

Computer Science Department  
Boston University

**Networks PhD Depth Exam**  
**November 2004**

1. Check that there are 5 questions over 6 pages including this page. Answer all questions. Turn in your answers by Friday, November 19, 4:00pm, at MCS-140G.
2. You are free to consult any notes, books and papers during the examination—make sure to include your references. Copying existing solutions or parts of them will be considered plagiarism, and in this case you will fail the exam.  

You must develop your own solutions, which might use existing ideas and techniques, as long as you cite them and clearly explain how these existing ideas/techniques fit within your own solution. Failure to do so and a feel of cut-and-paste in your answers will receive zero credit and you may fail the exam. 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!
8. The copyright to any new research question is owned by its author :)

**Networks PhD Depth Exam**  
**November 2004**  
**Question 1**

You are given a set of paths traversing a network. The paths are organized into a binary matrix  $A$  in which  $A_{ij} = 1$  if link  $i$  is traversed by path  $j$ . The number of paths is (as usual) considerably larger than the number of links.

1. What information about the network can you obtain from the rank of  $A$ ?
2. What information about the network can you obtain from the eigenspectrum of  $A$  (ie., the eigenvalues of  $A^T A$ )?
3. What information about the network can you obtain from the eigenvectors of  $A^T A$ ?
4. What information about the network can you obtain from the eigenvectors of  $AA^T$ ?
5. Assume you would like to extract the most commonly-used path segments in the set of traceroutes. What linear-algebraic operations could you perform on  $A$  to assist you?
6. Assume you are given some measurements on the set of links  $\vec{y}$  and are told that the corresponding measurements on the set of paths follow  $\vec{y} = A\vec{x}$ . Give an example of a commonly used metric for which this is true.
7. Is estimating  $\vec{x}$  hard in this situation? Explain why or why not. Describe a general strategy for estimating  $\vec{x}$ .
8. Now consider the matrix  $G = A^T$ . Assume you are given some measurements on the set of paths  $\vec{y}$  and are told that the corresponding measurements on the set of links follow  $\vec{y} = G\vec{x}$ . Give an example of a commonly used metric for which this is true.
9. Is estimating  $\vec{x}$  hard in this situation? Explain why or why not. Describe a general strategy for estimating  $\vec{x}$ .
10. Now assume that the equation  $\vec{y} = G\vec{x}$  holds, but that matrix multiplication should be interpreted in terms of  $(\min, \times)$  algebra. Give an example metric for which this is true.
11. Now consider the problem of estimating or discovering  $A$ , given  $\vec{x}$  and  $\vec{y}$ . Which of the three metrics (from answer 6, 8, or 10) would you use to attack this problem? Describe how you would attack this problem using that metric.
12. Now consider this problem: you can only measure some of the paths in  $\vec{y}$  and you want to estimate the values for the unmeasured paths. Assume we are back to standard linear algebra (ie.,  $(+, \times)$  algebra); how would you use the fact that  $\vec{y} = G\vec{x}$  to help you here?
13. Now consider this problem: Go back to the case of  $\vec{y} = A\vec{x}$ . Assume you can only measure some of the links in  $\vec{y}$  and you want to estimate the values for the for the unmeasured links. How would you use the fact that  $\vec{y} = A\vec{x}$  to help you? Consider also using the structure of  $A^T A$ .

**Networks PhD Depth Exam**  
**November 2004**  
**Question 2**

To study how damaging low-rate attacks could be on adaptive traffic sources, you decided to put your basic queueing theory knowledge to work. Let's say you model a legitimate TCP source with a Poisson arrival process of rate  $\lambda$  packets/second. This legitimate TCP rate  $\lambda$  depends on the observed loss rate  $p$  and the roundtrip time RTT. To model a low-rate (non-adaptive) attack which regularly injects a burst of packets, you use a batched Poisson process with rate  $\lambda_a^M$ , where  $M$  is a fixed burst size.

1. Given a certain buffer size and capacity for a bottleneck drop-tail link under attack, show the Markov chain for this system and compute the steady-state packet loss probability for given input rates  $\lambda$  and  $\lambda_a^M$ .
2. Given the current loss rate, one can update the input rate of the modeled TCP source  $\lambda(p)$ , iterating over the solution in part (1) assuming steady-state between rate updates. Write the analytical equations for this iterative approach and show whether it converges to a fixed point.
3. Use your iterative solution to study the potency of low-rate attacks by varying the attack rate  $\lambda_a^M$  and the attack magnitude  $M$  for various bottleneck buffer size and capacity values. Consider a potency metric that measures the wasted utilization per attack packet.
4. Summarize any insights you gained from your analysis, especially on the effectiveness of such low-rate attacks on high bandwidth-delay-product links.
5. On high bandwidth-delay-product highly multiplexed links, some have argued that efficient utilization can be maintained even for a small buffer size, say 10% of the bandwidth-delay product. Using your model, comment on the implications of such small buffers on the susceptibility of the network to low-rate attacks.
6. Summarize flaws (if any) in the above model and how they may affect the accuracy of your findings.

Now assuming that low-rate attacks are effective (at least under certain conditions), you are asked to:

7. Devise an end-to-end approach to overcome low-rate attacks.
8. Analyze the effectiveness (performance/cost) of your approach compared to in-network solutions such as the one described in the paper of Sun, Lui and Yau on your reading list.
9. Provide a sketch of how you would experimentally validate (e.g., via ns simulations) and implement (e.g., in Linux) your proposed end-to-end scheme. Discuss all pertinent issues and assumptions.

**Networks PhD Depth Exam**  
**November 2004**  
**Question 3**

You are asked to develop an end-system probe for discerning whether the congestion link between two end points is best modeled as a drop-tail buffer or a RED buffer. You may assume that there is only one congestion point on the path and that you are able to incorporate end-point functionality at any level.

1. Develop such an end-system probe by actively affecting and measuring end-to-end *delay jitter*.
2. Develop an analytical model that quantifies the expected change in delay jitter as a result of injected probe traffic under RED versus under Drop-Tail.
3. Comment on the marginal utility of your probing traffic, i.e. the accuracy of the inference per probe packet.
4. Provide a sketch of how you would experimentally validate (e.g., via ns simulations) and implement (e.g., in Linux) such a probing scheme. Discuss all pertinent issues and assumptions.
5. Identify techniques from prior work that could be used as an alternative approach for this inference problem. Specifically, discuss the pros and cons of this proposed scheme compared to existing Bayesian delay-based or loss-based inference techniques, for example the one described in the following paper:
  - Jun Liu, Mark E. Crovella (2001). Using Loss Pairs to Discover Network Properties. In: Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2001. pp. 127–138. <http://www.cs.bu.edu/faculty/crovella/paper-archive/imw-losspairs.pdf>

**Networks PhD Depth Exam**  
**November 2004**  
**Question 4**

The Border Gateway Protocol (BGP) is known to be susceptible to routing divergence due to potential conflicts among the routing policies of different administrative domains/systems (ASes). You will attempt here to provide a solution to this divergence problem. You may refer to the following seminal paper (on the reading list of CS 556):

- T. Griffin and G. Wilfong. *An Analysis of BGP Convergence Properties*. ACM SIGCOMM 1999. <http://citeseer.ist.psu.edu/333036.html>

You could also refer to the following paper:

- T. Griffin, F.B. Shepherd, and G. Wilfong. *Policy Disputes in Path-Vector Protocols*. IEEE ICNP 1999. <http://www.ieee-icnp.org/1999/papers/1999-3.pdf>

Your solution should be based on the local detection of divergence by each AS. You can consider only one destination, which should eventually become the root of the set of paths, denoted by  $\pi$ , from all other ASes. The basic idea is to add at each AS node a measure of divergence in the form of a price. This price should increase whenever BGP diverges. Whenever the new path of a node  $u$ , denoted by  $best(u, \pi)$ , has a lower rank than its current (previous) path, denoted by  $\pi(u)$ , the price of the node is increased. However, if a non-divergent alternative path is found, the price of  $u$  is allowed to decrease. The following pseudo-code specifies the set of condition/action for node  $u$  to locally infer divergence:

$$\pi(u) = [u, \pi(next(\pi(u)))] \rightarrow price(u) = max(price(u), price(next(\pi(u))))$$

$$\pi(u) > best(u, \pi) \text{ and } next(\pi(u)) \neq next(best(u, \pi)) \rightarrow \pi(u) = best(u, \pi); price(u) = price(u) + 1$$

$$\pi(u) < best(u, \pi) \text{ or } next(\pi(u)) = next(best(u, \pi)) \rightarrow \pi(u) = best(u, \pi); price(u) = price(next(\pi(u)))$$

The first action enforces that the price of a node is never smaller than the price of its next node along its path to the destination (root). The second action increases the price since the new best path has a lower rank than  $u$ 's currently chosen path. The third action is enabled when the new best path is ranked higher than the current path, or when the rank of the new path is lower but the next node is the same as that of the current path—in both cases, the price of  $u$  may decrease.

1. Show that for any node  $u$ , if  $\pi(u)$  changes infinitely often, then  $price(u)$  grows indefinitely.
2. Design, analyze, and discuss implementation issues for a solution that makes use of the above divergence inference algorithm to break the cycles and ensure convergence. Specifically, show the pseudo-code for the set of condition/action at each AS node, discuss how BGP would need to be modified or extended, prove that your solution would converge to a stable state starting from any state, and what experiments would you carry out to evaluate the performance of your solution.
3. Illustrate the successful operation of your proposed protocol on a BAD GADGET configuration.
4. How does your proposed protocol compare to existing solutions to BGP divergence?
5. The pricing function defined above increases linearly by one regardless of the exact ranking order of the paths, i.e. how much lower in rank the new path is. Discuss the implications of choosing a *ranking-dependent* increase/decrease pricing function, what this function might look like and why, and any changes that need to be made to your solution.

**Networks PhD Depth Exam**  
**November 2004**  
**Question 5**

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

- directed diffusion
- maximum likelihood estimation
- permutation scanning
- principal component analysis (PCA)
- probabilistic counting with stochastic averaging (PCSA)
- QR matrix decomposition
- randomized rounding
- threshold-based random walk
- wavelet denoising

Make a case for the one technique above that could be best applied to each of the following problems. Do not argue for a technique beyond those listed above, although you may mention other methods in your writeup. When several techniques on the list are applicable, make a case why your choice is superior. Do not mention any techniques that are not applicable. For each case, you should also briefly describe how you intend to apply the technique to the problem at hand. If there are any weaknesses in applying the method to the problem, briefly describe that as well.

1. Consider a multicast-enabled overlay network with synchronized clocks in which measurement packets are periodically multicast from each host to all other hosts. Also assume that the routes between hosts are known. Which method would be most applicable for identifying the loss rates on network-internal links?
2. Suppose this same set of hosts are participating in a Chord ring. A denial-of-service attack they are trying to jointly prevent is one in which a malicious host with a valid IP address requests many very large files. Assuming the Chord hosts have enough resources to monitor all users and can share this information, which technique might they employ to thwart such an attacker?
3. Describe how one of the techniques above might be adapted for use on the Chord ring to propagate urgent messages when a concurrent attack by many such malicious hosts is underway.
4. After the attack subsides, the log files of the overlay hosts are a measurement gold mine. Which technique could be employed to determine the approximate magnitude of the attack, both in terms of bytes downloaded and the number of attackers?
5. Consider the sliding-window join problem studied in the Das, Gehrke, Riedewald paper on your reading list. To prove their Theorem 2, they use the fact that there exists an optimal integral solution to their integer programming formulation to give a polynomial-time algorithm. Describe how they might have employed a method above if this fact were not true.