Delay-Tolerant Networking
Internet-Draft
Intended status: Informational
Expires: February 8, 2020

E. Birrane
Johns Hopkins Applied Physics Laboratory

August 7, 2019

Asynchronous Management Architecture
draft-ietf-dtn-ama-00

Abstract

This document describes an asynchronous management architecture (AMA) suitable for providing application-level network management services in a challenged networking environment. Challenged networks are those that require fault protection, configuration, and performance reporting while unable to provide humans-in-the-loop with synchronous feedback or otherwise preserve transport-layer sessions. In such a context, networks must exhibit behavior that is both determinable and autonomous while maintaining compatibility with existing network management protocols and operational concepts.

Table of Contents

1. Introduction

The Asynchronous Management Architecture (AMA) provides application-layer network management services over links where delivery delays prevent timely communications between a network operator and a managed device. These delays may be caused by long signal propagations or frequent link disruptions (such as described in [RFC4838]) or by non-environmental factors such as unavailability of network operators, administrative delays, or delays caused by quality-of-service prioritizations and service-level agreements.

An AMA is necessary as the assumptions inherent to the architecture and design of synchronous management tools and techniques are not valid in challenged network scenarios. In these scenarios, synchronous approaches either patiently wait for periods of bi-directional connectivity or require the investment of significant time and resources to evolve a challenged network into a well-connected, low-latency network. In some cases such evolution is merely a costly way to over-resource a network. In other cases, such evolution is impossible given physical limitations imposed by signal propagation delays, power, transmission technologies, and other phenomena. Asynchronous management of asynchronous networks enables large-scale deployments, distributed technical capabilities, and reduced deployment and operations costs.

The rationale and motivation for asynchronous management is captured in [BIRRANE1], [BIRRANE2],[BIRRANE3]. The properties and feasibility of such a system are taken from prototyping work done in accordance with [I-D.irtf-dtnrg-dtnmp].

1.1. Scope

This document describes the motivation, service definitions, desirable properties, roles/responsibilities, system model, and logical data model that form the AMA. These descriptions should be of sufficient specificity that implementations conformant to this architecture will operate successfully in a challenged networking environment.

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

It is assumed that any challenged network where network management would be usefully applied supports basic services (where necessary) such as naming, addressing, integrity, confidentiality, authentication, fragmentation, and traditional network/session layer functions. Therefore, these items are outside of the scope of the AMA and not covered in this document.

While possible that a challenged network may interface with an unchallenged network, this document does not address the concept of network management compatibility with synchronous 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 seven sections that, together, describe an AMA suitable for enterprise management of asynchronous networks: terminology, motivation, service definitions, desirable properties, roles/responsibilities, logical data model, and system model. The description of each section is as follows.

- Terminology - This section identifies those terms critical to understanding the proper operation of the AMA. 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 as providing a novel and useful alternative to current network management approaches. Specifically, this section describes common network functions and how synchronous mechanisms fail to provide these functions in an asynchronous environment.

- Service Definitions - This section defines asynchronous network management services in terms of terminology, scope, and impact.

- Desirable Properties - This section identifies the properties to which an asynchronous management system should adhere to effectively implement service definitions in an asynchronous environment. These properties guide the subsequent definition of the system and logical models that comprise the AMA.

- Roles and Responsibilities - This section identifies the roles in the AMA and their associated responsibilities. It provides the

terminology and context for discussing how network management services interact.

- Logical Data Model - This section describes the kinds of data that should be represented in deployment asynchronous management system.

- System Model - This section describes data flows amongst various defined Actor roles. These flows capture how the AMA system works to provide asynchronous network management services in accordance with defined desirable properties.

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) - The role associated with a managed device, responsible for reporting performance data, enforcing administrative policies, and accepting/performing actions. Agents exchange information with Managers operating either on the same device or on a remote managing device.

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

- Variables (VARs) - Information that is computed by an 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 3.14.

- Controls (CTRLs) - Operations that may be undertaken by an Actor to change the behavior, configuration, or state of an application or protocol managed by an AMP.

- Literals (LITs) - A literal represents a 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 (MACs) - A named, ordered collection of Controls.

- Manager - A role associated with a managing device responsible for configuring the behavior of, and receiving information from, Agents. Managers interact with one or more 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 Agent state.

- Report (RPT) - A typed, ordered collection of data values gathered by one or more Agents and provided to one or more 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 structure of a Report (RPT). This is the schema for a Report, generated by a Manager and communicated to one or more Agents.

- Rule - A unit of autonomous specification that provides a stimulus-response relationship between time or state on an Agent and the Controls to be run as a result of that time or state.

- State-Based Rule (SBR) - A state-based rule is any rule in which the rule stimulus is triggered by the calculable internal state of the Agent.

- 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. Reports only contain typed data values and the identity of the Table Template (TBLT) to which they confirm.

- 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 only considers relative time as it is known on the Agent.

3. Motivation

Challenged networks, to include networks challenged by administrative or policy delays, cannot guarantee capabilities required to enable synchronous management techniques. These capabilities include high-rate, highly-available data, round-trip data exchange, and operators "in-the-loop". The inability of current approaches to provide

network management services in a challenged network motivates the need for a new network management architecture focused on asynchronous, open-loop, autonomous control of network components.

3.1. Challenged Networks

A growing variety of link-challenged networks support packetization to increase data communications reliability without otherwise guaranteeing a simultaneous end-to-end path. Examples of such networks include Mobile Ad-Hoc Networks (MANets), Vehicular Ad-Hoc Networks (VANets), Space-Terrestrial Internetworks (STINTs), and heterogeneous networking overlays. Links in such networks are often unavailable due to attenuations, propagation delays, occultation, and other limitations imposed by energy and mass considerations. Data communications in such networks rely on store-and-forward and other queuing strategies to wait for the connectivity necessary to usefully advance a packet along its route.

Similarly, there also exist well-resourced networks that incur high message delivery delays due to hardware, software, or human limitations. Some examples of these networks are networks with understaffed operations centers and where data volume and management requirements exceed the real-time cognitive load of operators and/or their associated operations console software support. Also, networks that restrict user access to existing bandwidth due to policy create functionally similar situations to that of link disruption and delay.

Independent of the reason, when a node experiences an inability to communicate it must rely on autonomous mechanisms to ensure its safe operation and ability to usefully re-join the network at a later time. Additionally, nodes in a sparsely populated network may often be disconnected, making the concepts of "connected network" and "instantaneous connectivity" either impractical or impossible.

Specifically, challenged networks exhibit the following properties that may violate assumptions built into current approaches to synchronous network management.

- Links may be uni-directional.
- Bi-directional links may have asymmetric data rates.
- 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.

3.2. Current Approaches and Their Limitations

Network management tools in unchallenged networks provide mechanisms for communicating locally-collected data from Agents to Managers, typically using a "pull" mechanism where data must be explicitly requested by a Manager in order to be transmitted by an Agent.

Management approaches that rely on timely data exchange, such as those that rely on negotiated sessions or other synchronized acknowledgment, do not function in challenged network environments. Familiar examples of TCP/IP based management via closed-loop, synchronous messaging do not work when network disruptions increase in frequency and severity. While no protocol delivers data in the absence of a networking link, protocols that eliminate or drastically reduce overhead and end-point coordination require smaller transmission windows and continue to function when confronted with scaling delays and disruptions in the network.

A legacy method for management in unchallenged networks today 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 asynchronous notification messages, called traps, based on predefined triggering events. Thus, Managers can query Agents for status information, send new configurations, and be informed when specific events have occurred. Traps and queryable data are defined in one or more 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 in inappropriate for use in a challenged networking environment. SNMP relies on sessions with low round-trip latency to support its "pull" model. The SNMP trap model provides some Agent-side processing, however because the processing has very low fidelity and traps are typically "fire and forget," the underlying transport protocol that supports reliable, in-order message delivery is required. Adaptive modifications to SNMP to support challenged networks would alter the basic function of the protocol (data models, control flows, and syntax) so as to be functionally incompatible with existing SNMP installations. Therefore, this approach is not suitable for an asynchronous network management system.

The Network Configuration Protocol (NETCONF) provides device-level configuration capabilities [RFC6241] to replace vendor-specific

command line interface configuration software. The XML-based protocol provides a remote procedure call (RPC) syntax such that any exposed functionality on an Agent can be exercised via a software application interface. NETCONF places no specific functional requirements or constraints on the capabilities of the Agent, which makes it a very flexible tool for configuring a homogeneous network of devices.

NETCONF places specific constraints on any underlying transport protocol: a long-lived, reliable, low-latency sequenced data delivery session. This is a fundamental requirement given the RPC-nature of the operating concept, and it is unsustainable in a challenged network. Aspects of the data modeling associated with NETCONF may apply to an asynchronous network management system, such that some modeling tools may be used, even if the network control plane cannot.

Just as the concept of a loosely-confederated set of nodes changes the definition of a network, it also changes the operational concept of what it means to manage a network. When a network stops being a single entity exhibiting a single behavior, "network management" becomes large-scale "node management". Individual nodes must share the burden of implementing desirable behavior without reliance on a single oracle of configuration or other coordinating function such as an operator-in-the-loop.

4. Service Definitions

This section identifies the services that must exist between Managers and Agents within an AMA. These services include configuration, reporting, parameterized control, and administration.

4.1. 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.

New configurations received by an Agent must be validated to ensure that they do not conflict with other configurations or would otherwise prevent the Agent from effectively working with other Actors in its region. 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).

4.2. 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).

4.3. 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 effect 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 AMA: triggers based on the general 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.

4.4. Administration

Administration services enforce the potentially complex mapping of configuration, reporting, and control services amongst Agents and Managers in the network. Fine-grained access controls that specify which Managers may apply which services to which Agents may be necessary 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.

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 AMP messages.

5. Desirable Properties

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 AMA.

5.1. 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 roundtrip 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 roundtrip 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 make their own determinations in relation to the information that should be sent to Managers. 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.

5.2. 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. AMPs should minimize PDUs whenever practical, to include packing and unpacking binary data, variable-length fields, and pre-configured data definitions.

5.3. 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.

5.4. 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. The Agent should calculate the smaller data set on its own and transmit that instead. 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.

5.5. Autonomous Operation

AMA 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 are preferred as they can be flexible without incurring a set of problematic requirements or concerns.

6. Roles and Responsibilities

By definition, Agents reside on managed devices and Managers reside on managing devices. This section describes how these roles participate in the network management functions outlined in the prior section.

6.1. Agent Responsibilities

Application Support
: Agents MUST collect all data, execute all procedures, populate all reports and run operations required by each application which the Agent claims to manage. Agents MUST report supported applications 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.

Autonomous Control
: Agents MUST determine, without Manager intervention, whether a procedure should be invoked. Agents MAY also invoke procedures on other devices for which they act as proxy.

User 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.

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

Regional Proxy
: Agents MAY perform any of their responsibilities on behalf of other network nodes that, themselves, do not have an Agent. In such a configuration, the Agent acts as a proxy for these other network nodes.

6.2. Manager Responsibilities

Agent Capabilities Mapping
: Managers MUST understand what applications are managed by the various Agents with which they communicate. Managers should not attempt to request, invoke, or refer to application information for applications not managed by an Agent.

Data Collection
: Managers MUST receive information from Agents by asynchronously configuring the production of reports and then waiting for, and 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 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 Translation

Managers should provide some interface to other network management protocols. Managers MAY accomplish this by accumulating a repository of push-data from high-latency parts of the network from which data may be pulled by low-latency parts of the network.

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.

7. Logical Data Model

The AMA 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 an asynchronously managed network.

The elements of the AMA 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 AMA: (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 types, named values that cannot be modified once they have been defined.

Data that are 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 AMA. 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 AMA, the structure of the ordered collection is treated separately from the values that populate such a structure.

The AMA 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 AMA may define common RPTTs. Additionally, users within a network may define their own RPTTs that are useful in the context of a particular deployment.

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 syntactical 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 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 AMA enables similar capabilities without requiring that the Manager be in the processing loop of the Agent. Command execution in the AMA happens through the use of controls and macros.

A Control (CTRL) represents a parameterized, predefined procedure that can be run on an Agent. CTRLs do not have a return code as there is not the same concept of sequential execution in an asynchronous model. Parameters can be provided when running a command from a Manager, pre-configured as part of an autonomy response 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 AMA 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. An asynchronous 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-effecting components.

Often, a series of controls must be executed in concert 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 AMA 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 in response to the state of the Agent. This mechanism in the AMA is the "rule" and can key activated based on Agent 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 turn off a heater if some internal temperature is greater than a threshold:

```
IF (current_temp > maximum_temp) THEN turn_heater_off
```

Rules should be allowed to construct their stimuli from the full set of EDD values and VARs available to the network management system. Similarly, macro responses should be allowed to include controls from all applications known by the Agent. This enables an expressive capability to have multiple applications monitored and managed by the Agent.

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:

```
(((current_time - base_time) % 24_hours) == 0)
```

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 the the aforementioned AMA 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" - it's 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 AMA logical model. Operands include any AMA 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 AMA cooperate to perform network management services.

8.1. Control and Data Flows

The AMA 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.

AMA Control and Data Flows

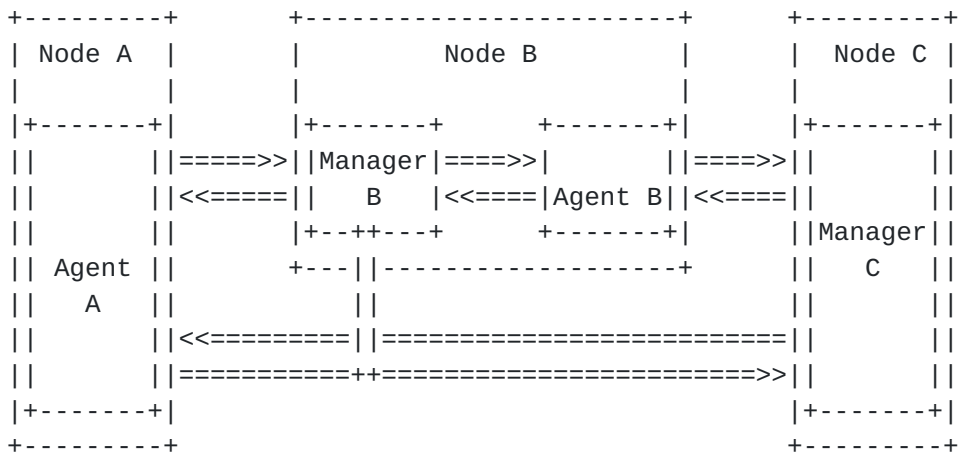


Figure 1

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 differing 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.

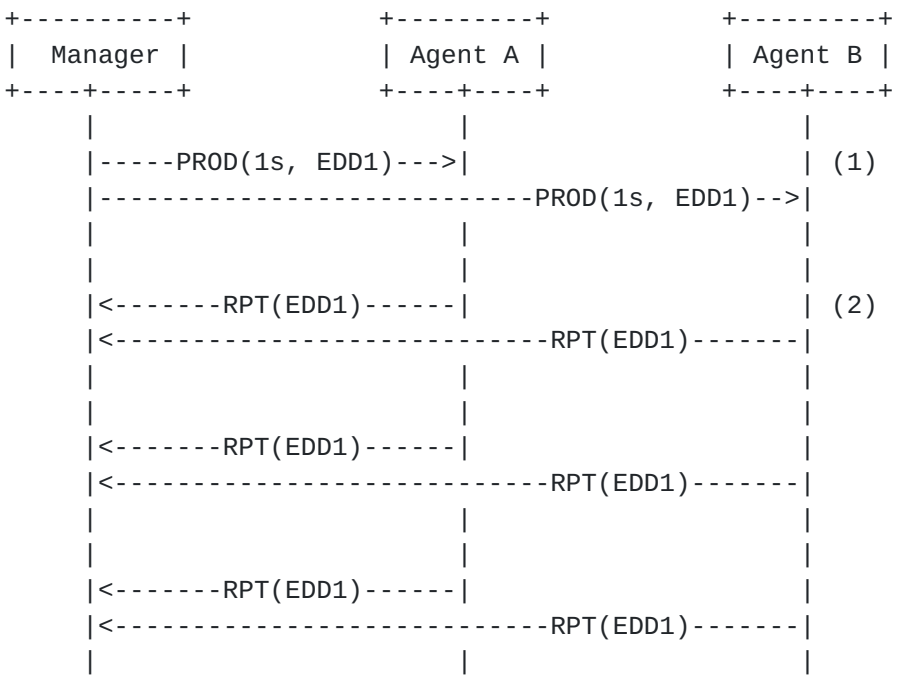
Term	Definition	Example
EDD#	EDD definition.	EDD1
V#	Variable definition.	V1 = EDD1 + V0.
DEF([ACL], ID,EXPR)	Define id from expression. Allow managers in access control list (ACL) to request this id.	DEF([*], V1, EDD1 + EDD2)
PROD(P,ID)	Produce ID according to predicate P. P may be a time period (1s) 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



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

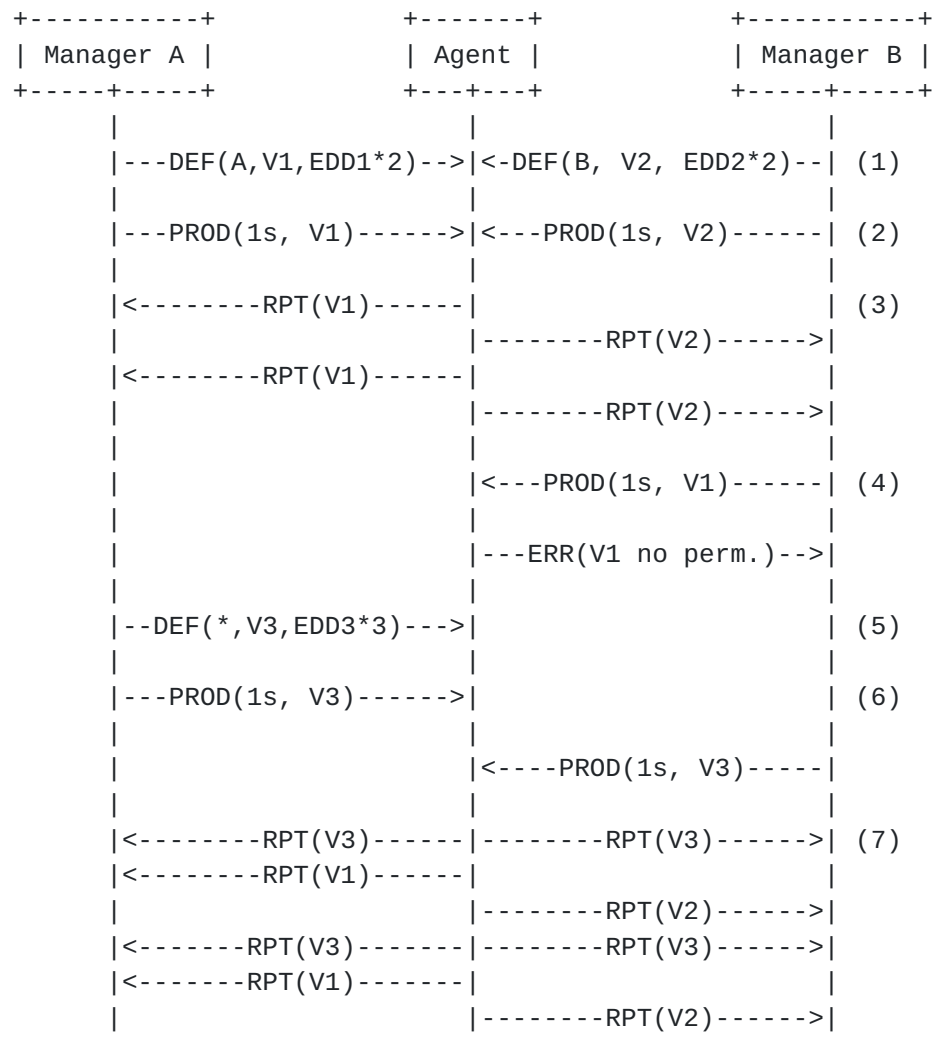
Figure 2

In this figure, the Manager configures Agents A and B to produce EDD1 every second in (1). At some point in the future, 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).

8.2.3. 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 3.

Multiplexed Management Control Flow



Complex networks require multiple Managers interfacing with Agents.

Figure 3

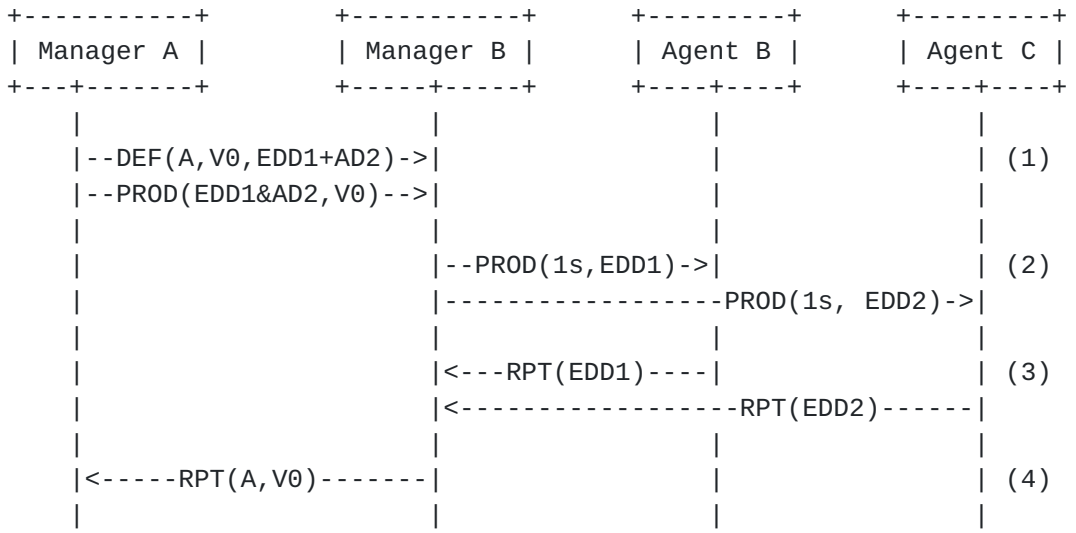
In more complex networks, any Manager may choose to define custom Reports and Variables, and Agents may need to accept such definitions from multiple Managers. Variable definitions may include an ACL that describes who may query and otherwise understand these definitions. In (1), Manager A defines V1 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 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).

8.2.4. Data Fusion

In some networks, Agents do not individually transmit their data to a Manager, preferring instead to fuse reporting data with local nodes prior to transmission. This approach reduces the number and size of messages in the network and reduces overall transmission energy expenditure. The AMA 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 4.

Data Fusion Control Flow



Data fusion occurs amongst Managers in the network.

Figure 4

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 AMA 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 AMA. 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. and R. Cole, "Management of Disruption-Tolerant Networks: A Systems Engineering Approach", 2010.

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

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

[I-D.irtf-dtnrg-dtnmp]
Birrane, E. and V. Ramachandran, "Delay Tolerant Network Management Protocol", draft-irtf-dtnrg-dtnmp-01 (work in progress), December 2014.

[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, April 2007.

[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>.

Author's Address

Edward J. Birrane
Johns Hopkins Applied Physics Laboratory

Email: Edward.Birrane@jhuapl.edu