OpenADR 2.0 Deployment Architectures: Options and Implications

Ulrich Herberg, Daisuke Mashima, Jorjeta G. Jetcheva, Sanam Mirzazad-Barijough
Fujitsu Laboratories of America
1240 East Arques Avenue
Sunnyvale, CA, 94085, USA
{uherberg, dmashima, jjetcheva, sanam}@us.fujitsu.com

***Abstract*—OpenADR 2.0, an internationally-recognized standard for Automated Demand Response (ADR), defines the interaction between an ADR server and client, but does not specify all the possible multi-tier deployment architectures that are valid relative to the standard's specification. In this paper, we analyze the properties of a number of OpenADR-based architectures that have been proposed for deployment by ADR vendors, in terms of interoperability (compliance with the standard), scalability, complexity, and security, with the goal of helping utilities and third party DR aggregators make informed decisions about their planned ADR deployments to ensure high performing, future-proof, and secure DR services.**

I. Introduction

Demand Response (DR) is a service utilized by utility companies, Independent System Operators (ISOs) [1], and third party service providers [2], [3], to signal electricity customers to curtail their electricity usage temporarily in order to alleviate peak demand and to balance electricity supply and demand [4], [5]. While DR has been in use for many years, it was typically triggered manually by calling, emailing, or texting electricity customers. More recently, complexity of balancing the power grid has increased due to the introduction of intermittent renewable technologies and distributed energy (generation) resources (DERs) [6]. Moreover, due to ever increasing peak demand and peak electricity prices, automated DR (ADR) has attracted significant attention [7], leading to the development of OpenADR 2.0 [8], an internationally recognized standard for ADR. OpenADR 2.0 is developed by the OpenADR Alliance, with certification programs released in mid-2013, and published by the International Electrotechnical Commission (IEC) as Publicly Available Specification (PAS) [9]. Since then, multiple vendors have received official certification for their OpenADR servers and clients [10], and a number of utilities have started deploying or planning pilot projects using OpenADR-compliant products as found in [11].

The OpenADR specification defines only the basic interaction between an OpenADR server and client, and does not specify all the possible deployment architectures that are valid within the scope of the standard. This has led to the proliferation of proposed deployment architectures within the vendor community, some of which may not be compliant with the OpenADR 2.0 standard, and others whose performance and security implications may not be immediately obvious.

In this paper, we analyze the properties of the OpenADR-based architectures that have been proposed for deployment within the standard itself and by vendors of DR solutions or discussed informally within the OpenADR community. Our goal is to highlight the pros and cons of representative architectures in terms of interoperability, scalability, complexity, and security, and help utilities and third party DR aggregators make informed decisions about their planned ADR deployments to ensure high performing, future-proof, and secure DR services.

II. Overview of OpenADR

There are currently two profiles of OpenADR 2.0: 2.0a for simple devices (e.g., thermostats), and 2.0b for full-featured energy management solutions (e.g., energy management systems and DR aggregators). In the following, we discuss some specifications of these profiles.

A. Communication Model

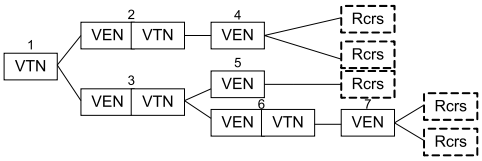

Figure 1. Example of tiered architecture of OpenADR nodes.

OpenADR defines a communication model between a DR server, also referred to as *Virtual Top Node (VTN)*, operated by an electric utility or other types of DR service providers, and DR clients, referred to as *Virtual End Nodes (VENs)*. A VEN is typically a gateway that controls one or more devices, called *Resouces*, that are the actual consumers of electricity and may participate in DR. Instead of a gateway, a VEN may also be a software module co-located on an end-device (e.g., a smart thermostat) directly controlled via OpenADR. Usually, a VEN controls Resources using protocols other than OpenADR, allowing the overall deployment to include both OpenADR-capable devices and devices supporting only other protocols.

OpenADR nodes are organized in a tree. An example is shown in Figure 1. In the simplest case, a tree has only one VTN-VEN pair, i.e., one tree hop (see Section III-A). More generally, there can be multiple hops, which may involve service providers for DR communication (see Section III-C). In such a case, intermediate nodes, such as node 2, 3, and 6 in Figure 1 may have both VTN and VEN functionalities.

As transport mechanisms, OpenADR supports HTTP and XMPP, the latter of which is best known as an online chat protocol. For HTTP, two communication modes are defined: PULL and PUSH. In HTTP PULL, communication is always initiated by a VEN, periodically polling for new messages. This allows connections from VENs that are behind firewalls and Network Address Translations (NATs). However, the PULL mode may cause significant communication overhead because of the periodic polling (refer to Section III-A for a study of the overhead). On the other hand, the HTTP PUSH mode allows both VEN and VTN to initiate communication whenever necessary, and thereby network traffic can be minimized. However, it requires a VEN to implement HTTP server functionality and to keep listening on an open port, which is often impractical when firewalls/NATs are used and might raise security concerns. PUSH mode is also implemented on XMPP so that we can avoid the issue with firewalls since XMPP provides client-initiated long-lasting TCP sessions that enable communication to and from behind firewalls and NATs.

B. Messages and Services

Message payloads in OpenADR are defined by an XML schema. OpenADR 2.0b provides four services, *EiEvent*, *EiReport*, *EiRegisterParty*, and *EiOpt*, and defines a different set of messages for each service. Among them, we provide some details of the first two services that are relevant to this paper.
EiEvent: DR events are sent from a VTN to a VEN to signal the VEN to reduce the electricity load of Resources attached to it. An event signal contains, amongst others, a start time, duration, and information about the amount of curtailment or updated electricity price. The VEN confirms or rejects participation in the event in its response message.
EiReport: Reports are usually sent from a VEN to a VTN to report the energy consumption or the status of Resources connected to the VEN. OpenADR supports history reports conveying a series of data points recorded in the past, and telemetry reports used for real-time reporting, which are periodically sent in a certain interval. The VEN first registers all of its reporting capabilities (sampling frequency, unit of measurement, amount of data buffered, Resources controlled by it, etc.) to the VTN in a *METADATA report*. Based on the capability information, the VTN can request appropriate reports when necessary. In the 2.0b profile, no incremental/decremental report capability registration is supported, so all capabilities must be registered at once. In other words, whenever changes in the capabilities occur (e.g., a Resource is added or removed), this registration process has to be repeated by sending all reporting capabilities.

C. Security

For all message exchanges in OpenADR, use of Transport Layer Security (TLS) with client authentication is mandated for mutual authentication as well as message integrity and confidentiality protection. The OpenADR 2.0 specification requires all nodes (both VTNs and VENs) to be equipped with public/private key pairs and digital certificates issued by a trusted Certificate Authority (CA), which implies that vendors have to pay nominal per-device cost for issuance and management of certificates. Communicating peers are required to authenticate each other by using the digital certificates.

Regarding authentication of VENs, the OpenADR specification requires to use the identifier of the VEN (*venID*) and some unique information derived from the VEN's digital certificate, e.g., a SHA fingerprint of the certificate. In order for a VTN to verify that a sender of an incoming message is actually the VEN whose venID is claimed in the payload, the VTN should perform validation of a one-to-one mapping between the venID and the digital certificate.

III. Architectures

In this section, we discuss the various proposed OpenADR deployment architectures and analyze each option in terms of interoperability (compliance with the OpenADR specification), scalability, complexity, and security.

A. Basic Two-Tier Architecture

The basic architecture of OpenADR is specified in [8], [12] and consists of a single VTN communicating with one or multiple VENs.

This architecture assumes that the utility/ISO directly communicates with all DR customers, and has full transparency and direct control of all VENs, i.e., it can send events to each individual VEN, and moreover, individual Resources attached to each specific VEN are enrolled with the VTN.

As DR proliferation in the residential and small and medium-sized business (SMB) sector increases, the total number of DR customers and thus VENs may become significant, e.g., in the range of tens and hundreds of thousands, and even millions. This configuration would pose a significant scalability challenge for this architecture. In this regard and to motivate the later discussions about advanced architectures, by using our OpenADR 2.0b-certified implementation, we measured the amount of typical network traffic for OpenADR communication when distributing DR events using the EiEvent service and reporting energy consumption via the EiReport service. This experiment was conducted in the local area network, and we measured the packet sizes in TLS sessions associated to these services. As the focus of this evaluation is placed on evaluating and estimating the volume of data communication, and not on the response time / delay, etc., an extrapolation to larger numbers of participating nodes is justified. The results are summarized in Figure 2 and Figure 3.

Assuming that one DR event is issued every day, we measured the amount of network traffic between a VTN and a VEN, and extrapolated the results to increasing number of VENs. All measurements are based on packet dump of TLS traffic and include overhead for TCP and TLS handshake. Regarding the polling frequency for the PULL mode, we considered a one second interval, which is expected to be a requirement for Fast DR or ancillary service implementation, to a one hour interval, which is usually sufficient for typical DR services. Regarding the number of nodes, we consider up

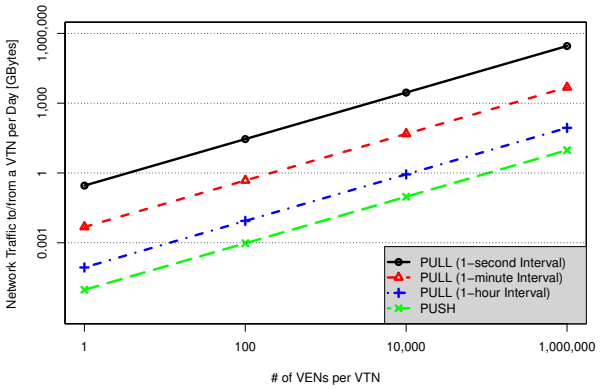

Figure 2. Estimated communication overhead for DR event distribution (in logarithmic scale)

to 1,000,000 nodes. Given that large utility companies in the US have millions of customers, this number is not unrealistic.

The difference in overhead between HTTP PUSH and XMPP PUSH was negligible, thus in the figure we only show the result of HTTP PUSH. The overhead of the PULL mode is significantly higher than PUSH since, regardless of whether a DR event is prepared for a VEN or not, VENs are required to inquire the VTN at a pre-determined interval for new messages. In the case of one second interval polling, 86,399 polling requests return empty responses while only one per day returns DR event information. As a result, including event acknowledgment messages from VENs, approximately 100 TByte traffic per day could be generated in the case of one million VENs.

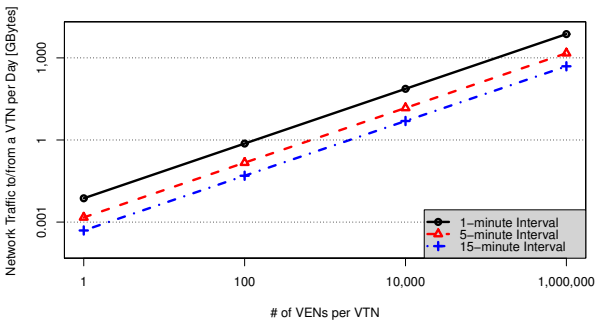

Figure 3. Estimated communication overhead for periodic telemetry reporting (in logarithmic scale)

To measure communication overhead for reporting, we focused on periodic telemetry reports, each of which conveys one meter reading. We considered three reporting intervals, 1 minute, 5 minutes, and 15 minutes. The report traffic contains report data and associated metadata sent by a VEN and confirmation from a VTN. The amount of traffic would exceed the order of 1 TByte when a VTN is handling over one million VENs even with a five-minute reporting interval. Also, if a single report payload contains more data points, the overhead will become even larger.

In addition to consuming bandwidth, sending a large number of individual messages results in increased latency at a VTN as messages are sent serially by the network interface. Moreover, the number of TCP states and database connections that a VTN needs to handle and computational cost for frequent TLS handshake are also issues to be considered. These problems can be alleviated by well-established engineering efforts, such as a load balancer, and / or alternative system architectures, which will be discussed later.

B. Two-Tier Architecture with XMPP

When XMPP is used instead of HTTP, communication is not established directly between a VTN and a VEN, but both are communicating as XMPP clients via an XMPP server (Figure 4). Communication between the VTN and XMPP server, as well as between the XMPP server and the VEN are respectively secured via TLS. However, the lack of end-to-end security implicitly requires that the XMPP server must be under control of and fully trusted by the entity operating the VTN (e.g., the utility or ISO).

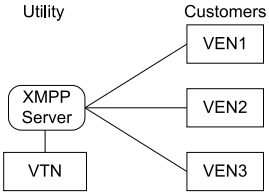

Figure 4. XMPP server architecture

In addition, as TLS is used pairwise between the VEN and the XMPP server as well as the XMPP server and the VTN, the VTN cannot access the VEN's digital certificate used for the TLS handshake (with the XMPP server). Therefore, it is not straightforward for the VTN to verify the mapping between the venID found in the incoming message payload and the digital certificate, which may require additional mechanisms on the VTN and/or XMPP server to meet the security requirements described in Section II-C.

C. Basic Three-Tier Architecture

The basic three-tier architecture of OpenADR is depicted in Figure 1 (e.g., the tree consists of nodes 1, 2, and 4) and includes a DR aggregator or other type of 3rd party such as a cloud-based Building/Home Energy Management System (BEMS/HEMS) provider between the utility and the customers [13].

The utility has a single OpenADR connection to the immediate intermediary (e.g., a DR aggregator), which in turn serves both as a VEN (towards the utility) and as a VTN (towards the customers). This use case is advantageous for utility companies that are interested in outsourcing customer recruitment and infrastructure deployment for communicating with a large number of customer VENs. The utility may also want to rely on sophisticated algorithms, devised by specialized DR aggregators, for selecting optimal customers to participate in specific events.

From an architecture standpoint, the constraints on the aggregator are the same as in the two-tier architecture. In

addition, the three-tier architecture gives rise to a new end-to-end security concern since OpenADR ensures mutual authentication and message integrity only for each hop; an end-customer receiving a DR event from the aggregator cannot directly verify whether this event was initially sent out by the trustworthy utility. Recently, one solution for this issue has been proposed to establish verifiable DR signal distribution path in multi-hop DR communication [14].

Concerning interoperability, the interface inside the aggregator between the VEN towards the utility and the VTN towards the customers is not specified in any standard. Therefore, each aggregator could interpret incoming events differently and then create equivalent events that are sent to the customers. For instance, if the utility sends an event to the aggregator with a `LOAD_DISPATCH` signal of 10 MW, the aggregator could select a certain number of its customers and then send events to each of them with a particular amount of required curtailment (e.g., 5kW). In this way, a DR aggregator has to "translate" the incoming event and then create new events sent to the end customers. For some of the payload elements of the DR event, such as marketContext (i.e., definition of a DR program), duration of the event, ramp-up and recovery, etc., it is unspecified how this translation has to be performed by the aggregator, which will likely lead to a different outcome for each implementation of translation by an aggregator.

D. Hybrid Device Deployment Architecture

Deployment scenarios may arise where some Resources within the customer premises are OpenADR-compliant (e.g., thermostats with built-in OpenADR VEN) and are only controllable through OpenADR, while others are not (e.g., a Smart Energy Profile (SEP) 2.0-based refrigerator). Non-OpenADR-compliant devices are typically implemented as Resources, whereas OpenADR-compliant devices are OpenADR VENs as software module on the same Resource. In order to provide grouping of Resources, both types of devices may need to be controlled by a single gateway device, e.g., HEMS or BEMS, which is seen as a VEN from the utility's perspective. This motivates the architecture shown in Figure 5.

This architecture is fully legitimate in terms of the OpenADR specification and allows higher flexibility for selection of end devices while maintaining a single point of control for customers' convenience. From the utility's point of view, it needs to handle only one VEN per customer's premise. On the other hand, its downside is that it may be too complex to implement on some of the more resource-constrained home gateway devices, as a VTN implementation is a lot more complex than a VEN.

E. Vendor Cloud

Figure 6 shows an architecture with a device vendor cloud interface in-between the utility and customers. The device vendor provides a single cloud-based VEN, whereas the customers' devices (e.g., thermostats) manufactured by the vendor are represented as OpenADR Resources. This implies that OpenADR itself is only used between the utility VTN and

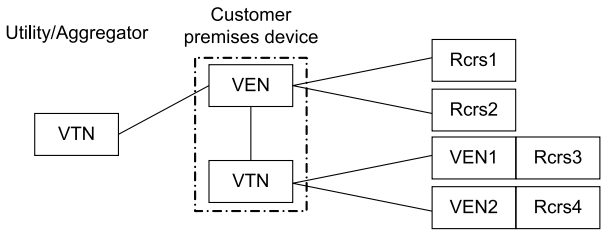

Figure 5. Hybrid device deployment architecture

the vendor-owned cloud-based VEN, and another (often proprietary) protocol enables communication between the VEN and the customers' devices[1]. When the utility sends an event to the VEN, the VEN translates the incoming signal into this other protocol and sends it out to the Resources. This use case is in deployment by several smart thermostat companies [15].

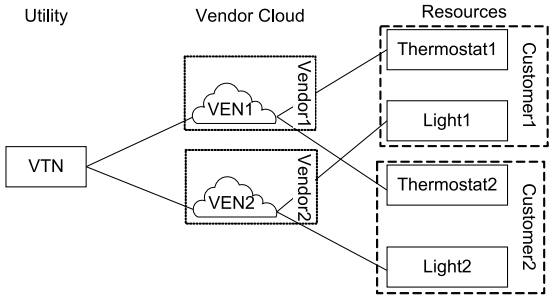

Figure 6. Vendor cloud architecture

As the VTN has only one OpenADR connection per vendor, this architecture provides significant complexity and scalability advantages to the utility. Instead, the responsibility for handling DR-related coordination is delegated to device vendors. In particular, when the cloud-based VEN receives an event from the utility VTN, the VEN has to create a payload in the proprietary format, select customers to send the event to, and then distribute the event to the selected customers.

Segmenting control across multiple DR service provider entities based on device type, and thus having each device in a customer's home be controlled by a separate DR provider may be undesirable as customers' burden for the curtailment cannot be optimally amortized. For example, one customer can experience multiple concurrent curtailments (one per vendor), whereas another may experience none, when it may have been less disruptive to curtail both but by a smaller amount. In order to avoid this, the utility would have to group Resources inside a single household accessible via VENs from different vendors, and to coordinate events sent via these vendors. This may eliminate the benefits of this architecture, which are to insulate the utility from having to deal with the scalability and complexity issue of managing all customer devices directly.

When the utility wants to receive a report from a particular customer device (rather than just aggregated/summarized reports from the vendor), another technical issue must be considered. As explained in Section II-B, all report capabilities of

[1]While proprietary protocols are predominant for current products on the market, protocols such as SEP2.0 may provide a standard-based alternative in the future.

a VEN, including information of Resources associated with it, must be registered through a single METADATA report. However, owing to the lack of support for incremental/decremental report registration in OpenADR 2.0, whenever a customer device is added or removed from the list (e.g., because the customer turns off the thermostat), then the complete list of all available reports must be resent. We generated sample METADATA report payloads based on the OpenADR 2.0b schema. With typical parameters, one METADATA report contains about two kilobytes of data per registered report (in XML, encoded in ASCII). Assuming that the vendor has 1,000,000 customers, each METADATA report is about two gigabytes in size. Since it can be assumed that customer devices will be added or removed to/from the vendor on a frequent basis, this may practically exclude reporting for this architecture, unless OpenADR is amended to support incremental report registration.

One of the main reasons to publish the OpenADR 2.0 standard was to avoid interoperability problems between proprietary DR standards: customers should be able to choose their devices from any vendor; as long as the device is OpenADR certified, the customer can be confident that the device is compliant with the standard. However, by introducing a cloud service and a proprietary protocol, this interoperability is not guaranteed anymore.

In terms of security, OpenADR mandates TLS with client authentication. This means that the cloud-based VEN needs only a single certificate, lowering the cost for the vendor significantly. On the other hand, it requires that the vendor uses its own authentication, confidentiality and integrity protection schemes for communication with end devices. Some of the thermostat vendors mentioned in [15] do not use encryption at all, or at least no client authentication, exposing their system to security threats.

F. Vendor Cloud per Household

To mitigate the drawback of the architecture discussed in Section III-E, let us discuss one alternative depicted in Figure 7. In this architecture, instead of using a single VEN endpoint for all customers, one VEN instance is running in the cloud per household (or per customer). Each customer device then represents a Resource attached to this VEN using a proprietary protocol, whereas the VTN and the VENs communicate via OpenADR. Similar to the architecture discussed in Section III-E, this configuration may be beneficial for using existing costumer devices as DR Resources. Also, a customer does not need to acquire a hardware-based VEN, reducing initial costs for setting up DR.

From a vendor's perspective, it is feasible to increase the number of VEN instances on the cloud. On the other hand, since each VEN requires its own security certificate, the vendor has higher expenses for the certificates, and managing one VEN per customer may be more complex than setting up a single VEN for all customers. On the utility side, this architecture faces the same scalability issues as the basic two-tier architecture (Section III-A) when the number of customers

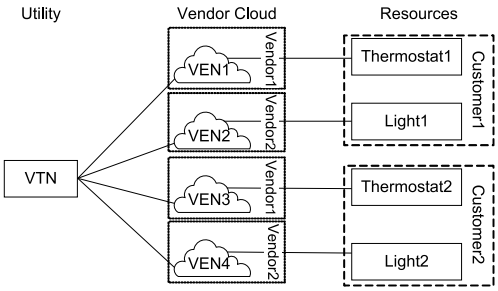

Figure 7. Vendor cloud per household architecture

increases. The situation may be even worse since multiple VENs could be allocated in a single customer's premise.

Compared to the vendor cloud architecture presented in Section III-E, this architecture gives the utility more control, as each household is represented as a VEN (per vendor). Therefore, DR events can be targeted to individual or groups of households. Also, the issues with incremental and decremental report registrations, described in Section III-C, do not exist, as each VEN can register its own reports. Note that even though this architecture shares the same interoperability issues with the vendor-based cloud configuration, the failure of a single VEN in the cloud results only in the disconnection of a single household associated with this VEN and not all the customers.

G. VEN Application Server Architecture

This architecture, depicted in Figure 8 and proposed in [16], defines a new entity (besides the VTN/VEN), which is intended to serve as a single "application server" or communication end-point for a group of VENs (i.e., in terms of TCP, DNS name, and certificate for TLS). The rationale is that this architecture allows for reduced number of communication streams at the VTN, which can communicate with one end-point per group of end-points. However, the flexibility of Resource grouping, which requires each group to be associated with a different VEN, is maintained. This architecture may be appealing for a customer that has multiple VENs under their control, for example in a large-scale industrial facility.

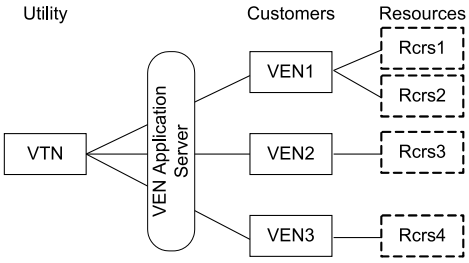

Figure 8. VEN application server architecture

Although not explicitly mentioned by the authors in [16], another motivation for this architecture may be to reduce the number of required digital certificates for VENs. Namely, only one certificate would be deployed on the VEN application server, rather than one per VEN, lowering the installation and operational cost of the system. While it seems attractive, the one-to-one mapping between certificate fingerprint and venID described in Section III-B cannot be maintained as multiple

venIDs would be associated with the same certificate fingerprint. If there is no such mapping, it becomes more difficult for the VTN to detect venID spoofing: any VEN from the group behind the application server could use any other venID from this group of VENs. To avoid such potential security issues, the VTN would be required to do additional checks for incoming OpenADR messages, verifying that messages with different venIDs originate from the associated IP address / DNS name, and therefore from behind the VEN application server and not from separate end-points. This could be done by adding the DNS name into the Common Name field of the x.509 certificate, and with a verification of the sender IP address towards this field. However, these are outside of the OpenADR specification and implementing them leads to increased system installation and maintenance complexity.

H. XMPP Server-to-Server Architecture

In the OpenADR Alliance, use of an external XMPP server to offload the CPU and memory requirements from a VTN onto an external entity by using XMPP server-to-server communication [17] is discussed. Such an architecture is depicted in Figure 9. In this architecture, the VTN is connected to the utility XMPP server, which itself has one (or more) connections to other trusted XMPP servers, e.g., hosted by DR aggregators. As the resource requirements are shifted from the utility VTN to an external entity in a tree-like structure, this solution has similar scalability advantages as the three-tier architecture described in Section III-C. XMPP server-to-server connections are not established on-the-fly, but rather a manual setup of certificates and trust relationships is required.

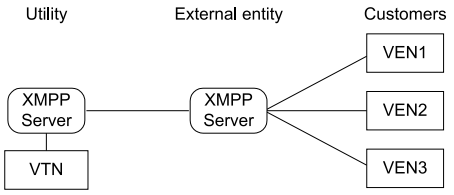

Figure 9. XMPP server-to-server architecture

Regarding security, responsibilities for authentication and secure communication are completely shifted from the utility XMPP server towards the external XMPP server, which is outside of utility's control: as described in Section III-B, authentication of VENs is done at the immediate XMPP server that VENs are connected to. In this scenario, the utility XMPP server can only verify that incoming payload comes from the "trusted" external XMPP server. The utility has no way of verifying the authenticity of the VEN or to match the venID with a certificate fingerprint. Thus, if the utility suspects that a VEN is compromised etc., the only countermeasure the utility can take is to revoke the external XMPP server's certificate, thereby blocking all VENs connected via the XMPP server.

Setting up the trust relationship via multiple XMPP servers and handling revocation of complete XMPP servers increases complexity of this architecture compared to the one presented in Section III-B. In order to alleviate the complete shift of trust from the utility to an external entity, a utility could require VENs to use XML Signatures (as defined in the OpenADR 2.0 specification). However, this may not be realistic in case of resource-constrained VENs, and so far no certified product implements this feature (to the best of our knowledge). Use of XMPP extensions for end-to-end confidentiality protection like [18] would help, but they are not defined in OpenADR 2.0b, resulting in compatibility issues.

IV. Conclusion

In this paper, we presented our analysis of the properties of the OpenADR-based architectures that have been proposed for deployment within the standard itself and also by vendors of DR solutions, and highlighted the pros and cons of each architecture in terms of interoperability (compliance with the OpenADR specification), scalability, complexity, and security. We hope that our observations and conclusions would help utility companies and third party DR aggregators make informed decisions about their planned ADR deployments to ensure high performing, future-proof, and secure DR services. As a future work, it is desired to conduct experiments for each architecture to provide more quantitative evaluations and comparisons in terms of performance and overhead.

References

[1] Federal Energy Regulatory Commission, "Regional Transmission Organizations (RTO)/Independent System Operators (ISO)," http://www.ferc.gov/industries/electric/indus-act/rto.asp.

[2] EnerNOC, Inc., http://www.enernoc.com.

[3] Comverge, Inc., http://www.comverge.com.

[4] M. Albadi and E. El-Saadany, "A Summary of Demand Response in Electricity Markets," *Electric Power Systems Research*, vol. 78, no. 11, pp. 1989–1996, 2008.

[5] V. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. Hancke, "Smart Grid Technologies: Communication Technologies and Standards," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 529–539, 2011.

[6] S. Kiliccote, P. Sporberg, I. Sheikh, E. Huffaker, and M. A. Piette, "Integrating renewable resources in California and the role of automated demand response," 2010, Lawrence Berkeley National Laboratory, Berkeley, CA.

[7] E. Koch and M. A. Piette, "Architecture Concepts and Technical Issues for an Open, Interoperable Automated Demand Response Infrastructure," in *Grid Interop Forum*, 2007.

[8] OpenADR Alliance, "OpenADR 2.0 Profile Specification B Profile," 2013, http://www.openadr.org/specification.

[9] International Electrotechnical Commission (IEC), "PAS 62746-10-1: [...] OpenADR 2.0b Profile Specification," 2014.

[10] OpenADR Alliance, "OpenADR 2.0 Certified Products," http://www.openadr.org/certified-products.

[11] Demand Response Research Center, http://drrc.lbl.gov/.

[12] OpenADR Alliance, "OpenADR 2.0 Profile Specification A Profile," 2012, http://www.openadr.org/specification.

[13] E. Koch and S. Kiliccote, "Role of Standard Demand Response Signals for Advanced Automated Aggregation," in *Proceedings of the GridInterop*, 2011.

[14] D. Mashima, U. Herberg, and W.-P. Chen, "Enhancing Demand Response Signal Verification in Automated Demand Response Systems," in *Proceedings of the 5th Innovative Smart Grid Technologies Conference*. IEEE PES, 2014.

[15] Navigant Research, "Automated Demand Response," 2014.

[16] IPKeys Technologies LLC, "OpenADR 2.0 VEN Concepts," 2013, available through OpenADR Profile WG mailing list archive.

[17] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," 2004, http://tools.ietf.org/html/rfc3920.

[18] "Off-the-Record Messaging Protocol Version 3," https://otr.cypherpunks.ca/Protocol-v3-4.0.0.html.