Workgroup: Delay-Tolerant Networking
Internet-Draft: draft-ietf-dtn-dtnma-05
Published: 12 March 2023
Intended Status: Informational
Expires: 13 September 2023
Authors: E.J. Birrane
Johns Hopkins Applied Physics Laboratory
S.E. Heiner
Johns Hopkins Applied Physics Laboratory
E. Annis
Johns Hopkins Applied Physics Laboratory

DTN Management Architecture

Abstract

The Delay-Tolerant Networking (DTN) architecture describes a type of challenged network in which communications may be significantly affected by long signal propagation delays, frequent link disruptions, or both. The unique characteristics of this environment require a unique approach to network management that supports asynchronous transport, autonomous local control, and a small footprint (in both resources and dependencies) so as to deploy on constrained devices.

This document describes a DTN management architecture (DTNMA) suitable for managing devices in any challenged environment but, in particular, those communicating using the DTN Bundle Protocol (BP). Operating over BP requires an architecture that neither presumes synchronized transport behavior nor relies on query-response mechanisms. Implementations compliant with this DTNMA should expect to successfully operate in extremely challenging conditions, such as over uni-directional links and other places where BP is the preferred transport.

This Internet-Draft will expire on 13 September 2023.

Copyright Notice

Table of Contents

1. Introduction

The Delay-Tolerant Networking (DTN) architecture, as described in [RFC4838], has been designed to cope with data exchange in challenged networks. Just as the DTN architecture requires new capabilities for transport and transport security, special consideration must be given for the management of DTN devices.

This document describes a DTN Management Architecture (DTNMA) providing configuration, monitoring, and local control of both application and network services on a managed device. The DTNMA is designed to provide for the management of devices operating either within or across a challenged network.

Fundamental properties of a challenged network are outlined in Section 2.2.1 of [RFC7228]. These properties include lacking end-to-end IP connectivity, having "serious interruptions" to end-to-end connectivity, and exhibiting delays longer than can be tolerated by end-to-end synchronization mechanisms (such as TCP). It is further noted that the DTN architecture was designed to cope with such networks.

NOTE: These challenges may be caused by physical impairments such as long signal propagations and frequent link disruptions, or by other factors such as quality-of-service prioritizations, service-level agreements, and other consequences of traffic management and scheduling.

Device management in these environments must occur without human interactivity, without system-in-the-loop synchronous function, and without requiring a synchronous underlying transport layer. This means that managed devices need to determine their own schedules for data reporting, their own operational configuration, and perform their own error discovery and mitigation.

Certain outcomes of device self-management should be determinable by a privileged external observer (such as a managing device). In a challenged network, these observers may need to communicate with a managed device after significant periods of disconnectivity. Non-deterministic behavior of a managed device may make establishing communication difficult or impossible.

The desire to define asynchronous and autonomous device management is not new. However, challenged networks (in general) and the DTN environment (in particular) represent unique deployment scenarios and impose unique design constraints. To the extent that these environments differ from more traditional, enterprise networks, their management may also differ from the management of enterprise networks. Therefore, existing techniques may need to be adapted to operate in the DTN environment or new techniques may need to be created.

NOTE: The DTNMA is designed to leverage any transport, network, and security solutions designed for challenged networks. However, the DTNMA should operate in any environment in which the Bundle Protocol (BPv7) [RFC9171] is deployed.

1.1. Scope

This document describes the desirable properties of, and motivation for, a DTNMA. This document also provides a reference model, service descriptions, autonomy model, and use cases to better reason about ways to standardize and implement this architecture.

This is not a normative document and the information herein is not meant to represent a standardization of any data model, protocol, or implementation. Instead, this document provides informative guidance to authors and users of such models, protocols, and implementations.

The selection of any particular transport or network layer is outside of the scope of this document. The DTNMA does not require the use of any specific protocol such as IP, BP, TCP, or UDP. In particular, the DTNMA design does not assume the use of either IPv4 or IPv6.

NOTE: The fact that the DTNMA must operate in any environment that deploys BP does not mean that the DTNMA requires the use of BP to operate.

Network features such as naming, addressing, routing, and security are out of scope of the DTNMA. It is presumed that any operational network communicating DTNMA messages would implement these services for any payloads carried by that network.

The interactions between and amongst the DTNMA and other management approaches are outside of the scope of this document.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.3. Organization

The remainder of this document is organized into the following nine sections, described as follows.

* Terminology - This section identifies terms fundamental to understanding DTNMA concepts. Whenever possible, these terms align in both word selection and meaning with their use in other management protocols.

* Challenged Network Overview - This section describes important aspects of challenged networks and necessary approaches for their management.

* Desirable Design Properties - This section defines those properties of the DTNMA that must be present to operate within the constraints of a challenged network. These properties are similar to the specification of system-level requirements of a DTN management solution.

*Current Network Management Approaches - This section provides a brief overview of existing network management approaches. Where possible, the DTNMA adopts concepts from these approaches. The limitations of current approaches from the perspective of the DTNMA desirable properties are identified and discussed.

*Motivation for New Features - This section provides an overall motivation for this work, to include explaining why a management architecture for challenged networks is useful and necessary.

*Reference Model - This section defines a reference model that can be used to reason about the DTNMA independent of an implementation. This model identifies the logical elements of the system and the high-level relationships and behaviors amongst those elements.

*Desired Services - This section identifies and defines the DTNMA services provided to network and mission operators.

*Logical Autonomy Model - This section provides an exemplar data model that can be used to reason about DTNMA control and data flows. This model is based on the DTNMA reference model.

*Use Cases - This section presents multiple use cases accommodated by the DTNMA architecture. Each use case is presented as a set of control and data flows referencing the DTNMA reference model and logical autonomy model.

2. Terminology

This section defines terminology that either is unique to the DTNMA or is necessary for understanding the concepts defined in this specification.

*Constants (CONST): Typed, immutable value referred to by a semantic name. Constants allow substituting a meaningful name for a fixed value. For example, using the constant PI_5_DIGIT_PRECISION rather than the literal value 3.14159.

*Controls (CTRLs): Procedures run by a DA to change the behavior, configuration, or state of an application or protocol managed by that DA. This includes procedures to manage the DA itself, such as to have the DA produce performance reports or to apply new management policies.

*DTN Management: Management that does not depend on stateful connections, timely delivery of management messages, or closed-loop control.

*DTNMA Agent (DA): A role associated with a managed device, responsible for reporting performance data, accepting policy directives, performing autonomous local control, error-handling, and data validation. DAs exchange information with DMs operating either on the same device and/or on remote devices in the network.

*DTNMA Manager (DM): A role associated with a managing device responsible for configuring the behavior of, and eventually receiving information from, DAs. DMs interact with one or more DAs located on the same device and/or on remote devices in the network.

*Externally Defined Data (EDD): Typed information made available to a DA by its hosting device, but not computed directly by the DA itself.

*Literals (LITs): Typed information whose name is the literal expression of its value. For example, the number 4 is a Literal value.

*Macros (MACROs): Named, ordered collections of Controls and/or other Macros.

*Operators (OPs): Mathematical functions used to calculate variable values and construct expressions to evaluate DA state.

*Reports (RPTs): Typed, ordered collections of data values gathered by one or more DAs and provided to one or more DMs. Reports comply to the format of a given Report Template.

*Report Templates (RPTTs): Named, ordered collection of data names that represent the schema of a Report. Templates are generated by a DM and communicated to other DMs and DAs.

*Rules: Unit of autonomous specification that provides a stimulus-response relationship between time or state on a DA and the actions or operations to be run as a result of that time or state.

*State-Based Rule (SBR): Any Rule triggered by the calculable, internal state of the DA.

*Tabular Report (TBL): Typed collection of data values organized in a tabular way in which columns represent homogeneous types of data and rows represent unique sets of data values conforming to column types. Tabular Reports are a specialization of a Report and comply to the format of a given Tabular Report Template.

*Tabular Report Template (TBLT): Named, typed, ordered collection of columns that comprise the structure for representing tabular data values. This template forms the structure of a Tabular Report.

*Time-Based Rule (TBR): A specialization, and simplification, of a State-Based Rule in which the rule stimulus is triggered by relative or absolute time on a DA.

*Variables (VARs): Typed information computed internal to a DA.

3. Challenged Network Overview

The DTNMA provides network management services able to operate in a challenged network environment, such as envisioned by the DTN architecture. This section describes what is meant by the term "challenged network", the important properties of such a network, and observations on impacts to conventional management approaches.

3.1. Challenged Network Constraints

Constrained networks are defined as networks where "some of the characteristics pretty much taken for granted with link layers in common use in the Internet at the time of writing are not attainable." [RFC7228]. This broad definition captures a variety of potential issues relating to physical, technical, and regulatory constraints on message transmission. Constrained networks typically include nodes that regularly reboot or are otherwise turned off for long periods of time, transmit at low or asynchronous bitrates, and/or have very limited computational resources.

Separately, a challenged network is defined as one that "has serious trouble maintaining what an application would today expect of the end-to-end IP model" [RFC7228]. This definition includes networks where there is never simultaneous end-to-end connectivity, when such connectivity is interrupted at planned or unplanned intervals, or when delays exceed those that could be accommodated by IP-based transport. Links in such networks are often unavailable due to attenuation, propagation delays, mobility, occultation, and other limitations imposed by energy and mass considerations.

NOTE: Because challenged networks might not provide services expected of the end-to-end IP model, devices in such networks might not implement networking stacks associated with the end-to-end IP model. This means that devices might not include support for certain transport protocols (TCP/UDP), web protocols (HTTP), or even internetworking protocols (IPv4/IPv6).

By these definitions, a "challenged" network is a special type of "constrained" network, where the constraints are related to end-to-

end connectivity and delays. As such, "all challenged networks are constrained networks ... but not all constrained networks are challenged networks ... Delay-Tolerant Networking (DTN) has been designed to cope with challenged networks" [RFC7228].

Solutions that work in constrained networks might not be solutions that work in challenged networks. In particular, challenged networks exhibit the following properties that impact the way in which the function of network management is considered.

- No end-to-end path is guaranteed to exist at any given time between any two nodes.
- Round-trip communications between any two nodes within any given time window may be impossible.
- Latencies on the order of seconds, hours, or days must be tolerated.
- Links may be uni-directional.
- Bi-directional links may have asymmetric data rates.
- The existence of external infrastructure, software, systems, or processes such as a Domain Name Service (DNS) or a Certificate Authority (CA) cannot be guaranteed.

3.2. Topology and Service Implications

The set of constraints that might be present in a challenged network impact both the topology of the network and the services active within that network.

Operational networks handle cases where nodes join and leave the network over time. These topology changes may or may not be planned, they may or may not represent errors, and they may or may not impact network services. Challenged networks differ from other networks not in the present of topological change, but in the likelihood that impacts to topology result in impacts to network services.

The difference between topology impacts and service impacts can be expressed in terms of connectivity. Topological connectivity usually refers to the existence of a path between an application message source and destination. Service connectivity, alternatively, refers to the existence of a path between a node and one or more services needed to process (often just-in-time) application messaging. Examples of service connectivity include access to infrastructure elements such as a Domain Name System (DNS) or a Certificate Authority (CA).

In networks that might be partitioned most of the time, it is less likely that a node would concurrently access both an application endpoint and one or more network service endpoints. For this reason, network services in a challenged network should be designed to allow for asynchronous operation. Accommodating this use case often involves the use of local caching, pre-placing information, and not hard-coding message information at a source that might change when a message reaches its destination.

NOTE: Oner example of rethinking services in a challenged network is the securing of BPv7 bundles. The BPSec [RFC9172] security extensions to BPv7 do not encode security destinations when applying security. Instead, BPSec requires nodes in a network to identify themselves as security verifiers or acceptors when receiving and processing secured messages.

3.2.1. Management Implications

Network management approaches must adapt to the topology and service impacts encountered in challenged networks. In particular, the ways in which "managers" and "agents" in a management architecture operate must consider how to operate with changes to topology and changes to service endpoints.

When connectivity to a manager cannot be guaranteed, agents must rely on locally available information and use local autonomy to react to changes at the node. Architectures that rely on external resources such as access to third-party oracles, operators-in-the-loop, or other service infrastructure may fail to operate in a challenged network.

In addition to disconnectivity, topological change can alter the associations amongst managed and managing devices. Different managing devices might be active in a network at different times or in different partitions. Managed devices might communicate with some, all, or none of these managing devices as a function of their own local configuration and policy.

NOTE: These concepts relate to practices in conventional networks. For example, supporting multiple managing devices is similar to deploying multiple instances of a network service -- such as a DNS server or CA node. Selecting from a set of managing devices is similar to a sensor node practice of electing cluster heads to act as privileged nodes for data storage and exfiltration.

Therefore, a network management architecture for challenged networks should:

1. Support a many-to-many association amongst managing and managed devices, and

2. Allow "control from" and "reporting to" managing devices to function independent of one another.

3.3. Management Special Cases

The following special cases illustrate some of the operational situations that can be encountered in the management of devices in a challenged network.

*One-Way Management. A managed device can only be accessed via a uni-directional link, or a via a link whose duration is shorter than a single round-trip propagation time.

*Summary Data. A managing device can only receive summary data of a managed device's state because a link or path is constrained by capacity or reliability.

*Bulk Historical Reporting. A managing device receives a large volume of historical report data for a managed device. This can occur when a managed device rejoins a network or has access to a high capacity link (or path) to the managed device.

*Multiple Managers. A managed device tracks multiple managers in the network and communicates with them as a function of time, local state, or network topology. This includes challenged networks that interconnect two or more unchallenged networks such that managed and managing devices exist in different networks.

These special cases highlight the need for managed devices to operate without presupposing a dedicated connection to a single managing device. To support this, managing devices must deliver instruction sets that govern the local, autonomous behavior of managed devices. These behaviors include (but are not limited to) collecting performance data, state, and error conditions, and applying pre-determined responses to pre-determined events. Managing devices in a challenged network might never expect a reply to a command, and communications from managed devices may be delivered much later than the events being reported.

4. Desirable Design Properties

This section describes those design properties that are desirable when defining a management architecture operating across challenged links in a network. These properties ensure that network management capabilities are retained even as delays and disruptions in the network scale. Ultimately, these properties are the driving design principles for the DTNMA.

NOTE: These properties may influence the design, construction, and adaptation of existing management tools for use in challenged

networks. For example, the properties the DTN architecture [RFC4838] resulted in the development of BPv7 [RFC9171] and BPSec [RFC9172]. The DTNMA may result in the construction of new management data models, policy expressions, and/or protocols.

4.1. Dynamic Architectures

The DTNMA should be agnostic of the underlying physical topology, transport protocols, security solutions, and supporting infrastructure of a given network. Due to the likelihood of operating in a frequently partitioned environment, the topology of a network may change over time. Attempts to stabilize an architecture around individual nodes can result in a brittle management framework and the creation of congestion points during periods of connectivity.

NOTE: The DTNMA must run in every environment in which BP bundles may be used, even though the DTNMA does not require the use of BP for its transport.

The DTNMA should not prescribe any association between a DM and a DA other than those defined in this document. There should be no logical limitation to the number of DMs that can control a DA, the number of DMs that a DA should report to, or any requirement that a DM and DA relationship implies a pair.

NOTE: Practical limitations on the relationships between and amongst DMs and DAs will exist as a function of the capabilities of networked devices. These limitations derive from processing and storage constraints, performance requirements, and other engineering factors. While this information is vital to the proper engineering of a managed and managing device, they are implementation considerations, and not otherwise design constraints on the DTNMA.

4.2. Hierarchically Modeled Information

The DTNMA should use data models to define the syntactic and semantic contracts for data exchange between a DA and a DM. A given model should have the ability to "inherit" the contents of other models to form hierarchical data relationships.

NOTE: The term data model in this context refers to a schema that defines a contract between a DA and a DM for how information is represented and validated.

Many network management solutions use data models to specify the semantic and syntactic representation of data exchanged between managed and managing devices. The DTNMA is not different in this regard - information exchanged between DAs and DMs should conform to one or more pre-defined, normative data models.

A common best practice when defining a data model is to make it cohesive. A cohesive model is one that includes information related to a single purpose such as managing a single application or protocol. When applying this practice, it is not uncommon to develop a large number of small data models that, together, describe the information needed to manage a device.

Another best practice for data model development is the use of inclusion mechanisms to allow one data model to include information from another data model. This ability to re-use a data model avoids repeating information in different data models. When one data model includes information from another data model, there is an implied model hierarchy.

Data models in the DTNMA should allow for the construction of both cohesive models and hierarchically related models. These data models should be used to define all sources of information that can be retrieved, configured, or executed in the DTNMA. This includes supporting DA autonomy functions such as parameterization, filtering, and event driven behaviors. These models will be used to both implement interoperable autonomy engines on DAs and define interoperable report parsing mechanisms on DMs.

NOTE: While data model hierarchies can result in a more concise data model, arbitrarily complex nesting schemes can also result in very verbose encodings. Where possible, data identifications schemes should be constructed that allow for both hierarchical data and highly compressible data identification.

4.3. Adaptive Push of Information

DAs in the DTNMA architecture should determine when to push information to DMs as a function of their local state.

Pull management mechanisms require a managing device to send a query to a managed device and then wait for a response to that specific query. This practice implies some serialization mechanism (such as a control session) between entities. However, challenged networks cannot guarantee timely round-trip data exchange. For this reason, pull mechanisms must be avoided in the DTNMA.

Push mechanisms, in this context, refer to the ability of DAs to leverage local autonomy to determine when and what information should be sent to which DMs. The push is considered adaptive because a DA determines what information to push (and when) as an adaptation to changes to the DA's internal state. Once pushed, information might still be queued pending connectivity of the DA to the network.

NOTE: Even in cases where a round-trip exchange can occur, pull mechanisms increase the overall amount of traffic in the network and

preclude the use of autonomy at managed devices. So even when pull mechanisms are feasible they should not be considered a pragmatic alternative to push mechanisms.

4.4. Efficient Data Encoding

Messages exchanged between a DA and a DM in the DTNMA should be defined in a way that allows for efficient on-the-wire encoding. DTNMA design decisions that result in smaller message sizes should be preferred over those that result in larger message sizes.

There is a relationship between message encoding and message processing time at a node. Messages with little or no encodings may simplify node processing whereas more compact encodings may require additional activities to generate/parse encoded messages. Generally, compressing a message takes processing time at the sender and decompressing a message takes processing time at a receiver. Therefore, there is a design tradeoff between minimizing message sizes and minimizing node processing.

NOTE: There are many ways in which message size, number of messages, and node behaviors can impact processing performance. Because the DTNMA does not presuppose any underlying protocol or implementation, this section is focused solely on the compactness of an individual message and the processing for encoding and decoding that individual message.

There is no advantage to minimizing node processing time in a challenged network. The same sparse connectivity that benefits from store-and-forward transport provides time at a node for data processing prior to a future transmission opportunity.

However, there is a significant advantage to smaller message sizes in a challenged network. Smaller messages require smaller periods of viable transmission for communication, they incur less re-transmission cost, and they consume less resources when persistently stored en-route in the network.

NOTE: Naive approaches to minimizing message size through general purpose compression algorithms do not produce minimal encodings. Data models can, and should, be designed for compact encoding from the beginning. Design strategies for compact encodings involve using structured data instead of large hash values, reusable, hierarchical data models, and exploiting common structures in data models.

4.5. Universal, Unique Data Identification

Elements within the DTNMA should be uniquely identifiable so that they can be individually manipulated. Further, these identifiers

should be universal - the identifier for a data element should be the same regardless of role, implementation, or network instance.

Identification schemes that are relative to a specific DA or specific system configuration might change over time. In particular, nodes in a challenged network may change their status or configuration during periods of partition from other parts of the network. Resynchronizing relative state or configuration should be avoided whenever possible.

NOTE: Consider the common technique for approximating an associative array lookup. A manager wishing to perform an associative lookup for some key K1 will:

1. Query a list of array keys from an agent.

2. Find the key that matches K1 and infer the index of K1 from the returned key list.

3. Query the discovered index on the agent to retrieve the desired data.

Ignoring the inefficiency of two round-trip exchanges, this mechanism will fail if the agent changes its key-index mapping between the first and second query. While this is unlikely to occur in a low-latency network, it is more likely to occur in a challenged network.

4.6. Runtime Data Definitions

The DTNMA should allow for the definition of new elements to a data model as part of the runtime operation of the management system. These definitions may represent custom data definitions that are applicable only for a particular device or network. Custom definitions should also be able to be removed from the system during runtime.

The custom definition of new data from existing data (such as through data fusion, averaging, sampling, or other mechanisms) provides the ability to communicate desired information in as compact a form as possible.

NOTE: A DM could, for example, define a custom data report that includes only summary information around a specific operational event or as part of specific debugging. DAs could then produce this smaller report until it is no longer necessary, at which point the custom report could be removed from the management system.

Custom data elements should be calculated and used both as parameters for DA autonomy and for more efficient reporting to DMs.

Defining new data elements allows for DAs to perform local data fusion and defining new reporting templates allows for DMs to specify desired formats and generally save on link capacity, storage, and processing time.

4.7. Autonomous Operation

The management of applications by a DA should be achievable using only knowledge local to the DA because DAs might need to operate during times when they are disconnected from a DM.

DA autonomy may be used for simple automation of predefined tasks or to support semi-autonomous behavior in determining when to run tasks and how to configure or parameterize tasks when they are run. In either case, a DA should provide the following features.

* Stand-alone Operation - Pre-configuration allows DAs to operate without regular contact with other nodes in the network. The initial configuration (and periodic update) of a DA autonomy engine remains difficult in a challenged network, but removes the requirement that a DM be in-the-loop during regular operations. Sending stimuli-and-responses to a DA during periods of connectivity allows DAs to self-manage during periods of disconnectivity.

* Deterministic Behavior - Operational systems might need to act in a deterministic way even in the absence of an operator in-the-loop. Deterministic behavior allows an out-of-contact DM to predict the state of a DA and to determine how a DA got into a particular state.

* Engine-Based Behavior - Operational systems might not be able to deploy "mobile code" [RFC4949] solutions due to network bandwidth, memory or processor loading, or security concerns. Engine-based approaches provide configurable behavior without incurring these concerns.

* Authentication, Authorization, and Accounting - The DTNMA does not require a specific underlying transport protocol, network infrastructure, or network services. Therefore, mechanisms for authentication, authorization, and accounting must be present in a standard way at DAs and DMs to provide these functions if the underlying network does not. This is particularly true in cases where multiple DMs may be active concurrently in the network.

Features such as deterministic processing and engine-based behavior do not preclude the use of other Artificial Intelligence (AI) and Machine Learning (ML) approaches on a managed device.

NOTE: The deterministic automation of the DTNMA can monitor and control AI/ML management applications on a managed device. Using multiple levels of autonomy is a well-known method to balance the flexibility of a highly autonomous system with the reduced risk of a deterministic system.

5. Current Network Management Approaches

Several network management solutions have been developed for both local-area and wide-area networks. Their capabilities range from simple configuration and report generation to complex modeling of device settings, state, and behavior. Each of these approaches are successful in the domains for which they have been built, but are not all equally functional when deployed in a challenged network.

Early network management tools designed for unchallenged networks provide synchronous mechanisms for communicating locally-collected data from devices to operators. Applications are managed using a "pull" mechanism, requiring a managing device to explicitly request the data to be produced and transmitted by a managed device.

NOTE: Network management solutions that pull large sets of data might not operate in a challenged environment that cannot support timely, round-trip exchange of large data volumes.

More recent network management tools focus on message-based management, reduced state keeping by managed and managing devices, and increased levels of system autonomy.

This section describes some of the well-known, standardized protocols for network management and contrasts their purposes with the desirable properties of the DTNMA. The purpose of this comparison is to identify elements of existing approaches that can be adopted or adapted for use in challenged networks and where new elements must be created specifically for this environment.

5.1. Simple Network Management Protocol (SNMP)

The de facto example of a pull architecture is the Simple Network Management Protocol (SNMP) [RFC3416]. SNMP utilizes a request/response model to set and retrieve data values such as host identifiers, link utilization metrics, error rates, and counters between application software on managing and managed devices. Data may be directly sampled or consolidated into representative statistics. Additionally, SNMP supports a model for unidirectional push notification messages, called traps, based on predefined triggering events.

SNMP managing devices can query agents for status information, send new configurations, and request to be informed when specific events

have occurred. SNMP devices separate the representations for data modeling (Structure of Management Information (SMI) [RFC2578] and the Management Information Base (MIB) [RFC3418]) and messaging, sequencing and encoding (the SNMP protocol [RFC3416]).

Separating data models from messaging and encoding is a best practice in subsequent management protocols and likely necessary for the DTNMA. In particular, SNMP MIBs provide well-organized, hierarchical Object Identifiers (OIDs) which support the compressibility necessary for challenged DTNs.

While there is a large installation base for SNMP, several aspects of the protocol make it inappropriate for use in a challenged network. SNMP relies on sessions with low round-trip latency to support its "pull" mechanism. Complex management can be achieved, but only through careful orchestration of real-time, end-to-end, managing-device-generated query-and-response logic.

The SNMP trap model provides some low-fidelity Agent-side processing. Traps are typically used for alerting purposes, as they do not support a local agent response to the initiating event. In a challenged network where the delay between a managing device receiving an alert and sending a response can be significant, the SNMP trap model is insufficient for event handling.

5.2. YANG-Based Protocols

Yet Another Next Generation (YANG) [RFC6020] is a data modeling language used to model the configuration and state data of managed devices and applications. A number of network management protocols have been developed around the definition, exchange, and reporting associated with YANG data models. Currently, YANG represents the standard for defining network management information.

5.2.1. The YANG Data Model

The YANG model defines a schema for organizing and accessing a device's configuration or operational information. Once a model is developed, it is loaded to both the client and server, and serves as a contract between the two. A YANG model can be complex, describing many containers of managed elements, each providing methods for device configuration or reporting of operational state.

The YANG module itself is a flexible data model that could be used for capturing the autonomy models and other behaviors needed by the DTNMA. The YANG schema provides flexibility in the organization of data to the model developer. The YANG schema supports a broad range of data types noted in [RFC6991]. YANG supports the definition of parameterized Remote Procedure Calls (RPCs) to be executed on

managed nodes as well as the definition of push notifications within the model.

The YANG modeling language continues to evolve as new features are needed by adopting management protocols. Two evolving features that might be useful in the DTNMA are notifications and schema identifiers.

- YANG notifications [RFC8639] and YANG-Push notifications [RFC8641] allow a client to subscribe to the delivery of specific containers or data nodes defined in the model, either on a periodic or "on change" basis. These notification events can be filtered according to XPath [xpath] or subtree [RFC6241] filtering as described in [RFC8639] Section 2.2.

- YANG Schema Item iDentifiers (SIDs) [I-D.ietf-core-sid] are proposed to be 63-bit identifiers used for more efficiently identification of YANG data elements for use in constrained environments.

While the YANG model is currently the standard way to describe management data, there are concerns with its unmodified use in the DTNMA, as follows.

1. Size. Data nodes within a YANG model are referenced by a verbose, string-based path of the module, sub-module, container, and any data nodes such as lists, leaf-lists, or leaves, without any explicit hierarchical organization based on data or object type. Existing efforts to make compressed identifies for YANG objects (such as SIDs) are still relatively verbose (~8 bytes per item) and do not natively support ways to glob multiple SIDs.

2. Protocol Coupling. A significant amount of existing YANG tooling presumes the use of YANG with a specific management protocol. The emergence of multiple YANG-based protocols may make these presumptions less problematic in the future. Work to more consistently identify different types of YANG modules and their use has been undertaken to disambiguate how YANG modules should be treated [RFC8199].

3. Agent Control. YANG RPCs execute commands on a device and generate an expected, structured response. RPC execution is strictly limited to those issued by the client. Commands are executed immediately and sequentially as they are received by the server, and there is no method to autonomously execute RPCs triggered by specific events or conditions.

5.2.2. YANG-Based Management Protocols

YANG defines the schema for data used by network management protocols such as NETCONF [RFC6241], RESTCONF [RFC8040], and CORECONF [I-D.ietf-core-comi]. These protocols provide the mechanisms to install, manipulate, and delete the configuration of network devices.

5.2.2.1. NETCONF

NETCONF is a stateful, XML-based protocol that provides a RPC syntax to retrieve, edit, copy, or delete any data nodes or exposed functionality on a server. It requires that underlying transport protocols support long-lived, reliable, low-latency, sequenced data delivery sessions.

NETCONF connections are required to provide authentication, data integrity, confidentiality, and replay protection through secure transport protocols such as SSH or TLS. A bi-directional NETCONF session must be established before any data transfer can occur. All of these requirements make NETCONF a poor choice for operating in a challenged network.

5.2.2.2. RESTCONF

RESTCONF is a stateless RESTful protocol based on HTTP. RESTCONF configures or retrieves individual data elements or containers within YANG data models by passing JSON over REST. This JSON encoding is used to GET, POST, PUT, PATCH, or DELETE data nodes within YANG modules.

RESTCONF is a stateless protocol because it presumes that it is running over a stateful secure transport (HTTP over TLS). Also, RESTCONF presumes that a single pull of information can be made in a single round-trip. In this way, RESTCONF is only stateless between queries - not internal to a single query.

5.2.2.3. CORECONF

CORECONF is an emerging stateless protocol built atop the Constrained Application Protocol (CoAP) [RFC7252] that defines a messaging construct developed to operate specifically on constrained devices and networks by limiting message size and fragmentation. CoAP also implements a request/response system and methods for GET, POST, PUT, and DELETE.

Currently, the CORECONF draft [I-D.ietf-core-comi] is archived and expired since 2021.

5.3. Autonomic Networking

The future of network operations requires more autonomous behavior including self-configuration, self-management, self-healing, and self-optimization. One approach to support this is termed Autonomic Networking [RFC7575].

In particular, there is a large and growing set of work within the IETF focused on developing an Autonomic Networking Integrated Model and Approach (ANIMA). The ANIMA work has developed a comprehensive reference model for distributing autonomic functions across multiple nodes in an autonomic networking infrastructure [RFC8993].

This work, focused on learning the behavior of distributed systems to predict future events, is an exciting and emerging network management capability. This includes the development of signalling protocols such as GRASP [RFC8990] and autonomic control planes [RFC8368].

Both autonomic and challenged networks require similar degrees of autonomy. However, challenged networks cannot provide the complex coordination between nodes and distributed supporting infrastructure necessary for the frequent data exchanges for negotiation, learning, and bootstrapping associated with the above capabilities.

There is some emerging work in ANIMA as to how disconnected devices might join and leave the autonomic control plane over time. However, this work is solving an important, but different, problem than that encountered by challenged networks.

6. Motivation for New Features

The future of network management will involve autonomous and autonomic functions operating on both managed and managing devices. However, the development of distributed autonomy for coordinated learning and event reaction is different from a managed device operating without connectivity to a managing node.

Management mechanisms that provide DTNMA desirable properties do not currently exist. This is not surprising since autonomous management in the context of a challenged networking environment is an emerging use case.

In particular, a management architecture is needed that provides the following new features.

1. Open Loop Control. Freedom from a request-response architecture, API, or other presumption of timely round-trip communications. This is particularly important when managing

networks that are not built over an HTTP or TCP/TLS infrastructure.

2. Standard Autonomy Model. An autonomy model that allows for standard expressions of policy to guarantee deterministic behavior across devices and vendor implementations.

3. Compressible Model Structure. A data model that allows for very compact encodings by defining and exploiting common elements of data schemas.

Combining these new features with existing mechanisms for message data exchange (such as BP), data representations (such as CBOR) and data modeling languages (such as YANG) will form a pragmatic approach to defining challenged network management.

7. Reference Model

There are a multitude of ways in which both existing and emerging network management protocols, APIs, and applications can be integrated for use in challenged environments. However, expressing the needed behaviors of the DTNMA in the context of any of these pre-existing elements risks conflating systems requirements, operational assumptions, and implementation design constraints.

7.1. Important Concepts

This section describes a network management concept for challenged networks (generally) and those conforming to the DTN architecture (in particular). The goal of this section is to describe how DTNMA services provide DTNMA desirable properties.

NOTE: This section assumes a BPv7 underlying network transport. Bundles are the baselined transport protocol data units of the DTN architecture. Additionally, they may be used in a variety of network architectures beyond the DTN architecture. Therefore, assuming bundles is a convenient way of scoping DTNMA to any network or network architecture that relies on BPv7 features.

Similar to other network management architectures, the DTNMA draws a logical distinction between a managed device and a managing device. Managed devices use a DA to manage resident applications. Managing devices use a DM to both monitor and control DAs.

NOTE: The terms "managing" and "managed" represent logical characteristics of a device and are not, themselves, mutually exclusive. For example, a managed device might, itself, also manage some other device in the network. Therefore, a device may support either or both of these characteristics.

The DTNMA differs from some other management architectures in three significant ways, all related to the need for a device to self-manage when disconnected from a managing device.

1. Pre-shared Definitions. Managing and managed devices should operate using pre-shared data definitions and models. This implies that static definitions should be standardized whenever possible and that managing and managed devices may need to negotiate definitions during periods of connectivity.

2. Agent Self-Management. A managed device may find itself disconnected from its managing device. In many challenged networking scenarios, a managed device may spend the majority of its time without a regular connection to a managing device. In these cases, DAs manage themselves by applying pre-shared policies received from managing devices.

3. Command-Based Management. Managing devices communicate with managed devices through an envisioned command and control interface. Unlike other network management approaches where managers locally construct datastores and databases for bulk updates, the DTNMA presumes that managed device databases are managed through a command-based interface. This, in part, is driven by the need for DAs to receive updates from both remote management devices and local autonomy.

7.2. Model Overview

A DTNMA reference model is provided in Figure 1 below. In this reference model, applications and services on a managing device communicate with a DM which uses pre-shared definitions to create a set of directives that can be sent to a managed device's DA. The DA provides local monitoring and control of the applications and services resident on the managed device. The DA also performs local data fusion as necessary to synthesize data products (such as reports) that can be sent back to the DM when appropriate.

DTNMA Reference Model

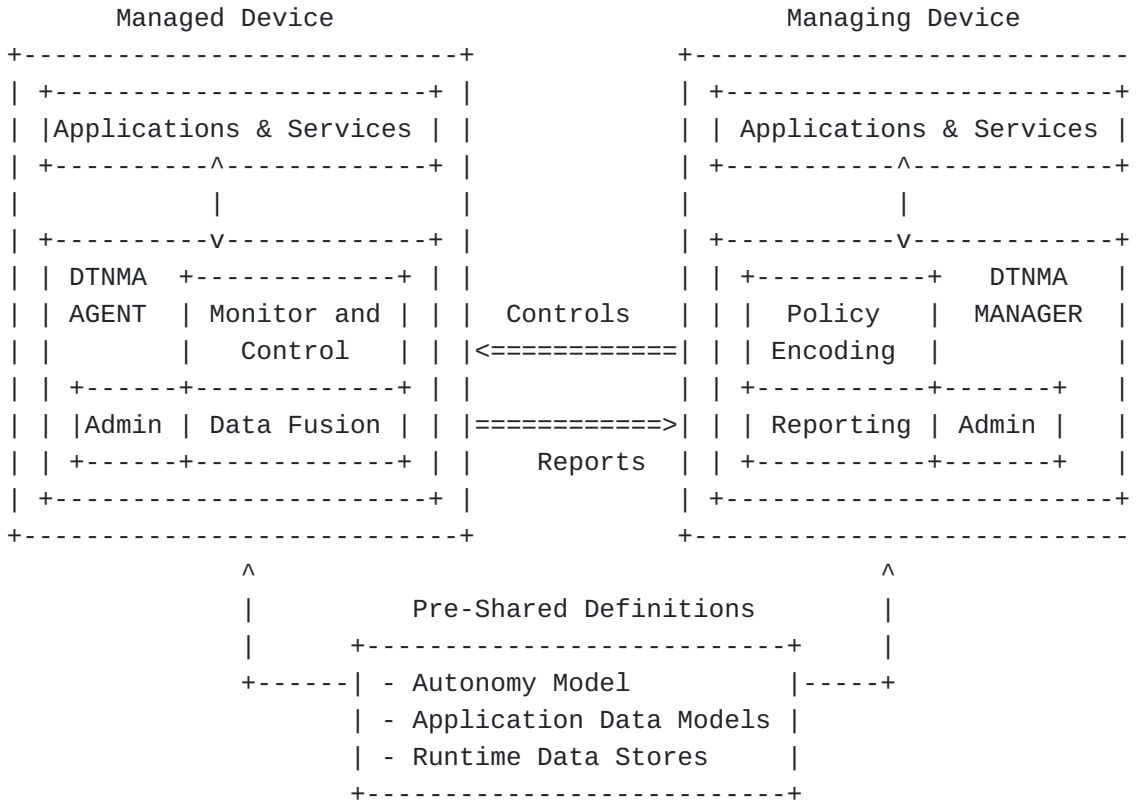

Figure 1

This model preserves the familiar concept of "managers" resident on managing devices and "agents" resident on managed devices. However, the DTNMA model is unique in how the DM and DA operate. The DM is used to pre-configure DAs in the network with management policies. it is expected that the DAs, themselves, perform monitoring and control functions on their own. In this way, a properly configured DA may operate without a timely, reliable connection back to a DM.

7.3. Functional Elements

The reference model illustrated in Figure 1 implies the existence of certain logical elements whose roles and responsibilities are discussed in this section.

7.3.1. Managed Applications and Services

By definition, managed applications and services reside on a managed device. These software entities can be controlled through some interface by the DA and their state can be sampled as part of periodic monitoring. It is presumed that the DA on the managed device has the proper data model, control interface, and permissions to alter the configuration and behavior of these software applications.

7.3.2. DTNMA Agent (DA)

A DA resides on a managed device. As is the case with other network management approaches, this agent is responsible for the monitoring and control of the applications local to that device. Unlike other network management approaches, the agent accomplishes this task without a regular connection to a DTNMA Manager.

The DA performs three major functions on a managed device: the monitoring and control of local applications, production of data analytics, and the administrative control of the agent itself.

7.3.2.1. Monitoring and Control

DAs monitor the status of applications running on their managed device and selectively control those applications as a function of that monitoring. The following components are used to perform monitoring and control on an agent.

Rules Database
: A DA monitors the state of the managed device looking for pre-defined stimuli and, when encountered, issuing a pre-defined response. The tuple of stimulus-response is termed a "rule". Within the DTNMA, these rules are the embodiment of policy expressions received from DMs and evaluated at regular intervals by the autonomy engine. The rules database is the collection of active rules known to the DA.

Autonomy Engine
: The DA autonomy engine is configured with policy expressions describing expected reactions to potential events. This engine is configured by managers during periods of connectivity. Once configured, the engine may function without other access to any managing device. This engine may also reconfigure itself as a function of policy.

Application Control Interfaces
: DAs must support control interfaces for all managed applications. Control interfaces are used to alter the configuration and behavior of an application. These interfaces may be custom for each application, or as provided through a common framework such as provided by an operating system.

7.3.2.2. Data Fusion

DAs generate new data elements as a function of the current state of the managed device and its applications. These new data products may take the form of individual data values, or new collections of data used for reporting. The logical components responsible for these behaviors are as follows.

Application Data Interfaces

DAs must support mechanisms by which important state is retrieved from various applications resident on the managed device. These data interfaces may be custom for each application, or as provided through a common framework such as provided by an operating system.

Data Value Generators

DAs may support the generation of new data values as a function of other values collected from the managed device. These data generators may be configured with descriptions of data values and the data values they generate may be included in the overall monitoring and reporting associated with the managed device.

Report Generators

DAs may, as appropriate, generate collections of data values for transmission to managers. Reports can be generated as a matter of policy or in response to the handling of critical events (such as errors), or other logging needs. The generation of a report is independent of whether there exists any connectivity between a DA and a DM. It is assumed that reports are queued on an agent pending transmit opportunities.

7.3.2.3. Administration

DAs must perform a variety of administrative services in support of their configuration. The significant such administrative services are as follows.

Manager Mapping

The DTNMA allows for a many-to-many relationship amongst DTNMA Agents and Managers. A single DM may configure multiple DAs, and a single DA may be configured by multiple DMs. Multiple managers may exist in a network for at least two reasons. First, different managers may exist to control different applications on a device. Second, multiple managers increase the likelihood of an agent encountering a manager when operating in a sparse or challenged environment.

Data Verifiers

DAs might handle large amounts of data produced by various sources, to include data from local managed applications, remote managers, and self-calculated values. DAs should ensure, when possible, that externally generated data values have the proper

syntax (e.g., data type and ranges) and any required integrity and confidentiality.

Access Controllers

DAs support authorized access to the management of individual applications, to include the administrative management of the agent itself. This means that a manager may only set policy on the agent pursuant to verifying that the manager is authorized to do so.

7.3.3. Managing Applications and Services

Managing applications and services reside on a managing device and serve as the both the source of DA policy statements and the target of DA reporting. They may operate with or without an operator in the loop.

Unlike management applications in unchallenged networks, these applications cannot exert timely closed-loop control over any managed device application. Instead, these applications must be built to exercise open-loop control by producing policies that can be configured and enforced on managed devices by DAs.

NOTE: Closed-loop control in this context refers to the practice of waiting for a response from a managed device prior to issuing new directives to that device. These "loops" may be closed quickly (in milliseconds) or over much longer periods (hours, days, years). The alternative to closed-loop control is open-loop control, where responses from a managed device and directives to the managed device are independent of one another.

7.3.4. DTNMA Manager (DM)

A DM resides on a managing device. This manager provides an interface between various managing applications and services and the DAs that enforce their policies. In providing this interface, DMs translate between whatever native interface exists to various managing applications and the autonomy models used to encode management policy.

The DM performs three major functions on a managing device: policy encoding, reporting, and administration.

7.3.4.1. Policy Encoding

DMs translate policy directives from managing applications and services into standardized policy expressions that can be recognized by DAs. The following logical components are used to perform this policy encoding.

Application Control Interfaces

DMs must support control interfaces for managing applications. These control interfaces are used to receive desired policy statements from applications. These interfaces may be custom for each application, or provided through a common framework, protocol, or operating system.

Policy Encoders

DAs implement a standardized autonomy model comprising standardized data elements. The open-loop control structures provided by managing applications must be represented in this common language. Policy encoders perform this encoding function.

Policy Aggregators

DMs collect multiple encoded policies into messages that can be sent to DAs over the network. This implies the proper addressing of agents and the creation of messages that support store-and-forward operations. It is recommended that control messages be packaged using BP bundles when there may be intermittent connectivity between DMs and DAs.

7.3.4.2. Reporting

DMs receive reports on the status of managed devices during periods of connectivity with the DAs on those devices. The following logical components are needed to implement reporting capabilities on a DM.

Report Collectors

DMs receive reports from DAs in an asynchronous manner. This means that reports may be received out of chronological order and in ways that are difficult or impossible to associate with a specific policy from a managing application. DMs collect these reports and extract their data in support of subsequent data analytics.

Data Analyzers

DMs review sets of data reports from DAs with the purpose of extracting relevant data to communicate with managing applications. This may include simple data extraction or may include more complex processing such as data conversion, data fusion, and appropriate data analytics.

Application Data Interfaces

DMs must support mechanisms by which data retrieved from agent may be provided back to managing devices. These interfaces may be custom for each application, or as provided through a common framework, protocol, or operating system.

7.3.4.3. Administration

Managers in the DTNMA must perform a variety of administrative services in support of their proper configuration and operation. This includes the following logical components.

Agent Mappings
The DTNMA allows DMs to communicate with multiple DAs. However, not every agent in a network is expected to support the same set of Application Data Models or otherwise have the same set of managed applications running. For this reason, DMs must determine individual DA capabilities to ensure that only appropriate controls are sent to a DA.

Data Verifiers
DMs handle large amounts of data produced by various sources, to include data from managing applications and DAs. DMs should ensure, when possible, that data values received from DAs over a network have the proper syntax (e.g., data type and ranges) and any required integrity and confidentiality.

Access Controllers
DMs should only send controls to agents when the manager is configured with appropriate access to both the agent and the applications being managed.

7.3.5. Pre-Shared Definitions

A consequence of operating in a challenged environment is the potential inability to negotiate information in real-time. For this reason, the DTNMA requires that managed and managing devices operate using pre-shared definitions rather than relying on data definition negotiation.

The three types of pre-shared definitions in the DTNMA are the DA autonomy model, managed application data models, and any runtime data shared by managers and agents.

Autonomy Model
A DTNMA autonomy model represents the data elements and associated autonomy structures that define the behavior of the agent autonomy engine. A standardized autonomy model allows for individual implementations of DAs, and DMs to interoperate. A standardized model also provides guidance to the design and implementation of both managed and managing applications.
NOTE: A standardized autonomy model is required for the interoperable encoding of policy statements. However, the DTNMA does not standardize a specific transport of those policy statements between agents and managers. The DTNMA also does not specify any transport-related encoding.

Application Data Models

As with other network management architectures, the DTNMA presupposes that managed applications (and services) define their own data models. These data models include the data produced by, and controls implemented by, the application. These models are expected to be static for individual applications and standardized for applications implementing standard protocols.

Runtime Data Stores

Runtime data stores, by definition, include data that is defined at runtime. As such, the data is not pre-shared prior to the deployment of DMs and DAs. Pre-sharing in this context means that DMs and DAs are able to define and synchronize data elements prior to their operational use in the system. This synchronization happens during periods of connectivity between DMs and DAs.

8. Desired Services

This section provides a description of the services provided by DTNMA elements on both managing and managed devices. These service descriptions differ from other management descriptions because of the unique characteristics of the DTNMA operating environment.

Predicate autonomy, asynchronous data transport, and intermittent connectivity require new techniques for device management. Many of the services discussed in this section attempt to provide continuous operation of a managed device through periods of no connectivity.

8.1. Local Monitoring and Control

DTNMA monitoring is associated with the agent autonomy engine. The term monitoring implies timely and regular access to information such that state changes may be acted upon within some response time period. Within the DTNMA, connections between a managed and managing device are unable to provide such a connection and, thus, monitoring functions must be handled on the managed device.

Predicate autonomy on a managed device should collect state associated with the device at regular intervals and evaluate that collected state for any changes the require a preventative or corrective action. Similarly, this monitoring may cause the device to generate one or more reports destined to the managing device.

Similar to monitoring, DTNMA control results in actions by the agent to change the state or behavior of the managed device. All control in the DTNMA is local control. In cases where there exists a timely connection to a manager, received controls a are still run through the autonomy engine. In this case, the stimulus is the direct receipt of the control and the response is to immediately run the

control. In this way, there is never a dependency on a session or other stateful exchange with any remote entity.

8.2. Local Data Fusion

DTNMA Fusion services produce new data products from existing state on the managed device. These fusion products can be anything from simple summations of sampled counters to complex calculations of behavior over time.

Fusion is an important service in the DTNMA because fusion products are part of the overall state of a managed device. Complete knowledge of this overall state is important for the management of the device, particularly in a stimulus-response system whose stimuli are evaluated against this state.

In-situ data fusion is an important function as it allows for the construction of intermediate summary data, the reduction of stored and transmitted raw data, and otherwise insulates the data source from conclusions drawn from that data.

While some fusion is performed in any management system, the DTNMA requires fusion to occur on the managed device itself. If the network is partitioned such that no connection to a managing device is available, fusion must happen locally. Similarly, connections to a managing device might not remain active long enough for round-trip data exchange or may not have the bandwidth to send all sampled data.

NOTE: While data fusion is an important function within the DTNMA, it is expected that the storage and transmission of raw (or pre-fused) data remains a capability of the system. In particular, raw data can be useful for debugging managed devices, understanding complex interactions and underlying conditions, and tuning for better performance and/or better outcomes.

8.3. Remote Configuration

DTNMA configuration services must update the local configuration of a managed device with the intent to impact the behavior and capabilities of that device. The change of device configurations is a common service provided by many network management systems. The DTNMA has a unique approach to configuration for the following reasons.

The DTNMA configuration service is unique in that the selection of managed device configurations must occur, itself, as a function of the state of the device. This implies that management proxies on the device store multiple configuration functions that can be applied as needed without consultation from a managing device.

This approach differs from the management concept of selecting from multiple datastores in that DTNMA configuration functions can target individual data elements and can calculate new values from local device state.

When detecting stimuli, the agent autonomy engine must support a mechanism for evaluating whether application monitoring data or runtime data values are recent enough to indicate a change of state. In cases where data has not been updated recently, it may be considered stale and not used to reliably indicate that some stimulus has occurred.

8.4. Remote Reporting

DTNMA reporting services collect information known to the managed device and prepare it for eventual transmission to one or more managing devices. The contents of these reports, and the frequency at which they are generated, occurs as a function of the state of the managed device, independent of the managing device.

Once generated, it is expected that reports might be queued pending a connection back to a managing device. Therefore, reports must be differentiable as a function of the time they were generated.

When reports are sent to a managing device over a challenged network, they may arrive out of order due to taking different paths through the network or being delayed due to retransmissions. A managing device should not infer meaning from the order in which reports are received, nor should a given report be associated with a specific control or autonomy action on a given managed device.

8.5. Authorization

Both local and remote services provided by the DTNMA affect the behavior of multiple applications on a managed device and may interface with multiple managing devices. It is expected that transport protocols used in any DTNMA implementation support security services such as integrity and confidentiality.

Authorization services enforce the potentially complex mapping of other DTNMA services amongst managed and managing devices in the network. For example, fine-grained access control can determine which managing devices receive which reports, and what controls can be used to alter which managed applications.

This is particularly beneficial in networks that either deal with multiple administrative entities or overlay networks that cross administrative boundaries. Allowlists, blocklists, key-based infrastructures, or other schemes may be used for this purpose.

9. Logical Autonomy Model

An important characteristic of the DTNMA is the shift in the role of a managing device. In the DTNMA, managers configure the autonomy engines on agents, and it is the agents that provide local device management. One way to describe the behavior of the agent autonomy engine is to describe the characteristics of the autonomy model it implements.

This section describes a logical autonomy model in terms of the abstract data elements that would comprise the model. Defining abstract data elements allows for an unambiguous discussion of the behavior of an autonomy model without mandating a particular design, encoding, or transport associated with that model.

9.1. Overview

Managing autonomy on a potentially disconnected device must behave in both an expressive and deterministic way. Expressivity allows for the model to be configured for a wide range of future situations. Determinism allows for the forensic reconstruction of device behavior as part of debugging or recovery efforts.

The DTNMA autonomy model is built on a stimulus-response model in which the autonomy system responses to pre-identified stimuli with pre-configured responses. Stimuli are identified using simple predicate logic that examines aspects of the state of the managed device. Responses are implemented by running one or more procedures on the managed device.

As with many such systems, behavior can be captured using the construct:

IF stimulus THEN response

NOTE: The use of predicate logic and a stimulus-response system does not conflict with the use of higher-level autonomous function or the incorporation of machine learning. The DTNMA recommended autonomy model allows for the use of higher levels of autonomous function as moderated and controlled by a more deterministic base autonomy system.

By allowing for a multi-tier autonomy system, the DTNMA may increase the adoption of higher-functioning autonomy because of the reporting, control, and determinism of the underlying predicate system.

DTNMA Autonomy Model

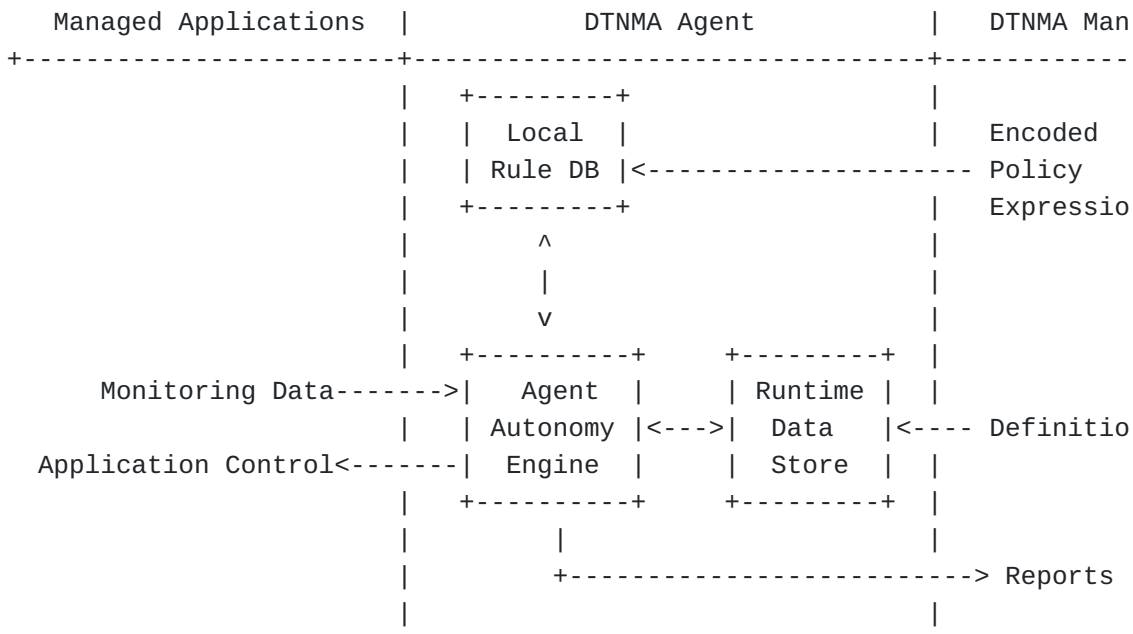

Figure 2

The flow of data into and out of the agent autonomy engine is illustrated in Figure 2. In this model, the autonomy engine stores the combination of stimulus conditions and associated responses as a set of "rules" in a rules database. This database is updated through the execution of the autonomy engine and as configured from policy statements received by managers.

Stimuli are detected by examining the state of applications as reported through application monitoring interfaces and through any locally-derived data. Local data is calculated in accordance with definitions also provided by managers as part of the runtime data store.

Responses to stimuli are run as updated to the rules database, updated to the runtime data store, controls sent to applications, and the generation of reports.

9.2. Model Characteristics

There are a number of ways to represent data values, and many data modeling languages exist for this purpose. When considering how to model data in the context of the DTNMA autonomy model there are some modeling features that should be present to enable functionality. There are also some modeling features that should be prevented to avoid ambiguity.

Traditional network management approaches favor flexibility in their data models. The DTNMA stresses deterministic behavior that supports forensic analysis of agent activities "after the fact". As such, the

following statements should be true of all data representations relating to DTNMA autonomy.

* Strong Typing - The predicates and expressions that comprise the autonomy services in the DTNMA should require strict data typing. This avoids errors associated with implicit data conversions and helps detect misconfiguration.

* Acyclic Dependency - Many dependencies exist in an autonomy model, particularly when combining individual expressions or results to create complex behaviors. Implementations that conform to the DTNMA must prevent circular dependencies.

* Fresh Data - Autonomy models operating on data values presume that their data inputs represent the actionable state of the managed device. If a data value has failed to be refreshed within a time period, autonomy might incorrectly infer an operational state. Regardless of whether a data value has changed, DTNMA implementations must provide some indicator of whether the data value is "fresh" meaning that is still represents the current state of the device.

* Pervasive Parameterization - Where possible, autonomy model objects should support parameterization to allow for flexibility in the specification. Parameterization allows for the definition of fewer unique model objects and also can support the substitution of local device state when exercising device control or data reporting.

* Configurable Cardinality - The number of data values that can be supported in a given implementation is finite. For devices operating in challenged environments, the number of supported objects may be far fewer than that which can be supported by devices in well-resourced environments. DTNMA implementations should define limits to the number of supported objects that can be active in a system at one time, as a function of the resources available to the implementation.

* Control-Based Updates - The agent autonomy engine changes the state of the managed device by running controls on the device. This is different from other approaches where the behavior of a managed device is updated only by updated configuration values, such as in a table or datastore. Altering behavior via one or more controls allows checking all pre-conditions before making changes as well as providing more granularity in the way in which the device is updated. Where necessary, controls can be defined to perform bulk updated of configuration data so as not to lose that update modality.

9.3. Data Value Representation

The expressive representation of data values is fundamental to the successful construction and evaluation of predicates in the DTNMA autonomy model. This section describes the characteristics of data representation for this model, both as individual data values and ways to aggregate these values into collections.

There is a useful distinction that can be made regarding the way in which data values are assigned in the context of an autonomy system. This section discusses four categories of assigning strategies and proposes mnemonics to differentiate each.

NOTE: The assignment and naming of data values are different from the base type of the data value. The DTNMA assumes common data types (e.g., integer, real, string, byte) would be supported in any operational autonomy model.

The four categories of value assignment can be derived by determining whether values are calculated internal or external to the autonomy model and whether, once calculated, these values can be changed.

	Immutable	**Mutable**
Internally Defined	CONST	VAR
Externally Defined	LIT	EDD

Table 1: Data Value Categories and Mnemonics

Constants (CONST) - Constant data values are named values that are defined in the context of the autonomy model. Both the name and the value of the constant are fixed and cannot be changed. An example of a constant would be defining the numerical value PI to 2 digits of precision (PI_2_DIGITS = 3.14).

Literals (LIT) - Literal data values are those whose name and value are the same. These values are used to represent atomic values that are too simple to be represented a constant. For example, the number 4 is a literal value. The name "4" and the value 4 are the same and inseparable. Literal values cannot change ("4" could not be used to mean 5) and they are defined external to the autonomy model (the autonomy model is not expected to redefine what 4 means).

Variables (VAR) - Variables are named data values defined by the autonomy model itself. They can be added and removed as a function of the function of the autonomy model, and the autonomy model is the sole determiner of their value. An example of a variable in an

autonomy model would be the number of times that a particular predicate evaluated to true.

Externally-Defined Data (EDD) - External data values are those provided to the autonomy model from its hosting environment. These values are the foundation of state-based autonomy as they capture the state of the managed device. The autonomy model treats these values as read-only inputs. Examples of externally defined values include temperature sensor readings and the instantaneous data rate from a radio.

9.4. Data Reporting

The DTNMA autonomy model should, as required, report on the state of its managed device (to include the state of the model itself). This reporting should be done as a function of the changing state of the managed device, independent of the connection to any managing device. Queuing reports allows for later forensic analysis of device behavior, which is a desirable property of DTNMA management.

There are at least four useful categories of reporting mechanism that should be present in the DTNMA These categories can be distinguished by whether the reported data share a common structure or not, and whether the report mechanism represents a scheme or data adherent to that schema.

	Schema	Values
Common Structure	TBLT	TBL
Mixed Structure	RPTT	RPT

Table 2: Data Reporting Mechanisms and Mnemonics

9.4.1. Tabular Reports (TBLs) and Tabular Report Templates (TBLTs)

Relational database tables provide collection, filtering, and reporting efficiencies when representing series of data collections that share a common syntactic structure and semantic meaning. Tables have a fixed structure identified by one or more vertical columns. They are populated by zero or more data collections, with one row per represented data collection.

To the extent that DTNMA reporting includes data collections similarly adhering to a common structure, these reports can be modeled similarly to tables. Such reports are called Tabular Reports (TBLs).

Every TBL is populated in accordance to a pre-defined schema, which is termed the Tabular Report Template (TBLT). This template defines

the columns that comprise the TBL and associated constraints on data values for those columns.

Dissimilar to relational database tables, TBLs are reporting mechanisms. They represent a report generated at a specific moment in time. Therefore, a managed device may produce and queue for transmission multiple TBLs for the same TBLT.

9.4.2. Reports (RPT) and Report Templates (RPTT)

Not all reportable data collections are efficiently represented in a tabular structure. In cases where there is no processing or encoding advantage to a tabular report, a non-tabular representation is needed. This representation is termed the DTNMA report (RPT).

A RPT is a snapshot of a collection of data values at a given moment in time. The type, number, order, and other details of these data values is given by a schema called the Report Template (RPTT).

Separating the structure (RPTT) and content (RPT) of a general purpose reporting mechanism reduces the size of generated traffic, which is an important property of the DTNMA.

9.5. Command Execution

The agent autonomy engine requires that managed devices issue commands on themselves as if they were otherwise being controlled by a managing device. The ability to support this type of commanding in the autonomy model is one of the unique requirements of the DTNMA. This approach is not dissimilar to the concept of Remote Procedure Calls (RPCs) that are sometimes used in low-latency, high-availability approaches to network management mechanisms.

Command execution in the DTNMA happens through the use of controls and macros.

Controls (CTRL) - A control represents a parameterized, predefined procedure that is run by the agent autonomy engine. CTRLs are conceptually similar to RPCs in that they represent parameterized functions run on the managed device. However, they are conceptually dissimilar from RPCs in that they do not have a concept of a return code as they must operate over an asynchronous transport. The concept of return code in an RPC implies a synchronous relationship between the caller of the procedure and the procedure being called, which might not be possible within the DTNMA.

NOTE: The use of the term Control in the DTNMA is derived in part from the concept of Command and Control (C2) where control implies the operational instructions that must be undertaken to implement (or maintain) a commanded objective. The DA autonomy engine controls

a managed device to allow it to fulfill some purpose as commanded by a (possibly disconnected) managing device.

For example, attempting to maintain a safe internal thermal environment for a spacecraft is considered "thermal control" (not "thermal commanding") even though thermal control involves sending commands to heaters, louvers, radiators, and other temperature-affecting components.

Even when CTRLs are received from a managing device with the intent to be run immediately, the control-vs-command distinction still applies. The CTRL run on the managed device is in service of the command received from the managing device to immediately change the local state of the device.

The success or failure of a CTRL may be handled locally by the agent autonomy engine. Otherwise, the externally observable impact of a CTRL can be understood through the generation and eventual examination of data reports produced by the managed device.

Macros (MACRO) - A Macro represents an ordered sequence of CTRLs execution. They may be implemented as a set of CTRLs, or as a mixed set of both MACRO and CTRL objects. Similar to CTRLs, a MACRO object should support parameterization and should not support a return code back to a caller.

9.6. Predicate Autonomy

The core function of the agent autonomy engine is to apply predetermined responses to predetermined state on a managed device. This involves the ability to calculate predicate expressions and the ability to associate the positive evaluation of these expressions with command execution.

9.6.1. Expressions

There are a few instances within the DTNMA autonomy model where a value must be calculated by the model itself, to include the following.

- *Calculating the value of a VAR.

- *Evaluating a predicate to see if it is true.

In cases such as these, the DTNMA must support an efficient, configurable syntax for defining expressions, calculating the value of these expressions based on the local state of the managed device, and using the calculated value in an appropriate way.

Expression (EXPR) - An Expression is a combination of operators and operands used to construct a numerical value from a series of other data values in the autonomy model.

Operator (OP) - An Operator represents a operation performed on at least one operand and returning a single result that, itself, can be used as an operand to some other operator. OPs may represent simple (+, -) or complex (sin, avg) mathematical functions or custom functions defined for the managed device.

Operands may be built from any autonomy model object that can be associated with a data value, to include the CONST, LIT, VAR, and EDD types, the result of an OP, and the result of a fully evaluated EXPR.

Predicate Expression (PRED) - A Predicate Expression is an EXPR whose evaluated data value is interpreted in a logical way as being either true or false.

9.6.2. Rules

A stimulus-response system associated stimulus detection with a commanded response. In the DTNMA, this relationship is captured through the definition of rules. These rules may be defined as focused on either the state of the managed device or optimized to only examine how time has passed on the managed device.

State-Based Rules (SBRs) - A state-based rule is one whose stimulus is indicated when a given PRED evaluates to true. Since the PRED is a combination of sampled and calculated data values on the managed device, evaluation of the PRED is evaluating the relevant state of the device. A SBR is one of the form:

IF PRED THEN MACRO

Time-Based Rules (TBRs) - A time-based rule is a specialization of a SBR that is optimized to only consider the passage of time on the managed device. A TBR is one of the form:

EVERY interval THEN MACRO

10. Use Cases

Using the autonomy model mnemonics defined in Section 9, this section describes flows through sample configurations conforming to the DTNMA. These use cases illustrate remote configuration, local monitoring and control, multiple manager support, and data fusion.

10.1. Notation

The use cases presented in this section are documented with a shorthand notation to describe the types of data sent between managers and agents. This notation, outlined in Table 3, leverages the mnemonic definitions of autonomy model elements defined in Section 9.

Term	Definition	Example
ID	DTNMA Object Identifier.	V1, EDD2
EDD#	Enumerated EDD definition.	EDD1
V#	Enumerated VAR definition.	V1 = EDD1 + V0
ACL#	Enumerated Access Control List.	ACL1
DEF([ACL],ID,EXPR)	Define ID from expression. Allow managers in ACL to see this ID.	DEF([ACL1], V1, EDD1 + EDD2)
PROD(P,ID)	Produce ID according to predicate P. P may be a time period (1s) or an expression (EDD1 > 10).	PROD(1s, EDD1)
RPT(ID)	A report containing data named ID.	RPT(EDD1)

Table 3: Terminology

These notations do not imply any implementation approach. They only provide a succinct syntax for expressing the data flows in the use case diagrams in the remainder of this section.

10.2. Serialized Management

This nominal configuration shows a single DM interacting with multiple DAs. The control flows for this scenario are outlined in Figure 3.

Serialized Management Control Flow

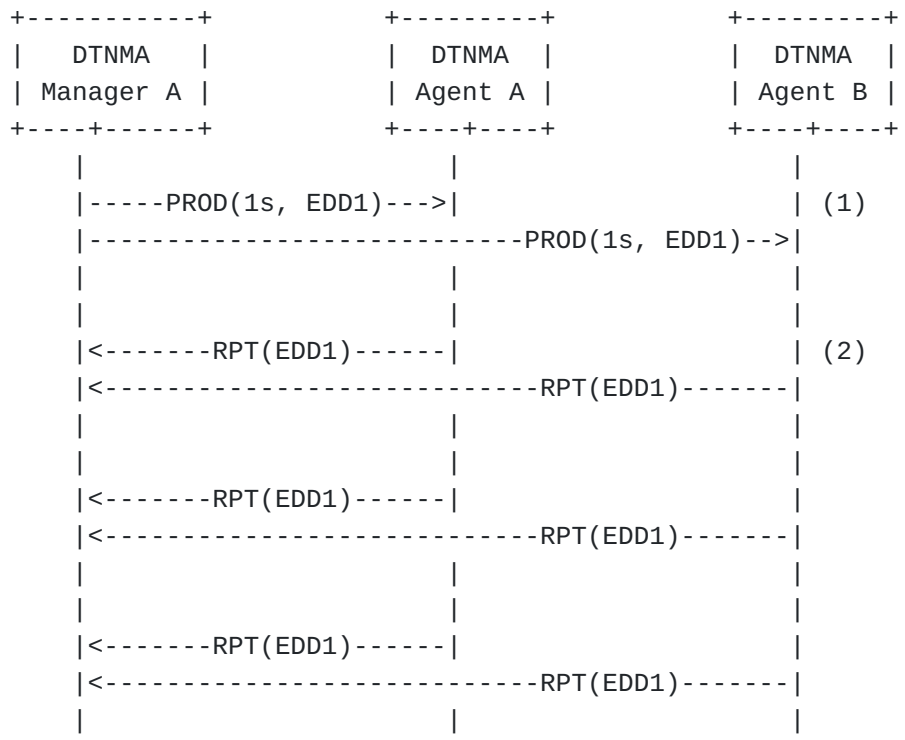

Figure 3

In a serialized management scenario, a single DM interacts with multiple DAs.

In this figure, the DTNMA Manager A sends a policy to DTNMA Agents A and B to report the value of an EDD (EDD1) every second in (step 1). Each DA receives this policy and configures their respective autonomy engines for this production. Thereafter, (step 2) each DA produces a report containing data element EDD1 and sends those reports back to the DM.

This behavior continues without any additional communications from the DM and without requiring a connection between the DA and DM.

10.3. Intermittent Connectivity

Building from the nominal configuration in Section 10.2, this scenario shows a challenged network in which connectivity between DTNMA Agent B and the DM is temporarily lost. Control flows for this case are outlined in Figure 4.

Challenged Management Control Flow

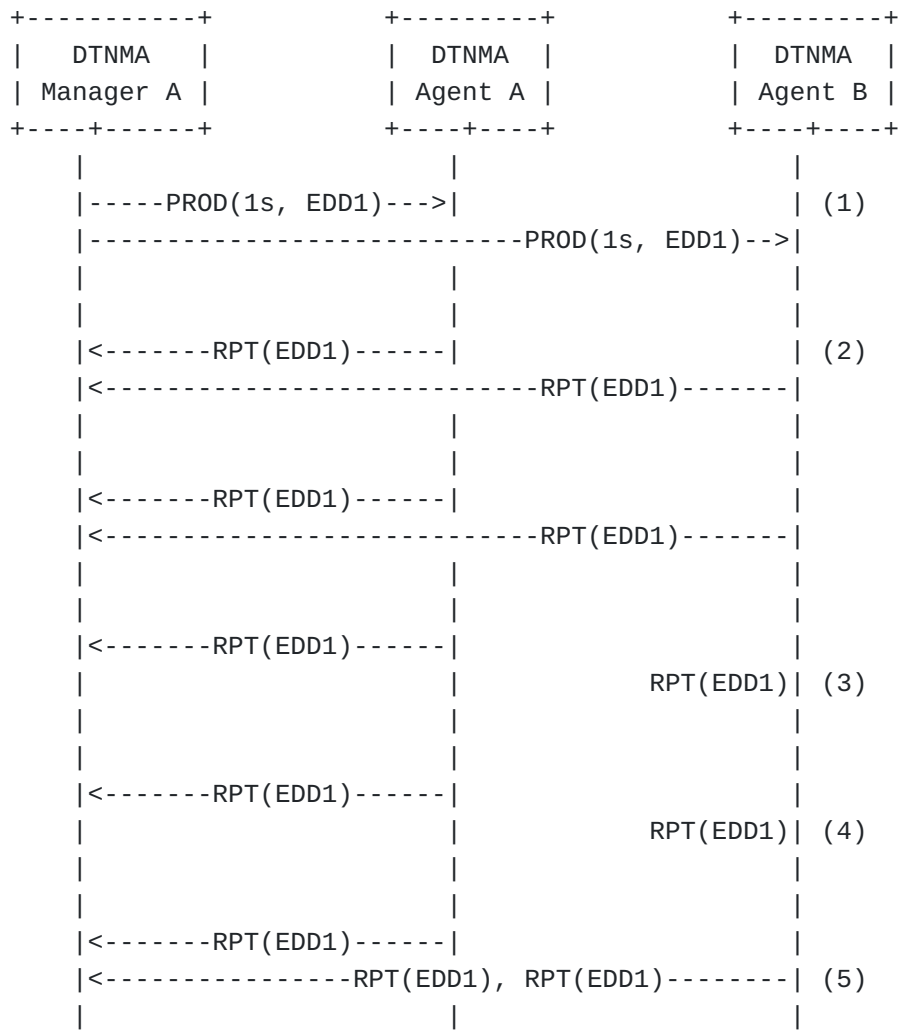

Figure 4

In a challenged network, DAs store reports pending a transmit opportunity.

In this figure, DTNMA Manager A sends a policy to DTNMA Agents A and B to produce an EDD (EDD1) every second in (step 1). Each DA receives this policy and configures their respective autonomy engines for this production. Produced reports are transmitted when there is connectivity between the DA and DM (step 2).

At some point, DTNMA Agent B loses the ability to transmit in the network (steps 3 and 4). During this time period, DA B continues to produce reports, but they are queued for transmission. This queuing might be done by the DA itself or by a supporting transport such as BP. Eventually, DTNMA Agent B is able to transmit in the network again (step 5) and all queued reports are sent at that time. DTNMA Agent A maintains connectivity with the DM during steps 3-5, and continues to send reports as they are generated.

10.4. Open-Loop Reporting

This scenario illustrates the DTNMA open-loop control paradigm, where DAs manage themselves in accordance with policies provided by DMs, and provide reports to DMs based on these policies.

The control flow shown in Figure 5, includes an example of data fusion, where multiple policies configured by a DM result in a single report from a DA.

Consolidated Management Control Flow

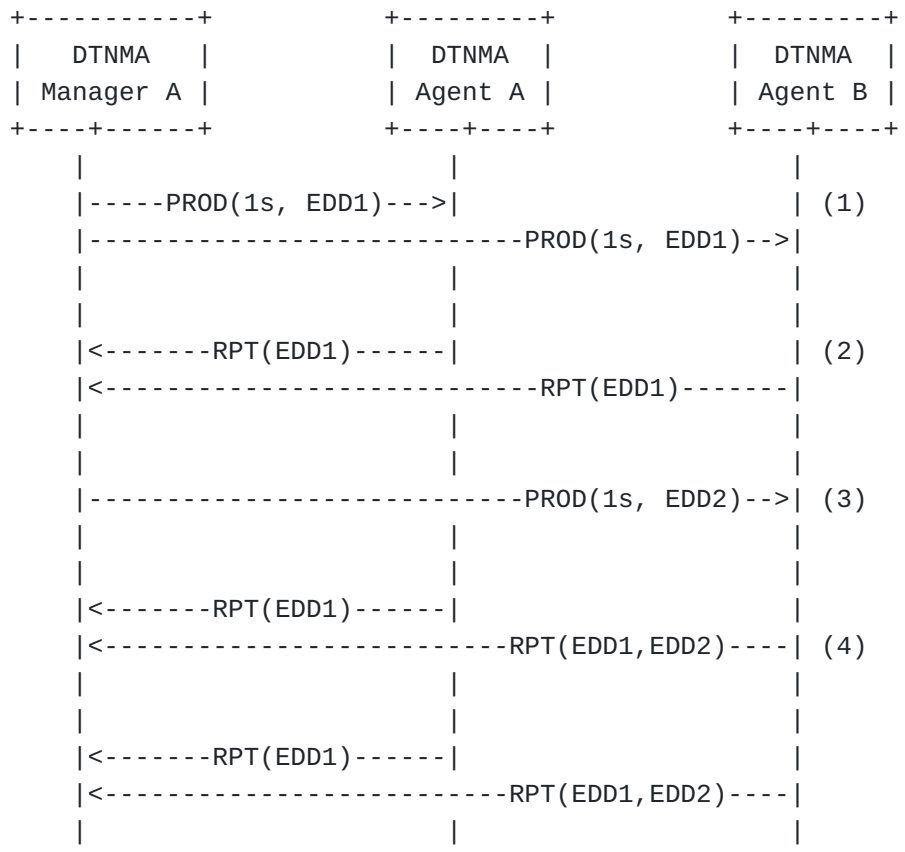

Figure 5

A many-to-one mapping between management policy and device state reporting is supported by the DTNMA.

In this figure, DTNMA Manager A sends a policy statement in the form of a rule to DTNMA Agents A and B, which instructs the DAs to produce a report with EDD1 every second (step 1). Each DA receives this policy, which is stored in its respective Rule Database, and configures its Autonomy Engine. Reports are transmitted by each DA when produced (step 2).

At a later time, DTNMA Manager A sends an additional policy to DTNMA Agent B, requesting the production of a report for EDD2 every second (step 3). This policy is added to DTNMA Agent B's Rule Database.

Following this policy update, DTNMA Agent A will continue to produce EDD1 and DTNMA Agent B will produce both EDD1 and EDD2 (step 4). However, DTNMA Agent B may provide these values to the DM in a single report rather than as 2 independent reports. In this way, there is no direct mapping between the single consolidated report sent by DTNMA Agent B (step 4) and the two different policies sent to DTNMA Agent B that caused that report to be generated (steps 1 and 3).

10.5. Multiple Administrative Domains

The managed applications on a DA may be controlled by different administrative entities in a network. The DTNMA allows DAs to communicate with multiple DMs in the network, such as in cases where there is one DM per administrative domain.

Whenever a DM sends a policy expression to a DA, that policy expression may be annotated with authorization information. One method of representing this is an ACL.

The use of an ACL in this use case does not imply the DTNMA requires ACLs to annotate policy expressions. ACLs in this context are for example purposes only.

The ability of one DM to access the results of policy expressions configured by some other DM will be limited to the authorization annotations of those policy expressions.

An example of multi-manager authorization is illustrated in Figure 6.

Multiplexed Management Control Flow

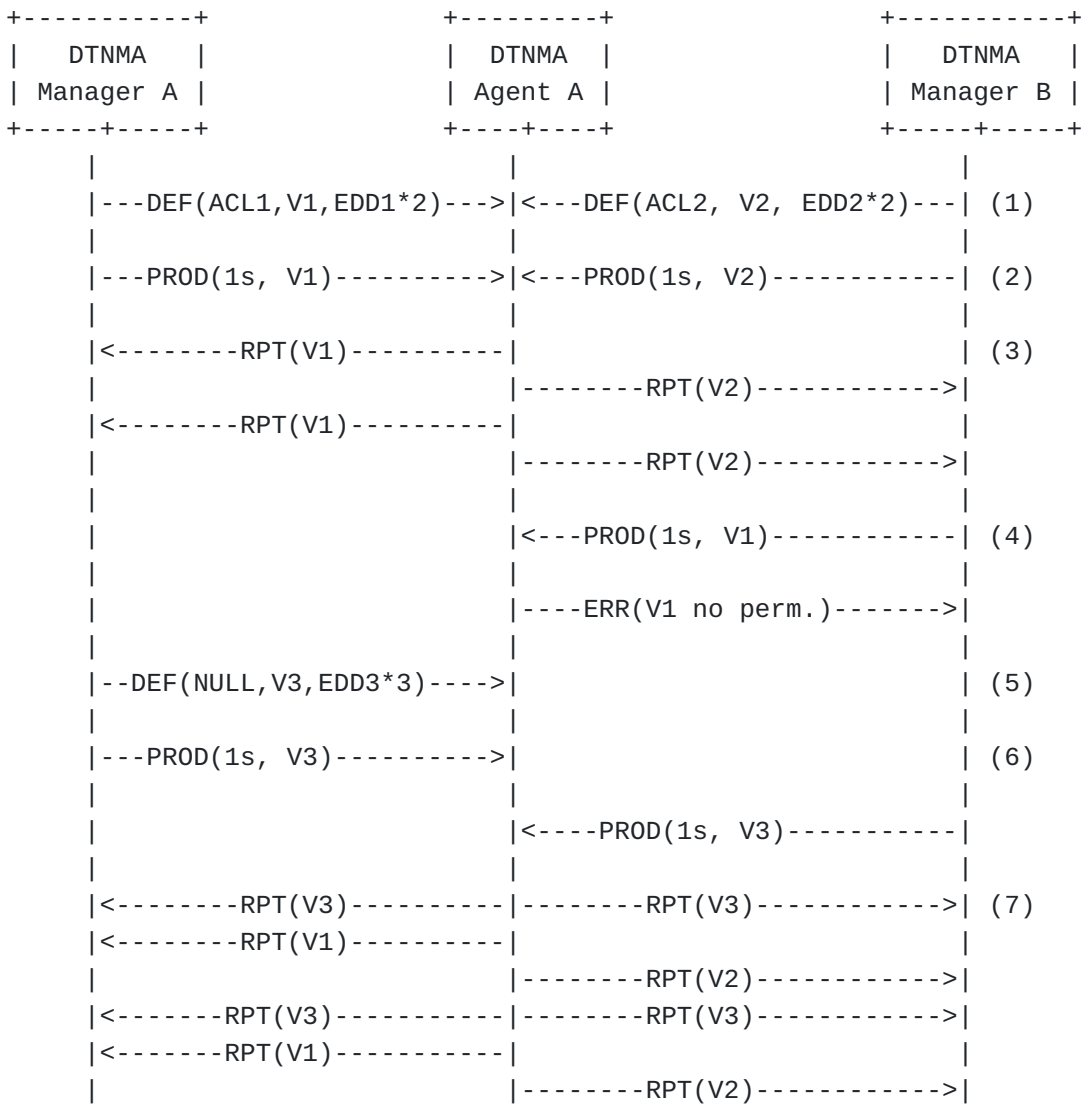

Figure 6

Multiple DMs may interface with a single DA, particularly in complex networks.

In this figure, both DTNMA Managers A and B send policies to DTNMA Agent A (step 1). DM A defines a VAR (V1) whose value is given by the mathematical expression (EDD1 * 2) and provides an ACL (ACL1) that restricts access to V1 to DM A only. Similarly, DM B defines a VAR (V2) whose value is given by the mathematical expression (EDD2 * 2) and provides an ACL (ACL2) that restricts access to V2 to DM B only.

Both DTNMA Managers A and B also send policies to DTNMA Agent A to report on the values of their VARs at 1 second intervals (step 2). Since DM A can access V1 and DM B can access V2, there is no authorization issue with these policies and they are both accepted

by the autonomy engine on Agent A. Agent A produces reports as expected, sending them to their respective managers (step 3).

Later (step 4) DM B attempts to configure DA A to also report to it the value of V1. Since DM B does not have authorization to view this VAR, DA A does not include this in the configuration of its autonomy engine and, instead, some indication of permission error is included in any regular reporting back to DM B.

DM A also sends a policy to Agent A (step 5) that defines a VAR (V3) whose value is given by the mathematical expression (EDD3 * 3) and provides no ACL, indicating that any DM can access V3. In this instance, both DM A and DM B can then send policies to DA A to report the value of V3 (step 6). Since there is no authorization restriction on V3, these policies are accepted by the autonomy engine on Agent A and reports are sent to both DM A and B over time (step 7).

10.6. Cascading Management

There are times where a single network device may serve as both a DM for other DAs in the network and, itself, as a device managed by someone else. This may be the case on nodes serving as gateways or proxies. The DTNMA accommodates this case by allowing a single device to run both a DA and DM.

An example of this configuration is illustrated in Figure 7.

Data Fusion Control Flow

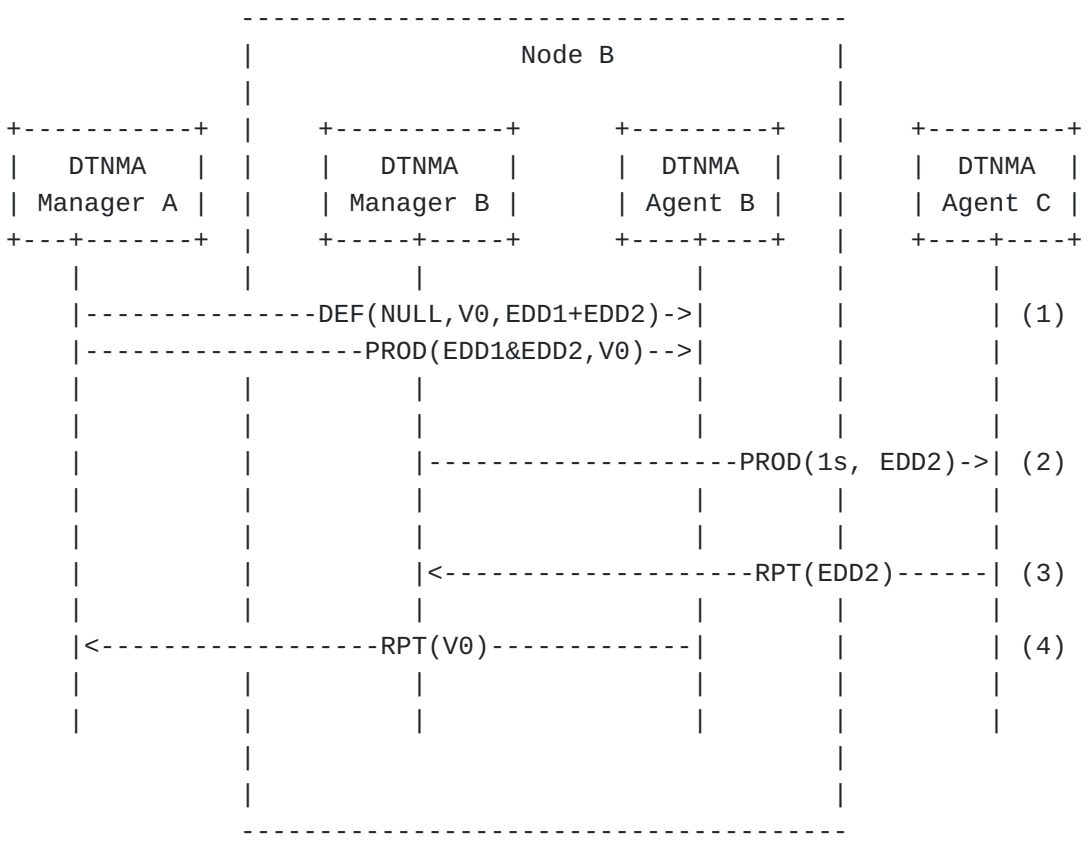

Figure 7

A device can operate as both a DTNMA Manager and an Agent.

In this example, we presume that DA B is able to sample a given EDD (EDD1) and that DA C is able to sample a different EDD (EDD2). Node B houses DM B (which controls DA C) and DA B (which is controlled by DM A). DM A must periodically receive some new value that is calculated as a function of both EDD1 and EDD2.

First, DM A sends a policy to DA B to define a VAR (V0) whose value is given by the mathematical expression (EDD1 + EDD2) without a restricting ACL. Further, DM A sends a policy to DA B to report on the value of V0 every second (step 1).

DA B can require the ability to monitor both EDD1 and EDD2. However, the only way to receive EDD2 values is to have them reported back to Node B by DA C and included in the Node B runtime data stores. Therefore, DM B sends a policy to DA C to report on the value of EDD2 (step 2).

DA C receives the policy in its autonomy engine and produces reports on the value of EDD2 every second (step 3).

DA B may locally sample EDD1 and EDD2 and uses that to compute values of V0 and report on those values at regular intervals to DM A (step 4).

While a trivial example, the mechanism of associating fusion with the Agent function rather than the Manager function scales with fusion complexity. Within the DTNMA, DAs and DMs are not required to be separate software implementations. There may be a single software application running on Node B implementing both DM B and DA B roles.

11. IANA Considerations

This informational document requires no registrations to be managed by IANA.

12. Security Considerations

Security within a DTNMA MUST exist in at least two layers: security in the data model and security in the messaging and encoding of the data model.

Data model security refers to the confidentiality of elements of a data model and the authorization of DTNMA actors to access those elements. For example, elements of a data model might be available to certain DAs or DMs in a system, whereas the same elements may be hidden from other DAs or DMs.

NOTE: One way to provide finer-grained application security is through the use of Access Control Lists (ACLs) that would be defined as part of the configuration of DAs and DMs. It is expected that many common data model tools provide mechanisms for the definition of ACLs and best practices for their operational use.

The exchange of information between and amongst DAs and DMs in the DTNMA is expected to be accomplished through some messaging transport. As such, security at the transport layer is expected to address the questions of authentication, integrity, and confidentiality.

13. Acknowledgements

Brian Sipos of the Johns Hopkins University Applied Physics Laboratory (JHU/APL) provided excellent technical review of the DTNMA concepts presented in this document.

14. Informative References

[I-D.ietf-core-comi] Veillette, M., Van der Stok, P., Pelov, A., Bierman, A., and I. Petrov, "CoAP Management Interface (CORECONF)", Work in Progress, Internet-Draft, draft-

ietf-core-comi-11, 17 January 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-11>.

[I-D.ietf-core-sid] Veillette, M., Pelov, A., Petrov, I., Bormann, C., and M. Richardson, "YANG Schema Item iDentifier (YANG SID)", Work in Progress, Internet-Draft, draft-ietf-core-sid-20, 1 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-sid-20>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, DOI 10.17487/RFC2578, April 1999, <https://www.rfc-editor.org/info/rfc2578>.

[RFC3416] Presuhn, R., Ed., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, DOI 10.17487/RFC3416, December 2002, <https://www.rfc-editor.org/info/rfc3416>.

[RFC3418] Presuhn, R., Ed., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, DOI 10.17487/RFC3418, December 2002, <https://www.rfc-editor.org/info/rfc3418>.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <https://www.rfc-editor.org/info/rfc4838>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol

(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <https://www.rfc-editor.org/info/rfc7575>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

[RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <https://www.rfc-editor.org/info/rfc8368>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

[RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990,

DOI 10.17487/RFC8990, May 2021, <https://www.rfc-editor.org/info/rfc8990>.

[RFC8993] Behringer, M., Ed., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", RFC 8993, DOI 10.17487/RFC8993, May 2021, <https://www.rfc-editor.org/info/rfc8993>.

[RFC9171] Burleigh, S., Fall, K., and E. Birrane, III, "Bundle Protocol Version 7", RFC 9171, DOI 10.17487/RFC9171, January 2022, <https://www.rfc-editor.org/info/rfc9171>.

[RFC9172] Birrane, III, E. and K. McKeever, "Bundle Protocol Security (BPSec)", RFC 9172, DOI 10.17487/RFC9172, January 2022, <https://www.rfc-editor.org/info/rfc9172>.

[xpath] Clark, J.C. and R.D. DeRose, "XML Path Language (XPath) Version 1.0", 1999.

Authors' Addresses

Edward J. Birrane
Johns Hopkins Applied Physics Laboratory

Email: Edward.Birrane@jhuapl.edu

Sarah E. Heiner
Johns Hopkins Applied Physics Laboratory

Email: Sarah.Heiner@jhuapl.edu

Emery Annis
Johns Hopkins Applied Physics Laboratory

Email: Emery.Annis@jhuapl.edu