Augmenting Cloud Architectures to Support Decentralized Applications

Michael Coughlin, Kelly Kaoudis, Eric Keller
University of Colorado, Boulder

***Abstract*—Despite the benefits of decentralized applications in terms of resilience and privacy, the overwhelming majority of applications with mainstream adoption are provided in a centralized manner. We argue that this is due to the direct benefits to the developer that centralization provides in terms of performance, monetization, and deployability. In this paper we introduce a new model, untrusted delegation, which joins the simplified deployment model of centralization with the benefits of decentralization. In this model, we decouple administrative ownership from administrative management, and leverage the existence of either a private cloud infrastructure, or a public cloud provider that acts as a neutral third party, that is augmented to support decentralization. Our initial prototype integrates with the Digital Ocean API and as a proof-of-concept, we can deploy Tor relay nodes with users only needing to sign up for a Digital Ocean account.**

I. Introduction

Decentralized applications consist of a distributed set of nodes and administrative domains that together cooperate to provide a service. As there is more than one administrative domain (application software and infrastructure is controlled by different parties), user privacy can be enhanced, since there is no single entity that has complete control and visibility over users' data and actions. The distribution of nodes allows for application performance and reliability to be enhanced, as there is an inherent diversity in execution environments and administrative control (*e.g.,* a single user cannot shut down the service with a router misconfiguration, and network communication can be optimized as there is no central bottleneck).

Because of these benefits, a number of these types of systems have already been implemented, including secure communication and idea-sharing (such as Tor [1], Freenet [2]), cryptocurrencies (Bitcoin [3]), distributed hash table based storage [4], [5], [6], web application platforms [7], web search [8], email [9], web services [10], [11], social networks [12], and a general call to re-decentralize the web [13].

Yet, the centralized (client-server) model has largely prevailed, especially for mainstream applications – few decentralized applications have gained mainstream adoption. The reasons for this, we believe, is due to the properties of centralization, which typically have a clear benefit to the application developer. In particular:

- Simple to deploy. The administrator/developer of a centralized application can acquire the computing resources needed, and deploy their software as they see fit. There is no need for coordination among many parties or reliance on users running software on their own device.
- Able to be monetized. The administrator/developer of a centralized application can put up gateways to its application in order to charge for the service, or leverage private user data to serve personalized ads. This ability to monetize is important as it provides the incentive to develop applications that people want to use.
- Provide better performance. With control over both the software and the infrastructure, the administrator/developer of a centralized application can optimize performance, such as by reducing latency through leveraging advances in data center networks.

We ask whether we can obtain similar benefits for decentralized applications. That is, can a developer create an application that can be run in a decentralized manner (that is not in their control), and where the application can have similar performance to running it in a centralized manner, be monetizable by the application developer, and still be easy to deploy?

The key lies in leveraging the cloud, which we argue can be augmented to support decentralized applications.

In this paper we describe a new cloud-supported approach we call *untrusted delegation*, which enables decentralized applications to be easier to deploy. In this model, a central entity is given credentials to deploy an application to a cloud service on a user's behalf, but is not trusted with these credentials, as all of its actions are logged and the credentials are both time-limited and only scoped to allows for the deployment of the application, and are enforced by the cloud service. The high-level idea for our untrusted delegation model is for users to sign up for an account from some cloud provider (increasing the number of administrative domains, thus increasing the effectiveness of decentralization), and delegate temporary management to an untrusted developer (centralizing management, simplifying deployment). The cloud provider facilitates this by enforcing access and enabling the users to check whether the application software is indeed out of the developers control and visibility.

In the remainder of the paper, we discuss properties and shortcomings of existing application models and introduce the untrusted delegation model (Section II), discuss the associated challenges and how we can address them in our architecture (Section III), and conclude (Section IV).

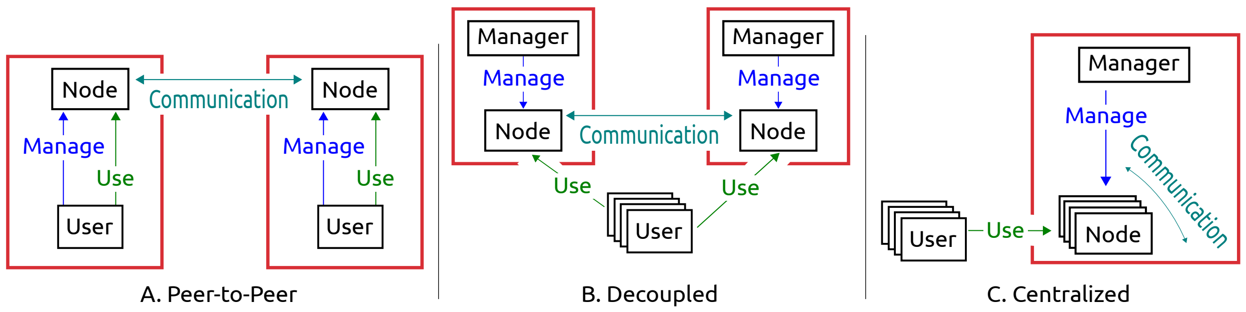

Fig. 1: **Existing Models** Red boxes indicate the party that owns the application node.

II. Application Models

Broadly speaking, applications are built atop a collection of distributed nodes which provide functionality and communicate with one another. In this section, we overview existing models and describe our untrusted delegation model.

A. Existing Models

In **peer-to-peer** (P2P) applications (Figure 1(A)), such as Freenet [2], users must administer their own nodes, which, networked together, comprise the application. With this, there is potential for increased resistance to faults, since there is no central bottleneck to limit individual connections or act as a single point of failure. These benefits can scale as the number of users (and thus nodes) increases. The collaborative ownership and management of these applications also provides some possible privacy benefits to the users, as there is no single entity which has complete visibility over the users' actions or control over their data. This distributed ownership, however, can also be a problem, as the application relies on users to provide hardware. Without an incentive from the application to keep these nodes online (such as a monetary award, as is done by Bitcoin), these applications will experience a great deal of churn in availability of nodes, as users will deactivate their nodes when they are not using the application.

As illustrated in Figure 1(B), the **decoupled model**, as used by Tor [1], separates the ordinary user from the node administrator. This introduces a new entity, the manager, who provisions and maintains nodes (managers will likely still be users, but the function is decoupled). This distributed nature allows for decoupled applications to inherit the benefits of P2P applications, but with more reliability. The downside is that entrusting all administrative responsibility to a select group of volunteers limits network growth and reduces user privacy, as user data is now controlled by this group (users still have some options in this model though, as they could choose to volunteer their own resources which may provide some control over their data). Just as in P2P applications, decoupled applications see increased returns with larger number of nodes, but in the decoupled case, resource contribution is limited to the managers, thus restricting the set of possible administrative domains.

The **centralized model** (Figure 1(C)), as is used by many cloud services like Google or Dropbox, takes the decoupled model to an extreme. A single manager, and therefore single administrative domain, provisions and maintains application nodes. This model is simpler to deploy and control compared to decentralized models (P2P and decoupled), as this responsibility is consolidated into the domain of a single manager, and performance is easier to optimize as the manager has direct control over all of the components (this model can still be implemented as a distributed system; we refer to centralization as a single controlling administrative domain). This model is also easier to monetize, as there is only a single point of access to the application, meaning that paid gateways, advertising or other monetization systems are easier to deploy and enforce. However, user privacy is decreased further, as now all of the user's data is controlled by a single domain, with no path for the user to control it other than what is provided by the application.

Because of these benefits, the centralized model has become the most popular application model for internet applications today. However, the drawbacks of this architecture are a direct result of its centralization. Any centrally controlled application has a single point of failure. A single malicious adversary or a power outage, hardware fault, or bug can bring down a centralized application. Second, as the application is controlled by a single entity, users must trust this entity, as it has complete application visibility and control over user data.

Current decentralized applications (both P2P and decoupled) have some advantages, such as increased fault resistance and potential for privacy, compared to centralized applications, but are impaired by issues with deployability, monetization, and performance (where centralized applications excel). We present a new model, untrusted delegation, which combines the benefits of distribution found in decentralized systems with the ease of deployment of centralized systems. Untrusted delegation allows for nodes to be owned by users, as in the P2P model, but hosted remotely, as is possible in the decoupled model. Nodes in a system following this model are centrally managed, but this management is temporary and not trusted, which is enforced by the underlying hosting platform that is used to provide the node resources.

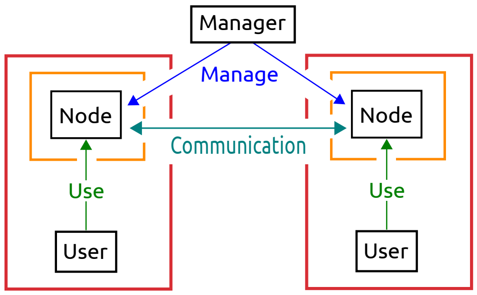

Fig. 2: **Untrusted Delegation.** Red boxes indicate ownership of application nodes and orange boxes indicate hosting of nodes for the owner.

B. *Untrusted Delegation*

Ownership: The distinguishing characteristic of our new model is its decoupling of resource ownership from resource management. In both the decoupled model and P2P model, the ownership was coupled to the managers (in the case of P2P, the users were the managers). In the untrusted delegation model, the ownership remains with the users, who are not the managers, with a separate entity performing the management (likely the developer). Finally, we allow (in fact rely on) application nodes to be hosted remotely, with ownership referring to control of a virtual resource.

Example Usage: Figure 2 illustrates our model, which can be compared to the existing models in Figure 1. A user with a funded cloud hosting account visits the central manager and delegates temporary account access. The manager uses this temporary access to deploy and configure an application node. These freshly provisioned nodes then can join the application and communicate with other nodes. As is shown by the red and orange boxes, the node resources are still owned by the user, but are hosted remotely.

Trust Model: We assume the same trust model as in other decentralized applications. This model trusts a collection of nodes as a whole while distrusting individual nodes and node resources. However, we deviate from this model slightly by introducing the central manager. The manager is untrusted, despite having temporary access to account credentials, as is discussed further in Section III. We also introduce the use of cloud providers, but we assume a diverse selection and use of cloud providers that can be trusted as a whole. Similarly, decentralized applications do not trust individual nodes, but require a wide assortment of hosting domains and underlying hardware to trust the collection of nodes as a whole. In addition, users must also trust the entity hosting their data, which can manifest in a number of different ways.

III. Untrusted Delegation Architecture

Shown in Figures 3 and 4 are the main components required for realizing the untrusted delegation model. Here, we discuss the challenges to implementing the untrusted delegation model and how these components address these challenges.

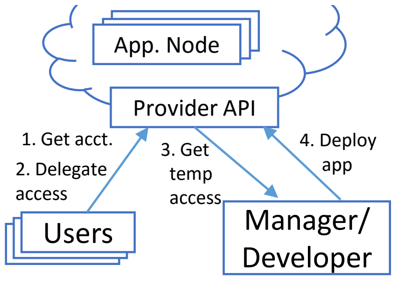

Fig. 3: **Delegation.** Users with funded cloud service accounts delegate access to the centralized manager (or application developer) using OAuth, which is enforced by the cloud provider's API. The manager/developer uses the resulting temporary access to deploy an application instance to the user's account.

A. *Temporary Delegation*

Challenge 1 – Central Management, Decentralized Ownership

The goal is for users to contribute resources (to achieve node growth proportional to user growth), retain administrative control (to increase the effectiveness of decentralization with a large number of administrative domains), but at the same time make use of a central manager (to increase deployability), and in some cases, also retain control over data access (to increase privacy). The key challenge is achieving central management with decentralized ownership. Restated, a central manager must access the user-owned resources without being considered a centralized administrative domain. This is especially important for data privacy, as users need to be able to control access to their data while still providing the central manager with the necessary access to deploy the application.

Cloud Providers Enforce Temporary Delegation

We use a delegation mechanism to enable a central manager to access user-owned resources without ownership, built using the OAuth protocol [14]. The cloud provider, a neutral third-party, can then enforce the restricted capabilities, and time-limits on access – without that third party, we would not have the ability to rescind access once it was granted. As illustrated in Figure 3, the user will instruct the cloud provider to give the central manager temporary control over the system using an OAuth authorization (the cloud provider prompts the user to allow the central manager access to their account before distributing credentials, similar to other OAuth applications). The central manager uses the delegated access to deploy an application node to a user's cloud account using the cloud provider's API (most likely as a virtual machine (VM)). As users retain administrative control over the deployed nodes, they can control the operation of the node by disabling it or removing network access.

B. *Community Verification*

Challenge 2 – Ensuring Application Correctness

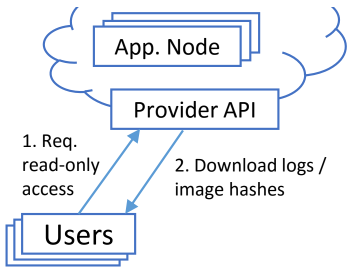

Fig. 4: **Verification.** Verifying identities, such as other users or a community entity, can be delegated read-only access to an account that is running an application instance. The cloud provider's API will then provide the needed information to verify that any running instances are running the correct software, such as hashes of the deployed VM images or logs of the instance's deployment.

By introducing a delegation system, we bridge the gap between ownership and administration. By leveraging the cloud provider to restrict access (both in capabilities and in time), we retain the principles of decentralization. What is not ensured is the correctness of the central manager's actions. We must be able to confirm correctness of nodes deployed by the manager, and guarantee that the manager only does exactly what is requested by the user with their account credentials, thus enabling the manager to be untrusted (*e.g.,* ensure that the manager does not install additional software that can access the user's data).

Cloud Providers Support Community Verification

The trust model of decentralized applications requires trust in the community as a whole. Individual nodes are untrusted, but the application as a whole is trusted. At least a certain number of nodes are assumed trustworthy. We utilize community verification to ensure that a deployed node is running the published software from the developer (we do not attempt to audit software for implementation correctness). The cloud provider acts as a neutral party to facilitate verification of deployed nodes. Otherwise, the central manager could utilize hiding techniques as it would completely control what software is installed and configured. As shown in Figure 4, the cloud provider allows for read-only access to verification information (*e.g.,* hashes of running VM images or hypervisor logs) using another OAuth exchange (similar to the central manager).

C. Proof of Concept: Deploying Tor Relays

To demonstrate our untrusted delegation system, we have built a proof-of-concept system to deploy Tor relays to user's cloud account (using Digital Ocean as the cloud provider). Tor is a representative decentralized application that, as highlighted by its developers, increases its security with more relay nodes [**?**]. Using Ruby API bindings, our prototype deploys a VM using an existing Ubuntu 14.04 image provided by Digital Ocean. CloudInit and Puppet are then used to install the Tor software and start the application. The Sinatra web application provides the web interface to users and manages the OAuth authorization and exchange. We also performed a survey of existing cloud service providers and found that most services have the capability to support our model, but very few exposed these capabilities in their APIs, and none of them supported all of our requirements (Digital Ocean supports enough functionality for us to create a prototype).

IV. Conclusion and Future Work

We have presented a new model for internet applications combining the advantages of decentralized models with the management and control advantages of centralized models. Our model of untrusted delegation allows for decentralized applications to receive contribution from all potential users, making the deployment of these applications much more feasible. With this architecture we provide a method for decentralized applications to achieve widespread adoption,allowing for the internet to progress towards a decentralized model. Going forward, we intend to improve our initial prototype to support community verification and to create a system that can support general applications. We also would like to explore more capabilities of hypervisors and cloud services to make the system more secure and easier to use.

Acknowledgments: This work was supported in part by NSF SaTC grant number 1406192.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document, Tech. Rep., 2004.
[2] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley, "Protecting free expression online with freenet," *Internet Computing, IEEE*, vol. 6, no. 1, pp. 40–49, 2002.
[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Consulted*, vol. 1, no. 2012, p. 28, 2008.
[4] C. T. Lesniewski-Laas, "Design and applications of a secure and decentralized distributed hash table," Ph.D. dissertation, Massachusetts Institute of Technology, 2010.
[5] A. Haeberlen, A. Mislove, and P. Druschel, "Glacier: Highly durable, decentralized storage despite massive correlated failures," in *Proc. Symposium on Networked Systems Design and Implementation (NSDI)*, Berkeley, CA, USA, 2005.
[6] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and B. Zhao, "OceanStore: An Architecture for Global-scale Persistent Storage," in *Proc. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.
[7] "ZeroNet: Decentralized websites using Bitcoin crypto and the BitTorrent network," http://zeronet.io/.
[8] "YaCy: Decentralized web search," http://yacy.net.
[9] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type iii anonymous remailer protocol," in *Security and Privacy, 2003. Proceedings. 2003 Symposium on*. IEEE, 2003.
[10] F. Glaser and L. Bezzenberger, "Beyond cryptocurrencies-a taxonomy of decentralized consensus systems," in *Twenty-Third European Conference on Information Systems (ECIS), Münster, Germany*, 2015.
[11] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*. IEEE, 1996.
[12] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: An online social network with user-defined privacy," in *Proc. ACM SIGCOMM*, 2009.
[13] L. CLARK, "Tim berners-lee: we need to re-decentralise the web," http://www.wired.co.uk/news/archive/2014-02/06/tim-berners-lee-reclaim-the-web, Feb. 2014.
[14] D. Hardt, "(The OAuth 2.0 authorization framework (IETF RFC 6749)," 2012.