Edge-Net: A Lightweight Scalable Edge Cloud

Justin Cappos
Tandon School of Engineering
NYU
Email: jcappos@nyu.edu

Matthew Hemmings
Email: discount.yoyos@gmail.com

Rick McGeer
US Ignite and
University of Victoria
Email: rick.mcgeer@us-ignite.org

Albert Rafetseder
NYU Tandon and
University of Vienna
Email:albert.rafetseder@univie.ac.at

Glenn Ricart
US Ignite
Email: glenn.ricart@us-ignite.org

***Abstract*—This paper describes *Edge-Net*, a lightweight cloud infrastructure for the edge. We aim to bring as much of the flexibility of open cloud computing as possible to a very lightweight, easily-deployed, software-only edge infrastructure.**

Edge-Net has been informed by the advances of cloud computing and the successes of such distributed systems as PlanetLab, GENI, G-Lab, SAVI, and V-Node: a large number of small points-of-presence, designed for the deployment of highly distributed experiments and applications. Edge-Net differs from its predecessors in two significant areas: first, it is a software-only infrastructure, where each worker node is designed to run part- or full-time on existing hardware at the local site; and, second, it uses modern, industry-standard software both as the node agent and the control framework. The first innovation permits rapid and unlimited scaling: whereas GENI and PlanetLab required the installation and maintenance of dedicated hardware at each site, Edge-Net requires only a software download, and a node can be added to the Edge-Net infrastructure in 15 minutes. The second offers performance, maintenance, and training benefits; rather than maintaining bespoke kernels and control frameworks, and developing training materials on using the latter, we are able to ride the wave of open-source and industry development, and the plethora of industry and community tutorial materials developed for industry standard control frameworks. The result is a global Kubernetes cluster, where pods of Docker containers form the service instances at each point of presence.

***Index Terms*—distributed systems, edge cloud**

I. Introduction

Distributed edge clouds have been the most successful research infrastructures in history. Their impact on the field can be measured by publication count. In the first five years of its existence, 2003-2008, a total of 68 papers at the six leading networking and distributed systems conferences (SIGCOMM, INFOCOMM, NSDI, OSDI, SIGMetrics, and SOSP) cited PlanetLab [22] as their experimental/operational infrastructure. Of these, 38 were at NSDI, of a total of perhaps 125 NSDI papers published over those years. During this period, when it was at its operational peak, it was the premiere networking and distributed systems experimental and observation platform. In fact, *every* submission to SOSP 2003 cited experiments on PlanetLab.

To date, no general-purpose distributed systems platform has matched the breadth and scope of PlanetLab. PlanetLab's website still shows 1353 nodes operating at 717 sites, and its MeasurementLab project with Google and the New America foundation remains one of the premier Internet observation platforms.

PlanetLab is dependent on bespoke software that has been made obsolete by recent developments in Cloud technology. PlanetLab uses a 35,000 line custom Linux kernel mod on its nodes, which must be maintained by the PlanetLab staff. The functionality of this kernel mod has now been completely replicated by standard improvements to the Linux kernel, and adds little value to PlanetLab.

The use of a custom kernel mod means that PlanetLab's node OS now trails current Linux distributions, because the kernel mod must be ported to each new Linux release. PlanetLab's nodes now run a dated Fedora Core release. The tools and software base available to PlanetLab experimenters through public repositories are thus limited; what should be an installation becomes a from-source porting effort for many modern pieces of software, meaning that deploying experiments has now become a non-trivial exercise.

PlanetLab's successor in spirit, GENI [18], offers greater capabilities than PlanetLab – in particular, the ability to create customized layer-2 networks across the wide area – but is similarly heavyweight and dependent upon GENI-only software. A GENI rack requires two weeks of installation and acceptance tests and it has been estimated that it requires at least 10% of an FTE for on-site maintenance. These factors limit the growth of the GENI infrastructure, since becoming a GENI site requires a substantial commitment from the host institution.

The deep capabilities of GENI come with their own cost; network programmability means that each experimenter who wishes to use GENI's network capabilities must program, or at least configure, a private layer-2 network between his experiment's sites [23], [25]. For a researcher wishing to explore advanced network programmability or protocols, this capability is an unmatched resource; for a distributed-systems researcher content to build systems on top of conventional layer-3 networks, it is unnecessary. For this reason, many GENI experiments now run on top of the routable "control

connection" rather than the private layer-2 data plane [9]. This is not a fundamental weakness of GENI: rather, it is evidence that many researchers are using GENI for PlanetLab-like experiments.

A summary of the testbeds for networking and distributed systems researchers, and their strengths and weaknesses, appears in Table I.

As can be seen from the table, Edge-Net offers the ease of use, scalability, and easy installation of Seattle [8] combined with the multi-purpose capabilities of PlanetLab.

Edge-Net occupies an important niche in the ecosystem of testbeds which is currently underserved: the niche of very wide-area, lightweight, multi-purpose distributed-systems testbeds.

II. Development and design considerations

Edge-Net began as an embedded infrastructure on GENI, the GENI Experiment Engine [3]–[5]. The power of GENI came with a cost – acquiring and configuring resources was a time-consuming and cumbersome task. For applications which required only PlanetLab-like resources, this was an unnecessary complication. The GEE offered single-click instantiation of a PlanetLab-like slice.

PlanetIgnite [6] was the next generation of the GENI Experiment Engine. The observation that inspired PlanetIgnite was that GENI was really only an authentication mechanism and source of Virtual Machines; but the GEE ran on a standard Ubuntu VM, which could run essentially anywhere. PlanetIgnite offered a downloadable image with automated registration, so a new site could joint PlanetIgnite in 15 minutes.

PlanetIgnite's central weakness was that it offered a single container to each service at a site; modern services are composed of container swarms, and it was desirable that the next generation infrastructure offer this. Further, PlanetIgnite used its own orchestration and management tools, which both represented a support burden and was unfamiliar to users. Edge-Net will remove these weaknesses.

Edge-Net is a bottom-up edge cloud designed to run on existing infrastructure (local Clouds, servers, personal computers, embedded systems), controlled by industry-standard advanced Cloud software. The design goal for Edge-Net is that any local virtual machine with Internet connectivity should be able to join the Edge-Net worldwide infrastructure in under five minutes, and any developer should be able to deploy his application across Edge-Net using standard, familiar Cloud technologies within five minutes.

Key elements of Edge-Net's technology are familiarity and simplicity. Edge-Net's worker nodes must be able to run on a wide variety of computing equipment without manual intervention or customization. This implies that the worker node must be as simple and make as few demands as possible on the underlying hardware. Further, the developer must be able to deploy his software across the Edge-Net infrastructure using familiar tools and technology, and of course the software must be able to be executed automatically with on-site resources and without developer intervention or customization.

One good choice for a Edge-Net worker node is an Ubuntu 16.04 VM running a Kubernetes Worker Node. This is an extremely well-tested cloud host environment from Google, runs any Docker container, and is the industry standard for Cloud deployments.

So Edge-Net is just a worldwide, multi-user Kubernetes cluster where nodes can join (or leave) at any time, simply by installing the Edge-Net software onto a standard VM on their site. Developers can use standard Kubernetes tools to deploy their service across Edge-Net, and are presented with the familiar elements of the Kubernetes UI – the Kubernetes dashboard and `kubectl`.

III. The Need for Ubiquity

Edge-Net is a bottom-up cloud. Edge-Net nodes are VMs which run on general-purpose computing nodes; these may be local Clouds, servers, or even personal or embedded computers running the appropriate software. Edge-Net is bottom up because edge Clouds must be orders of magnitude more ubiquitous than existing commercial Clouds. This is due to a theorem from physics and plane geometry. Getting a Cloud node within k milliseconds of a point on a plane requires that the Cloud node must be within k/c meters of the point where c is the speed of light. Thus, if a service is to be offered everywhere, there must be a Cloud node within k milliseconds of any point on the plane, and that requires covering the plane with circles of radius (k/c). The area of a circle is proportional to the square of its radius, so this amounts to covering the plane with circles of area $(k/c)^2$. Since the area of the plane is unchanged, reducing k by a factor of x requires increasing the number of circles (and thus the number of nodes) by a factor of x^2. Cutting latency by a factor of 10 means 100 nodes for every Cloud node today. This is an enormous and slow undertaking if done as a top-down commercial Cloud.

However, there is another model for building infrastructure. In the late eighties and early nineties, digital libraries were the hot topic, and they appeared to he a massive undertaking, with enormous disk farms and city-scale computing. There was no centralized, massive buildout; rather, a simple piece of software was made available for download by the National Center for Supercomputing Applications. Within a decade, the World Wide Web had spread to millions of sites and a digital library of massive, worldwide scale – admittedly uncurated and fragmented – by the local, individual action of millions of individuals and organizations.

Later systems that use a similar approach to growth include the renowned SETI@Home [2] distributed computation platform, and Seattle Testbed's viral deployment strategies that use existing end-user hardware to host testbed VM instances.

Edge-Net's vision is to replicate this experience: to build a world-girdling, always-available, next-to-you-wherever-you-are Cloud through the actions of millions of individuals and organizations.

IV. Applications and Services

The Edge-Net system is the platform for a new class of applications and services: Cloud-in-the-loop systems.

Testbed	Signature Strengths	Weaknesses	Installation Time	Scalability
GENI [18]	Programmable Layer-2 Networking	Narrow footprint	Two weeks	Limited
Seattle [8]	Extremely Broad Distribution Easy to use	Restricted Programming Environment Layer-3 only	Minutes	Very High
PlanetLab [22]	Broad Distribution Easy to use Multi-purpose Environment	Layer 3-only	Days	Moderate
Edge-Net	Very Broad Distribution Easy to use Standard Environment	Layer 3-only	$<$ 15 Minutes	Very High

TABLE I
STRENGTHS AND WEAKNESSES OF WIDE-AREA TESTBEDS

Lightweight devices (smartphones, sensors, actuators) interact with the real world (things and people) and can do some lightweight computation, but their computational power and/or battery power is soon exhausted. Conversely, Cloud systems are arbitrarily powerful, and so the natural pairing is lightweight devices with powerful Cloud systems: devices offload their storage and computing requirements to Cloud nodes. However, Cloud systems are often too far away in communication cost and/or latency; Google Cloud has 11 Points of Presence in North America in five regions (a sixth is planned). Distance is time; device-Cloud latency is on the order of tens of milliseconds or more. This means that a Cloud node and a device can have at most a few transactions per second, severely limiting the classes of applications which feature intimate device-Cloud interaction.

Reducing the latency between device and Cloud opens up a broad array of new services, including user-driven data-intensive visualizations [7], [10], [11] virtual and augmented reality projected into thin devices, remote surgery, collaborative design, and prediction and analysis for the Internet of Things including real-time video processing and analytics.

A. Wide-Area Distributed Systems

Applications for Edge-Net include wide-area messaging, distributed hash tables, distributed key-value stores, distributed blockchains such as Ethereum and Bitcoin, distributed filesystems such as Syndicate, overlay multicast, overlay routing, content-centric and named-data networking, among many others. Content-Centric [21] and Named Data Networking [28], when implemented at the overlay level, are essentially naming schemes and APIs overlaid on content-distribution networks. There has been a revival of interest in these systems in the past couple of years, largely due to the recognition that classic centralized Cloud systems do not have sufficient bandwidth and/or have too much latency to the edge for many services and applications. This forms the central motivation for the DARPA Dispersed Computing program [27]. Wide-area distributed systems are distinguished from Cloud distributed systems in that intercomponent latencies are three orders of magnitude or more higher than in classic Cloud distributed systems, and intermittent connectivity is a given. These considerations offer many fruitful directions for research, particularly on latency considerations for eventually-consistent wide-area systems [1]. Scalability of blockchain-like systems such as BigChainDB [17], are a particularly fecund research area. Further areas of investigation are autonomic control of wide-area systems.

B. Wide-Area and Edge Internet Measurement

Wide-area Internet Measurement remains an active field of research, with ACM Sigmetrics continuing as one of the leading systems conferences. Internet measurement relies on layer-3 connectivity to the routable Internet, not special-purpose layer-2 networking, so Edge-Net is very well suited for these tasks. Thanks to the standardized software that powers Edge-Net, its deployments may easily grow into the edges of consumer networks. This enables studying parts of the Internet that have been historically difficult to observe.

C. Generalized Gateway for Cyberphysical Systems

This is a variant of the mobile offload problem: in this case, for Cyber-physical systems (CPS). CPS are typically single-board computers with low-end processors, for reasons of both power conservation and cost. Raspberry PIs are particularly popular. This presents similar problems to those of cellphones: the systems don't have enough on-board processing to do the computation desired, but the Cloud is too far away for near real-time response. A further problem, present here, is one of security. In order to access the Cloud, the CPS devices must have Internet access. Putting the full networking stack into a CPS device opens up new network vulnerabilities, since they are designed to be underpowered and cheap, without the protections against viruses common in richer computing environments. The consequences were dramatically demonstrated by the webcam attack on DynDNS in October 2016 [13].

Edge-Net also runs on less powerful gateway devices and can thus provide a gateway functionality for CPS. On the one hand, it controls access from the outside inwards. On the other hand, it provides additional computational power for smart local sensing tasks.

D. Distributed Query Processing on Distributed Data Sets

This activity also represents a wide-area distributed system, but its extremely broad utility extends across the computing, social, and domain sciences. Distributed forensic processing

of log data is the key motivating problem behind DARPA's Dispersed Computing program. Some examples of large, distributed data sets in the domain sciences are radiotelescope measurements of pulsars, used to detect gravitational waves, and the outputs of climatological models examined in the Climate Model Intercomparison Project (CMIP). The central computational challenge in all of these data sets is running what amounts to a large, distributed MapReduce – search for data points of interest and then return those to a central site for processing.

E. Localized Systems

This is a shorthand term for systems which are offered everywhere, but where each local recipient must be served by a local server. Content Distribution Networks are perhaps the canonical example, but this has become a broad area of research with multiple systems offering a rich array of functionality. Specific examples are the NOWCasting weather system [14], [15] and the Ignite Distributed Collaborative Visualization System [7], [10], [11]. The latter is particularly instructive, since [7], [11] give a detailed analysis of the requirements for replicated computation in a fat-data thin-client interactive system.

1) Mobile Offload: Mobile offload is a special case of localized systems where the clients are mobile devices. The motivation is similar to the cases sketched for gateways and query processing above. Note particularly that the goal is to offload computation to standardized servers over standard layer-3 networking.

F. Any Linux-based GENI Experiment Which Doesn't Rely on the Private Network

Any Linux-based GENI Experiment which currently sends its traffic on the Control Plane could use Edge-Net in addition to GENI or instead of it, freeing up GENI resources and getting a broader geographic footprint as well.

V. Using Edge-Net

A. About Kubernetes

In the beginning, there was Borg. Borg was Google's container management system, which ran just about every application inside Google, highly efficiently. A full description was given in [26]. In 2014, Google released an open-source version of Borg for public use, called Kubernetes, or K8s. Kubernetes is now widely used in industry for cluster and cloud management, and there are copious tutorials and playgrounds available for people to get familiar with Kubernetes and its use. Indeed, the wide variety of training materials and answers on StackOverflow and similar sites was a strong motivation for choosing Kubernetes; there are more training materials available than there ever were for GENI, SAVI [16], PlanetLab, or other testbeds like G-Lab [19] and V-Node [20].

The notable feature of K8s is that it is a pure container infrastructure.

B. K8s Concepts

A K8s deployment is called a **service**. An instance of a service is called a pod. A pod is an ensemble of microservices, each of which is encapsulated in a Docker Container. A developer registers his pod either with the K8s command-line controller or with the web-based controller; assignment of pods to worker nodes can be done manually or via the K8s scheduler. A Daemon Set is a pod instance that should be continuously running; full-time services are of this form. A good example is a persistent, multi-tenant, distributed key-value store or a persistent monitoring service. A Namespace is the unit of isolation in K8s. Namespaces are groups of mutually-visible K8s services and daemon sets. Namespaces are created by the Kubernetes head node (the equivalent of the GENI management node), and refer to a collection of services. Namespaces are accessed by a certificate created by the head node when the namespace is created.

C. K8s Networking

It is anticipated that during the execution of a K8s Pod, that the worker node(s) on which the pod is running may fail, or the Pod may migrate across worker nodes in response to load, latency, external demand, and so on. As a result, K8s Pods are not addressed by IP address but by service name; the K8s proxy takes care of resolving Pod names to addresses. Edge-Net slices can host services on raw ports; however, port contention is managed by K8s. It is strongly recommended that Edge-Net users use the name resolution option rather than request direct access to external ports.

D. Control of a K8s Service

A K8s Service is controlled by the user from the command line of his personal computer or a web interface, which he can run locally through localhost or, in the alternative, can be given by the provider. In both cases the developer-facing tool (the web proxy or the `kubectl` command-line program) is the primary means of controlling, placing, running, and stopping Kubernetes pods. Typically, a pod is declared in a yaml file and created using a `kubectl` command, e.g. `$ kubectl create -f docs/user-guide/walkthrough/pod-nginx.yaml` And then run with `$ kubectl run` with appropriate parameters.

Once run, Pods can be entered using the exec command, stopped, started, exposed as a service, etc., using a command syntax very similar to Docker. One exception is that while Docker containers are bound to a single machine, Pods are bound to a cluster. In fact, to a K8s developer using a native K8s infrastructure, both VMs and physical machines are more or less irrelevant; assigning Pods to VMs or hardware (generally hardware; in a pure K8s environment VMs have little value) is the job of the K8s scheduler.

VI. Mapping GENI Concepts to Kubernetes

A K8s Service corresponds fairly closely to a GENI Slice. The major difference is one of perspective: a GENI Slice is

defined in terms of the operator's perspective (it is a bag to which the developer attaches resources) rather than from the developers (a Service is an organized collection of execution instances which together deliver a service to the end-user). The K8s Pod plays a role roughly equivalent to a GENI sliver. The rough equivalence is again primarily due to the operator vs. developer perspective; GENI defines a sliver as a resource which is attached to a slice; K8s defines a Pod as a collection of containers which form the unit of instantiable functionality for a Service. To see the difference, note that a GENI experimenter who wished to use K8s to deploy and organize his experiment might make the reasonable choice to deploy each Pod in a VM; in this case, GENI would see each Pod as a sliver. However, another reasonable choice, depending upon the resources consumed by each Pod, is to have multiple Pods in a VM. This would be the preferred option when the resources demanded by a Pod are relatively modest: not only is it much more resource-efficient, a Pod spins up very rapidly, on the order of seconds; in contrast, spinning up a VM on GENI takes about 15 minutes. In this case, GENI would continue to regard the VM as a sliver and the Pods, which form the actual unit of the service, are transparent to GENI. The unit of tenancy in K8s is the namespace; this is a collection of Pods that can be accessed through a single authorization certificate, and for our purposes can be regarded as isomorphic to a GENI Project.

There are three major issues confronting any distributed edge system: sharing worker nodes between multiple users, resource contention on the nodes, and, since network addresses are in perenially short supply, multiplexing popular ports on a single IP address. We address these here.

Central to our design is the Kubernetes *Namespace*. A Namespace is the unit of tenancy and trust in Kubernetes, and is the unit of tenancy and trust in Edge-Net. The Edge-Net namespace is an augmentation of the underlying K8s Namespace, and a set of services around it.

A. The Edge-Net Naming Architecture

Like PlanetLab, Edge-Net uses the public Internet for both user-slice and intra-slice communication. Due to the chronic scarcity of IPv4 addresses and the continued unreliability of IPv6 access and services, Edge-Net makes heavy use of DNS services as a means of both conserving port space and providing location-independence for localized services. Thus, architectural design of the namespace is particularly important. Neither PlanetLab nor GENI used their respective top-level namespaces for remote nodes, forcing experimenters to maintain often-changing configuration files. In contrast, Emulab [12], [24] adopted an experiment-centric namespace, where nodes were named `<nodename>.<experimentname>.<projectname>.emulab.net`, permitting experimenters to use name-based addressing in configuration files and scripts. We adopt a modified Emulab approach.

A K8s/Edge-Net Namespace is equivalent to an Emulab or GENI Project; names of slices and slivers can be chosen arbitrarily within a Namespace by the experimenter, just as they can on Emulab. At present, the Namespace name is chosen by Edge-Net, and is a mild transformation of the user's email address; it is our intent to permit experimenters to choose a namespace name freely in future.

There are two domains for Edge-Net: `edge-net.io` and `edge-net.org`. `edge-net.org` is used for administration: the portal, documentation and collateral. `edge-net.io` is used for sites, slices, and slice services such as DNS.

A sitename or a slicename will be chosen arbitrarily by the site or slice, respectively. A name is chosen by the object creator at the time of its creation, subject to availability and a mild test for suitability. Edge-Net administrators will regularly review site names for appropriateness. Sites will be under the subdomain `sites.edge-net.io`; slices under the subdomain `slices.edge-net.io`. Each VM will be numbered within a site, with `vm0.<sitename>.sites.edge-net.io` being an alias for `<sitename>.sites.edge-net.io`.

Each K8s daemon set or service will choose names arbitrarily for each sliver within its namespace. Each sliver will be named `<name>.<namespace>.slices.edge-net.io`, where `<name>` is chosen arbitrarily within the `namespace`.

B. DNS Resolution and Resource Contention

The Edge-Net Naming Architecture serves to protect user slices from contention and to offer a domain of trust and tenancy. Port multiplexing is achieved through the use of an `nginx` reverse proxy on each worker node. Resource contention is on a per-Pod basis on each node, enforced by existing K8s tools. We adopt a fair-share system across namespaces.

VII. Using Edge-Net

The goal in user experience design in Edge-Net is to offer as transparent an overlay on Kubernetes as possible; an experimenter should have the impression "it's just Kubernetes", and should be able to rely on the wealth of available Kubernetes tutorials, documentation, and services. The result is that the goal of the Edge-Net portal is to send the user to a Kubernetes dashboard as quickly as possible; the role of the Edge-Net portal is simply to manage namespaces and issue credentials to users.

The user flow through the portal is shown in Figure 1. The yellow boxes are non-portal maintained entities which provide services to the portal. The user logs in through an Identity Provider, which delivers the user's email address as the userid. If the user has not yet agreed to Edge-Net's Acceptable Use Policy, he is directed to a page to use and agree to the policy. Once he has agreed, Edge-Net's administrators approve his account and he is assigned a namespace. He is then directed to the Namespace Dashboard, where he can download a configuration file to use with Kubernetes. At this

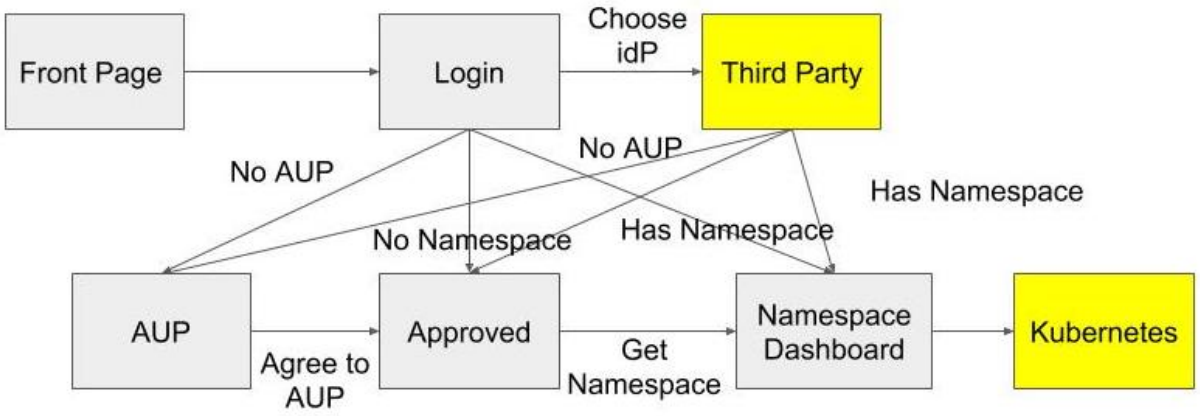

Fig. 1. User Flow Through the Edge-Net Portal

point he is directed to a standard Kubernetes dashboard, where he presents the credential and selects his namespace. He can then use the Kubernetes dashboard or the `kubectl` command line tool to work with his namespace; he is not required to use the Edge-Net portal again. The Edge-Net portal is simply an entity that generates namespaces and holds configuration files for namespaces.

This gives the portal/head node architecture given in Figure 2. The Kubernetes Head Node manages the cluster; aside from reports of which nodes are in the cluster, and their status, this is transparent to the portal. The user interacts with the portal to get credential and namespace information, but then works directly with the Kubernetes dashboard, shown in Figure 3 to allocate and manage daemon sets and services.

A. Adding a Node to Edge-Net

Adding a node to Edge-Net is quite similar to adding a node to its predecessor, `planet-ignite`. At present, there is a shell script which installs the Edge-Net software on an Ubuntu 14.04 or Ubuntu 16.04 VM; the site administrator downloads the script and runs it on a convenient VM. The script then prompts for the site name, downloads the Edge-Net software stack (a lightly modified version of the standard K8s worker node) and then transmits a join request to the Kubernetes head node. The head node tests for routability to the new worker node, and if the node can be reached it's added to the cluster and a DNS name is chosen for it.

VIII. A Federatable Architecture

The Portal/Head Node architecture of Edge-Net offers an attractive prospect for federating similar systems, with a global portal whose sole function is to manage configurations for the various namespaces. Federation has been a ubiquitous feature of distributed edge systems, for a number of reasons:

1) Edge clouds operate across a number of legal and administrative domains, and the usage of the systems are subject to differing constraints in each domain. The Acceptable Use Policy of a system such as PlanetLab, PlanetLab Europe, and Edge-Net are largely driven by legal constraints, and these vary across domains.
2) Some edge clouds have significant resource restrictions, and thus need to restrict access to a small subset of users.
3) Specific edge clouds access resources which should only be accessed by selected groups of users; for example, an IoT deployment

The architecture given here lends itself to such a federation, where each cluster maintains a separate set of namespaces and a subdomain name, but the global portal acts to maintain identity (or relationships with IdPs) and persistent storage for certificates. This federation is shown in Figure 4, and shows the advantage of a clean separation of concerns between portal and head node: the portal manages users and the head node clusters. This permits cluster administrators to grant access to users on their own clusters.

IX. Status, Conclusions, and Future Work

As of May 17, 2018, Edge-Net is currently deployed across 20 nodes: 16 on the GENI infrastructure and four on Canada's SAVI infrastructure. The EdgeNet portal is up and accepting users. Scalability and node addition tests are currently underway.

Many of the principles behind traditional cloud computing can also be applied to edge computing. Edge-Net does that while emphasizing ubiquity and low-latency access to the real-world IoT devices and end users who are spread out over broad areas. While such systems were served in the past by dedicated, real-time infrastructure, cloud principles suggest that elastic, shared infrastructure is often a more cost-effective, efficient, and higher reliability way to deliver such services. Edge-Net is intended as a lightweight research infrastructure which can spread broadly and quickly. A host of distributed system and locality-based applications and services will find it meets their needs. Edge-Net may evolve into a reference

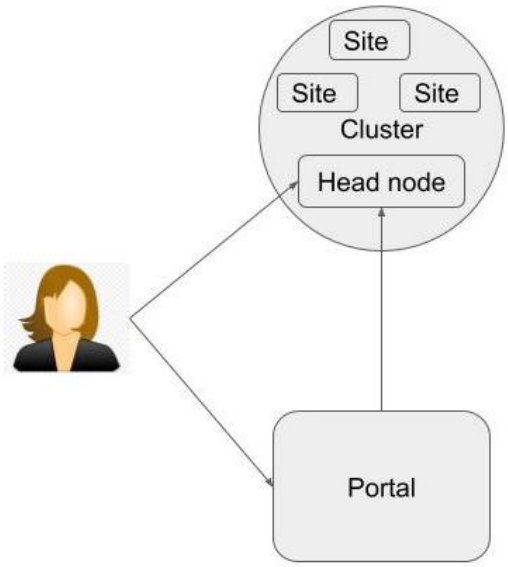

Fig. 2. Portal/Head Node Architecture

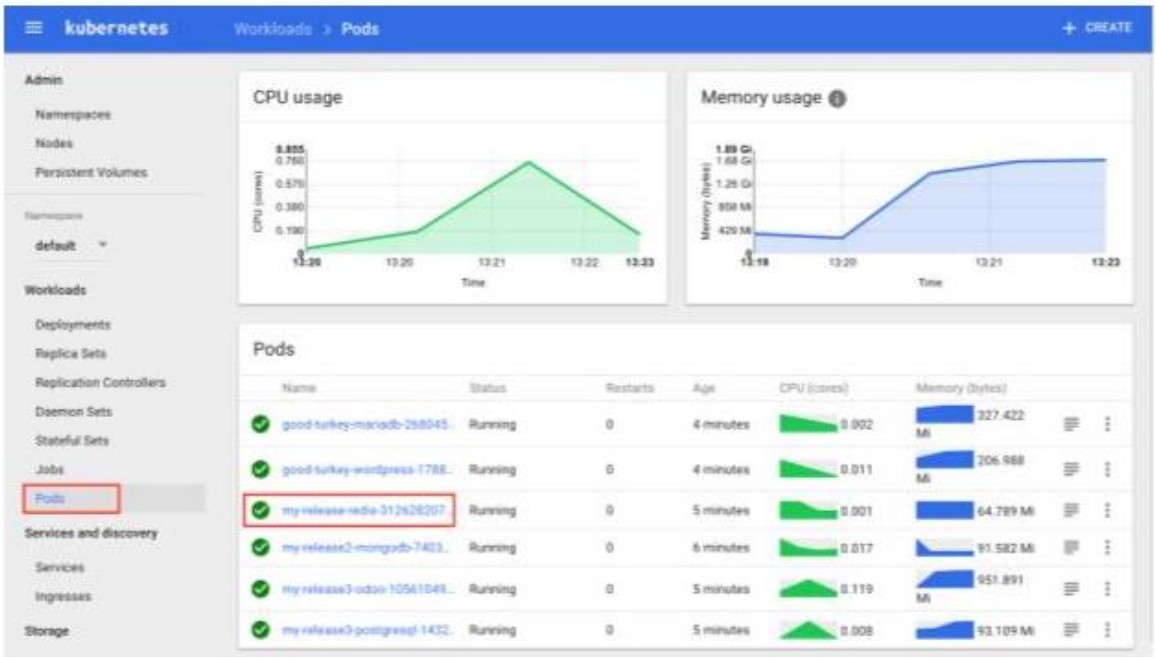

Fig. 3. Edge-Net Head Node Dashboard

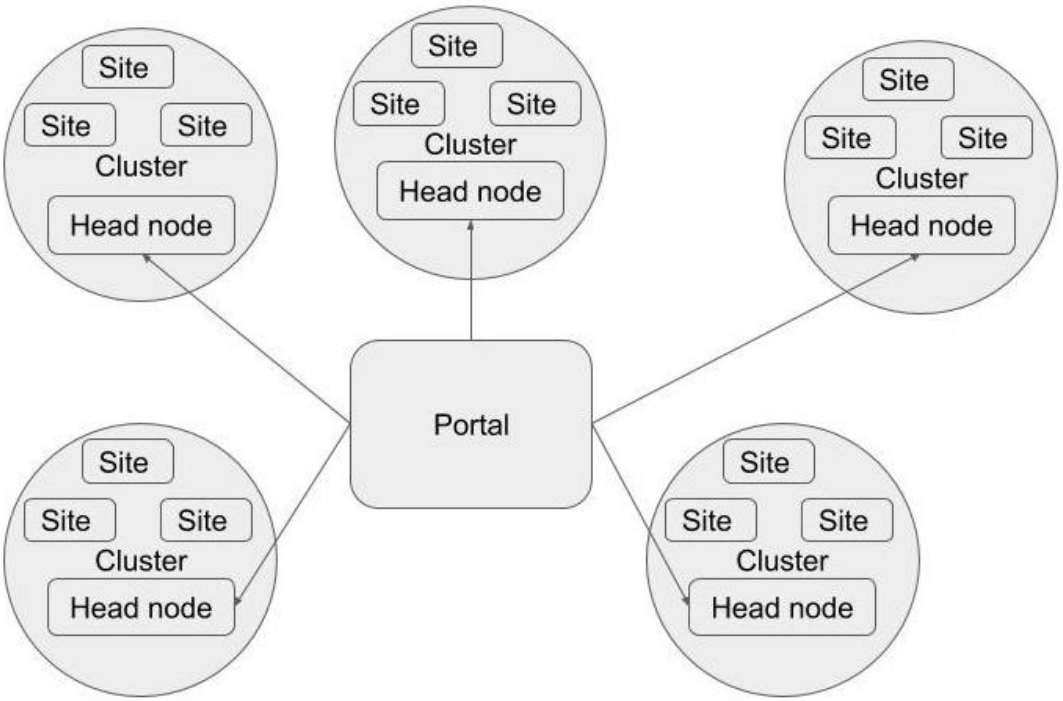

Fig. 4. Edge-Net Federation Architecture

implementation for future ubiquitous clouds-at-the-extreme-edge infrastructure.

Edge-Net is still in its infancy, with many planned features still to deploy. Many of these we have mentioned above: user-chosen namespaces and user administration of his own namespace, to permit multiple users to share a namespace as multiple users share Projects on GENI and Emulab; multiple head nodes; monitoring mechanisms such as PlanetFlow on PlanetLab; and certifications for trusted worker nodes. In addition, Edge-Net has adopted PlanetLab's microkernel approach to distributed systems; we encourage users to offer foundational services in slices, for the use of other services and applications running on Edge-Net.

ACKNOWLEDGEMENTS

A worldwide consortium is working with us to deploy and govern Edge-Net, and we thank our collaborators, particularly at JGN-X, PlanetLab, GENI, SAVI, and PlanetLab Europe. A full list will appear in the final camera-ready version of the paper, if accepted. This research has been sponsored by the National Science Foundation.

REFERENCES

[1] S. B. Ahsan and I. Gupta. The cat theorem and performance of transactional distributed systems. In *Proc. ACM PODC Workshop on Distributed Cloud Computing (DCC)*, 2016.

[2] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 2002.

[3] A. Bavier, J. Chen, J. Mambretti, R. McGeer, S. McGeer, J. Nelson, P. O'Connell, G. Ricart, S. Tredger, and Y. Coady. The geni experiment engine. In *Teletraffic Congress (ITC), 2014 26th International*, pages 1–6. IEEE, 2014.

[4] A. Bavier, J. Chen, J. Mambretti, R. McGeer, S. McGeer, J. Nelson, P. O'Connell, G. Ricart, S. Tredger, and Y. Coady. The GENI Experiment Engine. In *Proceedings of TRIDENTCOM'15*, 2015.

[5] A. Bavier and R. McGeer. The geni experiment engine. In *The GENI Book*, chapter 11. Springer-Verlag, New York, 2016.

[6] A. Bavier, R. McGeer, and G. Ricart. Planetignite: A self-assembling, lightweight, infrastructure-as-a-service edge cloud. In *International Teletraffic Congress*, 2016.

[7] S. Bhojwani, M. Hemmings, D. Ingalls, R. Krahn, J. Lincke, R. McGeer, M. Rder, , Y. Coady, and U. Stege. The ignite distributed collaborative scientific visualization system. In *Proceedings of IEEE CloudCom*, 2015.

[8] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson. Seattle: a platform for educational cloud computing. In *Proceedings of the 40th ACM technical symposium on Computer science education*, SIGCSE '09, pages 111–115, New York, NY, USA, 2009. ACM.

[9] R. Clark. Personal Communication.

[10] M. Hemmings, D. Ingalls, R. Krahn, R. McGeer, M. Rder, and U. Stege. Livetalk: A framework for collaborative browser-based replicated-computation applications. In *Proceedings of the International Teletraffic Congress*, 2016.

[11] M. Hemmings, R. Krahn, D. Lary, R. McGeer, M. Roeder, and G. Ricart. The ignite distributed collaborative scientific visualization system. In *The GENI Book*, chapter 19. Springer-Verlag, New York, 2016.

[12] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau. Large-scale virtualization in the emulab network testbed. In *USENIX Annual Technical Conference*, pages 113–128, 2008.

[13] B. Krebs. Hacked cameras, dvrs powered todays massive internet outage. https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/, October 2016.

[14] D. K. Krishnappa, D. Irwin, E. Lyons, and M. Zink. Cloudcast: Cloud computing for short-term mobile weather forecasts. In *2012 IEEE 31st*

International Performance Computing and Communications Conference (IPCCC), pages 61–70. IEEE, 2012.

[15] D. K. Krishnappa, D. Irwin, E. Lyons, and M. Zink. Cloudcast: Cloud computing for short-term weather forecasts. *Computing in Science & Engineering*, 15(4):30–37, 2013.

[16] A. Leon-Garcia and H. Bannazadeh. Savi testbed for applications on software-defined infrastructure. In *The GENI Book*, chapter 22. Springer-Verlag, New York, 2016.

[17] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto. Bigchaindb: A scalable blockchain database, 2016.

[18] R. McGeer, M. Berman, C. Elliott, and R. Ricci, editors. *The GENI Book*. Springer International Publishing, 2016.

[19] P. Mueller and S. Fischer. Europe's mission in next-generation networking with special emphasis on the german-lab project. In *The GENI Book*, chapter 21. Springer-Verlag, New York, 2016.

[20] A. Nakao and K. Yamada. Vnode and jgn-x. In *The GENI Book*, chapter 23. Springer-Verlag, New York, 2016.

[21] D. Perino and M. Varvello. A reality check for content centric networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pages 44–49. ACM, 2011.

[22] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir. Experiences building planetlab. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 351–366. USENIX Association, 2006.

[23] T. Rakotoarivelo, G. Jourjon, O. Mehani, M. Ott, and M. Zink. Walk through the geni experiment cycle. In *The GENI Book*, chapter 17. Springer-Verlag, New York, 2016.

[24] R. Ricci. Emulab. In *The GENI Book*, chapter 2. Springer-Verlag, New York, 2016.

[25] N. Riga, S. Edwards, and V. Thomas. The experimenter's view of geni. In *The GENI Book*, chapter 15. Springer-Verlag, New York, 2016.

[26] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 18:1–18:17, New York, NY, USA, 2015. ACM.

[27] S. Wagner. Dispersed computing. Technical Report DARPA-BAA-16-41, DARPA, July 2016.

[28] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. Technical Report NDN-0001, Parc, 2010.