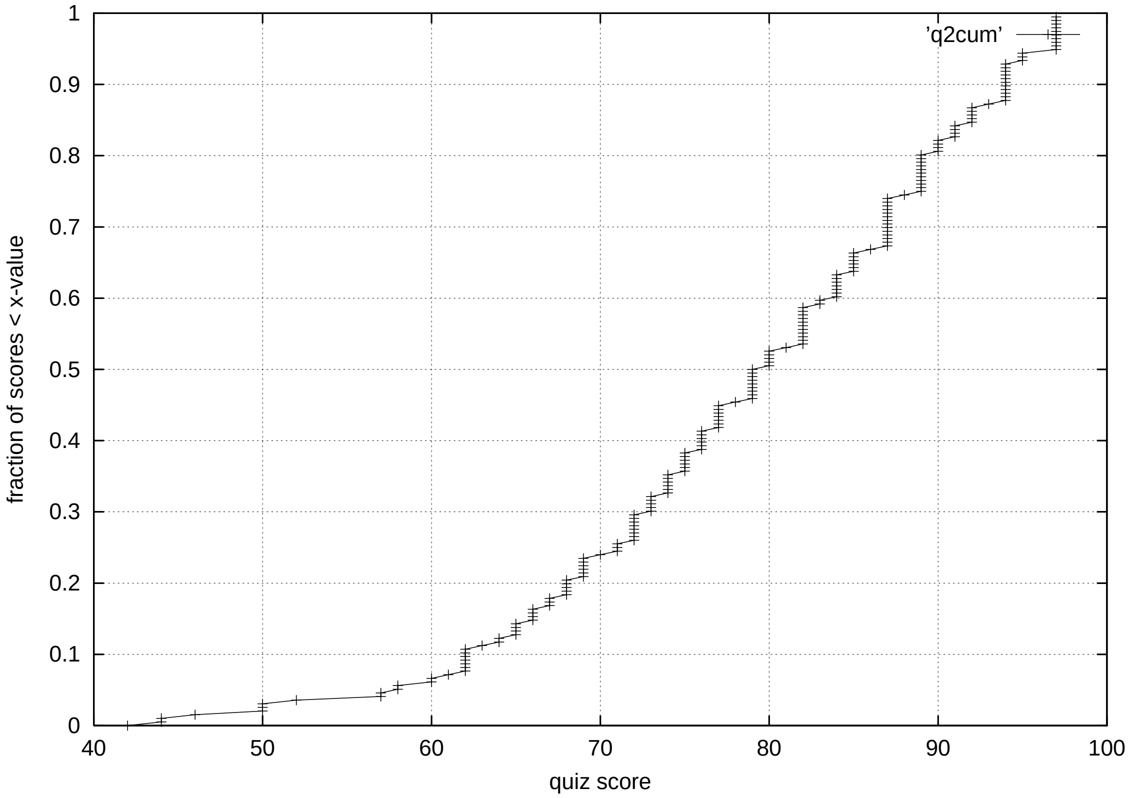
Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2003

Handout 27 - Quiz II Solutions

Cumulative Distribution



Mean: 78.5 / Standard Deviation 12.2 / Median 79

Name: Alyssa P. Hacker

I Reading questions

1. [10 points]: Which of the following describe activities that the Google system, as described in reading #12, performs as part of its processing of a particular client-submitted query?

(Circle ALL that apply)

A. Google executes the PageRank algorithm to find a ranking of pages pertinent for that particular query.

NO *At query time, Google uses the result of the PageRank algorithm, but that result was calculated by executing the algorithm at some earlier time, and on the entire data base.*

B. Pages get indexed in Barrels, in order of identifiers for the words appearing in the query.

NO *Pages do get indexed in barrels, but it does not happen in response to a specific query, it happens at indexing time.*

C. Google uses proximity of different query words on a page to improve precision.

YES *Proximity is one of the inputs to the decision of which web pages appear on the first response page.*

D. A Web Crawler collects pages containing the query keywords and puts them into a repository.

NO *Web crawling is done in the weeks and months before the query, not at query time.*

E. Google extracts lists of occurrences of the query words from pages and organizes them into "hit lists".

NO *Extraction of word occurrences is done as part of indexing, not as part of query handling.*

F. Pages containing all the query words in the client query are found from lists in barrels.

YES *This is the basic description of what happens when you make a quer.*

2. [10 points]: How can one be sure a compiler does not have a Trojan horse as described in "Trusting trust" (reading #14)?

(Circle ALL that apply)

A. Check the source code of the compiler line by line manually

B. Check the signature that came with the compiler to verify who produced the compiler

C. Run the compiler in a "sand box" separated from the network and other files

D. Use a theorem proving system on the source code of the compiler

E. None of the above

None of the suggestions guarantees absence of Trojan horses, so the correct answer is E. The inadequacy of answers A and D is one of the main points of the paper: there is no way to tell if source actually matches the object, so examining or proving things about the source assures you of nothing. Answer B simply shifts the question of whom we trust to not insert a Trojan horse; by itself it doesn't add any guarantee that one was not inserted. Running a compiler in a sandbox accomplishes little; a Trojan horse in the compiler can still affect the compiled code, so answer C does not apply.

Name:

3. [10 points]: The network file system (NFS), reading #11, adds a generation number to a UNIX inode to

(Circle ALL that apply)

A. allow the server to reuse an inode

YES *The generation number assures that when one client deletes a file and the server reuses its inode, any other clients that still have references to that file can be told that their reference is no longer valid.*

B. guarantee at-most-once semantics for `write` RPCs

NO *NFS does not provide at-most-once semantics.*

C. avoid tracking the file handles that a server has handed out to clients

YES *Rather than tracking down and invalidating stale file handles, it is easier to let clients use them and have the server report back that they are stale.*

D. delay writing modified data to disk till after sending a reply on `write` RPC

NO *Generation numbers have no connection with write dallying.*

4. [5 points]: On a machine configured with the default DNS domain "mit.edu", what is the context for the DNS name "ai."?

(Circle best answer)

A. mit.edu

B. the DNS root

C. ai.mit.edu

D. ai

The DNS name "ai." ends with a dot, which means that it is an absolute path name, so the context that must be used is the DNS root, answer B. Answer A is the context that would have been used had the name not ended in a dot. Answers C and D are contexts, but they would be used only if the name in question had at least one component appearing ahead of the name "ai".

II Monitoring ants

Alice has learned that ants are a serious problem in the dorms. To monitor the ant population she uses the motes of design project I. She spreads hundreds of motes in her dorm. Each mote can transmit a packet to another mote that is within its radio range. Motes forward messages on behalf of other motes to form a network that covers the whole dorm. The exact details of how this network of motes works are the topic of this question.

Each mote runs a small program that every 1 millisecond senses if there are ants present. Each time the program senses ants, it increments a counter, called *SensorCount*. Every 16 milliseconds the program sends the value of *SensorCount* and the mote's ID to the mote connected to Alice's desktop computer, which has ID A. After sending the message, the mote resets *SensorCount*.

For the rest of the quiz you should assume that only the radio consumes energy. The radio operates at a speed of 19.2 Kbits/s (a 916.50 Mhz transceiver). For transmitting the radio draws 12 milliamp at 3V. Although receiving draws 4.5 millamp, you may assume for this quiz that receiving is for free. The motes have a battery rated at 575 milliamphour (mAh).

5. [5 points]: If a mote transmits continuously, about how many hours can a mote transmit for?

(Circle the best answer)

A. 48 hours

B. 16 hours

C. 192 hours

D. 575 hours

The correct answer is A, 48 hours. Transmitting continuously, the mote consumes 12 milliamps. It has a battery with 575 milliamp-hours, so $\frac{575mAh}{12mA} \approx$ *48 hours*

6. [5 points]: How much energy (V x I x Time) does it take to transmit a single bit (1 watt-second = 1 joule)?

(Circle the best answer)

A. 36 millijoule

B. 1.9 microjoule

C. 1.9 millijoule

D. 36 microjoule

The correct answer is B, 1.9 microjoule. It takes $\frac{1}{19.2K}$ *seconds to transmit one bit. When transmitting, the mote consumes* $3V \times 12mA = 36mW$. *Finally,* $\frac{36mW}{19.2Ks^{-1}} \approx 1.9$ *milliWatt-milliseconds* $\approx$ *1.9 microjoules*

Name:

Because the radio range of the motes is only ten meters, the motes must cooperate to form a network that covers Alice's dorm. Motes forward messages on behalf of other motes to provide connectivity to A. To allow the motes to find paths and to adapt to changes in the network (e.g., motes failing because their batteries run down), the motes run a routing protocol. Alice has adopted the path-vector routing protocol from lecture. Each mote runs the following routing algorithm, which finds paths to mote A:

$n \leftarrow myID()$;
if $n = A$ **then** $path \leftarrow []$;
else $path \leftarrow NULL$;

procedure ADVERTISE()
 if $path \neq NULL$ **then** $transmit($**cons** $(n, path))$

procedure RECEIVE(p)
 if $n \in p$ **then return**
 else if $path = NULL \parallel first(p) = first(path) \parallel length(p) < length(path)$
  then $path \leftarrow p$

procedure TIMER()
 if $haveNotHeardFromRecently(first(path))$ **then** $path = NULL$

When a mote starts it initializes its variables n and $path$. Each mode has a unique ID, which the mote stores in the local variable n. Each mote stores its path to A into the $path$ variable. A path contains a list of mote IDs; the last element of this list is A. The first element of the list ($first(path)$) is the first mote on the path to A. When a mote starts it sets $path$ to NULL, except for Alice's mote, which sets *path* to the empty path ([]).

Every t seconds a mote creates a path that contains its own ID concatenated with the value of $path$, and transmits that path (see *advertise*) using its radio. Motes in radio range may receive this packet. Motes outside radio range will not receive this packet. When a mote receives a routing packet, it invokes the procedure *receive* with the argument set to the path stored in the routing packet. If a path contains n, then this routing packet circled back to n and the procedure *receive* just returns, rejecting the path. Otherwise, *receive* updates $path$ in three cases:

- if $path$ is NULL, because n doesn't have any path to A yet.
- if the first mote on $path$ is the mote from which we are receiving p, because that mote might have a new path to A.
- if p is a shorter path than $path$, because p is a better path to A than $path$. (For this quiz we assume shorter paths are better.)

A mote also has a timer. If the timer goes off, it invokes the procedure *Timer*. If since the last invocation of `Timer` a mote hasn't heard from the mote at the head of the path to A, it resets $path$ to NULL, because apparently the first node on $path$ is not reachable anymore.
Name:

The forwarding protocol uses the paths found by the routing protocol to forward reports from a mote to A. Since A may not be in radio range, a report message may have to be forwarded through multiple motes to reach A. This forwarding process works as follows:

procedure SEND$(counter)$
if $path \neq NULL$ **then** $transmit(first(path), (n, counter))$

procedure FORWARD$(nexthop, report)$
if $n \neq nexthop$ **then** $return$
if $n = A$ **then** $deliver(report)$
else $transmit(first(path), report)$

The procedure *send* creates a *report* (the mote's ID and the counter value) and transmits a report packet. The report packet contains the first hop on $path$, that is $first(path)$, and the report:

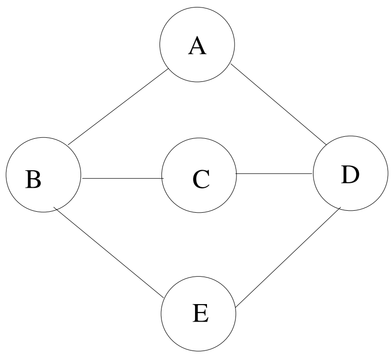
Nexthop	Source	SensorCount

report

The *nexthop* field contains the ID of the mote to which this packet is directed. The *source* field contains the ID of the mote that originated the message. The *sensorcount* field contains the sensor count. (If path is set to NULL, *send* has no mote to forward the report to and thus the mote drops the packet.)

When a mote receives a report packet, it calls *forward*. If a mote receives a report packet for which it is not the next hop, the mote ignores the packet. If a report packet has reached its final destination (A), *forward* delivers it to Alice's computer. Otherwise, the mote forwards the report by setting *nexthop* to the first mote on its $path$ variable. This process repeats until the report reaches A, which delivers the reports to Alice's computer.

For the subsequent questions assume the following arrangement of motes:



The circles represent motes with their node IDs. The edges connect motes that are in radio range of one another. For example, when mote A transmits a packet, it may be received by B and D, but not by C and E. Packets may be lost and motes may also fail (e.g., if their batteries run down). If a mote fails, it just stops sensing, transmitting, and receiving packets.

7. [10 points]: If motes may fail and if packets may be lost, what values could the variable $path$ contain at node E:

(Circle ALL that apply)

A. NULL

YES *If all the packets are lost, no routes are set up and E's path is NULL*

B. [B, C]

NO *All paths must lead to A, i.e., have A as the last entry.*

C. [D, A]

YES *Scenario: A advertises [A] to B and D, advertisement to B gets lost. D advertises [D,A] to E E accepts advertised path.*

D. [B, A]

YES *Scenario is as in C, except D and B are swapped*

E. [B, C, D, A]

YES *Scenario: A advertises [A] to B and D, advertisement to B gets lost. D advertises [D,A] to C and E, advertisement to E gets lost. C advertises [C,D,A] to B and D, D rejects path. B advertises [B,C,D,A] to E. E accepts advertised path.*

F. [C, D, A]

NO *E cannot receive advertisements from C, so it will never have a path that starts with C.*

G. [D, A, B, C, D, A]

NO *D and A occur twice in this path. For this path to occur at E, A must accept the path [B,C,D,A], then advertise the path [A,B,D,A] to D, which accepts the path and advertises the path [D,A,B,C,D,A] to E. But neither A nor D will accept the path, since a condition in receive rejects an advertised path if the node receiving the message is already in the path.*

8. [10 points]: If no motes fail and packets are not lost, what properties hold for Alice's routing and forwarding protocols with the given arrangement of motes?

(Circle ALL that apply)

A. Every mote's $path$ variable will contain a path to A.

YES/NO *After the routing path reaches a stable state, it is true that every mote has a path to A. But when the system starts, all paths (except A's) are NULL. Thus, depending on your interpretation, A may be circled or not circled.*

B. After the routing algorithm reaches a stable state, every mote's $path$ variable will contain a shortest path (in hops) to A.

YES *Receive always accepts a new path if it is shorter than its current path. Eventually this means that all motes will have a shortest path to A.*

C. After the routing algorithm reaches a stable state, the routing algorithm may have constructed a forwarding cycle.

NO *A forwarding cycle occurs if a packet can ever visit the same mote twice. This can't occur in the stable state because all routes follow the shortest path to A and the routes don't change.*

D. After routing algorithm reaches a stable state, the longest path is two hops.

YES *The longest shortest paths in the graph are all two hops long, so the longest routing path is two hops long.*

9. [10 points]: If packets may be lost but motes don't fail, what properties hold for Alice's routing and forwarding protocols with the given arrangement of motes?

(Circle ALL that apply)

A. Every mote's $path$ variable will contain a path to A.

NO *If all packets are lost, no mote has a routing path to A except A.*

B. The routing algorithm may construct forwarding cycles.

YES *Remember that a routine cycle occurs when a packet visits the same mote twice. Here is how a routing cycle can occur:*

(a) The system establishes the following path from C to A: [D,E,B,A], and sends a packet from C to D, then from D to E.

(b) The paths at B,C, and D timeout. They are then reestablished so that D has path [A], C has path [D,A], and B has path [C,D,A].

(c) E still has path [B,A], so it routes the packet to B.

(d) B routes to C, then D, and the packet has visited D twice.

This cycle ought to be temporary; the next steps of the routing algorithm will try to get rid of it. Unfortunately, routing packet losses can continue to frustrate the algorithm, so the cycle may persist or recur.

C. The routing algorithm may never reach a stable state.

YES *Routes may always time out and be established differently.*

D. When a mote sends a report packet using *send*, the report may or may not reach A.

YES *Report packets may be lost.*

A report is 13 bits: an 8-bit node ID and a 5-bit counter. A report packet is 21 bits: an 8-bit next hop and a report. Assume that your answer to question 6 (the energy for transmitting one bit) is j joules. Further assume that to start the radio for transmission takes s Joules. Thus, transmitting a packet with r bits takes $s + rj$ joules.

10. [5 points]: Assuming the routing algorithm reaches a stable state, no node and packet failures, how much *total* energy does it take for every node to send one report to A (ignoring routing packets)?

(Circle best answer)

A. $6(s + 21j)$

B. $4(s + 21j)$

C. $8(s + 21j)$

D. $5(s + 21j)$

In a stable state the routing protocol will construct a logical tree rooted at A. The intermediate nodes in the tree are B and D. C and E will form the second-level of this tree. A report from B results in one packet transmission from B to A, and a report from D results in one packet transmission from D to A. A report from C results in two packet transmissions (one to B or D, and then one to A). Similarly, a report from E also results in two packet transmissions. Thus, the total number of packet transmissions is 6.

11. [10 points]: What changes to the Alice's system will reduce the amount of energy needed for each node to send one report to A?

(Circle ALL that apply)

A. Add a nonce to a report so that Alice's computer can detect duplicates

NO *A nonce will make the packet bigger.*

B. Delay forwarding messages and piggyback them on a report message from this mote.

YES *By piggybacking multiple reports on a single report packet, the cost of starting the radio and transmitting the nexthop identifier can be amortized over multiple reports.*

C. Use 4-bit node IDs.

YES *There are only 5 nodes in the system, thus one can use 4-bit node IDs, which makes the packets shorter.*

D. Use a stop-and-wait protocol to transmit each report reliably from one mote to the next.

NO *This change may make Alice's system more reliable, but it won't save energy.*

Name:

To be able to verify the integrity of the reports, Alice creates for each mote a public-key pair. She manually loads the private key for mote i into mote i, and keeps the corresponding public keys on her computer. A mote signs the contents of the report (the counter and the source) with its private key and then transmits it. Thus, a signed report consists of a 5-bit counter, a node ID, and a signature. When Alice's computer receives a signed report, it verifies the signed report and rejects the ones that don't check out.

12. [10 points]: Assuming that the private keys on the motes are not compromised, and the `sign` and `verify` procedures are correctly implemented, what properties hold for Alice's plan:

(Circle ALL that apply)

A. A mote that forwards a report is not able to read the report's content.

NO *Reports are not sealed (they are only signed), so every mote can read a report.*

B. A mote that forwards a report may be able to duplicate a report without Alice's computer rejecting the duplicated report.

YES *A mote can duplicate the signed report. This duplicate will verify correctly, because the signature covers only the counter and the source.*

C. A mote that forwards a report can use its private key to verify the report.

NO *A signature is verified using a public key*

D. When Alice receives a report that verifies correctly, she knows that the report was signed by the mote that is listed in the report and that the report is fresh.

NO *The signatures will verify (see B), but Alice won't be able to determine if the report is fresh, since the report doesn't contain a nonce, and intermediate nodes can duplicate packets.*

End of Quiz II