NetSE Network Design and Engineering Meeting Report

Edited by Ellen Zegura

7 October 2008

Meeting background

Under the auspices of the Network Science and Engineering (NetSE) Council, a small, invitation-only meeting was held in August 2008 just before ACM SIGCOMM in Seattle. Invitees included those with long standing involvement in NetSE and the predecessor GENI effort; those with interest and partial prior involvement; and representatives from NSF and the GENI Project Office. A complete list of attendees is attached.

The meeting was organized around five sessions on Architecture, Adaptability, Accessibility, Accountability and Edge/Enterprise Networks. Each session consisted of 1-2 talks of 15 minutes, followed by discussion.

The Network Design and Engineering part of the NetSE intellectual space is relatively well trodden, as represented in the Research Plan produced by the Planning Group (editor: David Clark (and Scott?)) and the "Why We Dream of GENI" document written by Scott Shenker. The focus of this meeting was to extract a small number of themes that best exemplify the research challenges and opportunities in the area.

Themes

Theme 1: Complexity of Requirements

Exemplar question: Should a future Internet achieve five 9's reliability?

We have entered the age of the "ities" in system design, where performance requirements are augmented with other criteria such as security, manageability, adaptability, predictability. While it is easy to agree on the high level value of these criteria, it remains a significant challenge to figure out how to reflect them in design. David Clark observes that the problem starts with the criteria themselves, which are not, in fact, requirements. Instead, they are a wish list of ill-defined properties. Requirements are have-to-haves, while wishes are nice-to-haves; it is possible, even likely, that not all items on a wish list can be simultaneously satisfied. The ill-defined nature of these properties limits our ability to make substantive progress, since we are lacking even common agreement and terminology about what we are trying to accomplish.

Put in terms of the 5 9's question, we cannot answer the question without understanding (1) what it means and (2) what it costs, in real dollars and in the possible sacrifice of other desirata such as universal access. Neither of these is trivial.

Intellectual opportunities abound. [two versions below – one is statements and the other is questions – need to pick one or write a hybrid]

How are high level desirata unpacked to reveal specific, well-formed requirements? Given a collection of specific requirements, are they expressed in minimal form (i.e., does any one subsume another)? Given a collection of specific requirements, are they all simultaneously achievable? If not, which represent fundamental tradeoffs? Which requirements are related and suggest a modularity in the system? Can requirements that conflict be de-conflicted? What are the specifications of the mechanisms a system should implement to achieve the requirements?

Each high level criterion needs to be unpacked to reveal specific, well-formed requirements that are associated with the terminology. This process isn't easy; requirements tend to get "slippery" when one attempts to make them precise. After a collection of specific requirements have been identified, they must be considered in concert for interactions, both positive and conflicting. Positive interactions between components suggest modularity in network design; conflicting interactions may reveal fundamental tradeoffs.

Theme 2: Abundance of Technology

Exemplar question: How do we enable on-the-fly composition of network protocols across heterogeneous devices?

In addition to a long list of criteria for networks, we are experiencing tremendous advances in the technologies available to help satisfy requirements. The design space is rich and interesting and growing. More computation and storage per data unit are available, enabling paradigms such as store-carry-forward in mobile networks, where packets are stored in intermediate nodes while waiting for contacts with other mobile nodes. Programmability goes all the way down to the optical photon or radio-frequency waveform, making standards obsolete at the physical and medium-access-control layers and leading to the development of platforms such as software-defined radios. Edge networks are increasingly heterogeneous. Virtualization has decoupled physical and logical views of network resources and enabled unprecedented sharing.

The current Internet architecture is able to leverage advances in technologies that line up well with Internet modularity – physical/MAC layer and application layer advances. The current Internet has difficulty exploiting advances that do not fit well in the architecture and may violate design principles such as a common narrow waist, and sharply change the application of other principles such as the end-to-end argument. Greater processing and storage capabilities inside the network are both advances that have difficulty fitting into the Internet's modularity.

Programmability down to the physical layer suggests an essentially unlimited design space for networks. At the extreme, a network could be custom programmed, on-the-fly and based on current conditions, for every invocation of an application. How should this design space be explored? How should a software-driven network be modularized?

Similarly, sharply increased storage capabilities in the network – indeed, along any path in the network – offers the possibility that anyone can cache anything in the network and the challenge of how to use this richness of storage space to maximum effect. If we pollinate this semi-infinite in-network cache with the idea of on-the-fly, custom protocols from the previous paragraph, we find challenges in reconciling shared in-the-network caches with individualized protocol suites. Are these two concepts in conflict? Or are they pointers to a radical revision of how we modularize data networks?

Pushing these issues one step further, we can see the potential for truly disruptive technologies in our future. For instance, suppose quantum computers actually start to work? Quantum computing is a disruptive technology for networks, potentially rendering obsolete the most frequently used mechanisms for network security while simultaneously demanding new types of networking technology (quantum repeaters) to enable distributed quantum computation. How should future networks plan for disruptive technologies?

Theme 3: Need for Experimentation

It is well understood that experimentation, together with analysis and simulation, is a critical tool for understanding how to build good systems. Experimentation can reveal behavior that was previously unanticipated by analysis and simulation, where reality is always modeled with some approximations. Experimentation assists in the creation and refinement of models that, when well constructed, can lead to better analysis and simulations. Experimentation is not a panacea; indeed, the design of good experiments can be as challenging as the design of good systems. The challenges of good experimental design increase as the scale of the experiment increases, and our field would benefit from more science to add to the art of experimentation.

Experimentation is useful at many scales, from small in-the-lab experiments to Planet-lab scale experiments that utilize geographically distributed endpoints and wide-area paths.

The trend in programmability has as its limit the ability to change the network, from top to bottom, on the time scale of software compilation. This has two important consequences for experimentation. First, this rapid idea-test cycle could lead to a paradigm shift in the speed and style of innovation both in the research lab and in the commercial innovation sector. A low bar for large-scale experimentation can also mean a low bar for large-scale innovation. Second, full programmability produces a huge design space. The ability to experiment with point solutions in that large design space is essential for developing an understanding of the space.

Meeting Agenda

Session 1: Architecture

4-6pm	**Session 1: Architecture** **What are the challenges and opportunities in future network architectures?** Talk 1 – David Clark Talk 2 – Craig Partridge Discussion
6:00pm	**Break**
6:30-8pm	**Dinner and Session 2: Adaptability** **How can future networks adapt to changes in technology, applications and use?** Talk – Jen Rexford Discussion

Day 2: Monday, August 18th

8-9am	**Breakfast**
9-10:30am	**Session 3: Accessibility** **What are the design considerations for making networks and their information widely available and accessible?** Talk – Nick Feamster Discussion
10:30-11am	**Break**
11-12:30pm	**Session 4: Accountability** **What accountability primitives belong in networks?** Talk – David Andersen Discussion
12:30-1:30pm	**Lunch**
1:30-3pm	**Session 5: Enterprise Networks** **How are edge networks changing?** Talk – Amin Vahdat Discussion
3-4pm	**Meeting wrap-up**

Meeting Attendees

David Andersen, CMU
David Clark, MIT
Mike Dahlin, U Texas
Nick Feamster, GT
Darleen Fisher, NSF
Suzi Iacono, NSF
Dmitri Krioukov, CAIDA
Craig Partridge, BBN and GPO
Jennifer Rexford, Princeton
Amin Vahdat, UCSD
Ty Znati, NSF
Ellen Zegura, GT

My notes – DELETE AT THE END!!!

Free form writing: Our field has always benefited from experimentation. Experiments complement analysis. Experiments are essential for developing models that may make future experiments unnecessary. We convert our experimental observations into theories. Experimentation fills the gap between theory and reality. "Then a miracle occurs" is replaced by "then an experiment occurs". A very large design space benefits from exploration of point solutions – guy with a machete in unchartered territory. Experimentation for research purposes and experimentation for deployment purposes can be indistinguishable. That's a great thing – innovation at the speed of grad student or -entrepreneur compilation. We will ever reach a point where we can go from analysis to real system, working just as anticipated? Maybe. Should we try to do this? Sure. What are examples? Electrical circuit design, from VHDL to silicon? Complex systems always involve unchartered territory. Need to be able to explore easily to jump start and understand limits of models. Pattern – new phenomenon, experiments, formalisms, more experiments, … Build-learn-build-learn-build-deploy. Supporting innovation is the same as supporting experimentation (true??).