Association for Computing Machinery
Advancing Computing as a Science & Profession

Ubiquity, an ACM publication
November 2015

Ubiquity Symposium

The Internet of Things

Standards for Tomorrow

By Dejan Milojicic, Paul Nikolich, and Barry Leiba

Editor's Introduction

Over the decades, standards have been critical for defining how to interconnect computer and networking devices across different vendors so they can seamlessly work together. Standards have been critical, not only in networking and computer interfaces, but also at the operating system and systems software level. There are many examples, such as IEEE 802, POSIX, IETF, and W3C. There was always the question of the right time to standardize (not too early and not too late), and the time to complete a standardization project always seemed too long, but inevitable. However, the contemporary industry seems to be more dynamic and evolving than it has ever been, demanding more agile processes. Open source processes and software defined (networks, storage, data centers, etc.) offer alternatives to standards. In this article we attempt to envision the future role of standards, and how they will complement and enhance alternative choices toward the same goal. We first summarize traditional standards, then discuss alternatives and a couple of use cases, and conclude with some future directions and opportunities for standardization.

Ubiquity Symposium

The Internet of Things

Standards for Tomorrow

by Dejan Milojicic, Paul Nikolich, and Barry Leiba

Traditionally, an IEEE standard project is started when there is interest by enough people or entities to invest time, effort, and intellectual property to meet a specific market need. A working group is formed to develop a draft document. Document development is accomplished in face-to-face and electronic meetings and by "letter ballots." Everyone is given a fair chance to contribute. Owners of intellectual property that may become part of the standard must agree to make it available on a reasonable and non-discriminatory basis. When the working group decides the draft is sufficiently mature, it is then circulated for sponsor ballot and comment among any and all interested parties (i.e., not just the working group; this is known as the sponsor ballot group). It must achieve 75 percent approval and the working group must respond to all comments. The revised draft and comment responses are then recirculated to the sponsor ballot group. If there are comments, the working group must respond to them and recirculate the (possibly) revised draft and comment responses. This continues until there are no new comments and the 75 percent approval threshold is still met. When those conditions are met, the document is sent to the IEEE Standards Board for approval and ratification. This process has worked well in terms of ensuring a wide spectrum of interested parties can participate in the development of the document, as long as there is broad consensus. If broad consensus is not imminent, the process can take a very long time, occasionally resulting in a project being withdrawn.

Once a standard is ratified, manufacturers, service providers, and organizations may voluntarily design equipment, services, or processes to comply with the standards. Standard specifications alone are not enough to ensure useful implementations; the industry also must define and agree on test suites and engage in multi-vendor compliance/interoperability testing. This has been critical to the success of such standards as 802.3 Ethernet and 802.11 Wi-Fi.

Traditional Standards

Standards specify interfaces precisely and thoroughly enough to ensure multi-vendor interoperability, but only enough to allow vendor-dependent innovation. The core competitive approach to monetizing standards work involves a vendor getting their IP designed in to the standard to generate royalty income.

Networking. IEEE 802 and Internet Engineering Task Force (IETF) standards are used to build a large complex network of networks in a standardized, scalable way. The networking community uses 802 and IETF standards as building blocks for the interfaces and routing functions of network elements. Networks are built and managed using these network elements, whose core functions comply with their industry standards and are enhanced by individual suppliers' "added value" capabilities.

The current network infrastructure has proven to be remarkably adaptable, growing in size, complexity, and capability over the years. As a result of the growth into application areas that were not anticipated 30 years ago, certain requirements have emerged as high priorities that we did not think we needed in the 1980s: Requirements such as privacy, security, time criticality (bounded latency, accuracy, etc.), guaranteed message delivery, and an unbounded number of endpoints. These new requirements need to be addressed by amending the standards or creating new ones.

The standards community is responding to these emerging requirements quickly. For example, in the IEEE 802 LAN/MAN Standards organization many new projects/activities have recently been completed or started or are under active investigation. Examples include the 802.1 higher layer LAN protocol work on security and time sensitive networks; the 802.3 Ethernet work on interfaces for physical layers for intra-vehicular communications and power delivery; the 802.11 WLAN work on interfaces for wireless access in vehicular environments in addition to its typical pervasive applications; and the 802.15 WPAN work on interfaces for smart utility networks, medical devices, smart phone camera/optics, and mesh networks. 802 established the 802.24 Technical Advisory Group to coordinate 802 activities and define the emerging requirements for smart grid and Internet of Things (IoT) applications that are not met by existing 802 standards. Finally, 802 recently chartered an executive committee study group on privacy recommendations to identify requirements and suggest improvements to existing 802 protocols that may enhance privacy.

802 consists of more than 700 individual members, and operates in an open and transparent manner. This has enabled a wide range of intellectual property to be incorporated into 802-compliant products that are relatively low cost, high performance, and highly innovative. Fair, open, and transparent operation has been a key reason 802-standard-compliant interfaces and functions are pervasive in network equipment. Hundreds of individual experts worldwide, from large and small companies, continually bring their knowledge of the emerging needs of the networking marketplace, potential solutions, and their technical feasibility to 802. From this cauldron of ideas consensus emerges, projects are started, and standards ratified. Some are wildly successful in the market place, some die on the vine, but they are all produced with very high quality as a result of broad, exhaustive peer review.

Consisting of mostly fixed function dedicated hardware components, network infrastructure tends to evolve at a slower pace than software applications (although this is changing with software defined networking). This is good, because infrastructure, above all, must be highly reliable and scalable. That reliability and scalability is achievable because the underlying technology reaches a level of maturity that allows the entire spectrum of service infrastructure—component suppliers, equipment manufacturers, and network service providers—to understand how to build and deploy cost-effective solutions into the market.

Software applications, on the other hand, evolve at an entirely different pace, but benefit from standards in the same way. Standardized components allow complex functions to quickly be created in a reliable manner.

Operating system standards. In the ‘80s and ‘90s, we had quite a variety of operating systems, including different flavors of UNIX-like operating systems. This resulted in a strong need for standardization. One of the key operating systems standards is POSIX (Portable Operating System Interface) [1]. In a series of specifications that were developed more than 10 years, the POSIX standard defines the UNIX-like operating system core services (processes, signals, exceptions, timers, files, pipes, C libraries, etc.); real-time extensions; thread extensions, shells/utilities; and finally a more comprehensive specification for base definitions, interfaces, and commands and utilities. The latest document has close to 4,000 pages.

Despite some controversies (block sizes and some of the behavior defined), a number of operating systems were POSIX compliant in full (AIX, HP-UX, Solaris, Tru64, etc.) or in part (FreeBSD, Linux, etc.). In addition, Cygwin is an open source UNIX-like environment that runs on Windows operating systems and is POSIX compliant.

POSIX also defines conformance of applications to execute on the POSIX-compliant operating systems. POSIX defines strictly conforming applications, conforming applications, and conforming applications using extensions. The key goal is to guarantee portability of applications across the POSIX-compliant operating systems.

In practice, today POSIX is used as a standard against which operating systems are compared: whether they are compliant, non-compliant, or semi-compliant. However, it is also a somewhat aged standard. A lot of operating systems research and development has taken place since, and technology continues to advance. For example modern memory sizes warrant larger granularity of page sizes, NUMA (non uniform memory access) poses new opportunities and challenges for operating systems in terms of performance and scalability, which is not addressed by the standard, and new synchronization and communication primitives have emerged that are better suited for contemporary and next-generation hardware.

Middleware and application standards. Standards are critical for application-layer protocols as well. Organizations such as the World Wide Web Consortium (W3C), and the applications area, real-time applications, and infrastructure area in the IETF focus on those protocols in client-server and middlebox/proxy environments. Relevant applications (and associated protocols) include directory services (LDAP); email (SMTP, POP, IMAP); session initiation protocol (SIP), which is an enabler for voice over IP and other streaming applications; instant messaging (XMPP and SIMPLE); and various web services and applications (HTML, HTTP, XML, CSS, and many other web-related protocols and formats). The W3C's new HTML5 standard opens the door to many new and more flexible web-based applications. WebRTC—a joint project between the IETF and the W3C—provides a browser platform for streaming and real-time web applications, such as video calling, that do not require customized, proprietary software plug-ins.

Recent revelations about widespread "pervasive monitoring" of network traffic has raised awareness of security and privacy issues; the IETF, the W3C, and other organizations are putting an increased emphasis on these issues and on protocol-level mitigations and increased use of encryption to protect the privacy and integrity of users' information. See RFC 7258 / BCP 188, "Pervasive Monitoring Is an Attack" for background on this subject.

Alternatives to the Traditional Standards Development Process

Open-source processes. Open source represents a new way of generating product level software [2]. This is demonstrated through examples of the Linux operating system, virtualization tools such as Xen and Docker, Apache Web server, Mozilla Firefox browser, MySQL database, Joomla and Drupal content management systems, and many others.

Because of the large number of open source contributors these software packages can reach higher stability and robustness than industry-produced software. More importantly, and for the same reason, they also drive innovation.

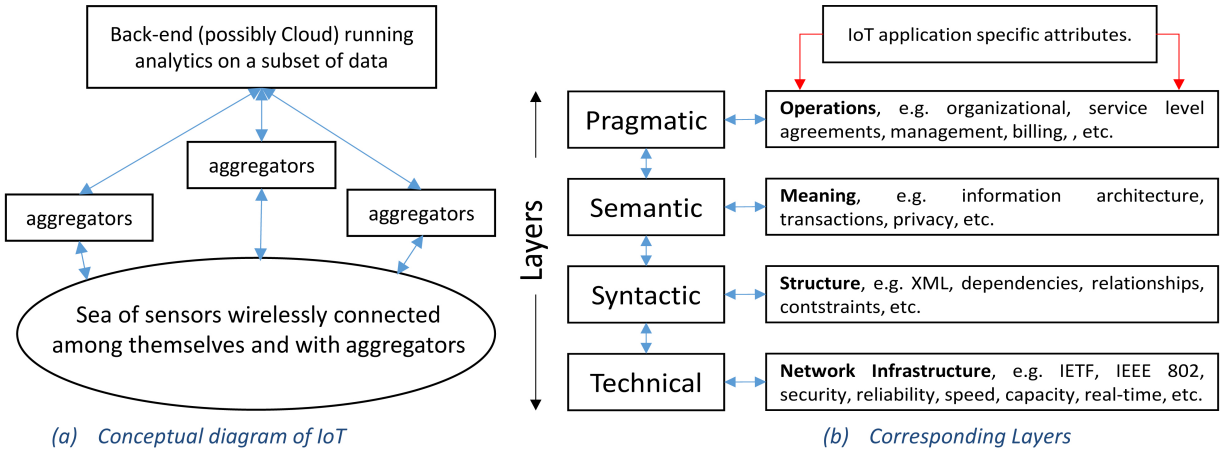
Typically the engineers doing the work create open source specifications. Running code is most important. Those who produce more running code get more decision making power. Not all open source is created in an equal way: there are community-driven models (Linux) as well as vendor driven ones (OpenStack).

Open source is an alternative to standards because the outcome is a complete package with executables and the source code. The software components are intentionally designed and implemented to be compatible with other software components. Where standards provide a mechanism for different implementations to interoperate, open-source implementations use collaborative development to build a single, open implementation.

Software defined. Software defined (networking, storage, data center, etc.) is a recent trend, which began with OpenStack's software defined compute and OpenFlow in the networking area. Networking has complex software stacks that were implemented in sophisticated high-end switches. Software defined networking (SDN) enables implementing these networking stacks on the standardized computing architectures avoiding high margins for networking equipment, and also offering other benefits such as quicker evolution, reconfiguration, and adaptation to varying needs.

The SDN approach has received interest from other areas—such as software defined storage and computing—where the storage solutions, such as storage area networks (SAN), can be delivered in a software-defined fashion that gives benefits similar to SDN. The same is true for data centers that also benefit from the ability to reconfigure data center and enable flexibility to the administrators and ultimately reduces cost to users.

SDNs are undergoing both classic standardization (Open Networking Foundation, ONF) and vendor-driven open source (Open vSwitch, Open Daylight). ONF is consensus standard, both Open vSwitch and Open Daylight are de-facto standards. Most software defined seems to push de-facto standards, i.e. standardize within a project level first and try to reach consensus with the rest of the ecosystem later.



(a) Conceptual diagram of IoT *(b) Corresponding Layers*

Figure 1. The "layering" concept: Complex systems require layers plus interoperability.

Two trends have highly helped towards software defined: virtualization and DevOps. Virtualization as an approach and set of principles helps abstract away the components and enables easier reconfiguration. The DevOps trend enables continuous beta deployment; by alternating development and deployment (and other operation aspects) it forces a quicker cycle from conception to design, implementation, deployment, and improvement.

Open Compute is an example of attempting to open up hardware specifications. Open Compute defines specs for server design, networking, storage, systems (hardware) management, power, and chassis.

Future Use Cases

Internet of Things (IoT). For the purpose of this article, IoT simply means the number of connected endpoints will increase by orders of magnitude over the current numbers of endpoints and they will consist of semi-autonomous devices, not people. Sensors and

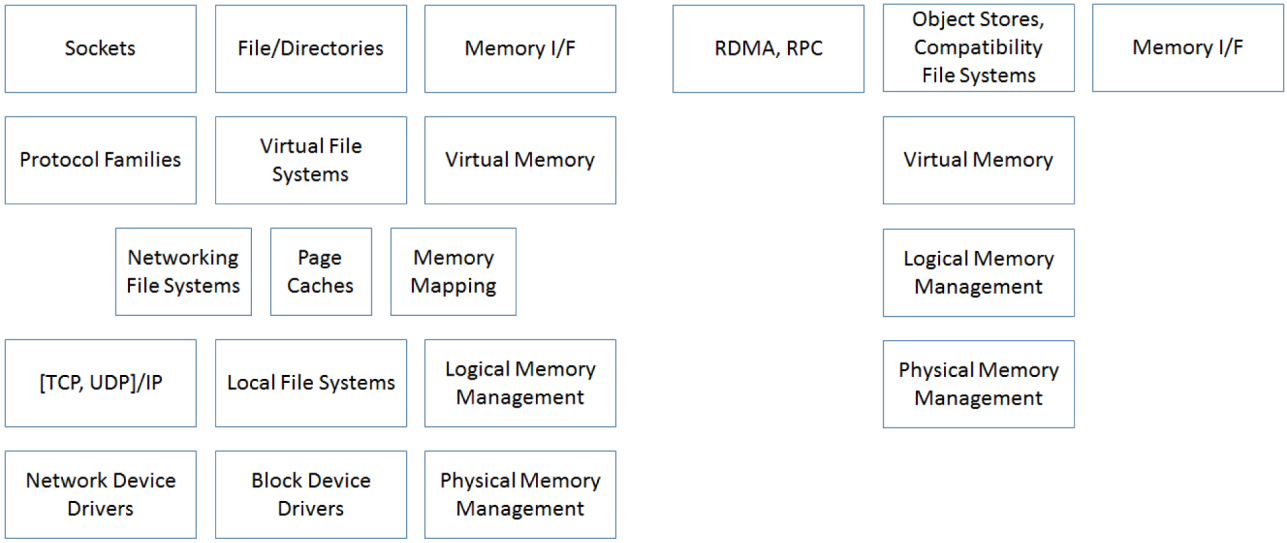
actuators, which act autonomously in a fully distributed manner or under centralized, automated control, will dominate these endpoints.

One method of organizing a complex system-of-systems like IoT is to develop a layer model as described in Figure 1. The layers are broadly divided into the technical, syntactic, semantic and pragmatic categories, each serving a particular purpose. This end-to-end solution and vertical stack will require some new form of standardization processes or an agreement across many vendors working on sensors, aggregators, and back-end analytics.

Next-generation operating systems. The next-generation operating systems will be influenced by new hardware features, such as nonvolatile memory, high-speed low-latency photonic interconnects, and heterogeneous multicores. These hardware features will flatten the operating system stack, eliminating the layers such as block device drivers and page caches for file systems (see Figure 2). The whole notion of serialization will go away and be replaced with byte-addressable objects.

Given these substantial changes, some form of standardization of new operating systems will have to take place. The need for a comprehensive but lengthy POSIX-like standard went away because of the consolidation of the operating systems space. In addition, the open-source communities are able to define, develop, and innovate operating systems and then maintain them much better than any single company or consortium can—no matter how successful or large it is.

Also the need for the application binary interoperability (ABI) write-once-run-anywhere, which seemed critical in the 1990s, went away. Some of these requirements went up the stack, in the programming languages (Java), Web interfaces, or new platforms (Hadoop). Similarly, the operating systems are being hidden by the new user interfaces, such as in case of Linux in Android.



(a) Traditional UNIX or Linux-like OS stack

(b) Flattened stack of next generation Operating Systems for data center clusters

Figure 2. Flattening and unification of the software stack inside data center systems as a result of the introduction of Non- Volatile Memory and Low-Latency Photonic Interconnects. On the left is the traditional Linux stack. Byte-addressable non-volatile memory will obviate the need for file systems and low-latency photonic interconnects will obviate the need for the complex protocols originally devised for the Internet. As a result, the number of interfaces and the need to standardize them will change. Each component on the left required a degree of standardization, which adds to the scope of standard and time to standardize.

Security design flaws and building codes. The center for secure design has brought up two intriguing approaches toward alternatives or complements to standardization [3]. In the first approach, a team of security practitioners from industry determined the top 10 software security design flaws, leading to principles of how to better design secure systems. It took one meeting and one document to reach some consensus around what these key design flaws are. Now other artifacts are being considered out of these design flaws. In a related effort, Carl Landwehr proposes the approach of civil engineering toward building codes to be used for more secure programming codes [4]. Both efforts prescribe, not precisely, but loosely, how to design and implement secure systems.

Summary

We have discussed how the standardization processes have worked in the past and how they may change in the future. We have provided some examples for existing standards and brought up how some new standards may evolve. We have also compared the past and future.

Today's standards development process requires a significant amount of face-to-face, real-time interchange among the working group members to build consensus. We see this becoming less important as alternative, non-real-time, non-face-to-face means of building consensus is made possible by novel social-network type interactions that are well documented and allow a much larger group of interested parties to participate. The challenge in this environment will be converging on consensus. Perhaps this will yield a "trial-and-error" approach where a wide variety of technical approaches can be prototyped, tested, and evaluated, with the approach that produces the best result being selected for incorporation into the standard. Furthermore, the standard will be subject to revision as new techniques surface to improve the basic functions defined by the standard.

This trial-and-error approach could work well for software, and as more and more of the network infrastructure becomes software defined, it is possible the network infrastructures built using these techniques will become highly adaptable, with a few basic architectures emerging that meet the basic needs of IoT systems—scalable, reliable, low cost, secure, private, etc.

Furthermore, some of the possible ways of producing documents quickly are to select and mentor a group of leaders who are skilled at getting volunteers to work efficiently and on a schedule, and who can maintain group momentum—this all boils down to motivation. If we can learn how to build highly motivated groups of volunteers this will produce big benefits. Focus is important, so the whole group knows exactly what its objective is. Additional helpful elements include reducing handling delays at each step in the production and optimizing the process to reduce steps, yet maintain openness and transparency. There needs to be broad consensus early in the process. Finally, if a project is languishing, the threat of termination should be real—sometimes that is the best motivator.

Our predictions are summarized in the table below and are self-explanatory. We believe standardization will continue to be an important process for industry and customers, but it will

have to evolve and modernize and keep up with the advances of and uses of modern technology.

Table 1. Comparison of Past and Future Standards.

Standardization Characteristics	Time Frame	
	Today	Future
Time-to-standardize	Long (a few years)	Much less (a few months)
Quality of standards	High	Sufficient
Transparency	High	Built into process
Reaching consensus	Complex, timely, and hard	Crowdsourced, power of numbers
Process	Writing documents, F2F meetings	Documents, working artifacts, models
Adoption	Usually required because of market	Preferable
Conformance	Strong, usually using testing	Relaxed

We are not trying to advocate open source versus standards, but rather emphasize the benefits and shortcomings of either and try to paint the picture of where the standards of future are going. One ultimate benefit of both approaches is in gathering expertise of people in good APIs or protocol design. Implementation itself could be very powerful, but also dangerous from the monoculture perspective. Heterogeneous and alternative implementations are a better safeguard against viruses and denial of service attacks.

Both standardization and open source will continue to be present in industry, but both will have to undergo changes toward becoming more agile, timely, and in different formats. There is obvious difference between the hardware and software is the former being much harder to change and latter easier. At the same time software defined is trying to narrow the gap.

About the Authors

Dejan Milojicic is a senior researcher and manager at HP Labs, Palo Alto, where he is responsible for systems software and systems management. He worked on standardization of software management within GGF and OMG. He is an IEEE Fellow and ACM Distinguished Engineer.

Paul Nikolich is the 802 LAN/MAN Standards Committee chairman and an IEEE Fellow. He's a self-employed consultant and investor in the communications technology fields. Nikolich has an M.S. from the Polytechnic Institute of New York.

Barry Leiba is a senior standards manager at Huawei Technologies, and has been working on standards in the IETF for more than 15 years. He served on the Internet Architecture Board from 2007 to 2009, and has been serving as the applications area director on the Internet Engineering Steering Group since 2012. Leiba is an IEEE senior member, and is associate editor in Chief of *IEEE Internet Computing* magazine.

DOI: 10.1145/2822533

References

[1] ISO/IEC/IEEE 9945 Information technology — Portable Operating System Interface (POSIX®) Base Specifications, Issue 7, Reference number ISO/IEC/IEEE 9945:2009(E)

[2] IEEE Computer Society 2022 Report. Section 3.2 The Open Intellectual Property Movement.

[3] IEEE Center for Secure Design. Avoiding the Top 10 Security Flaws.

[4] Landwehr, C. A building code for building code: putting what we know works to work. In *ACSAC '13 Proceedings of the 29th Annual Computer Security Applications*.