

# Elastic Infrastructure to Support Computing Clouds for Large-scale Cyber-Physical Systems

Douglas C. Schmidt  
EECS Department  
Vanderbilt University  
Nashville, TN

Email: [d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)

Jules White  
EECS Department  
Vanderbilt University  
Nashville, TN

Email: [d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)

Christopher D. Gill  
CSE Department  
Washington University  
St. Louis, MO

Email: [cdgill@cse.wustl.edu](mailto:cdgill@cse.wustl.edu)

**Abstract**—Large-scale cyber-physical systems (CPS) in mission-critical areas such as transportation, health care, energy, agriculture, defense, homeland security, and manufacturing, are becoming increasingly interconnected and inter-dependent. These types of CPS are unique in their need to combine rigorous control over timing and physical properties, as well as functional ones, while operating dynamically, reliably and affordably over significant scales of distribution, resource consumption, and utilization. As large-scale CPS continue to evolve and grow in scale and complexity they will impose significant and novel requirements for a new kind of cloud computing that is not supported by conventional technologies

To meet these requirements, cloud computing advances are needed to establish real-time computing, communication, and control foundations rigorously at scale. Likewise, advances are needed to apply these foundations in a flexible and scalable manner to different real-world large-scale CPS challenge problems. To support both foundational and experimental R&D, a new generation of elastic infrastructure must be designed, developed, and evaluated. This paper identifies challenges, opportunities, and benefits for this work and for the large-scale CPS it targets.

**Keywords**—Computing Clouds for Cyber-Physical Systems; Data Distribution Service; Middleware

## I. INTRODUCTION

Large-scale cyber-physical systems (CPS) are increasingly composed of applications and services deployed across a range of communication topologies, computing platforms, and sensing and actuation devices. Examples of these types of CPS include advanced air traffic management [1], supervisory control and data acquisition (SCADA) systems [2], next-generation smart power grids [3], remote health care delivery systems [4], integrated air and missile defense systems [?], and electronic trading systems [5]. The services and applications in large-scale CPS often form parts of multiple end-to-end cyber-physical flows that operate in mission- or safety-critical resource-constrained environments.

Each service within the end-to-end cyber-physical flows of large-scale CPS must process events belonging to other services or applications, while providing dependable quality-of-service (QoS) assurance (e.g., timeliness, reliability, and trustworthiness) within limited resource constraints or with

the ability to fail over to providers of last resort (e.g., a public utility in the case of a SCADA system or smart power grid).

Large-scale CPS have traditionally been designed and implemented using resources procured and maintained in-house. Significant fiscal and technological constraints, however, are motivating researchers and practitioners to consider alternatives that can still ensure mission- and safety-critical properties. In particular, the emergence of dependable- and increasingly commodity-computing clouds motivates design and operational considerations for large-scale CPS that include:

- **offering economic incentives**, e.g., pay-as-you-go and pay-as-you-grow models that emphasize computing, communication, and control as operating expenditures rather than capital expenditures;
- **consolidating and sharing hardware and software components** through multi-tenancy to reduce operating expenses, e.g., lower power consumption and hardware budget;
- **aggregating and disaggregating behaviors dynamically** to reduce risk, e.g., by minimizing contention and avoiding single points of failure; and
- **elastically auto-scaling computing, communication, and sensing/actuation resources** for real-time systems to ensure that shared system resources are used effectively and dependably without incurring unnecessary costs when resources are idle.

Despite the promise held by commodity cloud computing, however, supporting the timing and dependability requirements of large-scale CPS *at scale* is hard. This paper discusses a number of technical issues emerging in this context, including (1) precise auto-scaling of resources within local and system-wide constraints; (2) flexible optimization algorithms to balance real-time constraints with cost, scalability, utilization, and other (often conflicting) goals; (3) improved fault-tolerance fail-over to support real-time requirements; and (4) data provisioning, load balancing, and analysis algorithms that rely on- and may be used to optimize-physical properties of computations. This paper also explores key technical building blocks needed to create

a dependable and elastic infrastructure for large-scale CPS.

## II. THE EVOLUTION OF CPS IN SCALE AND COMPLEXITY

This section summarizes the evolution of CPS in terms of scale and complexity in terms of *QoS fidelity*, which ranges from low fidelity (e.g., "best effort" QoS) to high fidelity (e.g., stringent requirements on timeliness and dependability), *Degree of asset sharing*, which ranges from a low degree of sharing (e.g., each application or service is allocated a unique set of assets) to a high degree of sharing (e.g., assets are pooled amongst many applications and services), and *System scale*, which ranges from small scale (e.g., a dozen or so system components) to large scale (e.g., many thousands of system components).

### A. Overview of CPS

A cyber-physical system (CPS) is an integrated set of hardware and software that controls physical things (and which may or may not involve humans in the loop). CPS have historically involved a tight coupling and coordination between a system's computational elements (components implemented entirely in hardware or software) and physical elements (components that interact with the physical world). Traditional examples of CPS include anti-lock braking systems in automobiles [6] and automated pilot features in aircraft. These types of CPS typically exhibit high QoS fidelity, a very low degree of asset sharing, and a small number of system components.

Many CPS also have been used to control devices and/or processes in environments disconnected from networks. Although these types of stand-alone CPS are common, the next-generation of CPS [7] increasingly will use local area network (LAN) and/or wide area network (WAN) processing elements to control devices and interactions. These interactions may be influenced by physical environments (such as wind farms or hydro-electric power generators) or industrial environments (such as in chemical plants). More sophisticated emerging CPS (such as driverless cars and smart power grids) are also adaptive and intelligent, often solving problems as they occur in real time without direct human input.

Regardless of their scale and connectivity, CPS are time-sensitive since the right information or action delivered or performed too late results in an incorrect outcome. The QoS of a CPS thus has both a reliability dimension *and* a temporal one. In particular, system functionality must run in a timely manner.

Large-scale CPS must address requirements and challenges that aren't as relevant for traditional stand-alone CPS, including partial failure, higher latency and jitter due to shared communication links, and denial of service attacks. Security is an increasingly important QoS concern in CPS [8] since delivering information in a timely manner

is itself essential, but may be irrelevant if the information has been tampered with or compromised. Today's large-scale CPS typically exhibit higher QoS fidelity, a higher degree of asset sharing, and a larger number of system components than traditional CPS.

### B. Overview of Cloud Computing

Large-scale CPS have been developed in the past, primarily in the aerospace, defense, and power domains. These types of CPS, however, have been proprietary and expensive to develop and sustain. In recent years, therefore, the enormous commercial and government investment in commodity cloud computing environments has spurred an interest in leveraging these technologies as the basis for large-scale CPS.

Cloud computing provides applications with ubiquitous, convenient, and on-demand access to a shared pool of configurable computing resources across a network. The goal of this paradigm is to treat computing and communication as *utilities*. In particular, these capabilities are provided to applications as services, i.e., enabling the migration and scaling up/down of system computing, storage, and communication resources without requiring explicit involvement from applications.

The key characteristics of cloud computing environments typically include the following capabilities:

- *On-demand self-service provisioning*, which enables end-users of clouds to unilaterally provision computing capabilities, including networks, storage, and servers, which are often virtualized by generalizing the physical infrastructure and making it available as a set of managed components that are easier to use and control automatically.
- *Elastic resource pooling and multi-tenant models* where multiple applications run in the context of shared server and networking resources. Achieving these elastic capabilities requires a means to automatically and rapidly expand and contract the supply of computing and storage based on dynamically fluctuating levels of demand without adversely impacting essential QoS properties.
- *Managed operations* in which resource utilization can be controlled via some type of metering capability. These managed operations essentially "outsource" key hardware and software components and activities to third-party providers.

Most applications of commodity cloud computing environments focus on web hosting, where low cost (e.g., via resource sharing) and high availability (e.g., via replication) are critical QoS attributes. A key benefit of cloud computing in this domain lies in the economies of scale provided by multi-tenancy and elasticity, which involve the ability to have multiple applications and services sharing the same computing infrastructure, as well as the potential to expand and contract infrastructure as needed and on-demand. These types of cloud computing environments typically exhibit low

QoS fidelity, a high degree of asset sharing, and a large number of system components.

Although cloud computing is increasingly being adopted by individual consumers and by companies in certain industries, many classic implementations of cloud computing are at odds with CPS requirements, such as bounding latency and jitter, and avoiding priority inversions. In particular, unless managed carefully with respect to timing (e.g. as in ) and other criteria, virtualization may become detrimental in CPS due to higher overhead and jitter, as well as (hidden) scheduling issues. What is needed, therefore, are software and hardware infrastructures that can support the needs of next-generation large-scale CPS. These new large-scale CPS require high QoS fidelity and a high degree of asset sharing, and must support a large number of system components.

### III. THE EVOLUTION OF DESIGN AND OPERATIONAL PARADIGMS FOR CPS

During the past 40 years many paradigms relevant to large-scale CPS have come and gone. This section summarizes the evolution in the paradigms used to design and operate CPS at various levels of scale and complexity during this time.

#### A. Early Paradigms

In the 1970s and 1980s, there was a tendency to build CPS via a tightly-coupled design paradigm, where most elements of these CPS were proprietary and controlled or built by a single system integrator. These systems were designed in isolation with little reuse or sharing. Likewise, they were non-adaptive, e.g., if changes were made to requirements or the runtime environment many other parts of the systems could be adversely affected.

In general, a key limitation of such a tightly-coupled design paradigm for CPS was that small changes made to the software or hardware could affect the correctness of almost any other part of the system . Examples of these problematic changes include adjustments to requirements, implementation, infrastructure, operating systems, programming languages, middleware, and networks. As a result, these large-scale CPS were expensive to sustain and evolve, in addition to incurring vulnerabilities due to not being designed to connect to publically accessible networks, such as the Internet.

This tightly-coupled design paradigm also was problematic due to the ways in which developers and operators traditionally provisioned, scheduled, and certified CPS. The operational capabilities and characteristics of traditional CPS were typified by the need to obtain *all* the required resources. If such a provisioning process goes smoothly, traditional CPS usually work well. If not all of the resources are acquired, however, there could be major issues and a CPS simply might not work as needed.

The tight-coupling exhibited by such CPS was exacerbated by their stringent end-to-end QoS requirements, including bounded latency and absence of priority inversion. To meet these requirements, developers of traditional CPS typically hard-coded many implementation details, shared limited information between system components, and allocated resources statically. While this strategy works for small CPS in closed stand-alone environments, it simply doesn't scale up to meet the needs of large-scale CPS being developed and planned (e.g., based on emerging proposed industry standardization efforts, such as the Industrial Internet [9]). Moreover, it is not feasible to leverage commodity computing clouds as the basis for these types of CPS due to their reliance on statically provisioning and aversion to sharing.

#### B. Recent R&D Progress

Over the past decade, there have been tremendous advances in research and development for CPS, as well as evolution in the adoption and application of newer design paradigms. For example, cutting-edge CPS in both military and civilian domains are more layered and componentized than those of previous decades. In particular, modern large-scale CPS include layers of network, operating system, middleware, and programming language standardization and have become more robust at the infrastructure level. Moreover, advances in loosely coupled CPS software and system architecture have improved, so that when problems arise, properly programmed systems are able to cope through on-line adaptation.

A further benefit of these modern, less tightly coupled large-scale CPS is that solutions are potentially more cost-effective to evolve and retarget. Developers are less apt to have to backtrack and recertify an entire CPS when minor changes are made, which is a key cost-driver for sustainability in legacy CPS. Consequently, changes can be made to a CPS environment, requirements, and aspects of implementation, including those that are hidden behind component or module boundaries.

Modern large-scale CPS have also improved from an operational point of view. The majority of new loosely coupled large-scale CPS are being constructed via data-centric and reusable protocols. Event and messaging buses are more resilient in these types of large-scale CPS. When constructed properly, these large-scale CPS are designed to work appropriately even if they don't receive all resources in a timely manner, which enables dynamic allocation and management. There is the added benefit of better sharing support for resources, especially in environments with the ability to describe priorities and importance of information flow at multiple levels.

Some of the operating platforms that have evolved to support modern large-scale CPS have much in common with computing clouds. For example, the total ship com-

puting environment developed for the US Navy's DDG-1000 destroyer include advances in distributed resource management based on many of the technologies mentioned throughout this paper (and discussed further in [10]). While the scale of a DDG-1000 destroyer is not nearly as large as envisioned large-scale CPS (e.g. based on a continent-wide Industrial Internet), it serves as a good example of how metropolitan area network (MAN)-sized large-scale CPS can be developed reliably and securely.

#### IV. R&D TRENDS AND CHALLENGES FOR LARGE-SCALE CPS

Current trends and challenges within the domain of large-scale CPS are a hot topic of discussion. For example, the US National Science Foundation (NSF) recently convened stakeholders from academia, industry, and government at a workshop on research and implementation challenges at the intersection of Cloud Computing and CPS [11], from which a community report is currently being drafted. Topics discussed during this workshop included (1) the role of computing clouds in data collection, integration, analysis, and mining for CPS, (2) the roles of computing clouds in CPS control systems, (3) stability, safety, security, privacy, and reliability considerations in integrating cloud computing with CPS, and (4) programming models and paradigms for computing clouds that support CPS. When considering what is happening in this space now, it is useful to be familiar with approaches used by developers in the past and the insight those experiences provided when envisioning future directions.

##### A. *The Benefits and Limits of Elastic Hardware*

The CPS space is diverse and complicated, but it is reasonable to expect that some of the key answers can be found in research conducted on elastic hardware platforms in cloud computing environments. Elastic hardware refers to platforms with the ability to add or remove CPU capacity within a reasonable time frame and price. This technology enables cloud providers to add or subtract hardware without the need to change underlying logic or configurations of the software. Since programmers' time has become a precious commodity the flexibility enabled by elastic hardware is tremendously valuable.

One complication of elastic hardware is that most platforms have been utilized for hosting web applications in public cloud environments or data-centers. Although those environments have been relatively reliable for conventional web hosting services, they pale in comparison to the complexities and mission-criticalities of Industrial-Internet-style applications, where support for secure, real-time communications and failover are essential.

Elastic hardware is thus necessary, but not sufficient for building elastic applications that possess cyber-physical properties. There are a number of reasons why programming

elastic hardware for CPS is hard. The first is due to the fact that many programming models used by developers are inadequate. Developers tend to use complicated or obtrusive APIs, which are challenging to program. Conversely, there are solutions that are simple to program, but tend to have problems with respect to scalability and predictability. These solutions work well if timeliness is not a concern, but they are not a viable solution when timeliness is paramount.

Another issue is the general lack of understanding for real-time, concurrent network solutions. There are many inherent and accidental complexities in this area, including race conditions, deadlocks, priority inversions, and missed deadlines. The CPS development community needs to become more familiar with these issues so they can work more effectively to address them with the available tools.

Some operating platforms provide good support for multi-core solutions, but do not have sufficient support to transition seamlessly from multicore to distributed core. When this is the case, the system will work well up to 16 cores, (i.e., the current scale supported by high-end Intel or AMD multicore chip sets) and then start to degrade significantly when the system scales beyond that.

Finally, there is the long observed issue of inadequate support for QoS at scale. In this context, QoS refers to the ability to control systematic quality attributes (sometimes referred to as "para-functional properties"), including prioritization, failover and robustness, and system-wide resources in an end-to-end environment over various types of networking infrastructure. Approaches that work well for conventional web-based systems often do not work as well in the mission-critical CPS domain.

The impediments to programming elastic applications on elastic hardware described above affect the majority of computing systems, though they are particularly problematic for large-scale CPS. As a result, organizations may believe that since the traditional Internet works well for ecommerce or file sharing, it should work just as well for more complex large-scale CPS, until they ultimately discover that is not the case.

##### B. *Key Challenges for Elastic Large-scale CPS*

Large-scale CPS are increasingly used to connect people, data, and machines to enable access and control of mechanical devices in unprecedented ways. These types of CPS often integrate sophisticated machines embedded with sensors and sophisticated software to other machines (and end users) to extract data, make sense of it, and find meaning where it did not exist before. The overarching theme is that such machines-ranging from jet engines to gas turbines to medical scanners-connected via large-scale CPS have the analytical intelligence to self-diagnose and self-correct, so they can deliver the right information to the right people dependably at the right time.

Despite the promise of large-scale CPS, however, supporting the end-to-end QoS requirements is fundamentally hard and requires new advances in a number of key areas, including those discussed below.

1. *Precise auto scaling of resources* with an end-to-end focus needs to be a feature of CPS. Auto scaling is often thought about as adding cores when demand rises. Although this is certainly useful, it comes with the downside of not working properly from a system-wide perspective. Enabling technologies for large-scale CPS (such as the Industrial Internet [9]) require ways to scale up scheduling and auto scaling in a broad environment, to support precise behavior for end-to-end task chains. Stability and safety properties within mission-critical large-scale CPS require complex analysis to provide confidence that they will work as expected. Supporting this need calls for analysis examining reachability of states in a system, which is currently a particularly challenging part of the research space.

2. *Optimization algorithms that balance real-time constraints with cost* and other goals must be used. Often these problems can be solved by additional hardware, but not all developers have those resources available to them. Although deployment and configuration algorithms—along with services and infrastructure—are key to successful large-scale CPS, implementing these algorithms effectively is hard in domains where the cost commodity marginal basis is driven down. For example, the automotive industry needs to sell in volume, and thus cannot afford to spend thousands of dollars on high-end hardware in low-end to mid-level cars because the costs will not be recouped.

Another objective for large-scale CPS is co-scheduling or performing admission control and eviction of assorted task sets deployed on shared computing and communication resources to ensure that high priority operations take place at the appropriate time. These requirements are not typically met in conventional cloud computing environments, i.e., when these systems get overloaded, the QoS degrades and there is no way to prioritize between tasks.

Improved fault-tolerance fail-over that supports real-time requirements, which is crucial in environments with high probability of failures and attacks. One way to do this is semi-active replication [12], which is used so that running systems can fail-over rapidly and predictably. This replication style is designed to have some of the benefits of both the active replication and passive replication styles, including predictable fail over times and predictable behavior during program execution.

3. *Finer-grained and faster allocation of resources to enable CPS to be precisely scaled to meet demands driven by real-world phenomena.* Current elastic resource allocation approaches focus on virtual machines as the sole resource allocation unit. While virtual machines provide excellent isolation and resource partitioning properties, they incur significant allocation and startup costs. A single virtual

machine in a cloud may take tens of seconds to minutes to allocate and initialize resources for a CPS.

CPS are also influenced by a wide array of physical phenomena for which science has not developed accurate or fast predictive models. For example, predicting the exact load in a financial market even within a few minutes time is not a solved problem. Since it is hard to predict how the physical world will drive a CPS, it is hard to forecast far into the future the precise resource allocations that will be needed to meet CPS QoS goals.

Ensuring that CPS receive needed resources becomes hard when limited physical world predictability is combined with slow resource allocation. Either more precise predictive models are needed or cloud computing resource allocation must become more nimble to support timeliness and other QoS requirements. Research on faster and finer-grained resource allocation beyond virtual machines is thus needed to meet the challenge of producing fast and accurate predictive models for all physical systems that drive CPS.

4. *Data provisioning and load balancing algorithms that can take into account a variety of properties,* including geo-physical, when deciding where to migrate work. Cloud computing is generally considered so flexible that there is little distinction between where computation takes place and storage resides, which makes sense when there are no real-time QoS needs. As real-time QoS needs arise, however, exactly where parts of the system are located becomes more important. In these cases, affinity should be emphasized to reduce latency and jitter.

Storage is a key factor in CPS, as it does not do much good to virtualize storage if it then takes too long to move data from one node to another. At the same time, rebalancing and replication also need to happen. Taking physical dimensions into account in the context of load shaping is also beneficial and necessary. Developers must also discover ways to exploit physical characteristics of data and computation to better distribute work throughout CPS clouds.

In short, developers of large-scale CPS need a holistic approach. Advances in this area will be particularly challenging because success will require new approaches from a research point of view (which may be developed in isolation) to be combined across multiple system layers and to work seamlessly and effectively end-to-end in a CPS.

### C. Next-generation Challenges: Larger-Scale CPS

Over the past decade organizations have had greater success developing large-scale CPS. There has also, however, been a trend toward attempting to develop a new generation of more complex and large-scale CPS. Systems in this context are evolving towards ultra large-scale, i.e., they are pushing far beyond the size of even today's large-scale CPS by every measure, including: lines of code; amount of data stored, accessed, manipulated, and refined; number

of connections and interdependencies; number of hardware elements; number of computational elements, number of system purposes and user perception of these purposes; number of routine processes, interactions, and emergent behaviors; number of (overlapping) policy domains and enforceable mechanisms; and number of people involved in some way (see [13] for further discussion). Examples of these ultra-large-scale CPS are evolving in smart grid, Industrial Internet, and air traffic management domains.

The dynamic behavior of ultra-large-scale CPS can trigger transient overloads. There are numerous time critical tasks, and many resources depend on the environment. Often there are trade-offs and conflicts between the aforementioned resources. One of the most prominent challenges observed is integration with legacy systems and sub-systems.

The technologies historically used by system integrators to develop and sustain large-scale CPS have themselves incurred many challenges stemming from accidental and inherent complexities. For example, these technologies have tended to be highly heterogeneous in terms of programming languages, operating systems, middleware, and tooling. Likewise, technologies implemented several years ago may now be unusable in some environments due to rapid advances in the solution space.

Not surprising, it is tedious and error-prone to map problems and requirements from the problem space to the technologies that exist in the solution space. System integrators are ultimately responsible for trying to make these connections. These problems have recently become even harder to address because their requirements exceed the capabilities provided in today's commodity computing clouds.

Adding further complication, the U.S. government, which has been a major player in funding for large-scale CPS, has been forced to cut back significantly on research and development due to the fiscal constraints arising from sequestration. The following quote is attributed to Winston Churchill: "Gentleman, we've run out of money-it's time to start thinking," which serves as an accurate metaphor for what is happening in ultra-large-scale CPS domains today.

## V. A VISION FOR SOFTWARE INFRASTRUCTURES FOR LARGE-SCALE AND ULTRA-LARGE-SCALE CPS

This section outlines emerging research solutions and approaches for architecting large-scale CPS systems. The architecture covers the core components needed for CPS and specific technologies that can help to fill these gaps, such as the OMG's DDS.

### A. Key Requirements for Large-scale CPS Software Infrastructure

Meeting the challenges of large-scale CPS—including, but not limited to, approaches being discussed in the context

of proposed industry standards, such as the Industrial Internet [9], requires rethinking basic properties and principles commonly ascribed to cloud computing. Whatever the future of elastic cyber-physical systems software infrastructure may be, it should support the following requirements:

- Systems must be *flexible* as they must be able to replace, reuse, analyze, distribute, isolate, and then compose these pieces back together in a dependable way.
- Systems need to be *open* so that programmers do not program themselves into a corner with a solution that only works with commitment to a single vendor.
- Systems need to be *uniform* with respect to treating multicore and distributed core semantics in a common way. Uniformity keeps these two components transparent from the applications and services they run.
- Systems must be *scalable* as the demand for increasing scope rises. Solutions, such as load balancing algorithms, must take advantage of elastic hardware resources at the infrastructure level.

An important paradigm for meeting these requirements of large-scale CPS is *middleware*, which resides between applications and the underlying operating systems, networks, and hardware. Middleware provides services that are essential to design and operate large-scale CPS at scale. Below we discuss the key layers of large-scale CPS software infrastructure.

### B. Key Layers of Large-Scale CPS Infrastructure

Anyone who has taken a networking course knows that there are seven layers in the OSI stack and four layers in the Internet stack. In general, however, there's less familiarity of the layers within the middleware stack, which is essential for success in developing next-generation software infrastructure for large-scale CPS. The key layers are described briefly below.

*Operating systems* and *communication protocols* form essentially a hardware abstraction layer that allows higher-level services and applications to ignore differences in the underlying computing and networking hardware. *Host infrastructure middleware* is an operating system abstraction layer that abstracts away from the operating system and removes accidental complexities of the system's APIs. It amplifies programming software in a portable way. Examples of host infrastructure middleware include Java, Real-time Java, and Microsoft CLR.

The next level is *distribution middleware*, which allows for decoupling and abstracting the fact that there is a network between the sender and receiver of messages. Distribution middleware provides the ability to communicate across address and host boundaries in a way that is unobtrusive to the application. Examples of this type of middleware include SOAP, Web Services, CORBA and DDS. *Common middleware services* comprise the next layer.

After distribution middleware is implemented, it is easier to program across a network. The next challenge is deciding how to build reusable services that name the information, discover services, detect subscribers' presence, send events predictably, monitor the infrastructure, provide information durability, maintain historical data, record data flows and transactions, perform failover operations, etc., which fall within the realm of *common middleware services*.

*Domain-specific middleware services* form an important layer. These middleware services involve intellectual property or value added in a particular domain such as avionics, SCADA, C4ISR, air traffic management, and healthcare. This area is where the next generation of standards and capabilities must be researched and transitioned into practice.

### C. Promising Foundations Towards Elastic CPS Middleware: Data Distribution Service (DDS)

The Object Management Group's (OMG) Data Distribution Service (DDS) [15] supports many of the criteria for large-scale CPS software infrastructure mentioned above, i.e., it is flexible, open, uniform, and scalable. DDS offers a pattern language and a powerful software infrastructure for building loosely coupled, heterogeneous, evolvable, scalable, and dependable large-scale CPS.

DDS supports different types of information modeling, including relational modeling which uses a data-centric publish-subscribe abstraction in which events and their relationships to each other may be defined. It also supports object-oriented information modeling with its data local reconstruction layer.

DDS defines a global data space that enables publishers and subscribers to read and write topic data asynchronously, anonymously, and in a manner that is decoupled in time and space. It allows the production and consumption of data in this global data space and offers control over the ways in which information flows through a space, which is an essential capability for large-scale CPS.

DDS is also well suited for large-scale CPS in because of its rich set of QoS policies, which allow variables essential to delivering information in a timely and dependable manner to be controlled. There are about two-dozen QoS policies available in DDS, to handle priorities, deadlines, data durability, replication and redundancy, history, resource utilization and more. QoS policies that are particularly relevant to large-scale CPS include the ability to indicate latency and reliability bounds. Likewise, these QoS policies also support the ability to manage coherency issues and resource constraints.

DDS allows matching of publishers and subscribers in terms of QoS policies that are requested/offered (RxO). This distributed matching capability allows DDS implementations to decide how best to connect end-to-end flows among producers and consumers. When this capability is integrated

on top of intelligent communication infrastructure, it is able to provide enhanced control over the network core.

The ability to bridge diverse components of a CPS together is crucial. DDS provides many ways to bridge other technologies through the DDS data bus, which enables communication with other services and protocols in a way that can interoperate seamlessly with both legacy and new components and systems. There are also a number of standards available within the DDS ecosystem, such as language mappings for C++ and UML, and it is possible to take advantage of other standards as well, such as through mappings to RESTful services for tracking system status, and sending alerts and updates to key people.

When integrating large-scale CPS, no single vendor is likely to offer a sufficient set of services to realize an entire CPS. The potential to interwork and connect between different parties using a heterogeneous selection of middleware is thus both valuable and necessary. Interoperability protocols supported by DDS make it possible for services from different vendors to interoperate. There is also currently a vibrant research community focused on DDS [16], which further motivates its potential applicability as a context within which further refinement of policies and mechanisms for enforcing CPS semantics can be prototyped, explored, evaluated, and deployed.

### D. Promising Foundations towards Large-scale Sensing and Processing: Mobile Cloud Computing

As outlined in Section 5.1, a key goal of large-scale CPS systems is building systems that are flexible, open, uniform, and scalable. Mobile cloud computing is an emerging paradigm for building large-scale CPS that can sense the world through commodity mobile device sensors, aggregate this data in the cloud, and then make decisions that impact real world processes and decisions [17], such as vehicle navigation. These large-scale mobile cloud systems are attractive because of the ubiquitous deployment of existing mobile devices that can be used to rapidly construct large-scale CPS that sense and react to their environments in real-time. Moreover, these systems contain software APIs and distribution mechanisms that span huge numbers of devices in the Internet of Things. Cloud computing provides the additional scalability needed to adjust resource allocation and utilization as mobile devices join and leave the large-scale CPS.

## VI. CONCLUSION

Despite advances in elastic hardware and middleware technologies such as DDS, it is still hard to deploy CPS in cloud environments, making it necessary to investigate further advances in the state of the art for integrated elastic *hardware-software* infrastructures. It is unlikely that public clouds will serve as the basis of mission-critical large-scale CPS. It is more likely that private clouds will be used,

but that does not mean those systems will not benefit from standards and existing technologies.

What will matter most in developing new computing clouds for CPS is how a fundamental tension between multi-tenancy and elasticity on the one hand, and precision in the resulting CPS properties on the other hand, can be addressed. Virtualization may be beneficial if it can be afforded, but an alternative could be to run on the bare hardware using powerful integrative middleware technologies, such as those provided by (or perhaps evolved from) successful software infrastructure standards, such as DDS.

As an exemplar of the kind of flexible and efficient middleware that can be helpful for developing large-scale CPS, DDS is a particularly intriguing venue for further investigation because it is standards-based and includes a number of open-source solutions that facilitate the mixing and matching of capabilities and the ability to build infrastructure for dependable cyber-physical systems.

Combining the precise QoS fidelity needed by CPS with the scale and elastic sharing of resources offered by Cloud Computing is a promising direction for systems researchers. Although progress has been made, there remain many hard research challenges surrounding large-scale CPS, as discussed in the forthcoming report from the NSF Workshop on Cloud Computing for Cyber-Physical Systems [11].

In particular, thorough investigation of the trade-offs among the dimensions of QoS fidelity, system scale, and degree of asset sharing, is essential to separate accidental complexity that can be avoided from inherent complexity that must be managed. Experience so far developing large-scale CPS, and ongoing research towards even larger scale CPS, suggest that we have not yet reached the limits of those tradeoffs, but also suggest that new approaches, models, analyses, and techniques will be needed for further progress in large-scale CPS.

#### REFERENCES

- [1] J. Ding, J. Sprinkle, C. Tomlin, S. Sastry and J. Pournick. "Reachability Calculations for Vehicle Safety during Manned/Unmanned Vehicle Interaction." *AIAA Journal of Guidance, Control, and Dynamics*, 35(1):138-152, 2012.
- [2] S. Boyer, "SCADA: Supervisory Control And Data Acquisition (4th Edition)," International Society of Automation, 2009, ISBN 1936007096.
- [3] Q. Li, T. Cui, Y. Weng, R. Negi, F. Franchetti, M. Ilic, "An Information-Theoretic Approach to PMU Placement in Electric Power Systems," *IEEE Transactions on Smart Grid* 4(1), pp. 446-456, March 2013.
- [4] R. Bashshur, T. Reardon, and G. Shannon, "Telemedicine: A New Health Care Delivery System," *Annual Review of Public Health*, 21(1), pp. 613-637, May 2000.
- [5] M. Khalifa and R. Davison, "SME adoption of IT: the case of electronic trading systems," *IEEE Transactions on Engineering Management*, , 53(2), pp.275-284, May 2006.
- [6] M. Schinkel and K. Hunt, "Anti-lock braking control using a sliding mode like approach," *American Control Conference*, 2002.
- [7] Y. Le Gorrec, J. Magni, C. Doll, and C. Chiappa, *Modal Multimodel Control Design Approach Applied to Aircraft Autopilot Design*, *Journal of Guidance, Control, and Dynamics* 21(1), 1998.
- [8] M. Pajic and R. Mangharam, "Spatio-Temporal Techniques for Anti-Jamming in Embedded Wireless Networks", *EURASIP Journal on Wireless Communication and Networking*, 2010.
- [9] P. Evans and M. Annunziata, "Industrial Internet - Pushing the Boundaries of Minds and Machines," *General Electric*, available from [www.ge.com/docs/chapters/Industrial\\_Internet.pdf](http://www.ge.com/docs/chapters/Industrial_Internet.pdf).
- [10] P. Lardieri, J. Balasubramanian, D. C. Schmidt, G. Thaker, A. Gokhale, and T. Damiano, "A Multi-layered Resource Management Framework for Dynamic Resource Management in Enterprise DRE Systems," the *Journal of Systems and Software: special issue on Dynamic Resource Management in Distributed Real-Time Systems*, editors C. Cavanaugh and F. Drews and L. Welch, Vol 80, Issue 7, July 2007, pgs. 984-996.
- [11] ISIS, "NSF Workshop on Cloud Computing for Cyber-Physical Systems," available from [www.isis.vanderbilt.edu/workshops/cc4cps](http://www.isis.vanderbilt.edu/workshops/cc4cps).
- [12] J. Balasubramanian, A. Gokhale, A. Dubey, F. Wolf, C. Lu, C. Gill and D. Schmidt, "Middleware for Resource-Aware Deployment and Configuration of Fault-tolerant Real-time Systems," *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'10)*, April 2010.
- [13] Linda Northrop et al, "Ultra-Large-Scale Systems," *Software Engineering Institute*, 2005, available from [www.sei.cmu.edu/uls](http://www.sei.cmu.edu/uls).
- [14] A. Hakiri, P. Berthou, A. Gokhale, D. Schmidt, T. Gayraud, "Supporting End-to-End Quality of Service Properties in OMG Data Distribution Service Publish/Subscribe Middleware over Wide Area Networks," *Journal of Systems and Software* 86(10), October 2013.
- [15] D. C. Schmidt, A. Corsaro, and H. Van'T Hag, "Addressing the Challenges of Tactical Information Management in Net-Centric Systems with DDS," *CrossTalk special issue on Distributed Software Development*, May, 2008, pgs. 24-29.
- [16] A. Hakiri, P. Berthou, A. Gokhale, D. Schmidt, T. Gayraud, "Supporting End-to-End Quality of Service Properties in OMG Data Distribution Service Publish/Subscribe Middleware over Wide Area Networks," *Journal of Systems and Software* 86(10), October 2013.
- [17] J. White, S. Clarke, C. Groba, B. Dougherty, C. Thompson, and D.C. Schmidt, "R&D challenges and solutions for mobile cyber-physical applications and supporting Internet services," *Journal of internet services and applications*, 2010, 1(1), pgs. 45-56.