Workgroup: Delay-Tolerant Networking
Internet-Draft: draft-ietf-dtn-dtnma-01
Published: 10 July 2022
Intended Status: Informational
Expires: 11 January 2023
Authors: E.J. Birrane
Johns Hopkins Applied Physics Laboratory
E. Annis
Johns Hopkins Applied Physics Laboratory
S.E. Heiner
Johns Hopkins Applied Physics Laboratory

DTN Management Architecture

Abstract

This document describes the motivation for, and services required of, the management of devices deployed in a Delay-Tolerant Networking (DTN) environment. Together, this set of information outlines a conceptual DTN Management Architecture (DTNMA) suitable for deployment in any of the challenged and constrained DTN operational environments.

The DTNMA is supported by two types of asynchronous behavior. First, the DTNMA does not presuppose any synchronized transport behavior between managed and managing devices. Second, the DTNMA does not support any query-response semantics. In this way, the DTNMA allows for operation in extremely challenging conditions, to include over uni-directional links and cases where delays/disruptions prevent operation over traditional transport layers.

Status of This Memo

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 the DTN Management Architecture (DTNMA) designed to provide configuration, monitoring, and local control of both application and network services on a managed device operating either within or across a challenged network.

The structure of the DTNMA is derived from the unique properties of challenged networks are defined in [RFC7228]. These properties include cases where an end-to-end transport path may not exist at any moment in time and when delivery delays may prevent timely communications between a network operator and a managed device. 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. Importantly, these capabilities must be designed and implemented in a way that results in outcomes that are determinable by an outside observer as such observers may need to connect with a managed device after significant periods of disconnectivity.

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.

Ultimately, the DTNMA is designed to leverage any transport, network, and security solutions designed for challenged networks. However the DTNMA is designed to be usable in any environment in which the Bundle Protocol (BPv7) [RFC9171] may be deployed.

1.1. Scope

This document describes the motivation, services, desirable properties, roles/responsibilities, logical data model, and system model that form the DTNMA. These descriptions comprise a concept of operations for management in challenged networks

This document is not a normative standardization of a physical data model or any individual protocol. Instead, it serves as informative guidance to authors and users of such models and protocols.

The DTNMA is independent of transport and network layers. It does not, for example, require the use of BP, TCP, or UDP. Similarly, it does not pre-suppose the use of IPv4 or IPv6.

The DTNMA is not bound to a particular security solution and does not presume that transport layers can exchange messages in a timely manner. It is assumed that any network using this architecture supports services such as naming, addressing, routing, and security that are required to communicate DTNMA messages as would be the case with any other messages in the network.

While possible that a challenged network may interface with an unchallenged network, this document does not specifically address compatibility with other management approaches.

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 seven sections, described as follows.

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

* Motivation - This section provides an overall motivation for this work, to include explaining why this approach is a useful alternative to existing network management approaches.

* Desirable Properties - This section identifies the properties that guide the definition of the system and logical models that comprise the DTNMA.

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

* Roles and Responsibilities - This section identifies roles in the DTNMA and their associated responsibilities. It provides the context for discussing how services are provided for both managed and managing devices.

* Logical Data Model - This section describes the kinds of data, procedures, autonomy, and associated hierarchical structure inherent to the DTNMA.

* System Model - This section describes data flows amongst various defined DTNMA roles. These flows capture how the DTNMA system works to manage devices across a challenged network.

2. Terminology

* Actor - A software service running on either managed or managing devices for the purpose of implementing management protocols between such devices. Actors may implement the "Manager" role, "Agent" role, or both.

* Agent Role (or Agent) - A role associated with a managed device, responsible for reporting performance data, accepting/performing controls, error handling and validation, and executing any autonomous behaviors. DTNMA Agents exchange information with DTNMA Managers operating either on the same device or on a remote managing device.

* DTN Management - Management that does not depend on stateful connections or real time delivery of management messages. Such management allows for asynchronous commanding to autonomous managers running on managed devices. This management is designed to run in any environment conformant to the DTN architecture and/or in any environment deploying a BPv7 network.

* Externally Defined Data (EDD) - Information made available to a DTNMA Agent by a managed device, but not computed directly by the DTNMA Agent itself.

* Variables (VARs) - Typed information that is computed by a DTNMA Agent, typically as a function of EDD values and/or other Variables.

* Constants (CONST) - A Constant represents a typed, immutable value that is referred to by a semantic name. Constants are used in situations where substituting a name for a fixed value provides useful semantic information. For example, using the named constant PI rather than the literal value 3.14159.

* Controls (CTRLs) - Procedures run by a DTNMA Actor to change the behavior, configuration, or state of an application or protocol being managed within a DTN. Controls may also be used to request data from an Agent and define the rules associated with generation and delivery.

* Literals (LITs) - A Literal represents a typed value without a semantic name. Literals are used in cases where adding a semantic name to a fixed value provides no useful semantic information. For example, the number 4 is a Literal value.

* Macros (MACROs) - A named, ordered collection of Controls and/or other Macros.

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

*Operator (OP) - The enumeration and specification of a mathematical function used to calculate variable values and construct expressions to evaluate DTNMA Agent state.

*Report (RPT) - A typed, ordered collection of data values gathered by one or more DTNMA Agents and provided to one or more DTNMA Managers. Reports only contain typed data values and the identity of the Report Template (RPTT) to which they conform.

*Report Template (RPTT) - A named, typed, ordered collection of data types that represent the schema of a Report. This template is generated by a DTNMA Manager and communicated to one or more other DTNMA Managers and DTNMA Agents.

*Rule - A unit of autonomous specification that provides a stimulus-response relationship between time or state on an DTNMA Agent and the actions or operations to be run as a result of that time or state. A Rule might trigger actions such as updating a Variable, producing a Report or a Table, and running a Control.

*State-Based Rule (SBR) - Any Rule triggered by the calculable internal state of the DTNMA Agent.

*Synchronous Management - Management that assumes messages will be delivered and acted upon in real or near-real-time. Synchronous management often involves immediate replies of acknowledgment or error status. Synchronous management is often bound to underlying transport protocols and network protocols to ensure reliability or source and sender identification.

*Table (TBL) - A 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. Tables only contain typed data values and the identity of the Table Template (TBLT) to which they conform.

*Table Template (TBLT) - A named, typed, ordered collection of columns that comprise the structure for representing tabular data values. This template forms the structure of a table (TBL).

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

3. Motivation

Early work on the rationale and motivation for specialized management for the DTN architecture was captured in [BIRRANE1], [BIRRANE2], and [BIRRANE3]. Prototyping work done in accordance with the DTN Research Group within the IRTF as documented in [I-D.irtf-dtnrg-dtnmp] provides some of the desirable properties and necessary adaptations for this proposed management system for challenged networks.

The unique nature and constraints that characterize challenged networks require the development of new network capabilities to deliver expected network functions. For example, the distinctive constraints of the DTN architecture required the development of BPv7 [RFC9171] for transport functions and the Bundle Protocol Security Extensions (BPSec) [RFC9172] to provide end-to-end security. Similarly, a new approach to network management and the associated capabilities is necessary for operation in these challenged environments and when using these new transport and security mechanisms.

This section discusses the characteristics of challenged networks and how they may violate the assumptions made by non-DTNMA approaches about the operating environment.

3.1. Constrained and Challenged Networks

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, or 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, 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 attenuations, propagation delays, mobility, occultation, and other limitations imposed by energy and mass considerations.

These 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.

* Dependence on external infrastructure, software, systems, or processes such as Domain Name Service (DNS) or Certificate authorities (CAs) cannot be guaranteed.

Finally, it is noted that "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]).

Challenged networks differ from other kinds of constrained networks, in part, in the way that the topology and roles and responsibilities of the network may evolve over time. From the time at which data is generated to the time at which that data is delivered, the topology of the network and the roles assigned to various nodes, devices, and other actors may have changed several times. In certain circumstances, the physical node receiving messages for a given logical destination may have also changed.

Challenged networks cannot guarantee that a timely data exchange can be maintained between managing and managed devices. The topological changes characteristic of these networks can impact the path of messages, requiring the transport to wait to establish the incremental connectivity necessary to advance messages along their expected route. The BPv7 transport protocol implements this store-and-forward operation for DTNs.

3.2. Management of Challenged Networks

When topological change impacts the semantic roles and responsibilities of nodes in the network then local configuration and autonomy must be present at the node to determine and execute time-variant changes. For example, the BPSec protocol does not encode security destinations and, instead, requires nodes in a network to identify themselves as security verifiers or acceptors when receiving secured messages.

When applied to network management, the semantic roles of Agent and Manager may also change with the evolving topology of the network. Individual nodes must implement desirable behavior without relying on a single configuration oracle or other coordinating function such as an operator-in-the-loop and/or supporting infrastructure. These mechanisms cannot be supported by an asynchronous, challenged network.

The support for changing roles implies that there MUST NOT be a defined relationship between a particular manager and agent in a network. A network management architecture for challenged networks must support the association of multiple managers with a single agent, allow "control from" and "reporting to" managers to function independent of one another, and allow the logical role of a manager to be physically shared among assets and change over time.

Together, this means that a network management architecture suitable for challenged environments must account for certain operational situations.

* Managed devices that are only accessible via a uni-directional link, or via a link whose duration is shorter than a single round-trip propagation time.

* Links that may be significantly constrained by capacity or reliability, but at (predictable or unpredictable) times may offer significant throughput.

* Multi-hop challenged networks that interconnect two or more unchallenged networks such that managed and managing devices exist in different networks.

* Networks unable to support session-based transport. For example, when propagation delays exceed the Maximum Segment Lifetime (MSL) of the Transmission Control Protocol (TCP).

In these and related scenarios, managed devices need to operate with local autonomy because managing devices may not be available within operationally-relevant timeframes. Managing devices deliver instruction sets that govern the local, autonomous behavior of the managed device. These behaviors include (but are not limited to) collecting performance data, state, and error conditions, and applying pre-determined responses to pre-determined events. The goal is asynchronous and autonomous communication between the device being managed and the manager, at times never expecting a reply, and with knowledge that commands and queries may be delivered much later than the initial request.

3.3. Current Network Management Approaches and Limitations

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.

Generally, network management solutions that require managers and agents to push and pull large sets of data may fail to operate in a challenged (and thus, constrained) environment as a function of transmit power, bitrates, and the ability of the network to store and forward large data volumes over long periods of time.

Newer network management approaches are exploring the application of moe efficient message-based management, less reliance on end-to-end transport sessions, and increased levels of autonomy on managed devices. These approaches focus on problems different from those described above for challenged networks. For example, much of the autonomous network management work currently undertaken focuses more on well-resourced, unchallenged networks where devices self-configure, self-heal, and self-optimize with other nodes in their vicinity. While an important and transformational capability, such solutions will not be deployable in a challenged network environment.

This section describes some of the well-known, standardized protocols for network management and contrasts their purposes with the needs of challenged network management solutions.

3.3.1. Simple Network Management Protocol (SNMP)

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 manager to explicitly request the data to be produced and transmitted by an agent.

The de facto example of this 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 utilizations, error rates, and counters between application software on agents and managers. 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 managers can query agents for status information, send new configurations, and request to be informed when specific events have occurred. Traps and queryable data are defined in a data model known as Managed Information Bases (MIBs) which define the information for a particular data standard, protocol, device, or application.

While there is a large installation base for SNMP, there are several aspects of the protocol that make it inappropriate for use in a challenged network. SNMP relies on sessions with low round-trip latency to support its "pull" model that challenged networks cannot maintain. Complex management can be achieved, but only through craftful orchestration using a series of real-time, end-to-end, manager-generated query-and-response logic that is not possible in challenged networks.

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

Adaptive modifications to SNMP to support challenged networks and more complex application-level management would alter the basic function of the protocol (data models, control flows, and syntax) so as to be functionally incompatible with existing SNMP installations. This approach is therefore not suitable for use in challenged networks.

3.3.2. YANG Data Model and NETCONF, RESTCONF, and CORECONF

3.3.2.1. The YANG Data Model

Yet Another Next Generation (YANG) [RFC6020] is a data modeling language used to model configuration and state data of managed devices and applications. 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.

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 RPCs are used to execute commands on a device, generating an expected, structured response. However, 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.

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.

3.3.2.2. YANG-Based Management Protocols

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 the 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.

NETCONF uses verbose XML files to provide the ability to update and fetch multiple data elements simultaneously. These XML files are not easily or efficiently compressed, which is an important consideration for challenged networks.

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 requires the use of a secure transport such as TLS.

Unlike NETCONF, RESTCONF is stateless. However, the transfer of large data sets, such as configuration changes of many data elements, or the collection of information, depends greatly on the support of synchronous communication.

CORECONF is stateless, as RESTCONF is, and is built atop the Constrained Application Protocol (CoAP) [RFC7252] which defines a messaging construct developed to operate specifically on constrained devices and networks by limiting message size and fragmentation. CORECONF requires the use of DTLS or Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] to fulfil its security requirements. COAP supports a store and forward operation similar to DTN; however, it operates strictly at the application layer and requires specification of pre-determined proxies and moments of bi-directional communication.

CORECONF leverages the Concise Binary Object Representation (CBOR) [RFC8949] of YANG modules [I-D.ietf-core-yang-cbor] and provides further compressibility through the use of YANG Schema Item iDentifiers (SIDs) [I-D.ietf-core-sid]. While these design choices offer reductions in encoded data size, data compressibility is still dependent on underlying transport protocols and limited by the organization of the YANG schema.

3.3.2.3. Limitations of YANG-Based Approaches

YANG notifications are promising for challenged network management, defined as subscriptions to both YANG notifications [RFC8639]] and YANG PUSH notifications [RFC8641]. In this model, a client may subscribe to the delivery of specific containers or data nodes defined in the model, either on a periodic or "on change" basis. The notification events can be filtered according to XPath ([xpath]) or subtree ([RFC6241]) filtering as described in [RFC8639] Section 2.2.

While the YANG model provides great flexibility for configuring a homogeneous network of devices, it becomes a burden in challenged networks where concise encoding is necessary. 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]. All the 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.

Recent efforts for compression of the YANG model have used CBOR and SIDs to address YANG data nodes through integer identifiers. However, these compression strategies lack a formal hierarchical structure. The manual mapping of SIDs to YANG modules and data nodes limits the portability of these models and further increases the size of any encoding scheme.

3.3.3. The Future of Autonomous and Autonomic Network Management Solutions

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] and includes many recent efforts describe Autonomic architecture and protocols [RFC8993] as well as cite the gaps that exist between traditional and Autonomic Networking approaches [RFC7576]. Challenged networks require similar degrees of autonomy, however they lack the ability to depend on the complex coordination between nodes and the centralized and distributed supporting infrastructure that Autonomic networking proposes.

Policy-based management is a well-established approach that uses business and operations support systems to monitor and manage devices and networks in real-time. These systems leverage various, existing network management protocols and their supporting features, such as the use of YANG module classification types [RFC8199], to describe abstract services and support configuration of service level agreements. These services can then enact additional control over devices using network element modules. This approach is quite comprehensive but requires sufficient, supporting infrastructure and synchronous access, which cannot be provided by challenged networks.

3.3.4. Takeaways from Existing Network Management Protocols

While the protocols described above are useful and well-realized for different applications and networking environments, they simply do not meet the requirements for the management of challenged networks. However, that does not exclude features from each from contributing to the design of DTNMA.

The concept of a data model for describing network configuration elements has been used by many protocols to ensure compliance between managing and managed devices. A data model provides error checking and bounds operations, which is necessary when controlling mission critical devices.

The SNMP MIBs provide well-organized, hierarchical OIDs which support the compressibility necessary for challenged DTNs. YANG, NETCONF, and RESTCONF support notification abilities needed for DTN network management, but have limited features for describing autonomous execution and behavior.

CORECONF provides CBOR encoding and concise reference abilities using SIDs, but lack a hierarchical structure or authoritative planning to allocation. While this approach will become too verbose and prove limiting in the future, the encoding considerations from CORECONF can be used to inform the design of the DTNMA.

3.4. A Network Management Approach for DTNs

The DTNMA is designed with consideration for the constraints discussed in section Section 3.1. The DTNMA seeks to incorporate existing network management protocols and feature. However, there are core capabilities the DTNMA must provide in order to serve a challenged network that are not supported by these approaches.

The DTNMA proposes a data model that is that is designed for the compression required for a challenged network. The efficiency of data encoding is limited by the efficiency of the underlying data model. For this reason, naming schemes for the DTNMA must be hierarchical and patternable, supporting the level of compressibility needed by the resource-constrained devices that form a challenged network.

Autonomous behavior is required for the management of a DTN, which is characterized by link delays and disruptions. The constrained autonomy model of the DTNMA provides the deterministic management necessary for managed devices to detect and respond to events without intervention from an in-the-loop manager. The separation of remote and local, autonomous managing devices supports autonomous behavior even when synchronization is not feasible.

The sections below describe the desirable features of the DTNMA and build from e xisting protocols and mechanisms where possible, with adaptations made for the challenged networking environment.

4. Desirable Properties of an DTNMA

This section describes those design properties that are desirable when defining an architecture that must operate 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.

4.1. Asynchronous, Dynamic, and Highly Logical Architecture

An DTNMA built to support DTN must be agnostic of the underlying physical topology, transport protocols, security solutions, and supporting infrastructure. The DTNMA shall be limited to only the network management protocols, message structure, and information content, including but not limited to the type of objects to manage and the expected behavior and interaction upon access or execution of those objects. There shall be no prescribed association between between a manager and an agent other than those defined in the responsibilities associated with each in this document. There should be no limitation to the number of managers that can control an agent, the number of managers that an agent should report to, or any requirement that a manager and agent relationship implies a pair.

4.2. Model-derived and Hierarchically Organized Definition of Information

A means to define a shared contract between agent and manager has long been an approach to network management solutions. A model is a schema that defines this contract and defines all sources of information that can be retrieved, configured, or executed, as well as the various functions for parameterization, filtering, or event driven behavior. A model gives way to concise representation of information, intelligent suffixing, and patterning. The DTNMA model shall be designed with a limited set of object and data types to allow and be organized hierarchally to provide for highly compressible and concise encoding. This allows the agents and managers to infer context with limited link utilization necessary in DTN.

4.3. Intelligent Push of Information

Pull management mechanisms require that a Manager send a query to an Agent and then wait for the response to that query. This practice implies a control-session between entities and increases the overall message traffic in the network. Challenged networks cannot guarantee that the round-trip data-exchange will occur in a timely fashion. In extreme cases, networks may be comprised of solely uni-directional links which drastically increases the amount of time needed for a round-trip data exchange. Therefore, pull mechanisms must be avoided in favor of push mechanisms.

Push mechanisms, in this context, refer to the ability of Agents to leverage rule-based criteria to determine when and what information should be sent to managers. This could be based solely off logic applied to existing VARs or EDDs, based off operations applied to data elements, or triggered as a function of relative time. Such mechanisms do not require round-trip communications as Managers do not request each reporting instance; Managers need only request once, in advance, that information be produced in accordance with a predetermined schedule or in response to a predefined state on the Agent. In this way information is "pushed" from Agents to Managers and the push is "intelligent" because it is based on some internal evaluation performed by the Agent.

4.4. Minimize Message Size Not Node Processing

Protocol designers must balance message size versus message processing time at sending and receiving nodes. Verbose representations of data simplify node processing whereas compact representations require additional activities to generate/parse the compacted message. There is no asynchronous management advantage to minimizing node processing time in a challenged network. However, there is a significant advantage to smaller message sizes in such networks. Compact messages require smaller periods of viable transmission for communication, incur less re-transmission cost, and consume less resources when persistently stored en-route in the network. A DTN Management Protocol (DTNMP) should minimize PDUs whenever practical, to include packing and unpacking binary data, variable-length fields, and pre-configured data definitions.

4.5. Absolute Data Identification

Elements within the management system must be uniquely identifiable so that they can be individually manipulated. Identification schemes that are relative to system configuration make data exchange between Agents and Managers difficult as system configurations may change faster than nodes can communicate.

Consider the following common technique for approximating an associative array lookup. A manager wishing to do an associative lookup for some key K1 will (1) query a list of array keys from the agent, (2) find the key that matches K1 and infer the index of K1 from the returned key list, and (3) query the discovered index on the agent to retrieve the desired data.

Ignoring the inefficiency of two pull requests, this mechanism fails when the Agent changes its key-index mapping between the first and second query. Rather than constructing an artificial mapping from K1 to an index, an AMP must provide an absolute mechanism to lookup the value K1 without an abstraction between the Agent and Manager.

4.6. Custom Data Definition

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. Specifically, an Agent should not be required to transmit a large data set for a Manager that only wishes to calculate a smaller, inferred data set. These new defined data elements could be calculated and used both as parameters for local stimulus-response rules-based criteria or simply serve to populate custom reports and tables. Since the identification of custom data sets is likely to occur in the context of a specific network deployment, AMPs must provide a mechanism for their definition.

Aggregation of controls and custom formatting of reports and tables are equally important. Custom reporting provides the flexibility allowing the manager to define the desired format of all information to be sent over the challenged network from the agents, serving to both save link capacity and increase the value of returned information. Aggregation of controls allows a manager to specify a set of controls to execute, specifying both the order and criteria of execution. This aggregate set of controls can be sent as a single command rather than a series of sequential operands. In this case it is additionally possible to use outputs of one command to serve as an input to the next at the agent.

4.7. Autonomous Operation

DTNMA network functions must be achievable using only knowledge local to the Agent. Rather than directly controlling an Agent, a Manager configures an engine of the Agent to take its own action under the appropriate conditions in accordance with the Agent's notion of local state and time.

Such an engine 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. Wholly autonomous operations MAY be supported where required. Generally, autonomous operations should provide the following benefits.

* Distributed Operation - The concept of pre-configuration allows the Agent to operate without regular contact with Managers in the system. The initial configuration (and periodic update) of the system remains difficult in a challenged network, but an initial synchronization on stimuli and responses drastically reduces needs for centralized operations.

* Deterministic Behavior - Such behavior is necessary in critical operational systems where the actions of a platform must be well understood even in the absence of an operator in the loop. Depending on the types of stimuli and responses, these systems may be considered to be maintaining simple automation or semi-autonomous behavior. In either case, this preserves the ability of a frequently-out-of-contact Manager to predict the state of an Agent with more reliability than cases where Agents implement independent and fully autonomous systems.

* Engine-Based Behavior - Several operational systems are unable to deploy "mobile code" based solutions due to network bandwidth, memory or processor loading, or security concerns. Engine-based approaches provide configurable behavior without incurring these types of concerns associated with mobile code.

* Intelligent authentication, authorization, accounting (AAA), and error checking - A means of autonomous AAA, error checking, and validation of data and controls will be be required in all cases where agents or managers are disconnected from the rest of the network. In addition, there is a need to handle conflicts including messages that arrive out of order, or at the same time from different managers whose controls would otherwise conflict. The need to perform these operations still exists however they will need to be performed with context provided with controls sent or in accordance with pre-defined behavior and policy.

5. Services Provided by an DTNMA

The DTNMA provides a method of configuring DTNMA Agents with local, autonomous management functions, such as rule-based execution of procedures and generation of reports, to achieve expected behavior when managed devices exist over a challenged network. It further allows for dynamic instantiation and population of Variables and reports through local operations defined by the manager, as well as custom formatting of tables and reports to be sent back. This gives the DTNMA significant flexibility to operate over challenged networks, both providing new degrees of freedom over existing configuration based data models used in synchronous networks and allowing for more concise formatting over constrained networks. This architecture makes very few assumptions on the nature of the network and allow for continuous operation through periods of connectivity and lack of connectivity. The DTNMA deviates from synchronous management approaches because it never requires periods of bi-directional connectivity, and provides the manager flexibility to describe agent behavior that was unpredicted at the time of the data model creation.

This section identifies the services that a DTNMA would provide for management of challenged network resources. These services include configuration, reporting, autonomous parameterized control, and administration.

5.1. Configuration

Configuration services update Agent data associated with managed applications and protocols. Some configuration data might be defined in the context of an application or protocol, such that any network using that application or protocol would understand that data. Other configuration data may be defined tactically for use in a specific network deployment and not available to other networks even if they use the same applications or protocols.

With no guarantee of round-trip data exchange, Agents cannot rely on remote Managers to correct erroneous or stale configurations from harming the flow of data through a challenged network.

Examples of configuration service behavior include the following.

* Creating a new datum as a function of other well-known data:

  C = A + B.

* Creating a new report as a unique, ordered collection of known data:

  RPT = {A, B, C}.

* Storing predefined, parameterized responses to potential future conditions:

  IF (X > 3) THEN RUN CMD(PARM).

5.2. Reporting

Reporting services populate report templates with values collected or computed by an Agent. The resultant reports are sent to one or more Managers by the Agent. The term "reporting" is used in place of the term "monitoring", as monitoring implies a timeliness and regularity that cannot be guaranteed by a challenged network. Reports sent by an Agent provide best-effort information to receiving Managers.

Since a Manager is not actively "monitoring" an Agent, the Agent must make its own determination on when to send what Reports based on its own local time and state information. Agents should produce Reports of varying fidelity and with varying frequency based on thresholds and other information set as part of configuration services.

Examples of reporting service behavior include the following.

- Generate Report R1 every hour (time-based production).

- Generate Report R2 when X > 3 (state-based production).

5.3. Autonomous Parameterized Procedure Calls

Similar to an RPC call, some mechanism MUST exist which allows a procedure to be run on an Agent in order to affect its behavior or otherwise change its internal state. Since there is no guarantee that a Manager will be in contact with an Agent at any given time, the decisions of whether and when a procedure should be run MUST be made locally and autonomously by the Agent. Two types of automation triggers are identified in the DTNMA: triggers based on the internal state of the Agent and triggers based on an Agent's notion of time. As such, the autonomous execution of procedures can be viewed as a stimulus-response system, where the stimulus is the positive evaluation of a state or time based predicate and the response is the function to be executed.

The autonomous nature of procedure execution by an Agent implies that the full suite of information necessary to run a procedure may not be known by a Manager in advance. To address this situation, a parameterization mechanism MUST be available so that required data can be provided at the time of execution on the Agent rather than at the time of definition/configuration by the Manager.

Autonomous, parameterized procedure calls provide a powerful mechanism for Managers to "manage" an Agent asynchronously during periods of no communication by pre-configuring responses to events that may be encountered by the Agent at a future time.

Examples of potential behavior include the following.

- Updating local routing information based on instantaneous link analysis.

* Managing storage on the device to enforce quotas.

* Applying or modifying local security policy.

5.4. Authorized Administration, accounting, and error control

Administration services enforce the potentially complex mapping of auhorization to configuration, reporting, and control services amongst Agents and Managers in the network. Fine-grained access control can specify which Managers may apply which services to which Agents. This is particularly beneficial in networks that either deal with multiple administrative entities or overlay networks that cross administrative boundaries. Whitelists, blacklists, key-based infrastructures, or other schemes may be used for this purpose.

Other administrative services may place practical restrictions on the overall number of items that can be kept in a system. This includes items such as the number of rows kept by an Agent for a given table template or number of entries for a given report template.

Examples of administration service behavior include the following.

* Agent A1 only Sends reports for Protocol P1 to Manager M1.

* Agent A2 only accepts a configurations for Application Y from Managers M2 and M3.

* Agent A3 accepts services from any Manager providing the proper authentication token.

Note that the administrative enforcement of access control is different from security services provided by the networking stack carrying such messages.

6. DTNMA Roles and Responsibilities

By definition, Agents reside on managed devices and Managers reside on managing devices. There is however no pre-supposed architecture that connects managers and agents and therefore a single device could assume both roles. This section describes the responsibilities associated with each role and how these roles participate in network management.

6.1. Agent Responsibilities

Manager Mapping

Agents must receive messages from managers that govern application control, reporting, and autonomous behavior. Agents must maintain a list of managers which have delivered control messages along with a list of "report to" managers. The list of requested reports must be mapped to one or more managers.

Application Support

Agents MUST collect all data, execute all controls, populate all reports and run operations required by each application which the Agent manages. Agents MUST report supported applications by their data model so that Managers in a network understands what information is understood by what Agent.

Local Data Collection

Agents MUST collect from local firmware (or other on-board mechanisms) and report all data defined for the management of applications for which they have been configured. Agents must further use this information in the computation of variable expressions and rules-based autonomy.

Autonomous Control

Agents MUST determine, as previously prescribed by a manager, whether a procedure should be invoked.

Autonomous Reporting

Agents MUST determine, without real-time Manager intervention, whether and when to populate and transmit a given report or table targeted to one or more Managers in the network.

Custom Data Definition

Agents MUST provide mechanisms for operators in the network to use configuration services to create customized data definitions in the context of a specific network or network use-case. Agents MUST allow for the creation, listing, and removal of such definitions in accordance with whatever security models are deployed within the particular network.

Where applicable, Agents MUST verify the validity of these definitions when they are configured and respond in a way consistent with the logging/error-handling policies of the Agent and the network.

Consolidate Messages

Agents SHOULD produce as few messages as possible when sending information. For example, rather than sending multiple messages, each with one report to a Manager, an Agent SHOULD prefer to send a single message containing multiple reports.

Error Checking and State Control

Agents should perform error checking and validation of incoming manager messages as well as internally computed values. This includes but is not limited to validating the syntax of messages and controls according to the data model, preventing circular references in custom defined data, and verifying maximum nesting levels or table lengths have not been exceeded. This also includes control of internal agent operations and state. Finally there must be a means to handle conflicts such as messages that arrive out of order or messages from more than one authorized manager.

Authorized Administration and Accounting

The Agent shall provide authorized administration and accounting to restrict execution of controls, custom data definition, and reporting to only those authorized nodes. Both nominal and exception events shall be logged where applicable.

6.2. Manager Responsibilities

Agent Capabilities Mapping

Managers must maintain a list of supported models and managed applications. Managers MUST understand what applications are managed by the various Agents with which they communicate and maintain a list of those managed agents. Managers should not attempt to request, invoke, or refer to application information for applications not managed by an Agent. Agents must further maintain a list of all agents that are reporting to this manager.

Agent Messaging

Managers must generate and transmit control messages destined for agents. This includes all the control types, configuration, and parameterization described in the logical data model.

Data Collection

Managers MUST receive information from Agents asynchronously upon the configuration and production of reports by the local and other external managers, collecting responses from Agents over time. Managers MAY try to detect conditions where Agent information has not been received within operationally relevant time spans and react in accordance with network policy.

Custom Data Definitions

Managers should provide the ability to define custom data definitions. Any custom definitions MUST be transmitted to appropriate Agents and these definitions MUST be remembered to interpret the reporting of these custom values from Agents in the future.

Data Fusion

Managers MAY support the fusion of data from multiple Agents with the purpose of transmitting fused data results to other Managers within the network. Managers MAY receive fused reports from other

Managers pursuant to appropriate security and administrative configurations.

Error Checking and State Control

Managers should perform error checking and validation of incoming agent messages as well as internally configured controls for agents. This includes but is not limited to validating the syntax of messages and controls according to the data model, preventing circular references in custom defined data, and verifying maximum nesting levels or table lengths have not been exceeded. This also includes control of internal manager operations and state.

Authorized Administration and Accounting

The Manager shall provide authorized administration and accounting and send controls to only those agents for which it is authorized. It shall additionally validate incoming agent reports according to any defined restrictions. Both nominal and exception events shall be logged where applicable.

7. Logical Data Model

The DTNMA logical data model captures the types of information that should be collected and exchanged to implement necessary roles and responsibilities. The data model presented in this section does not presuppose a specific mapping to a physical data model or encoding technique; it is included to provide a way to logically reason about the types of data that should be exchanged in a DTN managed network.

The elements of the DTNMA logical data model are described as follows.

7.1. Data Representations: Constants, Externally Defined Data, and Variables

There are three fundamental representations of data in the DTNMA: (1) data whose values do not change as a function of time or state, (2) data whose values change as determined by sampling/calculation external to the network management system, and (3) data whose values are calculated internal to the network management system.

Data whose values do not change as a function of time or state are defined as Constants (CONST). CONST values are strongly typed, named values that cannot be modified once they have been defined.

Data sampled/calculated external to the network management system are defined as Externally Defined Data" (EDD). EDD values represent the most useful information in the management system as they are provided by the applications or protocols being managed on the Agent. It is RECOMMENDED that EDD values be strongly typed to avoid issues with interpreting the data value. It is also RECOMMENDED that the timeliness/staleness of the data value be considered when using the data in the context of autonomous action on the Agent.

Data that is calculated internal to the network management system is defined as a Variable (VAR). VARs allow the creation of new data values for use in the network management system. New value definitions are useful for storing user-defined information, storing the results of complex calculations for easier re-use, and providing a mechanism for combining information from multiple external sources. It is RECOMMENDED that VARs be strongly typed to avoid issues with interpreting the data value. In cases where a VAR definition relies on other VAR definitions, mechanisms to prevent circular references MUST be included in any actual data model or implementation.

7.2. Data Collections: Reports and Tables

Individual data values may be exchanged amongst Agents and Managers in the DTNMA. However, data are typically most useful to a Manager when received as part of a set of information. Ordered collections of data values can be produced by Agents and sent to Managers as a way of efficiently communicating Agent status. Within the DTNMA, the structure of the ordered collection is treated separately from the values that populate such a structure.

The DTNMA provides two ways of defining collections of data: reports and tables. Reports are ordered sets of data values, whereas Tables are special types of reports whose entries have a regular, tabular structure.

7.2.1. Report Templates and Reports

The typed, ordered structure of a data collection is defined as a Report Template (RPTT). A particular set of data values provided in compliance with such a template is called a Report (RPT).

Separating the structure and content of a report reduces the overall size of RPTs in cases where reporting structures are well known and unchanging. RPTTs can be synchronized between an Agent and a Manager so that RPTs themselves do not incur the overhead of carrying self-describing data. RPTTs may include EDD values, VARs, and also other RPTTs. In cases where a RPTT includes another RPTTs, mechanisms to prevent circular references MUST be included in any actual data model or implementation.

Protocols and applications managed in the DTNMA may define common RPTTs. Additionally, users within a network may define their own RPTTs that are useful in the context of a particular deployment.

Unlike tables, reports do not exploit assumptions on the underlying structure of their data. Therefore, unlike tables, operators can define new reports at any time as part of the runtime configuration of the network.

7.2.2. Table Templates and Tables

Tables optimize the communication of multiple sets of data in situations where each data set has the same syntactic structure and with the same semantic meaning. Unlike reports, the regularity of tabular data representations allow for the addition of new rows without changing the structure of the table. Attempting to add a new data set at the end of a report would require alterations to the report template.

The typed, ordered structure of a table is defined as a Table Template (TBLT). A particular instance of values populating the table template is called a Table (TBL).

TBLTs describes the "columns" that define the table schema. A TBL represents the instance of a specific TBLT that holds actual data values. These data values represent the "rows" of the table.

The prescriptive nature of the TBLT allows for the possibility of advanced filtering which may reduce traffic between Agents and Managers. However, the unique structure of each TBLT along may make them difficult or impossible to change dynamically in a network.

7.3. Command Execution: Controls and Macros

Low-latency, high-availability approaches to network management use mechanisms such as (or similar to) RPCs to cause some action to be performed on an Agent. The DTNMA enables similar capabilities without requiring that the Manager be in the processing loop of the Agent. Command execution in the DTNMA happens through the use of controls and macros.

A Control (CTRL) represents a parameterized, predefined procedure that can be run on an Agent. While conceptually similar to a "remote procedure call", CTRLs differ in that they do not provide numeric return codes. The concept of a return code when running a procedure implies a synchronous relationship between the caller of the procedure and the procedure being called, which is disallowed in an DTN management system. Instead, CTRLs may create reports which describe the status and other summarizations of their operation, and these reports may be sent to the Manager(s) calling the CTRL.

Parameters can be provided when running a command from a Manager, pre-configured as part of a response to a time-based or state-based rule on the Agent, or auto-generated as needed on the Agent. The success or failure of a control MAY be inferred by reports generated for that purpose.

NOTE: The DTNMA term control 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. A DTN management function controls an Agent to allow it to fulfill its commanded purpose in a variety of operational scenarios. 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 "commanding" heaters, louvers, radiators, and other temperature-affecting components. That said, CTRLs should be developed for specific autonomous and deterministic behavior where possible. Some controls may be designed to set configuration parameters or load complex policies, but there should be no assumption that it will be executed in real time.

Often, a series of controls must be executed in sequence to achieve a particular outcome. A Macro (MACRO) represents an ordered collection of controls (or other macros). In cases where a MACRO includes another MACRO, mechanisms to prevent circular references and maximum nesting levels MUST be included in any actual data model or implementation.

7.4. Autonomy: Time and State-Based Rules

The DTNMA data model contains EDDs and VARs that capture the state of applications on an Agent. The model also contains controls and macros to perform actions on an Agent. A mechanism is needed to relate these two capabilities: to perform an action on the Agent autonomously in response to the state of the Agent. This mechanism in the DTNMA is the "rule" and can be activated based on Agent internal state (state-based rule) or based on the Agent's notion of relative time (time-based rule).

7.4.1. State-Based Rule (SBR)

State-Based Rules (SBRs) perform actions based on the Agent's internal state, as identified by EDD and VAR values. An SBR represents a stimulus-response pairing in the following form: IF predicate THEN response The predicate is a logical expression that evaluates to true if the rule stimulus is present and evaluates to false otherwise. The response may be any control or macro known to the Agent.

An example of an SBR could be to initiate a thermal control self check if some internal temperature is greater than a threshold: IF (current_temp > maximum_temp) THEN thermal_control_self_check

Rules may construct their stimuli from the full set of values known to the network management system. Similarly, responses may be constructed from the full set of controls and macros that can be run on the Agent. By allowing rules to evaluate the variety of all known data and run the variety of all known controls, multiple applications can be monitored and managed by one Agent instance.

7.4.2. Time-Based Rule (TBR)

Time-Based Rules (TBR) perform actions based on the Agent's notion of the passage of time. A possible TBR construct would be to perform some action at 1Hz on the Agent.

A TBR is a specialization of an SBR as the Agent's notion of time is a type of Agent state. For example, a TBR to perform an action every 24 hours could be expressed using some type of predicate of the form: IF (((current_time - base_time) % 24_hours) == 0) THEN ... However, time-based events are popular enough that special semantics for expressing them would likely significantly reduce the computations necessary to represent time functions in a SBR.

7.5. Calculations: Expressions, Literals, and Operators

Actions such as computing a VAR value or describing a rule predicate require some mechanism for calculating the value of mathematical expressions. In addition to the aforementioned DTNMA logical data objects, Literals, Operators, and Expressions are used to perform these calculations.

A Literal (LIT) represents a strongly typed datum whose identity is equivalent to its value. An example of a LIT value is "4" - its identifier (4) is the same as its value (4). Literals differ from constants in that constants have an identifier separate from their value. For example, the constant PI may refer to a value of 3.14. However, the literal 3.14159 always refers to the value 3.14159.

An Operator (OP) represents a mathematical operation in an expression. OPs should support multiple operands based on the operation supported. A common set of OPs SHOULD be defined for any Agent and systems MAY choose to allow individual applications to define new OPs to assist in the generation of new VAR values and predicates for managing that application. OPs may be simple binary operations such as "A + B" or more complex functions such as sin(A) or avg(A,B,C,D). Additionally, OPs may be typed. For example, addition of integers may be defined separately from addition of real numbers.

An Expression (EXPR) is a combination of operators and operands used to construct a numerical value from a series of other elements of the DTNMA logical model. Operands include any DTNMA logical data model object that can be interpreted as a value, such as EDD, VAR, CONST, and LIT values. Operators perform some function on operands to generate new values.

8. System Model

This section describes the notional data flows and control flows that illustrate how Managers and Agents within an DTNMA cooperate to perform network management services.

8.1. Control and Data Flows

The DTNMA identifies three significant data flows: control flows from Managers to Agents, reports flows from Agents to Managers, and fusion reports from Managers to other Managers. These data flows are illustrated in Figure 1.

DTNMA Control and Data Flows

```
   +---------+       +------------------------+      +---------+
   | Node A  |       |         Node B         |      |  Node C |
   |         |       |                        |      |         |
   |+-------+|       |+-------+      +-------+|      |+-------+|
   ||       ||=====>>||Manager|====>>|       ||====>>||       ||
   ||       ||<<=====||   B   |<<====|Agent B||<<====||       ||
   ||       ||       |+--++---+      +-------+|      ||Manager||
   || Agent ||       +---||-------------------+      ||   C   ||
   ||   A   ||           ||                          ||       ||
   ||       ||<<=========||==========================||       ||
   ||       ||===========++========================>>||       ||
   |+-------+|                                       |+-------+|
   +---------+                                       +---------+
```

Figure 1

In this data flow, the Agent on node A receives Controls from Managers on nodes B and C, and replies with Report Entries back to these Managers. Similarly, the Agent on node B interacts with the local Manager on node B and the remote Manager on node C. Finally, the Manager on node B may fuse Report Entries received from Agents at nodes A and B and send these fused Report Entries back to the Manager on node C. From this figure it is clear that there exist many-to-many relationships amongst Managers, amongst Agents, and between Agents and Managers. Note that Agents and Managers are roles, not necessarily different software applications. Node A may represent a single software application fulfilling only the Agent role, whereas node B may have a single software application fulfilling both the Agent and Manager roles. The specifics of how these roles are realized is an implementation matter.

8.2. Control Flow by Role

This section describes three common configurations of Agents and Managers and the flow of messages between them. These configurations involve local and remote management and data fusion.

8.2.1. Notation

The notation outlined in Table 1 describes the types of control messages exchanged between Agents and Managers.

Term	Definition	Example
EDD#	EDD definition.	EDD1
V#	Variable definition.	V1 = EDD1 + V0.
DEF([ACL], ID,EXPR)	Define ID from expression. Allow managers in access control list (ACL) to request this ID.	DEF([*], 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 identified by ID.	RPT(EDD1)

Table 1: Terminology

8.2.2. Serialized Management

This is a nominal configuration of network management where a Manager interacts with a set of Agents. The control flows for this are outlined in Figure 2.

Serialized Management Control Flow

```
        +----------+            +---------+          +---------+
        |  Manager |            | Agent A |          | Agent B |
        +----+-----+            +----+----+          +----+----+
             |                       |                    |
             |-----PROD(1s, EDD1)--->|                    | (1)
             |----------------------------PROD(1s, EDD1)-->|
             |                       |                    |
             |                       |                    |
             |<-------RPT(EDD1)------|                    | (2)
             |<----------------------------RPT(EDD1)-------|
             |                       |                    |
             |                       |                    |
             |<-------RPT(EDD1)------|                    |
             |<----------------------------RPT(EDD1)-------|
             |                       |                    |
             |                       |                    |
             |<-------RPT(EDD1)------|                    |
             |<----------------------------RPT(EDD1)-------|
             |                       |                    |
```

Figure 2

In a simple network, a Manager interacts with multiple Agents.

In this figure, the Manager configures Agents A and B to produce EDD1 every second in (1). Upon receiving and configuring this message, Agents A and B then build a Report Entry containing EDD1 and send those reports back to the Manager in (2). This behavior then repeats this action every 1s without requiring other inputs from the Manager.

8.2.3. Challenged, DTN Management

This is a challenged configuration of network management where Agent B temporarily looses connectivity between the agent and the Manager. Flows in this case are outlined in Figure 3.

Challenged Management Control Flow

```
           +----------+            +---------+           +---------+
           |  Manager |            | Agent A |           | Agent B |
           +----+-----+            +----+----+           +----+----+
                |                       |                     |
                |-----PROD(1s, EDD1)--->|                     | (1)
                |----------------------------PROD(1s, EDD1)-->|
                |                       |                     |
                |                       |                     |
                |<-------RPT(EDD1)------|                     | (2)
                |<----------------------------RPT(EDD1)-------|
                |                       |                     |
                |                       |                     |
                |<-------RPT(EDD1)------|                     |
                |<----------------------------RPT(EDD1)-------|
                |                       |                     |
                |                       |                     |
                |<-------RPT(EDD1)------|                     |
                |<                                   RPT(EDD1)| (3)
                |                       |                     |
                |                       |                     |
                |<-------RPT(EDD1)------|                     |
                |<                                   RPT(EDD1)|
                |                       |                     |
                |                       |                     |
                |<-------RPT(EDD1)------|                     |
                |<----------------RPT(EDD1, EDD1, EDD1)-------| (4)
                |                       |                     |
```

Figure 3

In a challenged network, an agent must store and forward reports until links are available.

In this figure, the Manager configures Agents A and B to produce EDD1 every second in (1). Upon receiving and configuring this message, Agents A and B then build a Report Entry containing EDD1 and send those reports back to the Manager in (2). At step (3) the connection between Agent B and Manager is not available. The agent still generates the reports and stores locally using DTN protocols. At step (4) the link has been restored and all stored reports are successfully delivered to the manager.

8.2.4. Consolidated Message Management

This is a configuration of network management where Agent B has been configured to deliver two sets of data and demonstrates the Agent's responsibility to consolidate messages for transport. Flows in this case are outlined in Figure 4.

```
   Consolidated Management Control Flow

        +----------+            +---------+           +---------+
        |  Manager |            | Agent A |           | Agent B |
        +----+-----+            +----+----+           +----+----+
             |                       |                     |
             |-----PROD(1s, EDD1)--->|                     | (1)
             |----------------------------PROD(1s, EDD1)-->|
             |                       |                     |
             |                       |                     |
             |<-------RPT(EDD1)------|                     | (2)
             |<----------------------------RPT(EDD1)-------|
             |                       |                     |
             |                       |                     |
             |----------------------------PROD(1s, EDD2)-->| (3)
             |                       |                     |
             |                       |                     |
             |<-------RPT(EDD1)------|                     | (4)
             |<--------------------------RPT(EDD1,EDD2)----|
             |                       |                     |
             |                       |                     |
             |<-------RPT(EDD1)------|                     |
             |<--------------------------RPT(EDD1,EDD2)----|
             |                       |                     |
```

Figure 4

In a challenged network, Agents shall consolidate messages where possible.

In this figure, the Manager configures Agents A and B to produce EDD1 every second in (1). Upon receiving and configuring this message, Agents A and B then build a Report Entry containing EDD1 and send those reports back to the Manager in (2). At step (3), the manager configures Agent B to additionally report EDD2 every second. At step (4) Agent B proceeds to deliver EDD1 and EDD2 in the same report.

8.2.5. Multiplexed Management

Networks spanning multiple administrative domains may require multiple Managers (for example, one per domain). When a Manager defines custom Reports/Variables to an Agent, that definition may be tagged with an Access Control List (ACL) to limit what other Managers will be privy to this information. Managers in such networks should synchronize with those other Managers granted access to their custom data definitions. When Agents generate messages, they MUST only send messages to Managers according to these ACLs, if present. The control flows in this scenario are outlined in Figure 5.

Multiplexed Management Control Flow

```
    +-----------+            +-------+           +-----------+
    | Manager A |            | Agent |           | Manager B |
    +-----+-----+            +---+---+           +-----+-----+
          |                      |                       |
          |---DEF(A,V1,EDD1*2)-->|<-DEF(B, V2, EDD2*2)--| (1)
          |                      |                       |
          |---PROD(1s, V1)------>|<---PROD(1s, V2)------| (2)
          |                      |                       |
          |<--------RPT(V1)------|                       | (3)
          |                      |--------RPT(V2)------>|
          |<--------RPT(V1)------|                       |
          |                      |--------RPT(V2)------>|
          |                      |                       |
          |                      |<---PROD(1s, V1)------| (4)
          |                      |                       |
          |                      |---ERR(V1 no perm.)-->|
          |                      |                       |
          |--DEF(*,V3,EDD3*3)--->|                       | (5)
          |                      |                       |
          |---PROD(1s, V3)------>|                       | (6)
          |                      |                       |
          |                      |<----PROD(1s, V3)-----|
          |                      |                       |
          |<--------RPT(V3)------|--------RPT(V3)------>| (7)
          |<--------RPT(V1)------|                       |
          |                      |--------RPT(V2)------>|
          |<-------RPT(V3)-------|--------RPT(V3)------>|
          |<-------RPT(V1)-------|                       |
          |                      |--------RPT(V2)------>|
```

Figure 5

Complex networks require multiple Managers interfacing with Agents.

In more complex networks, any Manager may choose to define custom Reports and Variables, and Agents may need to accept such definitions from multiple Managers. Variable definitions may include an ACL that describes who may query and otherwise understand these definitions. In (1), Manager A defines V1 only for A while Manager B defines V2 only for B. Managers may, then, request the production of Report Entries containing these definitions, as shown in (2). Agents produce different data for different Managers in accordance with configured production rules, as shown in (3). If a Manager requests the production of a custom definition for which the Manager has no permissions, a response consistent with the configured logging policy on the Agent should be implemented, as shown in (4). Alternatively, as shown in (5), a Manager may define custom data with no access restrictions, allowing all other Managers to request and use this definition. This allows all Managers to request the production of Report Entries containing this definition, shown in (6) and have all Managers receive this and other data going forward, as shown in (7).

8.2.6. Data Fusion

Data fusion reduces the number and size of messages in the network which can lead to more efficient utilization of networking resources. The DTNMA supports fusion of NM reports by co-locating Agents and Managers on nodes and offloading fusion activities to the Manager. This process is illustrated in Figure 6.

Data Fusion Control Flow

```
               ---------------------------------------
               |                Actor B              |
               |                                     |
+-----------+  |    +-----------+      +---------+   |    +---------+
| Manager A |  |    | Manager B |      | Agent B |   |    | Agent C |
+---+-------+  |    +-----+-----+      +----+----+   |    +----+----+
    |          |          |                 |        |         |
    |------------------DEF(A,V0,EDD1+EDD2)->|        |         | (1)
    |------------------PROD(EDD1&EDD2,V0)-->|        |         |
    |          |          |                 |        |         |
    |          |          |--PROD(1s,EDD1)->|        |         | (2)
    |          |          |--------------------PROD(1s, EDD2)->|
    |          |          |                 |        |         |
    |          |          |<---RPT(EDD1)----|       |          | (3)
    |          |          |<--------------------RPT(EDD2)------|
    |          |          |                 |        |         |
    |<------------------RPT(A,V0)-----------|        |         | (4)
    |          |          |                 |        |         |
    |          |          |                 |        |         |
               |                                     |
               |                                     |
               ---------------------------------------
```

Figure 6

Data fusion occurs amongst Managers in the network.

In this example, Manager A requires the production of a Variable V0, from node B, as shown in (1). The Manager role understands what data is available from what agents in the subnetwork local to B, understanding that EDD1 is available locally and EDD2 is available remotely. Production messages are produced in (2) and data collected in (3). This allows the Manager at node B to fuse the collected Report Entries into V0 and return it in (4). While a trivial example, the mechanism of associating fusion with the Manager function rather than the Agent function scales with fusion complexity, though it is important to reiterate that Agent and Manager designations are roles, not individual software components. There may be a single software application running on node B implementing both Manager B and Agent B roles.

9. IANA Considerations

This protocol has no fields registered by IANA.

10. Security Considerations

Security within an DTNMA MUST exist in two layers: transport layer security and access control.

Transport-layer security addresses the questions of authentication, integrity, and confidentiality associated with the transport of messages between and amongst Managers and Agents in the DTNMA. This security is applied before any particular Actor in the system receives data and, therefore, is outside of the scope of this document.

Finer grain application security is done via ACLs which are defined via configuration messages and implementation specific.

11. Informative References

[BIRRANE1] Birrane, E.B. and R.C. Cole, "Management of Disruption-Tolerant Networks: A Systems Engineering Approach", 2010.

[BIRRANE2] Birrane, E.B., Burleigh, S.B., and V.C. Cerf, "Defining Tolerance: Impacts of Delay and Disruption when Managing Challenged Networks", 2011.

[BIRRANE3] Birrane, E.B. and H.K. Kruse, "Delay-Tolerant Network Management: The Definition and Exchange of Infrastructure Information in High Delay Environments", 2011.

[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://www.ietf.org/archive/id/draft-ietf-core-comi-11.txt>.

[I-D.ietf-core-sid] Veillette, M., Pelov, A., Petrov, I., and C. Bormann, "YANG Schema Item iDentifier (YANG SID)", Work in Progress, Internet-Draft, draft-ietf-core-sid-16, 24 June 2021, <https://www.ietf.org/archive/id/draft-ietf-core-sid-16.txt>.

[I-D.ietf-core-yang-cbor] Veillette, M., Petrov, I., Pelov, A., and C. Bormann, "CBOR Encoding of Data Modeled with YANG", Work in Progress, Internet-Draft, draft-ietf-core-yang-cbor-16, 24 June 2021, <https://www.ietf.org/archive/id/draft-ietf-core-yang-cbor-16.txt>.

[I-D.irtf-dtnrg-dtnmp] Birrane, E. and V. Ramachandran, "Delay Tolerant Network Management Protocol", Work in Progress, Internet-Draft, draft-irtf-dtnrg-dtnmp-01, 31 December 2014, <http://www.ietf.org/internet-drafts/draft-irtf-dtnrg-dtnmp-01.txt>.

[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>.

[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>.

[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>.

[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", DOI 10.17487/RFC7228, RFC 7228, 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>.

[RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, DOI 10.17487/RFC7576, June 2015, <https://www.rfc-editor.org/info/rfc7576>.

[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>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

[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>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", DOI 10.17487/RFC8949, STD 94, RFC 8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

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

[RFC9172] Birrane, E., III., 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

Emery Annis
Johns Hopkins Applied Physics Laboratory

Email: Emery.Annis@jhuapl.edu

Sarah E. Heiner
Johns Hopkins Applied Physics Laboratory

Email: Sarah.Heiner@jhuapl.edu