Boon Thau Loo Speaks Out
on His SIGMOD Dissertation Award, Better Networking Through Datalog, Life as an Assistant Professor, and More

by Marianne Winslett



http:// www.cis.upenn.edu/~boonloo/

Welcome to this installment of ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are in Alexandria, Virginia. I have here with me Boon Thau Loo, who is an Assistant Professor of Computer and Information Science at the University of Pennsylvania. Boon is the recipient of the 2007 ACM SIGMOD Dissertation Award for his thesis entitled "The Design and Implementation of Declarative Networks." Boon's PhD is from the University of California at Berkeley, where his advisors were Joe Hellerstein and Ion Stoica. So, Boon, welcome!

Thank you, Marianne.

Tell us about your thesis.

My thesis research is on declarative networking, which proposes a declarative framework that supports extensible specifications of networks. Our work is based on the insight that recursive queries, which have been traditionally used for querying graphs in databases, are a very natural fit for expressing network routing protocols, which are themselves based on recursive relationships among nodes in the network. We use a distributed version of Datalog, which we call *Network Datalog*, to specify queries that are compiled and executed by a distributed query processor to implement the network protocols.

What kinds of network protocols can you implement this way?

We can implement a wide variety of routing protocols. Initially we looked at how you can use this approach to build standard routing protocols, such as distance vector and path vector protocols. The interesting thing is that you can implement these protocols in a handful of lines of Network Datalog code, which is orders of magnitude less than the traditional implementation. We have also taken this one step further by showing how you can make use of this declarative framework for building more complex networks. For example, we specified the Chord

distributed hash table in 48 lines of code, which is two orders of magnitude more compact than the original C++ implementation.

How did the networking community react to your work?

They have reacted positively for the following reasons. First, our approach provides ease of programming. Not only can you build your routing protocols in a few lines of code, you can also go from one protocol to another by making small modifications. Second, using a declarative framework makes it easy for you to reason about the correctness properties of the resulting network. So, for example, by checking the safety properties of these recursive queries, we are essentially testing for the convergence properties of networks.

Had the networking community's experience with active networks make them nervous about having such flexible routing protocols?

I think it is indeed one of our challenges. Our approach leads to a restricted, safer instantiation of active networks. You can think of Network Datalog as an attempt to achieve a sweet spot between extensibility and safety. So on the one hand, we argue that we can use the declarative approach to prototype a wide variety of routing protocols. On the other hand, we can also ensure that these protocols are actually safe when you execute them in the network.

Does Network Datalog look different from regular Datalog?

The main difference between Network Datalog and regular Datalog is the use of location specifiers to specify data placement. As an example, consider a `link(@Src,Dst)` table, where each `link` tuple is stored based on the value of its `Src` field.

Is there an assumption that all nodes cooperate, i.e., if you ask somebody a query, they will definitely send you an answer and it will be correct?

In our original execution model, we assume that all nodes executing our queries are trusted. Instead of running a standard routing protocol, they run a general purpose query processor. If a node wants to start a new routing protocol, it injects the protocol as a distributed query, which is then disseminated through the network and executed in a distributed fashion at all the nodes.

Could a node attack by not answering the query properly?

Declarative networks reduce the likelihood that a network protocol designer implements the protocol in an unsafe way, e.g., so that it will never terminate or converge. They do not protect against malicious nodes in the network, i.e., nodes that can misbehave or execute something other than your query. One of my ongoing efforts addresses the security challenges of executing declarative networks in untrusted environments.

What was your experience in publishing this work? Did you send it to the networking community?

We decided right at the start that we needed to validate some of our ideas with the networking community. So, rather than try to write a paper on distributed recursive query processing per se, we instead submitted some of our initial work to networking conferences. One of the first venues where we published our work on declarative networking was at HotNets 2004, which is a workshop on emerging research directions in networking. Our position paper proposed the use of declarative queries to enable end-hosts to set up customized routes over the Internet. In the

following year, we published two additional papers targeted at the networking community: a SIGCOMM '05 paper that proposed the declarative framework for building safe, extensible routers, and a SOSP '05 paper that extended the declarative framework to support the rapid prototyping of complex overlay networks.

Based on our experience with declarative routing and declarative overlays, we then tried to ask more fundamental questions about executing recursive queries in this new environment. We looked at the semantics of recursive queries when you distribute and execute them over a network that is constantly changing. For example, we tried to understand what happens if you use incremental view maintenance techniques to maintain routing protocols, in the event that the network keeps changing. We came up with ways of reasoning that are based on "eventual consistency" semantics.

We also looked at the way traditional recursive queries are processed, which is known as semi-naive evaluation. With semi-naive evaluation, you make synchronous rounds of computation, with the goal of avoiding redundant computation. If you try to do this in a distributed environment, it doesn't quite work because you have nodes that could fail, and you could also have congestion in the network. It becomes very expensive if all the nodes must come to a consensus at each round before one can proceed onto the next round. To address this problem, we came up with what we call pipelined semi-naive evaluation. This approach relaxes semi-naive evaluation, but at the same time avoids this expensive consensus required at every round. The challenge there is to ensure that even though we relax semi-naive evaluation, one can still guarantee correct results, and compute them in an efficient manner.

The third topic that we looked at is the use of standard database query optimization techniques in the context of network protocols, such as magic sets rewriting and predicate reordering. This actually led to interesting connections with network routing optimizations. Let me give you an example: consider the path-vector routing protocol used in standard inter-domain routing protocol today. Suppose that you specify this protocol in a few lines of declarative Network Datalog, and then you apply standard database query optimizations, using predicate reordering and magic sets to limit the computation from sources to destinations. After applying the optimizations, you actually switch the protocol from path vector to dynamic source routing, which is intended for a wireless context! Intuitively, this makes a lot of sense, because in a wireless environment, you have nodes coming and going, and you have a high degree of mobility. It is actually very expensive to compute the routing tables proactively for all pairs of nodes. So the right thing to do is to compute routes in a more reactive fashion for selected source and destination nodes.

The interesting conclusion is that by applying a standard database query optimization, we can transform one network routing protocol into another. To take this one step further, we can make cost based decisions; depending on the statistics of the network and the degree of mobility, you can decide whether to use a more proactive protocol or a more reactive protocol---or in the best case, come up with a hybrid version based on the statistics of the network.

What are you working on now?

I am working on a variety of things, but I would like to focus on two, which are especially related to declarative networking. The first project is on declarative secure networks, which was started while I was visiting Microsoft Research Silicon Valley as a postdoc, six months prior to joining the University of Pennsylvania. This work was inspired by one of my MSR colleagues. He observed that the logic based languages that are being used in access control and trust management have a lot of similarities with Network Datalog. For example, instead of having

location specifiers that indicate the placement of data, instead you have security principals such as Bob and Alice make security-related assertions.

Based on this observation, we proposed a unified declarative language called *Secure Network Datalog* that can be used to specify declarative networks with security policies. We have explored using this language to implement secure network protocols in untrusted environments, including secure inter-domain routing protocols, DNSSEC, and secure routing in distributed hash tables. Recently, we also started looking at how we can exploit the fact that we have a common language and system that allows you to do both security and networks. For example, maybe this will make it easier to analyze the correctness or security properties of network protocols.

Another area of research that I am currently working on is applying declarative networking concepts to provide flexible network support for mobility. In a recent MobiArch '07 (co-located with SIGCOMM) workshop paper, we used the declarative approach for building extensible application-aware mobile networks that enables flexible addressing and naming of mobile devices, and session-aware quality-of-service routing. We are further looking into developing a suite of declarative wireless ad-hoc networks.

With a student at Penn, we have also recently looked at the anonymity aspect of network security: how can you build routing infrastructures that guarantee anonymity properties between source and destination nodes, but do so in an application-aware manner. So, in other words, each application can specify its own performance constraints in terms of latency, loss rates, and path throughput, and anonymous routes are set up based on the application constraints. Early results of this work appeared in HotSec '07.

In the case of anonymous MIX networks, whether your anonymity constraints could be met might depend on what other people's constraints were, in the sense that other people's traffic helps hide your own traffic.

We have focused primarily on Tor networks that implement onion routing for anonymous communication. Even without the use of mixes, we realize that there is an inherent tradeoff between performance and anonymity: network path diversity leads to increased anonymity at the expense of network performance. We argue that there exist several applications that desire not only customizable routing, but also the ability to automatically fine-tune performance and anonymity to meet their specific needs. We also believe that declarative networks and runtime optimizations can be useful in this context.

You just completed your first semester as an assistant professor. How did it go?

It has been an enjoyable, fulfilling, and exciting experience. I like to give the analogy of starting your own company. As an assistant professor, you are given initial funding by the university to start your research program. Based on this initial support, you have to develop a research program that can excite the funding agencies to provide additional resources, and attract good students to work with you.

The most enjoyable part of being a faculty member is interacting with students. In my first semester at Penn, I taught a research seminar titled "Networking Meets Databases". The goal of this seminar is to introduce students to current research at the intersection of databases and networking. The class attracted 15 students from a variety of backgrounds from our databases, networking, and security research groups. The topics covered include content-based networks and their implications on future Internet design, Internet-scale query processing, and data

management issues in sensor networks and delay-tolerant networks. To explore the synergies between the database and networking community, students read diverse papers from database and networking conferences. For example, to better understand cost-models used in network optimizations and publish/subscribe systems, students studied papers from NSDI, SIGMOD, SIGCOMM and VLDB. Two of the class projects from this seminar were published in MobiArch '07 and HotSec '07 respectively.

Do you have any other advice for graduate students?

I have two pieces of advice. First, if you are passionate about a particular area of research, you should persevere despite obstacles and setbacks. This advice is based on my own experiences gathered from my thesis work. It took about a year and a half before we got our first declarative networking paper into a workshop! We were putting together ideas from active networks and deductive databases, which were two areas of research that were considered no longer active. There was certainly a startup curve required to gain momentum in this work. So I think it is important that once you become excited about a certain area of research, you should persevere on it.

The other advice I have is for students to spend some time in industry. I myself immensely valued my time in summer internships. For example, working with the folks at Intel Research at Berkeley helped my thesis research a lot. I also spent a summer at the Intel Research Lab in Seattle, and later the postdoc at Microsoft Research. I think all of these experiences helped in terms of working with a wide variety of people, and also in grounding my work in actual problems that industry cares about.

If you magically had enough extra time to do one additional thing at work that you are not doing now, what would it be?

I would like to mention two additional things that I would like to do. First, it is important that we validate our research work via collaboration with the industry. To encourage the adoption of declarative networking ideas, I hope to work with actual Internet service providers, and commercial router vendors.

Second, I would like to explore collaboration opportunities with researchers in Asia. In addition to leveraging the huge talent pool, there are also emerging data management issues that arise from the proliferation of wireless communication devices, and the use of smart card and RFID technologies in emerging urban cities in Asia. As a start, I spent three weeks over the summer visiting two institutions in my home country: the National University of Singapore (NUS), and the Agency of Science, Technology, and Research (A*STAR). If time permits, I would like to visit on an annual basis.

If you could change one thing about yourself as a computer science researcher, what would it be?

I would like to strengthen my foundation in mathematics and statistics. Regardless of the research area you work in, a strong mathematics/statistics background will be really useful---e.g. in query optimization, network optimization, or machine learning/AI.

Thank you very much for talking with me today, Boon.

Thank you, it is my pleasure.