Amin Vahdat Biography

Amin Vahdat joined the faculty of the the Computer Science department at Duke University after receiving a PhD from the University of California, Berkeley in 1998. His research focuses on system support for highly available and high performance network services. As a graduate student he participated in the design and implementation of the UC Berkeley Network of Workstations project. Some of this work was later commercialized as part of the Inktomi search engine.

For his thesis work, Vahdat conceived, designed, and led the implementation of the WebOS project. WebOS laid the groundwork for a set of common abstractions to support the operation of large-scale network services. Vahdat's work was among the first to demonstrate the benefits of dynamically replicating web services across the wide area in response to changing global system characteristics.

At Duke University, Vahdat leads the Internet Systems Software Group (ISSG). Current projects include: i) the design and evaluation of a continuous consistency model for replicated services that bridges the semantic, performance, and availability gap between traditional strong and optimistic consistency models, ii) flexible control of tradeoffs between cost and any (or all) of performance, availability, and energy in large-scale utility services, iii) the Slice high performance network storage system, and iv) self-organizing, energy-aware system architectures for mobile environments.

Vahdat is a winner of the prestigious National Science Foundation Faculty Development CAREER award and a member of ACM, IEEE, and USENIX.

Selected Recent Publications:

- "The Costs and Limits of Availability for Replicated Services," Haifeng Yu and Amin Vahdat. Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), October 2001.
- "Managing Energy and Server Resources in Hosting Centers," Jeffrey S. Chase, Darrell Anderson, Prachi Thakar, Amin Vahdat, and Ronald Doyle. Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), October 2001.
- "Design and Evaluation of a Continuous Consistency Model for Replicated Services," Haifeng Yu and Amin Vahdat, Proceedings of the Fourth Symposium on Operating Systems Design and Implementation (OSDI), October 2000.
- "Active Names: Flexible Location and Transport of Wide-Area Resources," Amin Vahdat, Michael Dahlin, Thomas Anderson, and Amit Aggarwal. Proceedings of the Second USENIX Symposium on Internet Technologies and Systems, October 1999.
- "WebOS: Operating System Services For Wide Area Applications," Amin Vahdat, Thomas Anderson, Michael Dahlin, Eshwar Belani, David Culler, Paul Eastham, and Chad Yoshikawa. Proceedings of the Seventh Symposium on High Performance Distributed Computing, July 1998.

Three Grand Challenges in Computer Systems Research

Amin Vahdat
Department of Computer Science, Duke University

The computer systems research community deserves tremendous credit for many advances over the past thirty to forty years: sustained exponential improvements in CPU performance, memory and disk capacity, and network bandwidth, just to name a few. Looking forward however, the grand challenges for the research community are not likely to focus on performance. In fact, one could argue that the community as a whole has done such a good job developing a process for improving performance that the best performance-oriented research is actually taking place in industry rather than academic research settings.

In this short position paper, I will outline three key challenges for systems research in the years ahead and, in each case, some small steps that my research group is taking to address these challenges.

Availability and Reliability

Moving forward, system availability and reliability is likely to eclipse performance as the key criteria for evaluating computer systems. While measuring and improving performance is well understood, we do not yet have a precise definition for system availability. Network service "availability" is currently measured as the amount of time between reboots of a web server, not accounting either for failures in the middle of the network that prevent client access or for poor performance effectively making the service unusable. Even the long distance telephone network, which has phenomenally suffered less than 2 hours of downtime in the past 40 years, does not account for failed calls during times of peak demand. Much fundamental work remains to be done to understand and improve service availability. As a small sampling, we must: i) develop system benchmarks that realistically measure different aspects of availability; ii) evaluate how different system architectures and protocols behave with respect to availability rather than performance; and iii) investigate fundamental techniques, such as replication and redundancy, to mask the failures inherent to billion-node networks such as the Internet.

We have taken some initial steps to address these emerging issues. First, we argue that availability must be measured at the granularity of individual requests. Thus, each request has a performance tolerance. If the service is unavailable to respond in the alloted period, it must be effectively classified as unavailable for that request. Note that in this setting, availability must be measured end to end, and not from the standpoint of some centralized server. Even if a machine (web server) has not suffered a hardware/software outage, failures in the middle of the network, high levels of demand, or congestion may render the service effectively unavailable for some subset of clients. Similarly, availability is rarely an "all or nothing" switch for complex modern network services that are typically comprised of many cooperating components. An individual failure may not render a service entirely unavailable but may degrade the quality of the information returned to the client. Thus, not only must service availability be measured from the perspective of some required performance distribution, but each reply must reflect its quality of the reply along some continuous spectrum defined in a service-specific manner.

Self-Organizing Distributed Architectures

Emerging computer systems at two extremes of geographic scale—wide-area replicated services and wireless sensor networks—face similar challenges. In both instances, systems must adapt to dynamically changing network and service characteristics in a decentralized and scalable manner; for instance systems adapt to comply with pre-defined service availability targets or to achieve a given level of sensor resolution/confidence in a particular geographic location. Thus, a key goal of *self-organization* is to maintain target levels of availability, security, data consistency, quality of service, and performance in the face of individual failures, attacks, wide variations in performance, etc. A prerequisite to self-organization is scalable tracking of global system characteristics based on local observations. Next, based on pair-wise exchanges with a subset of global hosts, local decisions must be made to approximate the global optimum. Such architectures must be robust to individual failure, cannot assume knowledge of global group membership, and must operate based on partial and inaccurate views of system state.

We observe that improving availability, security, data consistency, quality of service, etc. typically entails an increase in overall system *cost*. This cost may be measured in any number of ways, including real dollars, energy (particularly important in the case of wireless sensor networks), and consumed system resources such as bandwidth or CPU time. For a particular demand level and for particular network conditions, the distributed service must be aware of the tradeoffs between cost and some other evaluation metric. As global system characteristics change (failures occur, network congestion increases, battery life diminishes), the distributed service must determine whether additional cost should be incurred to maintain desired system characteristics, e.g., target levels of availability. As one instance of the much more general problem, we have designed a scalable (10,000 nodes or more) network overlay able to build low-cost, delay-constrained adaptive topologies based entirely on local observations without global locking. As network conditions change, the overlay adapts to maintain target cost and delay constraints. We are currently adapting this structure for achieving target levels of end-to-end reliability.

Energy Consumption

As computation becomes more deeply embedded in our everyday lives (i.e., moving toward the well-articulated vision of ubiquitous computing) devices operating on very limited energy budgets become increasingly common. The utility of these devices will be determined by overall system lifetime rather than system performance. Improvements in battery capacity have not kept pace with other aspects of system performance, making low-power design an important challenge for future systems research. While such research has traditionally focused on hardware design, there are significant opportunities for improvements based on hardware/software cooperation. To explore this space, we have built an operating system (a modified version of Linux) that treats energy as a first class system resource, alongside more traditional OS resources such as the CPU, memory, disk, and network. The idea is that the OS allocates energy among competing tasks, rather than the individual more traditional resources. Thus, each task must operate within a given energy budget and all (e.g., CPU and disk) hardware access are charged within the context of this higher-level budget. By allocating energy based on individual task priority, we are able to improve overall system utility and lifetime.

Note that energy is not just a concern for battery-powered devices. The 140-machine cluster used by our research group currently incurs a $30,000/year energy bill (including the cost of cooling). While all machines are fully utilized during times of peak demand, only a handful are in use at any given time. Large-scale hosting services are typically also built around a cluster model for incremental scalability and improved reliability. Such services are typically apportioned for peak, rather than average-case, levels of demand. Thus, we have built a smart load balancing infrastructure that allocates cluster resources to competing applications based on current levels of demand, powering down machines that are not required to deliver target levels of reliability and performance.