

Computer Science Department
Boston University

Networks PhD Depth Exam
November 2003

1. Check that there are 5 questions over 8 pages including this page. Answer all questions. Turn in your answers by Friday, November 21, 4:00pm, at MCS-140F.
2. You are free to consult any notes, books and papers during the examination—make sure to include your references. You are NOT allowed to consult with any person during the examination.
3. You may typeset your answers (in which you should feel free to include hand-drawn figures), or you may neatly write your answers. Please answer each question in a separate writeup.
4. Although questions are of different length, they are of equal weight.
5. Please indicate the question number and your code number on each answer sheet. Do not write your name on the answer sheet.
6. If you have any doubt as to the interpretation of a question, make a reasonable assumption and explain your interpretation in your answer. No explanations will be given during the exam.
7. Your answers should show research maturity and depth, so you may support your answers not only by analysis, but possibly by measurements or simple simulations. Be creative!

Networks PhD Depth Exam
November 2003
Question 1

Consider a *traceroute session* between two points A and B, i.e. the set of hop-limited probes used to map the route, and the responses from these probes. When all of the packets that comprise a traceroute session all traverse the same path in the network, each of the individual probes (with increasing TTLs) trigger an ICMP response from a distinct router. The ordered sequence of reported router interfaces constitute the measured path from A to B.

However, load balancing and route fluctuation introduce error into this measurement process. Let us assume that there are two disjoint paths from A to B denoted by sets of vertices C_i and D_i , respectively. A traceroute session whose duration spans a routing change from the C path to the D path will observe a spurious “path” consisting of a subsequence of reported nodes from the C path followed by a subsequence of reported nodes from the D path. The crossover point from the C path to the D path, moreover, will now be implicitly believed to be an edge in the network topology, even though this edge may not really exist. For example, assuming paths of length 2, uncorrupted observations witness $\{C_1, C_2\}$ and $\{D_1, D_2\}$ as intermediate hops, while corrupted observations also witness $\{C_1, D_2\}$ and $\{D_1, C_2\}$.

- (a) Briefly outline the design of an experiment to estimate the magnitude of this problem in existing traceroute studies.
- (b) Your colleague claims that a “slow” traceroute, i.e. one which introduces temporal spacing between each probe along the path, is more likely to see a spurious path than a normal “fast” traceroute. Is she right, wrong, or does it depend on the characteristics of the path? Discuss.
- (c) Your colleague also claims that a “slow” traceroute tool could be a useful tool in characterizing routing behavior and routing changes along a path. What useful information could you compile about a path using this tool?
- (d) Implement a variant of traceroute which allows the user to specify the inter-packet spacing of probe packets.
- (e) Using your tool, conduct measurements across the Internet to either give preliminary evidence that there is something to the idea in (c), or that it is a bust.

Networks PhD Depth Exam
November 2003
Question 2

Please see the attached excerpt for the purpose of answering this question. You were given this excerpt which is Section 3 of a paper. You are asked to develop a full-blown routing protocol that makes use of the proposed hierarchy of so-called *neighborhood servers*. More specifically, you are asked to do the following:

- (a) Building on the given Section 3, develop a full specification of a protocol that allows a node to obtain a path to a destination node, assuming static conditions (i.e. links and nodes remain operational). Make sure to define how nodes are selected as neighborhood servers in your protocol, and describe the whole process from processing a new path request to its final resolution.
- (b) Complete Section 4 by extending your protocol to handle dynamic conditions, i.e. nodes and links can fail and recover. Make sure to address the issue of name-to-address mapping.
- (c) Design a set of experiments to evaluate the performance of your protocol. Specify a “baseline” protocol against which you will compare yours and justify your choice(s).
- (d) Comment on the applicability of this protocol in different network environments—can we use this protocol to replace BGP in the Internet? If so, what would be the pros and cons of such deployment? Since nodes may maintain only small neighborhoods, can we use this protocol in a resource-constrained environment such as a sensor network?

Outline any adaptations you may need to do to your protocol to accommodate different environments.

- (e) What would be the effect of topological characteristics on the formation of neighborhoods and the selection of neighborhood servers? For example, comment on the two cases when your protocol runs on a random Waxman-like router topology versus a power-law-like autonomous-system topology.

3 Neighborhood Hierarchical Protocol

Here we present our protocol for static conditions, i.e. all links and nodes stay operational. We present the case of dynamic network conditions in Section 4.

Each node has a unique identifier. `NodeIds` denotes the set of all node identifiers. For a node u , $nodeid(u)$ denotes the identifier of u . We denote by $NodeNeighbors(u)$ the set of identifiers of the neighbors of u .

Neighborhoods (i.e. the state of a subset of surrounding links) are maintained by special nodes called *neighborhood_servers*. Each neighborhood_server has a *locality*, which is a set of nodes around the neighborhood_server. A neighborhood_server maintains a *neighborhood* that consists of the nodes in its locality, links between these nodes and links outgoing from the locality¹. More formally, a neighborhood_server x maintains the following variables:

$Locality_x \subseteq NodeIds$. Nodes for which x maintains state (i.e. costs of outgoing links)

$Neighborhood_x$. Neighborhood of x .

$$= \{ \langle u, timestamp, expirytime, \{ \langle v, cost \rangle : v \in NodeNeighbors(u) \} \rangle : u \in Locality_x \}$$

The purpose of $Neighborhood_x$ is to obtain complete paths between nodes in $Locality_x$. Hence, which nodes to include in $Locality_x$ and which links to include in $Neighborhood_x$ are not arbitrary. $Locality_x$ and $Neighborhood_x$ must be connected, i.e. there should be a path in $Neighborhood_x$ between any two nodes in $Locality_x$. We note that $Neighborhood_x$ can contain links to nodes outside $Locality_x$. A node u is said to be *in the neighborhood* of a neighborhood_server x , if either u is in the locality of x , or $Neighborhood_x$ has a link from a node in the locality of x to u . The localities and neighborhoods of different neighborhood_servers can be identical, overlapping or disjoint.

In order to scale, we cannot have one large neighborhood. Thus, obtaining a complete path from a source to a far-away destination involves accumulating neighborhoods of a sequence of neighborhood_servers. To do this efficiently, neighborhood_servers are organized hierarchically. Specifically, we assign each neighborhood_server a hierarchy level from $0, 1, \dots$, with 0 being the top level in the hierarchy. A parent-child relationship between neighborhood_servers is defined as follows:

1. Every level i neighborhood_server, $i > 0$, has a parent neighborhood_server whose level is less than i .
2. If neighborhood_server x is a parent of neighborhood_server y then x 's locality contains y and y 's locality contains x .
3. The locality of a top level neighborhood_server contains all other top level neighborhood_servers.

In the hierarchy, a parent can have many children and a child can have many parents. The range of the parent-child relationship is extended to ordinary nodes; that is, if $Locality_x$ contains node u , u is said to be a child of x , and x is a parent of u . We assume that each node has at least one parent neighborhood_server.

For node u , we define an address to be a sequence $\langle x_0, x_1, \dots, x_t \rangle$ such that x_i for $i < t$ is a neighborhood_server-id, x_0 is a top level neighborhood_server-id, x_t is the identifier of u , and x_i is a parent of x_{i+1} . Since the parent-child relationship is many-to-many, a node may have many addresses. If a source node wants to find a path to a destination node, it first queries the name servers to obtain a set of addresses for the destination². Second, the source queries neighborhood_servers to obtain an accumulated neighborhood containing both itself and the destination node. Then, it chooses a path from this accumulated neighborhood.

¹Not all the links have to be included.

²Similar to what is done currently in the Internet.

Networks PhD Depth Exam
November 2003
Question 3

Attached to this exam is the first page from a very recently issued patent entitled:

WORLD-WIDE-WEB SERVER THAT FINDS OPTIMAL PATH BY SENDING MULTIPLE SYN+ACK
PACKETS TO A SINGLE CLIENT

Read the abstract of this patent and answer the following questions:

- (a) The patent claims that the proposed technique would result in the selection of the “optimal path” between a server and a client. As described in the patent’s abstract, what dimensions (or metrics) are being optimized by the patent?
- (b) OK, startups are out of fashion these days, but in a parallel universe one can still pretend! As the Director of Engineering of a startup company that decided to exclusively license this patent, you are tasked with milking this patent for all its worth. So, for EVERY interesting metric you can think of (as it relates to web content delivery), explain how (if at all possible) the idea presented in the patent could be applied to optimize that metric. For each such case, give a sketch of how this could be done (i.e., explain what architectural pieces need to exist and provide enough of a description to allow your ideas to be put to practice by your product development team.
- (c) While you have been thinking about optimizing the “path” between a server and a client, a bright intern working under your supervision suggested that the idea in the patent could be applied to the problem of load balancing a server farm. To do so, she suggested an architecture whereby a load-balancing box (typically called a TCP router), upon receiving a SYN packet from the client, would multicast that packet to all (or a random subset) of the servers in the farm. These servers would then respond directly with individual SYN+ACK packets to the client, who responds to the first such SYN+ACK with an ACK. The load-balancing box would then use information in that ACK to figure out which one of the servers it had originally selected is the one to which the TCP connection should be assigned. The box would then forward this ACK packet and all subsequent packets from this flow to that server (until the connection is closed). Discuss the pros and cons of this load balancer (compared to a traditional load balancer that collects feedback information, e.g., CPU load, number of pending requests, cache hit rates, I/O activities) from all servers in the farm in order to make its own choice of which server to select for an incoming connection.
- (d) Sketch the complete design of a web server farm that is CPU-load-balanced using the idea in part 3. Assume that the web server address is the address of the TCP router and that all servers in the farm masquerade their IP addresses in any responses to clients to be the TCP router’s address (also, they each have a second private IP address that is not advertised by DNS). Write and test code segments that will do the following:
 - (i) On the TCP proxy, listen to a client’s SYN packet on port 80 and replicate that packet to a list of servers on the subnet,
 - (ii) On the server, listen to a SYN packet and respond with a SYN+ACK in such a way that the set of servers are CPU load balanced,
 - (iii) On the TCP proxy, listen to the client’s ACK packet and determine the “winner” by printing “connection is assigned to server X”.

Networks PhD Depth Exam
November 2003
Question 4

Consider a load balancer (LB) which attempts to distribute two types of requests over two servers. Type-1 requests require a low response-time (latency) service, while type-2 requests require a high rate service. Assume that the LB periodically collects response-time and utilization statistics about both servers. Then, the LB selects the lowest response-time server for type-1 requests, while it directs type-2 requests to the lowest utilization server.

For the purpose of modeling this system and gaining insights into its performance, Bob suggested to model it as a *discrete-time* dynamical flow system of a fixed number of N streams of type-1 requests and M streams of type-2 requests. Different streams may be coming from different sites seeking service from that 2-server system. Bob's model then defines the state of the system by the current number of streams of each type, say x and y , served by one of the two servers (say, server 1). Note that this definition of system state suffices since the number of streams of each type on the other server (server 2) is simply $N - x$ and $M - y$.

Bob's model also assumes that the state of the system at the LB is updated every fixed period of time, say T , when the LB makes a new decision on where to direct each type of requests. The system is assumed to reach steady-state between two successive updates (assuming long enough T).

- (a) Assume that each stream generates requests at the rate of 1 request/time unit according to a Poisson process. Also, assume that the service time of each request consists of a fixed processing time, P_1 and P_2 , at each one of the two servers, respectively. Moreover, each request waits in a FCFS queue to be served for an exponentially distributed time, with averages $1/\mu_1$ and $1/\mu_2$, at each one of the two servers, respectively. For the purpose of this question, all queues are assumed of infinite size. The LB follows a non-greedy approach, where it only directs a fraction $0 < \alpha \leq 1$ of the streams of each type from one server to the other best server at a given update instant.

Write the difference equations that describe the system as per Bob's model. Derive the (necessary and sufficient) stability conditions of this system, i.e. identify the range of system parameters (P_i, μ_i, N, M) for which the set of system equations converge to equilibrium from any initial state and for any α —for the purpose of this question, consider the *desired equilibrium* to be the system state where all type-1 streams are all directed to the lowest response-time server and stay there, and all type-2 streams are all directed to the highest rate server and stay there. That is, at the *desired equilibrium* point, the system state is $(x = N, y = 0)$ if server 1 provides the lowest response time and server 2 provides the highest rate (or offers the lowest utilization). *In answering this and remaining parts of this question, you may resort to numerically solving the difference equations to support your answers obtained analytically.*

Refine your stability conditions to be a function of the starting/initial state (but still independent of the value of α). Note that a different starting system state could be the result of new streams coming in, existing streams terminating, etc.

- (b) Alice proposes a modification to Bob's model to examine the use of round-robin (RR) scheduling at both servers to replace FCFS. In each server's RR scheduler, requests from type-1 streams are placed in one queue and requests from type-2 streams are placed in another queue. The service rate is equally divided between the two queues—To simplify your analysis, you may assume that the service times at each queue at an RR scheduler are exponentially distributed.

Rewrite the difference equations that describe the system as per Alice's model. Compare Alice's system to that of Bob's in terms of the set of starting states for which each system converges to the *desired equilibrium*—you want to choose the system that gives you the largest convergence region!

- (c) Further refine the stability conditions obtained in parts (a) and (b) to illustrate the effect of α on the system behavior.

Networks PhD Depth Exam
November 2003
Question 5

Consider the following data structures which were discussed and used in one or more of the papers on our reading list (listed alphabetically):

- approximate reconciliation tree
- Bloom filter
- Chord distributed hash table
- ephemeral state
- Internet indirection infrastructure
- minwise-independent permutation
- multistage filter
- NICE tree
- MINT caricature
- small-world graph

Make a case for the data structure above that would be the best fit for each of the following applications (choose one for each application). Do not argue for a data structure beyond those listed above, although you may mention such a data structure in your writeup. When several data structures on the list are applicable, make a case why your choice is superior. Do not mention data structures that are not applicable. In most cases, it will also be necessary to briefly describe how the data structure will be used to solve the application. State any additional assumptions you make about each application clearly.

- (a) Consider a web cache that can store any one million of a billion different possible web pages, each with a unique URL. You would like to implement LFU (least frequently used) replacement at this cache, but you only have 20MB of room to store the frequencies in your data structure. Specify the data structure you would use.
- (b) Now consider a web cache in a network of caches. In the event that a webpage causes a miss in your cache, you wish to redirect the message to another cache which is likely to have the page. Specify the data structure a web cache would use to transmit its cache contents to its peers in such an architecture.
- (c) Two nodes in a sensornet have a set of 10,000 distinct events that they are capable of sensing. When they have jointly witnessed 8000 of the 10,000 events, they are to report to the base station. Suppose node A has sensed 5850 events and node B has sensed 6230. Assuming energy is at a premium (i.e. every transmitted byte counts), what data structure should they use to determine whether they have reached the threshold?
- (d) A useful service was provided by Monkeys.com until a few months ago. Their basic idea was to enumerate a set of Internet IP addresses that spammers used frequently. Spam blockers could check the list and automatically mark traffic from these sources as spam. Unfortunately, the spammers now use denial-of-service attacks to keep the site down most of the time. What data structure could be used to organize the spam addresses and make denial-of-service a tougher prospect?
- (e) Consider a fusion service in a sensor net. A user would like to upload a new algorithm for fusing, inside the network, different signals sensed by different sensors. What data structure could be used to make the processes of algorithm distribution and fusion flexible and resilient?



US006587438B1

(12) **United States Patent**
Brendel

(10) **Patent No.:** **US 6,587,438 B1**

(45) **Date of Patent:** **Jul. 1, 2003**

(54) **WORLD-WIDE-WEB SERVER THAT FINDS OPTIMAL PATH BY SENDING MULTIPLE SYN+ACK PACKETS TO A SINGLE CLIENT**

5,898,668 A	4/1999	Shaffer	370/230
5,928,330 A	7/1999	Goetz et al.	709/231
6,061,349 A *	5/2000	Coile et al.	370/389
6,104,717 A *	8/2000	Coile et al.	370/401

(75) Inventor: **Juergen Brendel**, Santa Clara, CA (US)

* cited by examiner

(73) Assignee: **Resonate Inc.**, Mountain View, CA (US)

Primary Examiner—Wellington Chin

Assistant Examiner—Jamal A. Fox

(74) *Attorney, Agent, or Firm*—Stuart T. Auvinen

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(57) **ABSTRACT**

An optimal path through the Internet to a client is determined by the server during connection establishment. During the 3-way handshake that establishes a connection, a web server ordinarily sends a single SYN+ACK packet to the client. Instead of sending just one SYN+ACK packet, the server is modified to send multiple SYN+ACK packets, each using a different path to the client. When the multiple SYN+ACK packets are sent from the server at the same time, the first packet that reaches the client used the fastest path through the Internet. The client responds to this first SYN+ACK packet with an ACK packet back to the server. The other SYN+ACK packets that use slower paths arrive at the client after the first SYN+ACK packet and are ignored by the client as being out-of-order. The server includes a different sequence number with each SYN+ACK packet. The client increments this sequence number and includes the incremented sequence number in the ACK packet. The server reads this incremented sequence number in the ACK packet to determine which packet reached the client first. The path used by this packet is then included in source-routing fields of all future packets in the connection.

(21) Appl. No.: **09/470,143**

(22) Filed: **Dec. 22, 1999**

(51) **Int. Cl.**⁷ **G08C 15/00**; H04J 1/16; H04J 3/14; H04L 12/26

(52) **U.S. Cl.** **370/238**; 370/254

(58) **Field of Search** 370/238, 229, 370/235, 232, 233, 234, 255, 256, 254, 389, 395.31, 395.32, 410, 395.2, 392, 238.1, 393.21

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,007,052 A *	4/1991	Flammer	370/389
5,193,151 A	3/1993	Jain	395/200
5,444,706 A	8/1995	Osaki	370/94.1
5,570,346 A	10/1996	Shur	370/17
5,633,861 A *	5/1997	Hanson et al.	370/232
5,727,002 A	3/1998	Miller et al.	371/32
5,774,660 A *	6/1998	Brendel et al.	709/201
5,802,106 A	9/1998	Packer	375/225

20 Claims, 6 Drawing Sheets

