CS 7260: Internet Architectures and Protocols January 24, 2006

Project Suggestions

Instructor: Prof. Nick Feamster *College of Computing, Georgia Tech*

Please do not link to this page from anywhere, and do not redistribute this handout electronically. Thanks!

This writeup contains some project suggestions, which are *not* complete specifications, and in some cases are intentionally vague. A lot of the fun in research is figuring out precisely what to work on and what questions to ask.

We strongly encourage you to come up with your own ideas, including by modifying the suggestions in this document, and run them by. Please send me email if you have any questions or comments about any of these suggestions. It's fine to do a project connected to your research or thesis work, or to do a joint project with another class (*e.g.*, Russ Clark's Network Management class) provided there's a strong networking aspect to it.

Note: *It's your job, not mine, to explain why your project has something new to offer. This means that you should do a diligent and convincing job of understanding and presenting previous relevant research.*

1 Routing, Traffic Engineering, and Modeling

1. **Scaling routing/forwarding to very large networks.** Traditional interdomain routing uses "hot potato" routing among the egress routers that have learned a route to a destination. This scheme requires *all* routers within an AS to maintain complete routing tables for all destinations, which may become untenable for very large routing tables.

   An alternate routing scheme would impose some degree of hierarchy within an AS by having only a subset of the routers (*e.g.*, the egress routers) maintain routes to all destinations, and having the rest of the routers simply forward the traffic for which they did not have routes to *some* egress router (which would then, in turn, forward the traffic to the proper egress router). Another possibility would be to divide routing table state across routers in the AS, so every router carried some subset of the table state; routers without the state could then forward traffic randomly to some other router in the AS, until the traffic reached a router that actually had the proper routing table entry.

   One project might be exploring the tradeoffs between maintaining complete routing table state, and the cost imposed by having each router only maintain a subset of the routing tables.

2. **Incentives for more specific routes.** To control routing table size, routers maintain routes for IP *prefixes* using a process called aggregation. Aggregation dramatically reduces routing table size because Internet addressing is typically hierarchical. Unfortunately, ggregation can hide important information, such as the fact that groups of destinations within a single prefix may not share fate (*e.g.*, the destinations may be geographically distributed). Perhaps the routing protocol could incorporate pricing or incentives to encourage ASes to carry more specific prefixes?

3. **Alternate billing structures and business models for routing.** Many of the problems with Internet routing today result from the fact that ASes selectively filter rotues. This practice results from the fact that advertising a BGP route is an explicit agreement to carry traffic for that route (*i.e.*, based on some previously arranged contract). Because these contracts are only established bilaterally (not globally), some AS that is downstream of the ASes that made such an arrangement might actually suffer from these ASes' decision to filter a path. An alternate approach might be to advertise *all* reachable paths, but affix prices or costs to each route. This attribute, plus some mechanism for accounting for when a path is used, might improve the performance and resilience of Internet paths. There may be ways to achieve this without source routing, but considering source routing is also an option nere.

4. **The "social cost" of restrictive peering.** Selective filtering of routes, or "restrictive peering", often creates wildly suboptimal paths in the Internet. For example, there was a time when the Internet-level path from my Cambridge grad school apartment to MIT (half a mile down the road) would traverse Washington, DC, simply because my ISP (Comcast) did not directly exchange traffic with MIT, and their respective upstream ISPs did not have a peering point in Cambridge, MA. In this project, you could attempt to quantify this effect: how do the myopic decisions made by ISPs increase the latencies seen by end users on various paths. This project is measurement-based, so part of the fun would be figuring out how to measure this effect.

5. **Inbound traffic engineering.** How effective is AS path prepending at controlling inbound traffic vs. other methods (*e.g.*, advertising more specific prefixes). I have access to a place where we can announce prefixes and routes with various types of attributes (as well as measure incoming traffic) that could help you study this question.

6. **Does rotue-flap damping cause Internet outages?** Intuition says that routing instability should follow an end-to-end path failures; *i.e.*, when a link fails, at some later point in time the routing protocol should detect this failure and propagate the information about the failed link. Some of my previous work has observed that routing instability can actually *precede* an end-to-end path failure. This phenomenon suggests that routing "features", such as route flap damping, might actually *cause* outages.

7. **(Sub)optimality of BGP.** How often does BGP select a suboptimal path? You can define "suboptimal" in any number of ways, but the most interesting ones relate to end-to-end delays or bandwidth. We have some resources that can help for this project, including tools.

8. **Evolution of the AS-level Internet topology.** The nature of Internet topology and the right models to describe it has been a topic of much active work. A recent paper (H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, W. Willinger, *Network Topology Generators: Degree-Based vs. Structural*, *SIGCOMM 2002*, Pittsburgh, PA) has one answer to this question and, equally importantly, references to much other work on this problem.

   As you will see from the paper, the jury is still out on this question, so it's worth looking at. A relatively new take on this problem would be to look at Routeviews tables over a long period of time and develop a model to describe how the current AS topology emerged (as well as a model to describe the topology itself). If we had insights into how we got to the current topology over time, we might gain valuable insight into a process that describes the Internet topology. You might also consider developing an "organic" model for the growth of interconnections between Internet ASs.

If you are interested in working on this project, you may also wish to speak with Prof. Constantine Dovrolis.

9. **Studying TCP implementations on routers.** Use TBIT to study various TCP implementations on *routers*. This is interesting because BGP4 runs over TCP, so it's worth knowing the properties of different router implementations. You should consider inventing new signatures of TCP implementations and incorporate them into Tbit, as you start playing with the tool.

10. **IGP convergence.** There are a bunch of different factors that affect the convergence of IGPs like OSPF (detection, flooding delays, route recursion, shortest-path calculation, etc.). How much does each of these factors contribute? For the worst offender(s), what's a good solution?

11. **Inferring router behavior.** Router vendors often sell routers that claim to implement congestion control mechanisms such as RED or fair queueing with a fixed number of queues. Your task in this project is to come up with end-to-end experiments to infer what a router is doing, treating the router implementation as a black box. Specific question to answer include: (1) What's the size of the router's buffer? (2) Does the router implement RED correctly and what are its parameters? (3) What does the algorithm to map flows to queues in fair queueing look like? Does the router have different queueing schemes for large and small packets?

    If you get the basic stuff working with experiments in a "clean room" (e.g., on a PC-based router running Linux or FreeBSD) with no other traffic, you might get ambitious and try your experiments on real-world routers and compare what they do with their specs. We should be able to provide you with a couple of real routers to work with.

12. **Forwarding with fast updates.** Good IP forwarding schemes tend to heavily compress forwarding tables on chip to scale to large sizes. This makes incremental updates problematic. A scheme was proposed in *SIGCOMM 2000* for this problem, but you may be able to come up with something better. This is a good project to sink your teeth into if you like algorithms and data structures.

13. **BGP communities.** What are all the possible things BGP can be used for? Many ISPs use communities to control re-advertisement besides the common ones (e.g., no export, no peer), what other published types of mechanisms are people using (and can you actually witness the effects of this in the routing table?) Can these ISPs community functions be used against them in some fashion? (e.g., DoS attack, etc). Maybe something related to multihoming?

2 Network Security

2.1 Infrastructure security

1. **Packet watchdogs.** A router should reject packets destined for hosts with no corresponding advertised route. More generally, a router should reject packets from sources that should not have a valid route through this router to the destination. Deploying packet filters only at routers in large ASes in the Internet "core" could eliminate most of these packets, but an AS must be able to construct these filters in the first place, which would require discovering the routes from the source to that AS. How can an AS determine the source and destination addresses it should see on packets arriving on a link? [1]

[1] This question has a dual for *route* filters in the control plane.

2. **Defending DNS against attack.** One way to attack DNS, which we discussed in class, is to poison DNS caches across the Internet with bad entries and long TTLs. Come up with a scalable and viable solution to this problem. First read Steve Bellovin's paper on DNS weaknesses, and also research what has been proposed so far.

3. **Tracking causality in networks.**

4. **Scalable, efficient, deployable interdomain routing security.** One of the major drawbacks of currently proposed solutions for securing BGP route advertisements (*e.g.*, S-BGP, soBGP, etc.) is that all of these solutions require deploying a public key infrastructure.

5. **Forwarding path assurances.** The current Internet routing infrastructure provides essentially no assurances about where traffic will actually go (*i.e.*, there's no "data plane security"). This shortcoming might be problematic for ISPs or end hosts that, say, want to ensure that their traffic does not traverse a specific ISP (*e.g.*, "don't send my packets through AT&T's network"). Design a system or protocol change that could provide such assurances.

6. **Design and simulation of an infrastructure worm.** "Traditional" worms infect and spread among end hosts. In class, we read a paper that modelled the spread of a very fast spreading end-host worm. You might consider doing an analogous project, but for a worm that infrects the routing infrastructure. How might such a worm spread, and which nodes would you attack? If the goal of such a worm would be to disable this infrastructure, how might you design a worm that does so in a way that still allows the worm to propagate? How might you defend against such a worm (*e.g.*, with device heterogeneity, protection of critical infrastructure, etc.)

2.2 Worms, viruses, botnets, and spam

1. **Virus epidemiology.** The course readings have a paper by Staniford et al. on virus propagation. This model does not account for "hubs" that are well-connected, which might be able to do substantially more damage than other nodes that aren't so well-connected. You could address the question of what the effects of a more targeted worm are, e.g., on critical infrastructure such as routers or DNS top-level servers.

   To be valuable, this project should attempt to garner data that you can use to validate the model. We don't have access to any such data, unfortunately, but this does not have to be a show-stopper; however, you'll need to be creative in both coming up with models, and showing somehow that they're realistic. (E.g., you may be able to model the propagation of email-based viruses and somehow show that the model approximates reality.)

2. **Virus throttling.** Recently, there has been a proposal that seeks to limit the number of concurrent outgoing connections (to five) in order to reduce the rate that a virus propagates. If you don't believe five as a magic number, can you come up with a better idea to set a rate limit? Figure out how well these kinds of approaches might work. See `http://www.usenix.org/events/sec03/tech/twycross.html`

2.3 Anti-censorship and anonymity

1. **Circumventing censorship with "viral" propagation.** Infranet (`http://nms.lcs.mit.edu/projects/infranet/`) is a system for thwarting web censorship. Code available on

sourceforge. Set it up as a service on Planetlab. One major drawback of Infranet, as it stands, is that *every* client must make contact with a "responder" on the opposite side of the firewall. This technique does not make a whole lot of sense for content that large numbers of clients want to retrieve (*e.g.*, cnn.com, nytimes.com, etc.) A more tenable solution might be to only require a small subset of clients to bring content across the firewall, and then design a CDN to allow the rest of the clients to retrieve the "forbidden" content from other clients that have already retreived it. One approach to this problem might involve the use of delay-tolerant networks (for example, one might imagine the content being brought across the firewall on a USB key, iPod, etc.)

2. **The "proxy discovery" problem for Web censorship.** Figure out a way to split its "responder" into two parts—one that's untrusted that just does forwarding, and one that is the actual covert channel end point. See the papers by Feamster *et al.* The second paper from the privacy workshop hasn't been implemented, nor have all the details been fleshed out. We'll use your software in deployment!

3. **Exploiting social networks to construct anonymous communications networks.** Anonymous communications networks (*e.g.*, Tor, Crowds, Infranet, etc.) are typically designed with the idea of hiding communications between parties. Another way to build such a communications network, however, would be to exploit known social networks to construct new communications "links" between other pairs of parties. In this project, you could design an anonymous communications network that can cloak communications between arbitrary parties under the guise of traffic between pairs of parties that are already known to communicate with one another.

4. **Anonymous (or censorship-resistant) VOIP.**

2.4 Other

1. **Click fraud.** Competing companies often employ large armies of botnets to submit clicks on a competitor's Web site, in an attempt to drive up their costs. This practice is known as *click fraud*. Devise a scheme or system for detecting or preventing click fraud.

2. **De-anonymizing web logs using timing attacks, common access patterns, etc.**

3 Network Management and Troubleshooting

1. **Static configuration analysis for VPNs.** This project involves further exploring the types of errors that static configuration analysis can detect. For example, you could investigate errors caused by the interactions between a network's BGP and internal routing configurations (see the background papers for more details on this problem). You could also analyze some other aspect of the network configuration (e.g., a network's VPN configuration). In any case, you'll be finding errors in the configurations of operational networks.

2. **Joint static/dynamic analysis to detect misconfiguration.** Incorrect routing information, whether introduced accidentally or maliciously, can have disastrous effects on communication. Unfortunately, misconfiguration and malice are far too common: network operators misconfigure the Internet's routing protocol in a way that is globally visible about once every

few days, and nefarious individuals (e.g., spammers) introduce bogus routing information into the global Internet on a daily basis.

Of course, the events that affect some AS are present in the routing messages received by that AS. The challenge lies in extracting the meaningful anomalies and faulty routing information from the vast majority of routing messages that reflect insignificant changes or actual updates in the network topology.

3. **A system for distributed network troubleshooting.** The Internet provides minimal facilities for a network operator to debug global reachability problems; worse yet, local actions by a single ISP (*e.g.*, filtering) can disrupt global connectivity in perplexing ways. The state-of-the-art for distributed troubleshooting is for a network operator to send an email to the NANOG mailing list (*e.g.*, "I can't reach prefix X? Can anyone else see this prefix?", "I'm announcing prefix X, but my customers can't see site Y. Who's filtering my route advertisements?"). In this project, you could design a distributed troubleshooting system to help network operators answer these questions more easily (perhaps with an overlay network). A successful project in this area could have some real operational impact (and perhaps earn you fame or fortune)!

4. **Design patterns for router configuration.** Configuring a network of routers to run a routing protocol is like writing a large distributed program. Unfortunately, unlike programmers, network operators have few tools that make routing configuration easier. In particular, operators much configure each router separately, rather than configuring the network as a whole. This project gives you opportunities to explore how an alternate "programming model" could help reduce the complexity of routing configuration. Much of a router's configuration is replicated exactly across every router, or modified only slightly. Can you design techniques that automtically recognized these "design patterns" and help a network operator reduce this repeated effort?

5. **Efficient queries on large datasets.** The Internet contains massive datasets: packet traces, traffic statistics, traces of routing protocols, etc. Internet measurement research relies on efficient ways to process queries over this data. Unfortunately, the sheer size of the data, in combination with the fact that it may also be distributed over many collection points, can make even simple queries difficult. The following two examples illustrate two challenging data management problems:

   - Longitudinal studies over large datasets. RouteViews maintains a massive archive of the Internet's routing messages, as well as snapshots of the ISP-level topology. Suppose you wanted to perform a query that involved looking at the routing update messages for a single IP prefix. Doing this today would require parsing all of the archived files and filtering the vast majority of data that you didn't find interesting.
   - One-way delay and outage queries. The hosts on the RON testbed are equipped to measure one-way delay and loss along the Internet paths between the testbed machines. Actually maintaining these statistics, however, requires a large "merge" operation, since no single host maintains information about any one path between a host pair.

   Due to the size and distributed nature of many Internet datasets, performing even simple, exploratory queries quickly becomes a major endeavor. Can you suggest data processing algorithms or techniques that would make the problems described above less frustrating?

6. **Effects of routing instability on end-to-end performance.** Study in more depth the extent to which routing instability degrades application-level end-to-end performance (continuing on previous work by Feamster *et al.*, Sigmetrics 2003). Along these lines, you could also ask questions about the extent to which overlay networks prevent degradation of application-level performance. Overlays take time to detect failures, and switching paths also imposes some degradation in and of itself (*e.g.*, packet reordering, etc.).

7. **Running an IGP on an overlay network to make it scale better.** It is widely held that routing overlay such as RON do not scale to large numbers of nodes because these overlays require active probing on the paths between every pair of nodes to quickly detect failures. However, you might be able to design a more scalable overlay network by only probing some set of these pairwise paths, and running a routing protocol (analogous to an IGP, or some shortest-paths routing protocol) to discover paths that are not probed directly. At what point (*i.e.*, network size) would such a solution become a more tenable option than direct probing of all paths, and how well does this technique work in practice?

8. **Incentives for keeping routing registries up-to-date.**

4 Network Architectures

1. **Implementation of the Routing Control Platform.** The Routing Control Platform (RCP) is a system that separates the logic of route selection from the actual forwarding of packets. RCP collects internal routing topology data and BGP routes to external destinations, and computes the routes that each router in an AS should select. A conventional RCP deployment would just "emulate" the outcome of a fully meshed iBGP configuration—*i.e.*, it would compute the route that each router would select in such a configuration and push those "answers" to the individual routers in the AS. In this project, you could augment an existing software router (*e.g.*, XORP, Quagga, etc.) to include RCP functionality.

2. **Support for experimentation with network architectures and protocols.** An ongoing project is attempting to set up a shared experimental networking infrastructure on PlanetLab, whereby researchers can experiment with software routers on a *virtual* network infrastructure on PlanetLab. This infrastructure (which is still in the works—you're welcome to help build it in any way you like) allows experimenters to specify a virtual network topology, incluing internal routing protocol adjacencies and iBGP sessions, as well as a schedule on which to fail links. The idea is that a network experimenter should be able to specify an experiment and seamlessly migrate it from an emulation environment (*e.g.*, Emulab) to a more realistic network scenario (*e.g.*, a virtual network infrastructure on Planetlab). This virtual network infrastructure requires various aspects of support:

   - *Underlay monitoring.* Studying failures in a virtual network infrastructure like Planetlab requires the ability to inject failures on virtual links, but it also requires the ability to *monitor* when underlying failures occur on the IP network. In this project, you could design and implement monitoring infrastructure that exposes the conditions of the underlying network.
   - *Interface to the "outside world".* A major advantage of a virtual network infrastructure would be its ability to interface to other, "real" neighboring networks on the Internet. For example, given the ability to advertise reachability for some set of destiantions to

neighboring networks, an experimenter could conceivably study the effects of routing instability (*i.e.*, within the virtual network) on *incoming* traffic. In this project, you could design and implement a way enable such a study.

5 The Last-Resort Department

If none of the above projects interests you and you aren't able to come up with one on your own, here are a few suggestions to get you started in finding a project that does. Find some problem/area in networking that you like (or better still, that truly excites you, and/or that seems of immediate and important relevance to you). Then address the following:

1. Has a solution been posed to this problem? What do you think of it?
2. If a solution has been proposed, has it been simulated and evaluated properly?
3. If the system has been simulated, does anyone understand its mathematical properties?
4. If the system has been simulated and theoretically modeled, has it been implemented and its performance studied under realistic conditions?
5. Is the solution scalable? Is it robust? Secure? How well does it hold up in the face of failures? Will it work in heterogeneous environments? Will it stand up to coming advances in technology?
6. If several solutions have been proposed, has anyone performed a comparative study of them? Why do some schemes work better than others? Can we characterize the conditions under which some schemes work better than others.

You'd be surprised at how many good PhD theses got their start this way. Good luck!