.33 = 33 % CPU Utilization

Disk Utilization:

Disk Idle time = 2 ms + 1 ms + 4 ms + 2 ms = 9 ms
U = 1 - (9 ms / 15 ms)
U = (6 ms / 15 ms)
U = 0.40 = 40 % CPU Utilization

Network Interface Utilization:

Network Interface Idle time = 2 ms + 6 ms + 1 ms + 2 ms = 11 ms

U = 1 - (11 ms / 15 ms)

U = (4 ms / 15 ms)

U = .27 = 27 % CPU Utilization

b. Throughput = (number of requests serviced by the system) / total runnning time

Throughput of server:

number of requests serviced by system = 1 request

Throughput of server = 1 request / 15 ms

Throughput of server = 0.7 requests / ms

If MPL = 2, there can be 2 requests that are concurrently serviced

Given Assumptions:

There will always be pending HTTP requests

The system has reached a stable and reliable state

Usage of CPU, Disk, and Network Interface if MPL = 2

Assume A and B represent two different requests and each part of the graph represents 1 ms

c. Utilizations if N = 2

1 cycle can be seen in the period from t=9ms to t=25ms, thus the length of the cycle is 16ms.

```
| t = 0ms           | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18
 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28
 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| CPU              | A | A | B | B |   |   |   |
  | A |    |    |    |    | A  | A  | B  | A  | A
 |    |    | B  | B  | B  | B  | A  |    |    |
 |    | A  | A  | B  | A  | A  |    |    | B  |
| Disk             |   |   | A | A | A | A | A |
```

```
A | B | B  | B  | B  | B  | B  |    |    |    |
 | A  | A  | A  | A  | A  | A  | B  | B  | B  | B
 | B  | B  |    |    |    |    |    |    |    |
| Network Interface |   |   |   |   |   |   |   |
  |   | A  | A  | A  | A  |    |    |    | B  | B
 | B  | B  |    |    |    |    |    | A  | A  | A
 | A  |    |    |    | B  | B  | B  | B  |    |
```

Utilizations:
CPU Utilization = 10 ms active time/16 ms total time per cycle = 10/16 = 62.5%
Disk Utilization = 12 ms active time/16 ms total time per cycle = 12/16 = 75%
Network Interface Utilization = 8 ms active time/16 ms total time per cycle = 8/16 = 50%

d. Throughput of server: 2 requests per 16 ms per cycle
Throughput = 2 requests/16ms per cycle = 0.125 requests per ms

e. If we can set N to any value, the disk utilization can be 100%, thus the maximum throughput of the web server can be represented as (1/8)*(4/3) = 1/6

f. No. The limit to the capacity of this system (the bottleneck) is the disk since it takes 6 ms to fetch the file per request, which is necessary for performing any operation after it under my assumptions.Thus, the slow fetching causes the disk to be the bottleneck resource, not the CPU.

g. Minimum speedup for each device
CPU = 5/4 = 1.25
Disk = 6/4 = 1.5

h. Resulting capacity
Minimum speedup is 16/13 = 1.23
(1/8)*(16/13) = 0.153 requests/ms

2. Big data

8 hours to run, 20% time on int operations, 35% time in I/O

Option 1: $1000

reduce the time of integer operations by 25 %

Option 2: $4000

faster I/O devices to reduce latency from 6 us to 5 us (1 - 5/6 = 0.17 = 17 % reduction in time

Amdahl's law

y = 1/((1-f(1-(1/x))) where y is overall performance/speedup, f is the fraction of time a resource is used in serving requests, and x is the performance of that resource x-fold (making the resource x-times faster)

a. Improving int operations

f = 20%, x = 1/(1-25%) = 1.33

y = 1/((1-0.2*(1-(1/1.33)))

y = 1.052

b. Improving I/O

f = 35%, x = 1/(1-17%) = 1.20

y = 1/((1-0.35*(1-(1/1.20)))

y = 1.062

c. Cost effectiveness?

Int operations:

1.052 speedup / $1000 = 0.001052 speedup/$

I/O:

1.062 speedup / $4000 = 0.000266 speedup/$

On a per dollar basis, improving the int operations would be more cost effective.

d. Both improvements:

1/((1/5)*(3/4)+0.35*5/6+0.45) = 1.121

e. Theoretical limit for processor speedup

1/(4/5) = 5/4 = 1.25

f. Theoretical limit for I/O speedup
1/(1-0.35) = 1.54

3. Cache accessing
Given assumption: latency of main memory is 10x latency of the cache

Optimization 1: reduce latency of main memory by 50%
Optimization 2: reduce latency of cache by 25%

a. cache hit rate = 95%
Opt 1:
before opt/after opt = ((.95*1) + (.05*10)) / ((.95*1)+(.05*5))
before opt/after opt = 1.2083
Opt 2:
before opt/after opt = ((.95*1) + (.05*10)) / ((.95*0.75)+(.05*10))
before opt/after opt = 1.196

b. Cache hit rate = r
((r*1)+(1-r)*10) / ((r*1)+(1-r)*5) = ((r*1+(1-r)*10)/((r*0.75)+(1-r)*10)
0.75*r + (1-r)*10 = r+(1-r)*5
0.75*r+10-10*r = r+5-5*r
5 = 5.25*r
r = 0.95238 = 95.238%

If the cache rate is greater than 95.238%, the the cache optimization would be better. Otherwise, the main memory optimization owuld be better.

c. Original/Optimized
((h*1)+(1-h)*10) / ((h*0.75)+(1-h)*5)
(h+10-10*h)/(0.75*h+5-5*h)
(10-9*h) / (5-4.25*h)

4. ad-hoc network
path with H hops (links between two nodes)
link between any two nodes is 10 Mbps
300 bytes carry metadata

P - fraction of lost packets on one link

Assuming 1 MB is about 10^6 bits

a. Effective throughput
packages = (10*10^6 bits)/((1500*8)/1s)) = 833.33 (no loss)
Throughput in bits = 833.3*(1-P)^H *1500*8 bits
Effective throughput = 833.3*(1-P)^H*1200*8 bits

b. H=1, P=0.01
Effective throughput = 833.3*(1-P)^H*1200*8 bits
Effective throughput = 833.3*(1-0.01)^1*1200*8 bits
Effective throughput = 7.92 Mbps

c. Maximum loss probability
H = 1, Effective bandwidth >6 Mbps for average-length path L
833.33*(1-P)^L*1200*8 > 6*10^6 bits

5. Finding capacity of general networks (MaxFlow)
0. finding feasible "path" b/w source and dest
1. compute "flow" capacity of that path
2. reducing capacities on links of new paths accordingly and eliminate all links that have capacity 0 (create a residual graph)
3. repeat to get additional flow capacities until all paths are exhausted
Capacity = sum of all flow capacities
C=f1+f2...fn

a. (see picture)

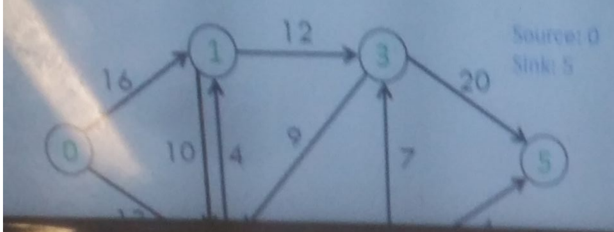
b. One algorithm you could use to choose to find which increase in connection would increase the capacity the most would be to just choose the bottleneck connection. Finding the bottleneck is more difficult. One way to do this may be to increase the capacity of the node connected to as many nodes before the output node as possible. This will increase the capacity because if one node before the output node receives too much traffic, it can route some traffic to the other node. In this case, using this algorithm, the best connection to increase would be the connection from 4 to 3. Increasing this connnection from 7 to 8 would increase the capacity to 24.

/* I found stack overflow links describing a the Ford fulkerson algorithm, but I couldn't understand it very well so I decided not to make it my official answer to this question */

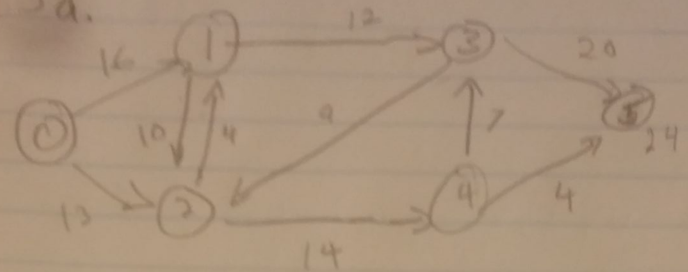
c. In general the capacity is limited based on the bottleneck connection with the lowest capacity. As you increase the capacity of the bottleneck connection, it will eventually reach a point where it wont be the lowest capacity and adding more capacity to the former bottleneck connection would not increase the overall capacity.

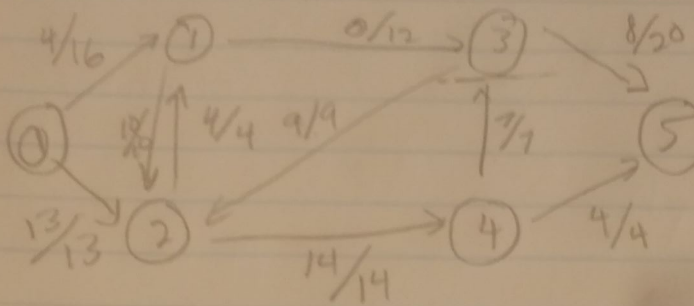
d. Performing a DoS attack would best at the connection from 1 to 5 as the capacity would now be 11 instead of 23 because attacking that 12 capacity connection would cause node 2 to receive much more traffic than anticipated as there is no alternative but to route to node 2.

nd egress links (e.g., by subjecting it to a Denial of Service
Which link would you choose and why?

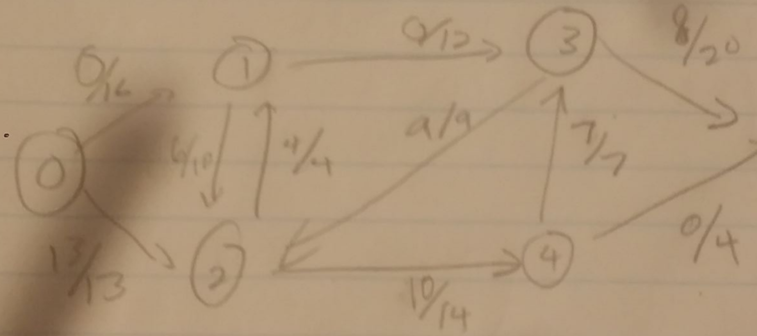


5a.

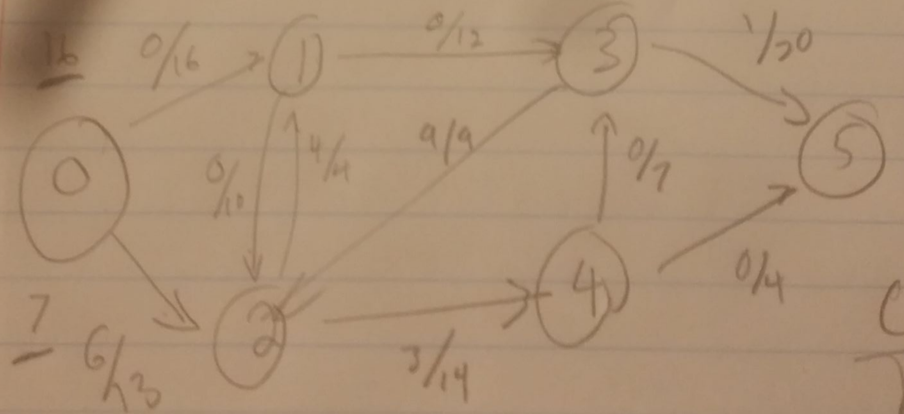




C=12



C=16



E

C=16+7

C=23