Session 3: Demystifying the network - the theory, models & testing

Transcribed by: M. Taimoor Tariq, Ammar Tahir (UIUC)

Automating network heuristic design and analysis

Anup Agarwal *(Carnegie Mellon University); Venkat Arun (MIT CSAIL); Devdeep Ray, Ruben Martins, Srinivasan Seshan (Carnegie Mellon University)*

Summary:

Community uses multiple heuristics e.g. CCAs, arguing about correctness is difficult. By using CEGIS search the input space (environment) and heuristic space (e.g. set of templates for CCAs) and metrics of interest, and guide the model through counter examples. CEGIS is slow however, using optimizations help with it e.g. guided pruning of CCA search space.

Questions:

1. **What impact interaction of multiple client has on this modeling**
   Jitter can e.g. be used to simulate other clients or model explicit interaction but it increases search space.
2. **Does dynamicity (e.g. non-congestive delay) make a difference? Would CCAs converge then?**
   It depends on the definition of a steady state region, having a larger region accounts for dynamicity.
3. **Can we model topologies?**
   It's not modeled yet but it can be interesting.
4. **Can we do this online instead of offline?**
   Heuristics built are offline, no component of the system is online. We are not learning any parameters through this.
5. **Not enough attention is given to short-term behavior, eg. when an application has to send in short bursty traffic. Have you looked at short term transient properties? E.g. performance during start up of a flow, pause and restart.**
   At the boundaries of the snapshot, link rate can vary arbitrarily. If not performing well in the start, it needs to do better towards the end of the snapshot. Don't have constraints on how quickly it will converge to the stable state but you can introduce some level of constraints to ensure that it converges to the stale state in a given amount of time.
6. **Interaction of different clients, how to handle unknown number of clients and what impact that would have**
   The way we model jitter, can also be an artifact or interaction from other flows. That increases the size of our modeling and slows down our system. It is a challenging problem and our solution is not a completely clean one yet
7. **How easy is it scalable to behaviors where it is topology dependent?**
   Not explicitly thought about it. Environment needs to be modeled accordingly. Currently we are working inside SMT, not sure if we can create a parametrized topology.
8. **Can you do this in real-time to speed it up? Is there a lot of benefit to doing it in 1 minute vs 1 hour if you are doing it offline anyway?**

The heuristics that we are building are offline. They are not learning anything. Once we have decided all the parameters, its a fixed heuristic. These optimizations help since otherwise these can run for days or months and it is not even sure if it will even complete. So we need a bound for how long it will run for.

CC-Fuzz: Genetic algorithm-based fuzzing for stress testing congestion control algorithms
***Devdeep Ray**, Srinivasan Seshan (Carnegie Mellon University)*

Automated tools for testing and designing congestion control algorithms. Use fuzzing to find issues in existing congestion control algorithms.
Existing techniques don't really test the code paths which actually lead to those problems. Manually generating these pathways is tedious. So basically, how to design scenarios to design scenarios which trigger bad behavior in CCAs. CC-Fuzz found new and existing vulnerabilities in BBR and TCP-Cubic.

Questions:

1. **Do you model one flow or more as well?**
   Yes, the system only has one flow but we do model cross traffic.
2. **Are you after implementation bugs or semantic bugs?**
   We are mostly testing implementation bugs,e.g. NS3 implementation. It is going to find bugs for both, bad implementation as well as bad design. For example, the TRO bug is because of the way BBR tracks the RTO cycles.

Minding the gap between Fast Heuristics and their Optimal Counterparts
***Pooria Namyar**, Behnaz Arzani (Microsoft Research); Ryan Beckett (Microsoft); Santiago Segarra (Microsoft Research); Himanshu Raj (Microsoft); Srikanth Kandula (Microsoft Research)*

Existing methods like simulated annealing fail to find adversarial inputs while testing heuristics because they do not take into account environment context. MetaOpt uses a two level optimization formulation where adversarial generative formulation tries to find adversarial input and heuristic tries maximize input.

Questions:

1. **How do you guarantee to find the worst case input?**
   The first level of optimization tries to maximize the gap
2. **How feasible is it to always find the optimal solution? Or is it always possible?**
   It takes time but it terminates when it is able to find an adversarial input.

End of Session Discussion

1. **You can in many cases have a small sequence that is good enough to identify the problem instead of a long sequence of events.**
   We try to minimize the amount of cross traffic to find interesting adversarial example.
2. **For the first paper will you be able to find a powerful enough CCA from search space and for the second paper will you be able to find a power adversary?**
   In formal methods what you get is accurate. With fuzzing it is easier to do but no guarantees. With the fuzzing approach, you can model acceptable traces using consensus on previous work.
3. **You are taking an optimization/verification approach with the model. The heuristic has to have some level of simplicity. How complex can that heuristic be? How complex of a heuristic can you accept and represent.**
   It just needs to be representable in a convex form. It depends a lot on the heuristic. Most of the heuristics are convex but representing them in convex, requires a lot of work. Moreover, instead of using convex constraints, you can also extend them to non-convex constraints. One idea, you don't need to model the heuristic in the solver. Suppose you have a combined objective, with two individual objectives, where the combined objective does well

4. **Does the vast majority of networks require these worst case heuristics?**
   In general if you choose one or the other, you are forced to make a tradeoff by definition of these techniques. A lot of the learnings we take can also be used for heuristics which cannot be incorporated into these mathematical models. I think the trend has been that the scalability of these algorithms will help out. Our search space will be higher and we will be able to search the space more. So it might not be a tradeoff anymore.

5. **Lower layers are probabilistic because of the complex nature of lower layers. One of the things these systems (presented in last session) are assuming is that the network is not adapting to what you are assuming. What happens when the underlying network is adapting to the traffic as well.**
   You can't gather traces from one protocol and use them for another. Some degree of causality model, what would happen on a different network Whether these works apply to a model where the network is adapting to what the network is seeing?

6. **In the game model, we can model such behavior as hidden information, so we will have a probabilistic proof.**

   Causality is important, our choice of picking CCA is driven by causality: pick CCA that satisfies the previous counter example.