Editor: Schahram Dustdar • dustdar@dsg.tuwien.ac.at

Web-Scale Service Delivery for Smart Cities

Fei Li, Michael Vögler, Sanjn Sehic, Soheil Qanbari, Stefan Nastic, Hong-Linh Truong, and Schahram Dustdar • *Vienna University of Technology*

Smart cities encompass services in diverse business and technological domains. Presently, most of these services are delivered through domain-specific, tightly coupled systems, which entail limited scalability and extensibility. The authors propose Web-scale service delivery that addresses these limitations and encourage the creation of novel services based on a domain-independent, cloud-based service-delivery platform.

Current smart-city services are typically provided in single domains — for example, building management, transportation, and so on. With such services, domain-specific application requirements drive all system component design and determine most technical choices, ranging from sensors and smart devices to middleware components and computing infrastructure. The service-delivery process is rigidly orchestrated by *domain solution providers* who develop applications and integrate subsystems from various vendors. This model leads to many closed vertical systems with tightly coupled hardware, networks, middleware, and application logics. Scalability and extensibility are intrinsically limited in such systems, and the closed relationships between stakeholders stifle the creation of novel services.

Web-scale service delivery for smart cities is a new methodology aimed at delivering open and scalable smart-city services by encouraging collaboration between stakeholders in the Internet of Things and clouds. It exploits the prevalence of computing resources and software services distributed on the Web to break up the closed vertical service-delivery model. Here, we present new stakeholder relationships and service-delivery models, as well as the research challenges involved in realizing them.

Two Motivating Services

We first analyze two representative smart-city services and examine the limitations of the closed service-delivery model.

Building Energy Management Systems

Managing energy for buildings is one of the most important services in smart cities. A *building energy management systems* (BEMS) project typically starts when a solution provider surveys the target building, which can either already exist or be in design. A project's scale can range from a small building to a large campus comprising various building types. So, the number of devices, the volume of data that will need processing, and application complexity could vary significantly. After surveying and designing for the specific building, the solution provider will acquire suitable hardware devices from original equipment manufacturers (OEMs), integrate them into an infrastructure solution, and develop analytical and control applications.

This process produces many vertically isolated BEMS[1] (often referred to as "silos"), which leads to two problems. The first is maintainability: the more silos that are provisioned, the more system instances the solution provider must maintain. System maintenance becomes a particularly painful process — hardware devices are monitored in separate systems, and software instances must be updated separately and

 Published by the IEEE Computer Society 1089-7801/13/$31.00 © 2013 IEEE

tested on site with specific hardware configurations. The second issue is extensibility: many campuses and building compounds expand continuously to accommodate new users; deployed BEMS might need to manage more buildings and facilities. In any case, BEMS ought to scale up accordingly. In the current silo-based service-delivery model, such expansion could require the solution provider to reconfigure the whole system from the bottom up or to add new isolated BEMS because of tight coupling among devices, middleware, and applications. Furthermore, the energy consumption data that individual BEMS collect are largely underutilized because data storage and processing modules are isolated. Today, a painstaking data cleaning and integration process is a prerequisite for conducting fine-grained energy consumption analysis on a large scale.[1]

In brief, BEMS exemplify a *vertical service delivery* model, by which a single solution provider is generally responsible for provisioning and maintaining the entire solution throughout its lifetime. This is the dominant model by which most current smart-city services are delivered. Scalability and extensibility in this model are inherently limited.

Public-Event Organization

Public events constitute an essential part of urban life. Some events might reoccur regularly, such as yearly city marathons or national day parades; some might be ad hoc, such as demonstrations. An event at any scale has certain effects on city dwellers. Participants or visitors want to know how to reach and leave the location. Those who aren't interested want to know how to avoid the event. Public services should be ready for expected or unexpected situations. Event organizers must address all these concerns.

A typical event-organization service is composed of data collections, organization plans, notification channels, and contingency plans. It runs through four phases in normal situations — planning, preparation, operation, and finishing. In addition, contingency operations might be carried out if unexpected situations arise. The information required for an efficient organization application is diverse; it can include relatively stable information such as maps, public transportation routes, communication channel capacity, and facility accessibility, as well as real-time data such as traffic, weather, public transportation status, and parking lot occupation. Although most such information is available through existing public services, these services are isolated and domain-specific. They operate their own information infrastructures, process the data in-house, and publish them via various public channels. The key challenge to developing event-organization services is accommodating the event specifics (scale, local resources, processes, safety concerns, and so on) by properly collecting domain-specific services and information sources. Such development requires significant effort, given that it might be used only once or need to be updated periodically for regular events. Furthermore, the computing resources required to provision such services are needed only during the events. Thus, the effort required to deliver such a service is justified only if the development and provisioning are efficient and cost-effective.

Public-event organization is a case of *third-party service* delivery: an application provider acquires access to existing IoT services and other information sources to develop applications for specific purposes. These information sources are highly diverse, ranging from public services such as public transportation status to commercial services such as mobile networks. The provider's focus is on developing the application logic because it doesn't own the information sources or, in most cases, have direct control of them. However, the provider must provision computing resources to ensure quality of service (QoS). Third-party smart-city applications, particularly those needing to incorporate multiple IoT and Web services, are still uncommon due to the challenges this use case exemplifies.

Stakeholders in Smart-City Service Delivery

At the center of Web-scale smart-city service delivery are domain-independent service-delivery platform providers, who present a new type of platform-as-a-service (PaaS)[2] offering that integrates IoT devices and infrastructures, processes data from a large amount of distributed data sources in real time, and lets applications employ both IoT and cloud resources on demand. The management of both IoT infrastructure and cloud resources is hidden from application providers. Platform providers must ensure the required provisioning of computing resources involved in service delivery. The platform also provides cloud services, including service metering, billing, and tenant management that will let stakeholders share resources and establish flexible business relationships. Figure 1 illustrates the key components in a service-delivery platform and the change in stakeholder relationships compared to vertical service delivery.

Such a platform's emergence will directly influence traditional domain-specific solution providers. With a cloud-based platform, solution providers can leverage cloud resources to integrate IoT infrastructure and develop domain-specific applications, thus enabling virtualized vertical

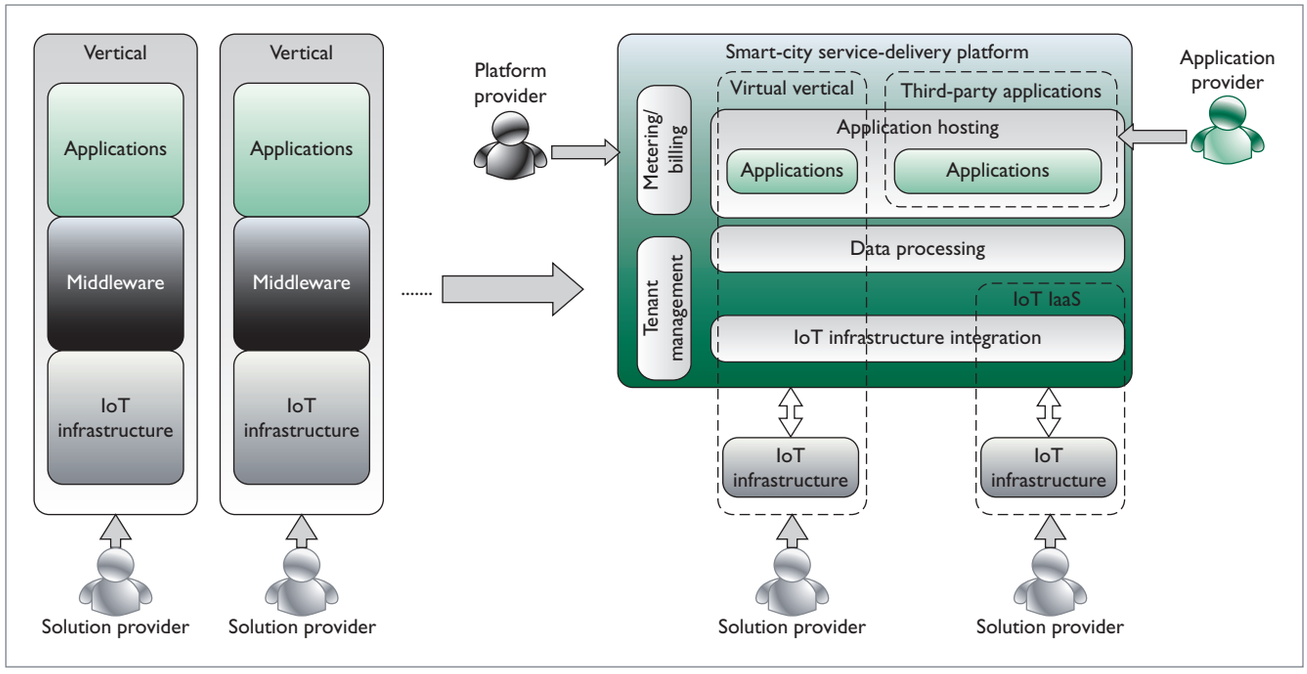

Figure 1. Emergence of service-delivery platform providers in smart cities. The platform is a new type of platform-as-a-service (PaaS) offering that integrates Internet of Things infrastructure and provides services for application providers to employ IoT and cloud resources on demand.

solutions, or *virtual verticals*. In virtual verticals, solution providers can reuse software services on the cloud and scale up services without investing in the computing infrastructure. In addition, other than the traditional role of providing vertically integrated IoT solutions, solution providers can also provide IoT infrastructure as a service (IoT IaaS) on the cloud to open IoT device capabilities to third-party application developers.

The platform will also benefit cloud application providers who specialized mainly in Web and cloud application development. The service-delivery platform lets these providers access IoT services to create novel applications for users. Application providers won't need domain-specific knowledge for managing IoT infrastructures because such infrastructures' capabilities are provided as services on the cloud, and the platform facilitates the important components for service delivery. Thus, application providers can focus on application logics and enjoy on-demand use of both cloud and IoT resources.

Service-Delivery Models

Collaboration between the aforementioned core stakeholders will enable open and scalable service-delivery models for smart cities. Figure 2 shows some typical service-delivery workflows.

Virtual Verticals

The virtual-vertical model corresponds with traditional vertical solution development: solution providers integrate IoT infrastructure (devices, networks, and so on) and develop application logics. The key difference is that the virtual verticals use computing resources and other necessary software services, such as data service and metering, on cloud platforms.

At first, IoT infrastructures are integrated through virtualization,[3] which is a set of commonly applied techniques for opening service interfaces to access device capabilities. A solution provider registers these IoT services to the platform for further use. Then, the provider can either offer them as IoT IaaS by configuring the services' metering and billing schemes, or use the services to develop virtual verticals. In the PaaS paradigm, virtually isolated system environments are ensured for applications via tenant management and application environment configuration. In the configured application-hosting environment, applications can scale up and extend their capabilities by employing the resources under an agreed tenant relationship, and cloud resource provisioning's elasticity can help the virtual verticals deal with fluctuating user demands. Finally, applications can also configure their metering and billing schemes to offer their services on the cloud.

This model is widely applicable to most existing vertical solutions. The first use case domain – BEMS – is a typical example. The platform can give each building a virtually isolated operation environment that's based on the IoT infrastructure

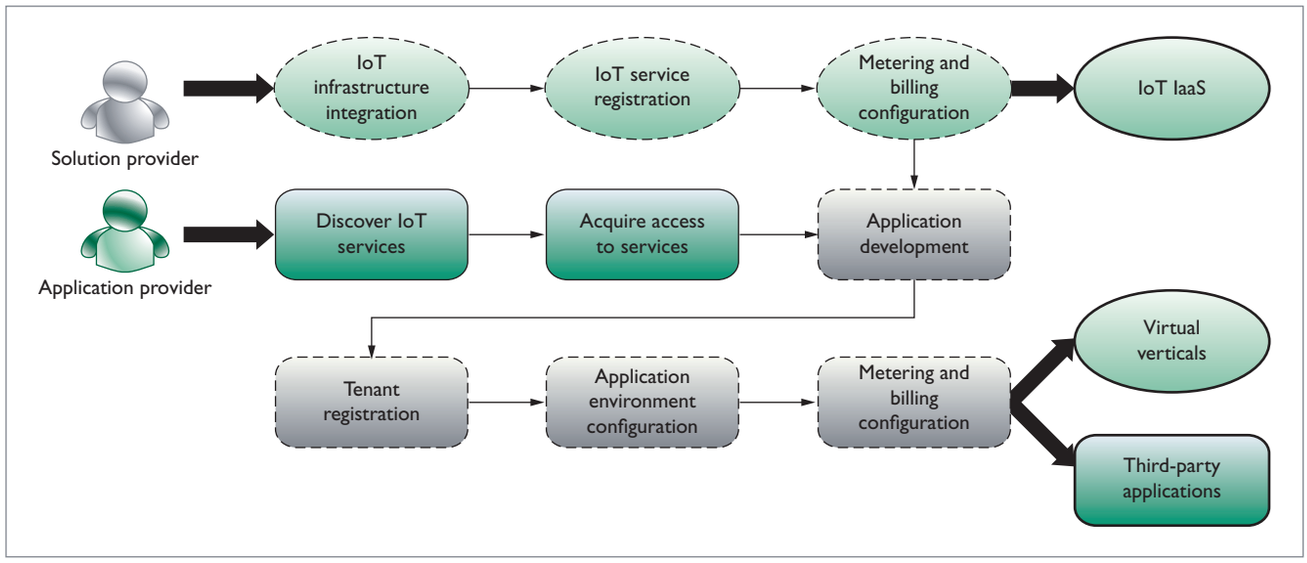

Figure 2. Service-delivery workflows. The workflows result in three service-delivery models — virtual verticals, third-party applications, and Internet of Things infrastructure as a service (IoT IaaS).

deployed in the building. The energy management system is delivered on the cloud and has access to buildings under its management, while sharing computing resources and data processing services with other services. This also allows data aggregation from multiple management domains for analysis.

Third-Party Applications

In traditional, vertical IoT service delivery, third-party application development isn't common due to tightly coupled system architectures and inflexible resource provisioning. On the smart-city service-delivery platform, third-party application developers can use platform services, including data and virtualized IoT services, thus mitigating the difficulties resulting from using IoT capabilities in third-party applications. The usual concerns of device maintenance and data acquisition in IoT services are decoupled from application development, so that application providers can focus on their business logics and quickly deliver new services.

Application development starts with discovering available IoT services and acquiring access, which might require application and IoT service providers to agree on service-level agreement (SLA) and billing terms. Similar to the virtual-vertical model, applications can have their own hosting environments configured with necessary virtualized IoT services, platform services, and computing resources. Furthermore, application providers can configure flexible usage models and billing schemes for applications so that the cost and revenue associated with them are monitored in real time. In this way, the platform fosters an environment for creating and delivering new applications as services.

Public-event organization is a typical case for using this model. Lower-level devices and domain-specific capabilities are publicly available. Application developers can access public information through the platform and acquire the specific data related to the event. Application providers are relieved from having to provision computing resources and from surging user demands. Furthermore, the PaaS paradigm gives specialized IoT solution providers an environment in which to serve multiple users through virtualization. For example, providers can use real-time public transportation monitoring in other applications.

Research Challenges

Several challenges remain before the proposed service-delivery models can be fully realized.

Effective Multitenancy and Resource Sharing

Smart-city services delivered as either virtual verticals or third-party applications must deal with a dynamic number of devices, varying amounts of real-time data, and ad hoc application usage on a shared cloud platform. Cross-layer planning methodologies are necessary to effectively provide each tenant with a virtually isolated, tailor-made environment with on-demand provisioning of such heterogeneous resources. Virtualization has been a hot topic in both cloud and IoT research, and it will play a critical role in resource sharing. However, research on each environment doesn't provide a coherent model for how to schedule or predict the use of both resource types. The key question is how to effectively and efficiently utilize heterogeneous and distributed computing resources for each service's specific functional and

nonfunctional requirements. Furthermore, the model must incorporate the ownership and physical environments of IoT infrastructure to share device capabilities when possible.

Programming Models

Programming support is inherent to PaaS. Because field environments are intrinsically volatile, and, in many cases, aren't under application providers' control, offering programming models that decouple applications from device specifics is essential. Such decoupling is needed not only to separate concerns at development time but also to accommodate IoT environments' volatility at runtime. Functional programming[4] can be investigated for smart-city services because it provides intrinsic scalability for accessing distributed services and managing real-time data. Another important aspect of programming models is to deal with the heterogeneity of IoT environments. When it comes to controlling IoT infrastructures, many functions require that control logics be executed in gateways, which usually offer nonstandard and vendor-specific runtime environments, posing significant challenges for Web application developers to provide control logics that are executable on these heterogeneous gateways. Furthermore, the capabilities for directly managing such environments and deploying applications on them might not be open to application developers, who are decoupled by the service-delivery platform. Therefore, it's necessary to investigate a generative programming approach that can describe device control logics on service-delivery platforms and automatically generate executables in different gateway environments.

Quality-Aware Real-Time Data Processing

Producing, processing, and consuming real-time data are essential to smart-city applications. To accommodate the volatile data quality in ubiquitous environments while delivering reliable services, the service-delivery platform must provide mechanisms to ensure real-time data's quality. Current research is focused on developing feasible metrics for real-time data by adapting conventional data quality metrics in databases and extending past work on quality of context.[5] However, two further challenges need to be addressed for quality-aware data processing in smart-city applications. The first is to use real-time data metrics to ensure required data quality for applications. Data quality assurance should be differentiated for different users in different situations. A range of assurance methods from statistical (if alternative data sources are uncommon) to selective (when alternative data sources are abundant) methods should be investigated for real-time data. The second challenge is to apply quality-assurance mechanisms to data services. This would require designing a novel data service interface and implementing the mechanisms for multiple data processing engines.

Metering and Service-Level Agreement

Metering is often provided on the cloud to measure how resources are used; this information contributes to system administration and billing. Compared to existing cloud offerings, the smart-city service-delivery platform must meter a broader range of resources in various application contexts. Solution and application providers should configure diverse usage patterns and billing models, such as subscription and pay-per-use. Because metering and billing aren't usually considered in vertically isolated IoT services, the service-delivery platform needs to provide on-demand metering and billing configurations for stakeholders. Configurability is the key to encouraging wider participation from small service providers. As regards billing, monitoring SLA fulfillment is also critical to charging for services and evaluating service providers' performance. A generic SLA framework for IoT services is still lacking owing to the dominantly closed service model. Established SLA practices in service-oriented architecture (SOA) research also need to be investigated to assess their feasibility for smart-city services. Furthermore, the SLA models in smart-city services are complicated because of diversity in application domains. So, the challenge is to establish a generic SLA model and corresponding assurance mechanisms at the platform level while enabling domain-specific SLAs for smart-city service providers.

Web-scale service-delivery models are the cornerstones of open and scalable services in smart cities. The proposed virtual verticals, IoT IaaS, and third-party applications will foster an ecosystem in which IoT solution providers, cloud platform providers, and application developers can collaboratively create novel services. The proposed delivery models are supported by a cloud-based service-delivery platform. Further research is needed to realize the platform and delivery models in the areas of heterogeneous resource management, programming models, real-time data processing, metering configuration, and SLA models.

Acknowledgments

This work is sponsored by the Pacific Controls Cloud Computing Lab (PC3L), a joint lab between Pacific Controls, Dubai, and the Distributed Systems Group at the Vienna University of Technology.

References

1. J. Cook, D. Smith, and A. Meier, "Coordinating Fault Detection, Alarm Management,

and Energy Efficiency in a Large Corporate Campus," *Proc. 2012 ACEEE Summer Study on Energy Efficiency in Buildings*, ACEEE, 2012, pp. 83–93.
2. P. Mell and T. Grance, "The NIST Definition of Cloud Computing (draft)," NIST special publication, vol. 800, 2011, p. 145.
3. D. Guinard et al., "Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services," *IEEE Trans. Services Computing*, vol. 3, no. 3, 2010, pp. 223–235.
4. S. Sehic et al., "A Programming Model for Context-Aware Applications in Large-Scale Pervasive Systems," *Proc. 8th IEEE Int'l Conf. Wireless and Mobile Computing, Networking and Communications* (WiMob 12), IEEE, 2012, pp. 142–149.
5. F. Li, S. Nastic, and S. Dustdar, "Data Quality Observation in Pervasive Environments," *Proc. 10th IEEE/IFIP Int'l Conf. Embedded and Ubiquitous Computing* (EUC 12), IEEE, 2012, pp. 602–609.

Fei Li is a postdoctoral researcher in the Distributed Systems Group at the Vienna University of Technology, and technical leader of the Pacific Controls Cloud Computing Lab. His research interests include cloud computing, the Internet of Things, and real-time data processing. Li has a PhD in computer science from the Beijing University of Posts and Telecommunications. Contact him at li@dsg.tuwien.ac.at.

Michael Vögler is a PhD candidate in the Distributed Systems Group at the Vienna University of Technology. His research is focused on cloud computing platform and enterprise services. Vögler has a Dipl.-Ing degree in computer science from the Vienna University of Technology. Contact him at voegler@dsg.tuwien.ac.at.

Sanjn Sehic is a PhD candidate in the Distributed Systems Group at the Vienna University of Technology. His research interests include ubiquitous computing and programming models for large-scale context-aware systems. Sehic has a Dipl.-Ing degree in computer science from the Vienna University of Technology. Contact him at ssehic@dsg.tuwien.ac.at.

Soheil Qanbari is a PhD candidate in the Distributed Systems Group at the Vienna University of Technology. His research interests include business models, pricing models, and billing mechanisms for cloud computing. Qanbari has an MS in computer science from Bahai Institute for Higher Education (BIHE). Contact him at qanbari@dsg.tuwien.ac.at.

Stefan Nastic is a PhD candidate in the Distributed Systems Group at the Vienna University of Technology. He's focused on smart applications and service governance for cloud computing. Nastic has a Dipl.-Ing in computer science from the Vienna University of Technology. Contact him at snastic@dsg.tuwien.ac.at.

Hong-Linh Truong is a postdoctoral researcher in the Distributed Systems Group at the Vienna University of Technology. His research interests include cloud elasticity and hybrid services. Truong has a PhD in computer science from the Vienna University of Technology. Contact him at truong@dsg.tuwien.ac.at.

Schahram Dustdar is a full professor of computer science and head of the Distributed Systems Group, Institute of Information Systems, at the Vienna University of Technology. His research interests include service-oriented architectures and computing, cloud and elastic computing, complex and adaptive systems, and context-aware computing. Dustdar is an ACM Distinguished Scientist and IBM Faculty Award recipient. Contact him at dustdar@dsg.tuwien.ac.at.