

# Towards Modeling Application Execution Platforms for Delay-Tolerant Networks

Anastasia Vöhringer  
Dresden University of Technology  
Faculty of Computer Science  
Dresden, Germany  
anastasia\_klara.voehringer@tu-dresden.de

Marius Feldmann  
D3TN GmbH  
Dresden, Germany  
marius.feldmann@d3tn.com

Iris Braun  
Dresden University of Technology  
Faculty of Computer Science  
Dresden, Germany  
iris.braun@tu-dresden.de

**Abstract**—In this paper, we present a first draft of MAEPOS (Meta-model for Application Execution Platforms Operated in Space), a meta-model for a systematic approach to design application execution platforms for Delay- and Disruption-Tolerant Networking (DTN). MAEPOS is a technology-independent description of an application execution platform for flexible instantiation with specific technologies. The feasibility of MAEPOS has been confirmed using a basic scenario in which the meta-model is instantiated for a DTN-supported platform operated in lunar orbit. The operation of a platform via DTN involves specific requirements, which MAEPOS conceptually implements. The intermediate results presented in the following form the basis for further work in which MAEPOS will be extended to design secure and reliable application execution platforms. Therefore, we provide a brief overview of the ideas for the automated mapping of MAEPOS to an application context and for a quantitative evaluation of MAEPOS.

**Index Terms**—Delay-Tolerant Networking, Disruption-Tolerant Networking, Space Cloud, Meta-model.

## I. INTRODUCTION

During the past three decades, solid technical solutions have been specified, implemented, and deployed to make networking possible in challenging environments with long link delays and intermittent connectivity. The relevant domain of Delay- and Disruption-Tolerant Networking (DTN) has produced several protocol standards, most significantly the Bundle Protocol (see IETF RFC 5050 and 9171), which has been used in several production-ready DTN protocol implementations such as the Interplanetary Overlay Network (ION), High-Rate Delay Tolerant Networking (HDTN), or  $\mu$ D3TN.

Although the DTN community has achieved many valuable results, one specific field has not been in the focus so far: The integration of applications into DTNs. If an application developer would like to create a distributed application intended to be operated within a DTN, several manual tasks have to be performed that require specific knowledge of the DTN domain in general and of at least one DTN protocol implementation specifically. This is totally different from how distributed applications operating in the Internet are developed nowadays. Due to the use of application execution environments such as

The intermediate results presented in this paper have been created in the context of the DARKSOL project. This project is co-funded by the European Union and co-financed from tax revenues on the basis of the budget adopted by Saxon State Parliament.

Kubernetes and approaches integrating development and operation processes with support from unified tooling (typically named *DevOps*), development focus can be fully directed to the application itself. Thus, developers do not have to care for infrastructure- or network-related aspects.

As it is desirable to achieve a similar level of abstraction when creating distributed applications that should be operated within a DTN such as the Solar System Internet, the project DARKSOL (<https://darksol.cloud>) aims to provide appropriate solutions with equivalent characteristics suitable for DTNs.

As part of the DARKSOL project, this work addresses the essential requirement of technology independence for the solutions developed. These should not be dependent on a single application execution platform such as Kubernetes, but should be generic enough so that they can be mapped to various existing or emerging platform technologies. By this, the project results should be applicable to various technological contexts. This should not only simplify the design of DTN-capable application execution platforms in general, but should specifically assist in addressing cross-cutting concerns such as DTN connectivity, security, or monitoring. It is intended to achieve this requirement by defining a meta-model that is capable of describing different instances of application execution platforms for delay-tolerant networks. In this paper, we present preliminary results on the way to an expressive meta-model focusing on defining application execution platforms for DTNs.

In a first step, a use case for applying the meta-model named MAEPOS (Meta-model for Application Execution Platforms Operated in Space) is described in section II to further motivate the presented work. Additionally, essential requirements for this meta-model are discussed and related work is summarized in section III. The current state of the defined meta-model is presented in section IV. A first basic validation of this meta-model is presented in section V by modeling a Kubernetes-based application execution platform designed for the Solar System Internet. The paper concludes with a summary and an outlook on future work in section VI.

## II. USE CASE AND REQUIREMENTS

### A. Use Case for the Defined Meta-model

In order to sketch a vision behind the meta-model, a use case is presented in the following, motivating the usage.

The new space company *ACME* is used to operate different applications in cloud infrastructures. Specifically, Kubernetes is used as a basis for application deployment and operations in order to abstract from infrastructure details.

Together with two partner companies, *ACME* has won a contract from the National Space Agency to develop and deploy two satellites in lunar orbit. The aim of these satellites is to optically monitor the lunar surface and apply image analysis algorithms to the created footage in a two-stage process. Rapid initial analysis should be performed directly on the satellites. A more in-depth analysis requiring extensive processing power shall be performed in a data center on earth. Thus, the images are processed on one of the satellites using a specific software provided by *ACME* and additionally transferred to Earth for further analysis. The second satellite provides communication and support services and can take over the workload of the first satellite if it fails. The satellites use the Solar System Internet communication services for transmitting data using DTN protocols to Earth.

*ACME* is responsible not only for developing the analysis software, but also for designing and creating a deployment and operations approach. The operations approach must include processes for providing up-to-date versions of the analysis software. As *ACME* is using Kubernetes in other contexts, developers propose to use Kubernetes also as an application execution platform on lunar satellites. As the hardware used for the satellites is powerful enough to execute a Kubernetes instance and as the resulting overhead is acceptable, the project partners as well as the National Space Agency accept this proposal.

After the decision has been made in the project, *ACME*'s developers start to investigate options to operate a Kubernetes cluster in space via DTN protocols. During their analysis of the state of the art, they discover *MAEPOS*, which they understand as a tool to model the DTN-enabled Kubernetes platform they intend to create and deploy. Using *MAEPOS*, they rapidly design a model of the desired Kubernetes-based execution environment. The model not only covers the base functionality including communication services, but also addresses cross-cutting concerns such as confidential and authenticated data transfer between the satellites and the ground stations, as well as availability monitoring.

Based on a set of standard components taken from a software catalog associated with several elements of the model, such as the implementation of a DTN protocol, monitoring tooling, and a time synchronization service, many parts of the platform are automatically integrated into a fundamentally working solution. *ACME* only has to make minor adaptations to the platform created in a model-driven manner to achieve a ready-for-production solution. The platform was deployed on lunar satellites and successfully used. During mission

operations, it cares for stable operations of the image analysis software which is updated several times during the mission.

After a use case for applying the meta-model has been sketched, a set of core requirements for this model will be discussed in the next section.

### B. Requirements

Based on the use case of *ACME* described above, several fundamental requirements can be identified for the *MAEPOS* meta-model. The proposed meta-model is designed to be technology-independent in order to provide maximum flexibility in the design and development of an instance. To reflect realistic cluster architectures, the model should enable the logical and physical distribution of the DTN-supported platform across multiple nodes connected by low-latency links. Within such a distributed platform, the applications should optionally be portable, configurable, and scalable.

Another important requirement is support for multi-tenancy, including the isolation of resources, data, and processes between different tenants within the DTN-supported platform. In addition, the meta-model must integrate DTN protocols for communication purposes to ensure that DTN functionality can be represented at every layer and node of the architecture. Given the possibility of interrupted connectivity – especially in space-based or remote environments – the meta-model must also support autonomous life-cycle management (LCM) so that the DTN supported platform can operate, adapt, and recover independently when administrative access is not available.

Finally, extensibility is a key design principle: the meta-model should be extensible to integrate cross-cutting concerns in the DTN-supported platform such as security, monitoring, encryption, and other operational aspects that are essential for robust and maintainable deployments.

## III. RELATED WORK

In the domain of modeling application execution platforms, there are various contributions with distinct areas of focus. In [1], the authors propose a structured meta-model that formally describes central elements of cloud infrastructures to make automated analysis and optimization of cloud resources possible. The model also incorporates variability points and optimization goals. The *Aeolus* project [2] presents an automated approach and a tool-chain to provision modern cloud environments with customized requirements and optimization targets. The approach has been theoretically and practically validated in industrial scenarios to ensure correctness and completeness. Various modeling approaches and languages exist. The systematic review in [3] offers a comprehensive comparison of existing cloud modeling languages, identifying *TOSCA* (Topology and Orchestration Specification for Cloud Applications) as one of the most widely adopted. The *TOSCA* standard [4] enables platform-independent modeling and management of cloud applications, capturing their topology, behavior, and dependencies. In [5], a methodical approach is presented to translate UML-based cloud designs

into TOSCA models by extending UML with the Cloud Application Modeling Language (CAML) and applying the CAML2TOSCA transformation tool.

Model-based abstractions for cloud architectures in delay-tolerant networking (DTN) environments remain largely underdeveloped. Existing contributions are mostly architectural proposals. For example, [6] explores how application execution via DTN can be enhanced through cloud technologies and containerization (e.g., Docker). The paper describes an experimental environment in which the High-Rate DTN (HDTN) system is deployed, integrated, and evaluated using container platforms such as Kubernetes and AWS.

The integration of security concepts into cloud modeling is a subject of ongoing research. In [7], a theoretical multi-layered security architecture is presented that adapts classical defense-in-depth principles to cloud environments. In [8], a higher level of abstraction is introduced to model security aspects in cloud applications, which proposes a novel domain-specific language (DSL) for model-driven specification and validation of security requirements. The DSL is compatible with standards such as CAML, TOSCA, and WS-Agreement and supports automated base security modeling based on the Cloud Controls Matrix and EU policy recommendations. These conceptual approaches may serve as a foundation for extending the MAEPOS meta-model with security concepts and implementation strategies relevant to the DTN context in future work.

#### IV. ABSTRACTION OF THE APPLICATION EXECUTION PLATFORM

##### A. The Model

The meta-model is based on the fusion of two technology domains. Terrestrial cloud technologies offer tools to run applications in a portable, scalable, and distributed manner. However, these components must be combined with space technologies to enable communication and operation via DTN. That is why MAEPOS contains components from both areas: the higher-level components of the terrestrial application execution platform, such as nodes, clusters, and cluster components with the components for communication via DTN such as DTN Nodes, Endpoints, or Clients. A Life-Cycle Management Controller and Support Services represented in the model provide additional functionalities to enable platform operations via DTNs.

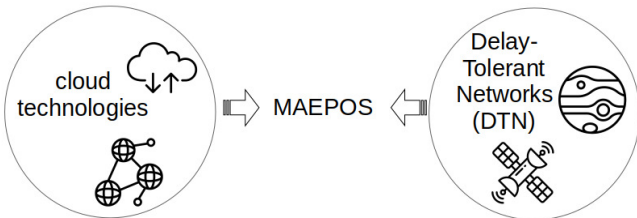


Fig. 1. Technology domains joined by MAEPOS

Before presenting the model, the basic assumptions made by the DARKSOL project are explained. Users, such as ACME, must check whether there is a low-latency connection comparable to the terrestrial Internet between the physical nodes that form the platform. The meta-model only supports the reliable design of a platform that can be externally controlled via DTNs. It does not support DTN-based communication between the software components that comprise the core of the application execution platform. Furthermore, applications running on the execution platform and communicating over DTN must be explicitly aware of these connections and align their processes with these conditions. Without these assumptions, it is unlikely that cluster technologies and applications run smoothly on the platform, which would require extensive adaptations of existing platform technologies.

MAEPOS consists of five hierarchical layers, each represented by a class from the top to the bottom in Fig. 2 (Application, Cluster, ClusterNode, Node and NodeInfrastructure). Several classes ensure the DTN capability of the Space Cloud, including DTNAgent with DTNEndpoint, LCMController and SupportService. The objects of these classes can be used on several hierarchical levels in instances of the meta-model. The exact structure and their role vary depending on the context. For example, an ArtifactRegistry is a SupportService that, if it is instantiated at the Node layer, will be storing virtual machine images used by an upper node for separation, while at the Cluster layer the ArtifactRegistry may contain container images for running specific applications.

**Application layer:** Applications are located within a cluster and can be executed through a ClusterComponent. In addition, an LCMController on the cluster layer can control the life-cycle of an application, i. e. starting, updating, and terminating it.

**Cluster layer:** A cluster consists of ClusterComponents that are distributed across one or more ClusterNodes. These ClusterComponents, in turn contain ClusterComponents or work together with other ClusterComponents. ClusterComponents can also use SupportServices. If a ClusterComponent has a DTNAgent, then this component can be reached via DTN and is called DTNClusterComponent. A ClusterNode can provide SupportServices that provide functional support for the cluster. One or more LCMControllers and DTNNodes can also be provided, which enable the lifecycle management of the applications and ClusterComponents via DTN and facilitate the DTN accessibility of the cluster.

**Node layer:** A node can be a physical or logical node and can itself contain nodes. A node can also contain a ClusterNode. This is a specific type of node that hosts part of a cluster or an entire cluster. Each node can have one or more SupportServices for additional functions and one or more LCMControllers for managing the life-cycle of the contained nodes or clusters. In addition, a node can also be accessible via DTN if it contains a DTNAgent. The nodes of a cluster are summarized as NodeInfrastructure, which provides networking for low-latency communication within the cluster.

**DTN:** Connectivity via DTN is provided by DTNAgents.

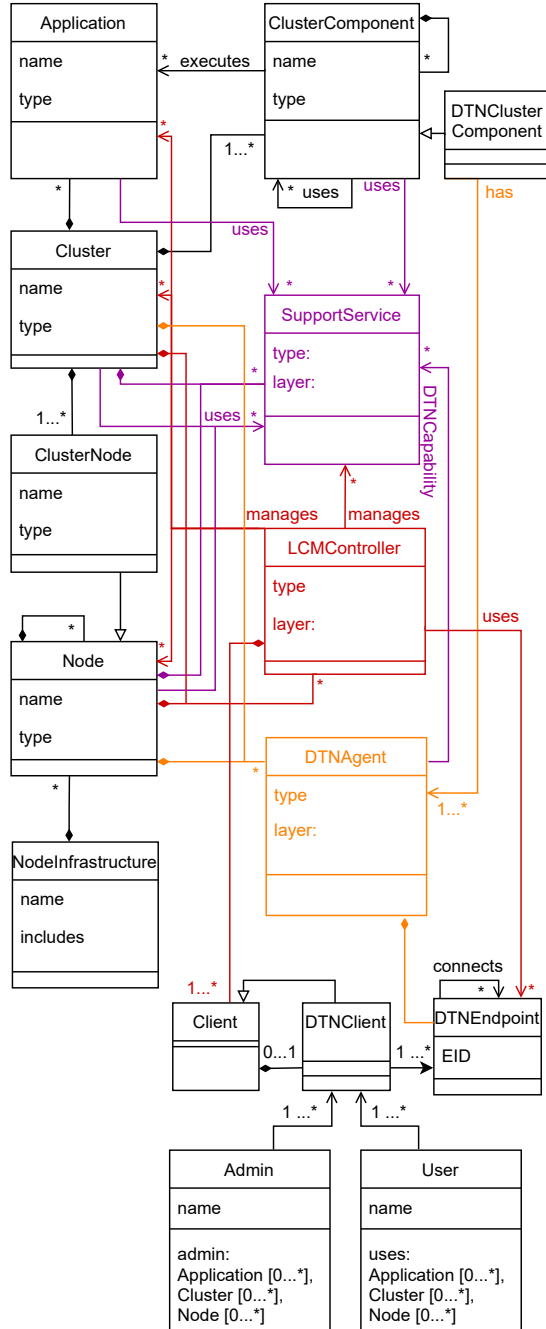


Fig. 2. Meta-model to describe DTN-enabled application execution platforms

These implementations of the DTN protocols can be located on any layer and, depending on this, may require one or more SupportServices to establish full DTN capability. A DTNAgent offers a DTNEndpoint characterized by an Endpoint Identifier (EID), the unique identifier for a logical communication target in the DTN. The DTN communication connection is established between an administrator or user and an LCMController of the desired layer. Both have a DTN client for this purpose, which in turn can manage several DTNEndpoints. The LCMController has one or more clients for the cluster-internal low-latency communication. If the LCMController has to communicate via a DTN, it has one or more DTN clients for this cluster-external high-latency communication.

After presenting the model, its applicability will be evaluated using an example. Further examples with different infrastructure conditions and technological challenges will expand the evaluation in the future.

## V. VALIDATION

### A. Validation Approach and Concrete Example

The example scenario is the one introduced in section II. After describing this instance of the meta-model, the general requirements for the meta-model are reviewed. The two satellites planned for the application execution platform in lunar orbit (see Fig. 3) are represented in the architecture by two physical nodes: the InfrastructureController and the ServerNode, which are capable of communicating with each other via a low-latency connection. The InfrastructureController node runs two containers, SupportService and CommunicationService. The SupportService container encapsulates four services, including a DNSServer for local name resolution to enable platform communication, a NTPServer for system time synchronization, a DHCPserver for dynamic IP address allocation, and an ArtifactRegistry, which manages the images used to provision the K8sCluster and the remaining components in a VM using the Linux distribution *Talos*. The CommunicationService container hosts two services: a DiscoveryService to enable the discovery of applications outside the platform and  $\mu$ D3TN, a DTN protocol implementation.

The server node runs a virtual machine using Talos as the operating system. Within this VM, a Kubernetes cluster, called K8sCluster is deployed. The TalosVM executes additional components that support the functionality of the Kubernetes cluster. These include an ArtifactRegistry, which stores container images and Helm charts for applications within the K8sCluster. Furthermore, the TalosVM contains a DiscoveryService, another instance of the DTN protocol implementation  $\mu$ D3TN, and a controller named AppLCM, which manages the life-cycle of applications within the Kubernetes cluster. An application named DemoApp is deployed as a container within the Kubernetes cluster. Additionally, the cluster includes Custom Resource DTN-Ingress/Egress, which enables DTN-based communication in the Kubernetes environment. The Ingress component facilitates incoming DTN communication, while the Egress component handles outgoing DTN traffic.

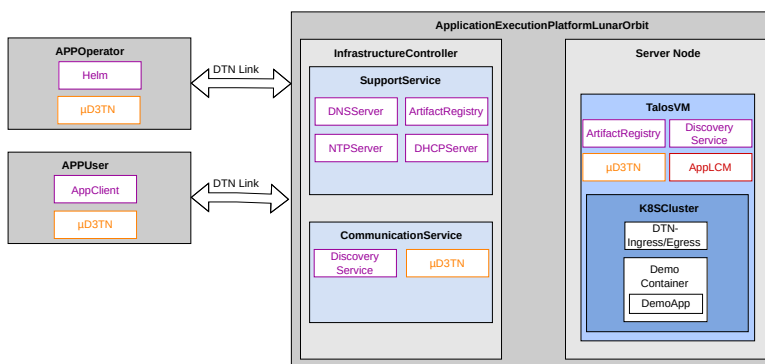


Fig. 3. Sketch of the execution platform as designed by ACME

Outside the platform, there is an admin named APPOperator, who manages the DemoApp in the Kubernetes cluster. There is also a user of the DemoApp application named APPUser. Both have an instance of the DTN protocol  $\mu$ D3TN installed. The admin uses Helm for managing the DemoApp and the user leverages an AppClient for interaction with the app. The admin on Earth uses Helm to package and send Kubernetes deployments as single bundles over DTN. This reduces communication overhead and allows the remote cluster to apply updates autonomously and reliably.

### B. Modeling an application execution platform for the Solar System Internet

The instantiation of MAEPOS with the application execution platform in lunar orbit is simplified (see Figure 4) by only showing a detailed representation of all components from ClusterNode onward. The components of the Node and NodeInfrastructure are only indicated by methods in the respective object. In addition, the instance is color-coded to match the architecture diagram (see Figure 3).

This are the most important features of the instance:

- The Kubernetes cluster is only located on the physical node named ServerNode. The other physical node provides supporting services and DTN capability.
- The LCMController AppLCM is located on the cluster layer, but manages the application layer. This is the concept behind the LCMController, which is positioned on a layer and manages the components of the next higher layer.
- The same is valid for the SupportService, which is storing artifacts for the next higher layer, for example, ArtifactRegistry on the ClusterNode stores the images for the DemoApp for the K8sCluster.
- To provision the cluster, all layers require an ArtifactRegistry and an LCMController, which use the artifacts to build the next higher layer.
- The LCMController in the cluster contains a DTNClient for DTN communication, for example with admin or user.
- Each layer that has to be accessible via DTN requires a DTNEndpoint, and a DTNAgent, as well as possibly

SupportServices such as a DiscoveryService. In this case, these components are provided on the cluster level by  $\mu$ D3TN as DTNAgent.

- DTN communication is given between the user’s DTNAgent and administrator’s DTNAgent and the DTNAgent of the ClusterNode.

After presenting an example of the instantiation of MAEPOS, the requirements introduced in Section II-B will be discussed in the next step.

### C. Discussion of Requirements

- **Technology independence:** The meta-model can be instantiated using different technologies. For example for orchestration Kubernetes, DockerSwarm or AmazonECS, for DTN  $\mu$ D3TN, IPN or HRDTN and so on.
- **Cluster distribution across low-latency nodes:** The mechanism by which Nodes can recursively contain Nodes and thus also ClusterNodes, which in turn can belong to the same Cluster, enables distributed clusters within the low-latency network.
- **Multi-tenancy incl. secure separation:** The ability to nest nodes indefinitely and equip them with separation solutions, as well as flexibly distributing communication and control components, allows any number of clients to be separated to any degree.
- **Support of DTN capability on every layer/node:** DTN capabilities can be established at the Node, ClusterNode, and Cluster level by the DTNAgent, including the DTNEndpoint and SupportServices.
- **Autonomous life-cycle management:** Autonomous life-cycle management can be operated by the LCMController at the Node, ClusterNode, and Cluster level.
- **Expandability with mechanisms for security, monitoring, encryption:** Expandability should be ensured by the layered and abstract structure of the meta-model and will be addressed in future work.

## VI. SUMMARY AND OUTLOOK

This paper has introduced an initial 5-layer meta-model for the design of DTN-enabled application execution platform,

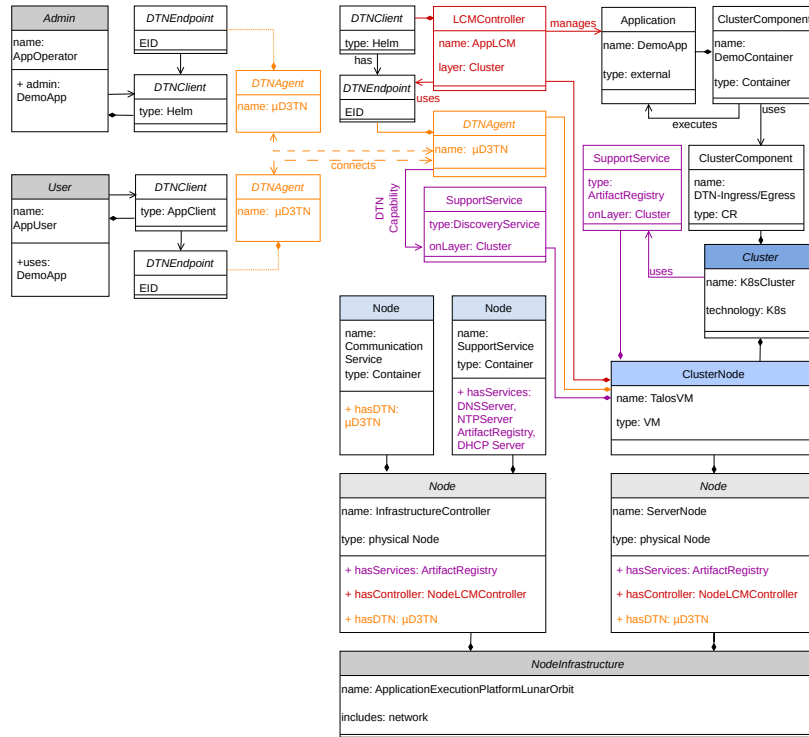


Fig. 4. Simplified example instance of the meta-model

termed MAEPOS. It was applied in a scenario that involves two satellites in lunar orbit to demonstrate its initial applicability. A broader spectrum of scenarios must be addressed by the meta-model to prove its general applicability in the future.

In the future, the application of MAEPOS will no longer be purely descriptive but will take the form of automated conceptualization followed by the provision of individually customized instances tailored to infrastructural conditions and technological requirements. First, MAEPOS will be described in a formal cloud modeling language. Next, a software interface will be created for a framework that integrates relevant technologies and, in combination with the meta-model, generates a concrete technology- and distribution-specific architecture. Then, this solution should be able to be automatically deployed over the generation of artifacts for life-cycle management of the DTN-supported platform.

To evaluate the extensibility of MAEPOS, an abstract mechanism that can be applied to various cross-cutting concerns will be designed. The aspects of security, observability, and reliability have not yet been sufficiently specified for DTN-supported platforms. The extension with these cross-cutting concerns will ensure secure, observable, and reliable instances of the platform. Furthermore, a DTN SOCKS-like protocol would be helpful to handle data from DTN connections according to how it is processed by the communication partner. This could also include standardizing Helm commands transmitted over BP. Ultimately, MAEPOS lays the foundation for comprehensive and standards-compliant design and

development of solutions that connect DTN technologies and terrestrial cloud components, enabling secure, reliable, and distributed application execution over DTN in the future.

## REFERENCES

- [1] K. Chatzirimou, K. Lano, and S. Zschaler, *Towards A Meta-Model of the Cloud Computing Resource Landscape*. 2013.
- [2] R. Di Cosmo, M. Lienhardt, R. Treinen, S. Zacchiroli, J. Zwolakowski, A. Eiche, and A. Agahi, "Automated synthesis and deployment of cloud applications," ASE '14, (New York, NY, USA), p. 211–222, Association for Computing Machinery, 2014.
- [3] A. Bergmayr, U. Breitenbücher, N. Ferry, A. Rossini, A. Solberg, M. Wimmer, G. Kappel, and F. Leymann, "A systematic review of cloud modeling languages," *ACM Comput. Surv.*, vol. 51, Feb. 2018.
- [4] OASIS Committee, "TOSCA version 1.0 standard – topology and orchestration specification for cloud applications." <https://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html>, 2013. OASIS Standard.
- [5] A. Bergmayr, U. Breitenbücher, O. Kopp, M. Wimmer, G. Kappel, and F. Leymann, "From architecture modeling to application provisioning for the cloud by combining UML and TOSCA," in *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, Scitepress, 2016.
- [6] B. LaFuente, R. Dudukovich, N. Kortas, E. Schweinsberg, B. Tomko, and A. Kotliarchuk, "Exploring new frontiers in space communications: enhancing delay tolerant networking through cloud and containerization," in *AIAA SCITECH 2024 Forum*, p. 1741, 2024.
- [7] T. Mavroeidakos, A. Michalakis, and D. D. Vergados, "Security architecture based on defense in depth for cloud computing environment," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 334–339, 2016.
- [8] K. Kritikos and P. Massonet, "An integrated meta-model for cloud application security modelling," *Procedia Computer Science*, vol. 97, pp. 84–93, 12 2016.