

tbd BOF
Internet-Draft
Intended status: Informational
Expires: November 10, 2014

M. Behringer
M. Pritikin
S. Bjarnason
Cisco
May 9, 2014

Autonomic Networking Use Case for Network Bootstrap
draft-behringer-autonomic-bootstrap-00

Abstract

This document describes a use case for a specific autonomic functionality: Securely bootstrapping new devices into a network, without the need for pre-staging. The goal is to illustrate a requirement from network operators, and to illustrate how autonomic approaches can solve this use case.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 10, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Problem Statement	3
3. Benefits of an Autonomic Solution	3
4. Intended User and Administrator Experience	4
5. Analysis of Parameters and Information Involved	4
5.1. Device Based Self-Knowledge and Decisions	4
5.2. Interaction with other devices	4
5.3. Information needed from Intent	5
5.4. Monitoring, diagnostics and reporting	5
6. Comparison with current solutions	5
7. Security Considerations	5
8. IANA Considerations	6
9. Acknowledgements	6
10. Change log [RFC Editor: Please remove]	6
11. Informational References	6
Authors' Addresses	7

1. Introduction

Autonomic Networking (AN) is a new way to manage network devices or functions, by making them self-managing. Rather than centralising all intelligence on a controller, the idea is to distribute intelligence across the network. The fundamental concepts of Autonomic Networking are described in [I-D.irtf-nmrg-autonomic-network-definitions]. A gap analysis for AN has been documented in [I-D.irtf-nmrg-an-gap-analysis].

This document describes a use case for a specific autonomic functionality: Securely bootstrapping new devices into a network, without the need for pre-staging. The goal is to illustrate a requirement from network operators, and to illustrate how autonomic approaches can solve this use case.

Bootstrapping in the context of this document refers to the process in which a new device joins a network in a secure way. A secure, zero-touch bootstrap process is defined here as a process without pre-staging, where the a new device can

- o [A] authenticate itself securely, such that no other entity can perform a man-in-the-middle attack or masquerade as an expected device. (see [RFC4949] for definitions of "man-in-the-middle attack" and "masquerade");

- o [B] authenticate the network securely, such that the device joins only the correct network.

It is required to bootstrap a device to make it securely manageable by a controller or network management system. There are networks where requirement [A] is a must, requirement [B] a should. In some networks both [A] and [B] are a must. In either case the behavior of a new device must be the same, to prevent possible downgrade attacks.

2. Problem Statement

Today, there are two main ways to bootstrap a device:

- o Pre-configuring the device with a minimum or full configuration, so that it can be at least securely reached and managed. This process can be secure, as defined in the introduction.
- o Using so called "zero-touch" methods. Many of those are derived from some form of DHCP interaction. DHCP however is not suitable to establish trust between the network and the new device. As a result, any device can join the domain through this process. Additionally DHCP, as many other bootstrap methods today, does not provide a mechanism for identifying the network.

In some networks it is not acceptable to not be able to authenticate new devices. Examples are:

- o In Metro Ethernet and Mobile RAN scenarios many network nodes are deployed outside traditional data-center like environments, for example in the basement of an office building. While still typically in a locked cabinet or room, the operator has less control in these environments, and there is increasing concern that hackers might be able to connect their own devices, to receive valuable configuration details and possibly infiltrate the network.
- o In industrial automation, certain wireless sensors and actuators fulfill mission critical tasks and it must be assured that only legitimate devices can connect.

3. Benefits of an Autonomic Solution

For a secure, zero-touch device enrolment, a device must act "autonomically". It cannot rely on a central entity such as a controller or DHCP server, because the problem is exactly to establish trust with this controller in the first place. Therefore, in this use case an autonomic solution is absolutely required.

4. Intended User and Administrator Experience

On the side of the new network element, an unskilled person should be able to connect the device following a simple cabling scheme, perhaps only supplying power to a wireless device. Once connected, the device must be powered on. There should be no need for any configuration tasks.

On the network side, the operator of the network must have full control over which element joins the network in which location. It must be possible to securely identify the new device, such that no other device can take its role. This can be achieved by validating unique device identifiers (UDI), for example serial numbers. While authorization of the UDI is required, it can be further automated, for example providing a white list of valid device UDIs. The actual process of enrolment and bootstrapping should be zero-touch for the operator.

5. Analysis of Parameters and Information Involved

5.1. Device Based Self-Knowledge and Decisions

Each device has self-knowledge about its own identity. This can be a unique device identifier such as a serial number. For a fully secure process, the device requires an IEEE 802.1AR [IDevID] credential.

5.2. Interaction with other devices

A new device interacts only with a neighbouring domain device. All communications are link local, therefore the new device does not require a routable IP address. The neighbouring device acts as a proxy for the below information flows.

A new device exchanges data through the neighboring proxy with two entities:

- o A "Registrar" device, acting as the trust anchor of the domain. The new device sends its own credentials to the Registrar, and after successful authentication, receives domain information, to enable subsequent enrolment to the domain. The Registrar sends all required information: a device name, domain name, plus some parameters for the enrolment.
- o A cloud service provided by the vendor to facilitate autonomic mechanisms. The new device may receive a signed authorization token from the vendor cloud service, telling the device its target domain. The authorization token can be obtained at the time of deployment or a priori at the time of white list authorization.

Based on the above interactions with a network a device can decide locally whether or not to join a domain, and a domain can decide whether to enrol a device or not.

5.3. Information needed from Intent

Intent is not required for this autonomic process.

5.4. Monitoring, diagnostics and reporting

The process can be monitored in several points:

The new device logs all transactions locally. Since it does not have a trust relationship with a domain at the beginning, it cannot report its data anywhere at the beginning of the process. Such data should be logged locally.

The proxy device logs all data relating to the connection of new devices, information received, etc. The proxy is also a potential attack point, since by nature it must listen to packets from unauthenticated nodes. All such information must be logged in the domain, and there must be local diagnostic tools to validate the state of each transaction and node.

The Registrar device logs all connection attempts, as well as successful connections. Also here, local diagnostic tools are required.

The vendor cloud service also logs all interactions with domains.

6. Comparison with current solutions

Today, the task of bootstrapping a new device is either done through pre-staging or in an insecure way, where any device could join the domain. An autonomic approach allows the process to be secure, while still being zero-touch.

The document [I-D.pritikin-bootstrapping-keyinfrastructures] outlines a possible solution to the use case described here.

7. Security Considerations

An autonomic approach can be used to make a bootstrap process secure. As only link local addresses are used in the process, only nodes directly adjacent to a potentially malicious node can be attacked. Denial of service prevention must be applied on the proxy nodes. The other network elements must be secured according to general best

practices, but require no specific security with regard to this approach.

8. IANA Considerations

This document requests no action by IANA.

9. Acknowledgements

This document has been written with the XML2RFC tool [RFC2629].

10. Change log [RFC Editor: Please remove]

version 0: Initial version.

11. Informational References

[I-D.irtf-nmrg-an-gap-analysis]

Behringer, M., Carpenter, B., and S. Jiang, "Gap Analysis for Autonomic Networking", draft-irtf-nmrg-an-gap-analysis-00 (work in progress), April 2014.

[I-D.irtf-nmrg-autonomic-network-definitions]

Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking - Definitions and Design Goals", draft-irtf-nmrg-autonomic-network-definitions-00 (work in progress), December 2013.

[I-D.pritikin-bootstrapping-keyinfrastructures]

Pritikin, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", draft-pritikin-bootstrapping-keyinfrastructures-00 (work in progress), January 2014.

[IDevID] IEEE Standard, , "IEEE 802.1AR Secure Device Identifier", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

Authors' Addresses

Michael H. Behringer
Cisco

Email: mbehring@cisco.com

Max Pritikin
Cisco

Email: pritikin@cisco.com

Steinthor Bjarnason
Cisco

Email: sbjarnas@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 22, 2014

M. Behringer
S. Bjarnason
Balaji. BL
T. Eckert
Cisco
June 20, 2014

An Autonomic Control Plane
draft-behringer-autonomic-control-plane-00

Abstract

In certain scenarios, for example when bootstrapping a network, it is desirable to automatically bring up a secure, routed control plane, which is independent of device configurations and global routing table. This document describes an approach for an "Autonomic Control Plane", which can be used as a "virtual out of band channel" - a self-managing overlay network, which is independent of configuration, addressing and routing on the data plane.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 22, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Problem Statement	3
3. Self-Creation of an Autonomic Control Plane	3
3.1. Preconditions	4
3.2. Adjacency Discovery	4
3.3. Authenticating Neighbors	4
3.4. Capability Negotiation	5
3.5. Channel Establishment	5
3.6. Context Separation	5
3.7. Addressing in the ACP	6
3.8. Routing in the ACP	6
4. Self-Healing Properties	7
5. Self-Protection Properties	7
6. Use Cases for the ACP	8
7. The Administrator View	8
8. Security Considerations	9
9. IANA Considerations	9
10. Acknowledgements	9
11. Change log [RFC Editor: Please remove]	9
12. References	9
Authors' Addresses	10

1. Introduction

Today, the management and control plane of networks typically runs in the global routing table, which is dependent on correct configuration and routing. Misconfigurations or routing problems can therefore disrupt management and control channels. Traditionally, an out of band network has been used to recover from such problems, or personnel is sent on site to access devices through console ports. However, both options are operationally expensive.

In increasingly automated networks either controllers or autonomic service agents in the network require a control plane which is independent of the network they manage, to avoid impacting their own operations.

This document describes a self-forming, self-managing and self-protecting "Autonomic Control Plane" (ACP) which is inband on the network, yet independent of configuration, addressing and routing problems. It therefore remains operational even in the presence of

configuration errors, addressing or routing issues, or where policy could inadvertently affect control plane connectivity. It serves as a "virtual out of band channel": An operator can use it to log into remote devices in such cases. And an SDN controller can use it to securely bootstrap network devices in remote locations, even if the network in between is not yet configured; no data-plane dependent bootstrap configuration is required. An example of such a secure bootstrap process is described in [I-D.pritikin-bootstrapping-keyinfrastructures]

2. Problem Statement

An "Autonomic Control Plane" (ACP) provides a solution to some of today's operational challenges. These fall into three broad categories:

- o Bootstrapping a network while devices are not yet configured. Bootstrapping a new device today requires all devices between the controller and the new device to be completely and correctly addressed, configured and secured. Therefore, bootstrapping a network happens in layers around the controller. Without console access it is not possible today to make devices securely reachable before having configured the entire network between.
- o Maintaining reachability of network devices even in the case of certain forms of misconfiguration and routing issues. For example: certain AAA misconfigurations can lock an administrator out of a device; routing or addressing issues can make a device unreachable; shutting down interfaces over which a current management session is running can lock an admin irreversibly out of the device. Traditionally only console access can help recover from such issues.
- o Data plane dependencies for NOC/SDN controller applications: Certain network changes are today hard to operate, because the change itself may affect reachability of the devices. Examples are address or mask changes, routing changes, or security policies. Today such changes require precise hop-by-hop planning; an ACP would simplify them.

3. Self-Creation of an Autonomic Control Plane

This section describes the steps to set up an Autonomic Control Plane, and highlights the key properties which make it "indestructible" against many inadvertent changes to the data plane, for example caused by misconfigurations.

3.1. Preconditions

Each autonomic device has a globally unique domain certificate, with which it can cryptographically assert its membership of the domain. The document [I-D.pritikin-bootstrapping-keyinfrastructures] describes how a domain certificate can be automatically and securely derived from a vendor specific Unique Device Identifier (UDI) or IDevID certificate.

3.2. Adjacency Discovery

Adjacency discovery exchanges identity information about neighbors, either the UDI or, if present, the domain certificate (see Section 3.1. This document assumes the existence of a domain certificate.

Adjacency discovery provides a table of information of adjacent neighbours. Each neighbour is identified by an globally unique device identifier (UDI).

The adjacency table contains the following information about the adjacent neighbours.

- o Globally valid Unique device identifier (UDI)
- o Link Local IPv6 address
- o Trust information
- o Validity of the trust

Adjacency discovery can populate this table by several means. One such mechanism is to discover using link local multicast probes, which has no dependency on configured addressing and is preferable in an autonomic network.

3.3. Authenticating Neighbors

Each neighbour in the adjacency table is authenticated. The result of the authentication of the neighbour information is stored in the adjacency table. We distinguish the following cases:

- o Inside the domain: If the domain certificate presented is validated to be in the same domain as that of the autonomic entity then the neighbour is deemed to be inside the autonomic domain. Only entities inside the autonomic domain will by default be able to establish the autonomic control plane. An ACP channel will be established.

- o Outside the domain: If there is no domain certificate presented by the neighbour, or if the domain certificate presented is invalid or expired, then the neighbour is deemed to be outside the autonomic domain. No ACP channel will be established.

3.4. Capability Negotiation

Autonomic devices have different capabilities based on the type of device and where it is deployed. To establish a trusted secure communication channel, devices must be able to negotiate with each neighbour a set of parameters for establishing the communication channel, most notably channel type and security type. The channel type could be any tunnel mechanism that is feasible between two adjacent neighbours, for example a GRE tunnel. The security type could be any of the channel protection mechanism that is available between two adjacent neighbours on a given channel type, for example IPSEC. The establishment of the autonomic control plane can happen after the channel type and security type is negotiated.

3.5. Channel Establishment

After authentication and capability negotiation autonomic nodes establish a secure channel towards their direct AN neighbours with the above negotiated parameters. In order to be independent of configured link addresses, these channels can be implemented in several ways:

- o As a secure IP tunnel (e.g., IPsec), using IPv6 link local addresses between two adjacent neighbours. This way, the ACP tunnels are independent of correct network wide routing. They also do not require larger than link local scope addresses, which would normally need to be configured or maintained. Each AN node MUST support this function.
- o L2 separation, for example via a separate 802.1q tag for ACP traffic. This even further reduces dependency against the data plane (not even IPv6 link-local there required), but may be harder to implement.

3.6. Context Separation

The ACP is in a separate context from the normal data plane of the device. This context includes the ACP channels IPv6 forwarding and routing as well as any required higher layer ACP functions.

In classical network device platforms, a dedicated so called "Virtual routing and forwarding instance" (VRF) is one logical implementation option for the ACP. If possible by the platform SW architecture,

separation options that minimize shared components are preferred. The context for the ACP needs to be established automatically during bootstrap of a device and - as necessitated by the implementation option be protected from being modified unintentional from data plane configuration.

In addition this provides for security, because the ACP is not reachable from the global routing table. Also, configuration errors from the data plane setup do not affect the ACP.

3.7. Addressing in the ACP

The channels explained above only establish communication between two adjacent neighbours. In order for the communication to happen across multiple hops, the autonomic control plane requires internal network wide valid addresses and routing. Each autonomic node must create a loop back interface with a network wide unique address inside the ACP context mentioned in Section 3.6.

We suggest to create network wide Unique Local Addresses (ULA) in accordance with [RFC4193] with the following algorithm:

- o Prefix FC01::/8
- o Global ID: a hash of the domain ID; this way all devices in the same domain have the same /48 prefix.
- o Subnet ID and interface ID: These can be either derived deterministically from the name of the device, or assigned at registration time of the device.

3.8. Routing in the ACP

Once ULA address are set up all autonomic entities should run a routing protocol within the autonomic control plane context. This routing protocol distributes the ULA created in the previous section for reachability. The use of the autonomic control plane specific context eliminates the probable clash with the global routing table and also secures the ACP from interference from the configuration mismatch or incorrect routing updates.

The establishment of the routing plane and its parameters are automatic and strictly within the confines of the autonomic control plane. Therefore, no manual configuration is required.

All routing updates are automatically secured in transit as the channels of the autonomic control plane are by default secured.

The routing protocol inside the ACP should be light weight and highly scalable to ensure that the ACP does not become a limiting factor in network scalability. We suggest the use of RPL as one such protocol which is light weight and scales well for the control plane traffic.

4. Self-Healing Properties

The ACP is self-healing:

- o New neighbors will automatically join the ACP after successful validation and will become reachable using their unique ULA address across the ACP.
- o When any changes happen in the topology, the routing protocol used in the ACP will automatically adapt to the changes and will continue to provide reachability to all devices.
- o If an existing device gets revoked, it will automatically be denied access to the ACP as its domain certificate will be validated against a Certificate Revocation List during authentication.

5. Self-Protection Properties

As explained in Section 3, the ACP is based on channels being built between devices which have been previously been authenticated based on their domain certificates. The channels themselves are protected using standard encryption technologies like IPsec which provide additional authentication during channel establishment, data integrity and data confidentiality protection of data inside the ACP and in addition, provide replay protection.

An attacker will therefore not be able to join the ACP unless having a valid domain certificate, also packet injection and sniffing traffic will not be possible due to the security provided by the encryption protocol.

The remaining attack vector would be to attack the underlying AN protocols themselves, either via directed attacks or by denial-of-service attacks. However, as the ACP is built using link-local IPv6 address, remote attacks are impossible. The ULA addresses are only reachable inside the ACP context, therefore unreachable from the data plane. Also, the ACP protocols should be implemented to be attack resistant and not consume unnecessary resources even while under attack.

6. Use Cases for the ACP

The ACP automatically enables a number of use cases which provide immediate benefits:

- o Secure bootstrap of new devices without requiring any configuration. As explained in Section 3, a new device will automatically be bootstrapped in a secure fashion and be deployed with a domain certificate. This will happen without any configuration, allowing a new device to be shipped directly to the end-user location without the need for any pre-provisioning.
- o Virtual-out-of-band (VooB) control plane which provides connectivity to all devices regardless of their configuration or global routing table. This makes it possible to manage devices without having to configure data plane services or to deploy a separate management network. It also simplifies management applications, because changes done by the applications cannot affect reachability of the devices.

7. The Administrator View

An ACP is self-forming, self-managing and self-protecting, therefore has minimal dependencies on the administrator of the network. Specifically, it cannot be configured, there is therefore no scope for configuration errors on the ACP itself. The administrator may have the option to enable or disable the entire approach, but detailed configuration is not possible. This means that the ACP must not be reflected in the running configuration of devices, except a possible on/off switch.

While configuration is not possible, an administrator must have full visibility of the ACP and all its parameters, to be able to do trouble-shooting. Therefore, an ACP must support all show and debug options, as for any other network function. Specifically, a network management system or controller must be able to discover the ACP, and monitor its health. This visibility of ACP operations must clearly be separated from visibility of data plane so automated systems will never have to deal with ACP aspect unless they explicitly desire to do so.

Since an ACP is self-protecting, a device not supporting the ACP, or without a valid domain certificate cannot connect to it. This means that by default a traditional controller or network management system cannot connect to an ACP. To make this possible for systems not supporting the ACP natively, the connection to the ACP must be manually established, through configuration. [EDNOTE: More details to be provided in a later version of this document.] Long term NMS

systems might become autonomic devices with domain certificates, and then automatically join the ACP.

8. Security Considerations

An ACP is self-protecting and there is no need to apply configuration to make it secure. Its security therefore does not depend on configuration.

However, the security of the ACP depends on a number of other factors:

- o The usage of domain certificates depends on a valid supporting PKI infrastructure. If the chain of trust of this PKI infrastructure is compromised, the security of the ACP is also compromised. This is typically under the control of the network administrator.
- o Security can be compromised by implementation errors (bugs), as in all products.

Fundamentally, security depends on correct operation, implementation and architecture. Autonomic approaches such as the ACP largely eliminate the dependency on correct operation; implementation and architectural mistakes are still possible, as in all networking technologies.

9. IANA Considerations

This document requests no action by IANA.

10. Acknowledgements

This work originated from an Autonomic Networking project at Cisco Systems, which started in early 2010. Many people contributed to this project and the idea of the Autonomic Control Plane, amongst which (in alphabetical order): Ignas Bagdonas, Parag Bhide, Alex Clemm, Toerless Eckert, Yves Hertoghs, Bruno Klauser, Max Pritikin, Ravi Kumar Vadapalli.

11. Change log [RFC Editor: Please remove]

00: Initial version.

12. References

[I-D.irtf-nmrg-an-gap-analysis]

Behringer, M., Carpenter, B., and S. Jiang, "Gap Analysis for Autonomic Networking", draft-irtf-nmrg-an-gap-analysis-00 (work in progress), April 2014.

[I-D.irtf-nmrg-autonomic-network-definitions]

Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking - Definitions and Design Goals", draft-irtf-nmrg-autonomic-network-definitions-00 (work in progress), December 2013.

[I-D.pritikin-bootstrapping-keyinfrastructures]

Pritikin, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", draft-pritikin-bootstrapping-keyinfrastructures-00 (work in progress), January 2014.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

Authors' Addresses

Michael H. Behringer
Cisco

Email: mbehring@cisco.com

Steinthor Bjarnason
Cisco

Email: sbjarnas@cisco.com

Balaji BL
Cisco

Email: blbalaji@cisco.com

Toerless Eckert
Cisco

Email: eckert@cisco.com

UCAN BOF
Internet-Draft
Intended status: Informational
Expires: December 25, 2014

D. Bogdanovic
Juniper Networks
June 23, 2014

Autonomic Networking in mobile wireless backhaul
draft-bogdanovic-nmrg-mobile-backhaul-use-case-00

Abstract

Mobile backhaul networks that utilize microwave technology in transport are suspicious to seasonal and/or meteorological changes. For those reasons throughput can vary significantly. This draft provides problem statement and how autonomic networking can be applied to the problem.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 25, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Definitions and Acronyms	2
2. Problem Statement	3
3. Intended User and Administrator Experience	3
4. Analysis of Parameters and Information Involved	4
4.1. Parameters each device can decide for itself	4
4.2. Information needed from policy intent	5
5. Interaction with other devices	6
5.1. Information needed from other devices	6
5.2. Monitoring, diagnostics and reporting	7
6. Comparison with current solutions	8
7. Security Considerations	8
8. IANA Considerations	8
9. Acknowledgements	8
10. Change log [RFC Editor: Please remove]	8
11. References	8
Author's Address	9

1. Introduction

Microwave technology is one of the workhorses in MBH networks today. Unlicensed microwave links can be set up in days rather than the weeks or months it might take to implement additional wireline capacity for backhaul. Even licensed links, while requiring some mildly time-consuming bureaucratic approval, still easily outpace the time-to-deployment of wireline alternatives. Fiber may offer unlimited bandwidth, but the tradeoff is in time and cost savings. Microwave's improvements in bandwidth, capacity, and reliability in the past few years have made it an ideal interim broadband connectivity solution during the often lengthy process of deploying fiber. In fact, these improvements go as far as to establish microwave as a legitimate permanent alternative to fiber. Although its many benefits, because of other restrictions that microwave links have, they can't be utilized at maximum. OSPF/MPLS networks that are overlayed on top of microwave transport and provide additional benefits of packet routing and switching to mobile backhaul.

1.1. Definitions and Acronyms

MBH: Mobile Backhaul

BTS: Base Transceiver Station

OSPF: Open Shortest Path First

MPLS: Multi Protocol Label Switching

PLMN: Public Land Mobile Networks

CoS: Class of Service

MTTR: Mean Time To Recovery

MNO: Mobile Network Operator

IDU: In Door Unit

ODU: Out Door Unit

SNMP: Simple Network Management Protocol

IP: Internet Protocol

IPv4: Internet Protocol version 4

IPv6: Internet Protocol version 6

2. Problem Statement

Mobile backhaul (MBH) networks utilize microwave networks to transport traffic back to Public Land Mobile Networks (PLMN). Base transceiver stations (BTS) and/or eNodeBs connect to a device that has multiple microwave connections to PLMN. Not each link has the same throughput and the quality of the link varies with different factors, like air temperature, precipitation, vegetation, etc. Today those networks are overlayed with OSPF/MPLS networks and although OSPF provides automatic redirection of traffic in case of primary link failure, it reduces network throughput, as microwave link bandwidth slowly degrades, due to rain, snow, tower bending due to wind and/or temperature, vegetation growing. During the link degradation period, the throughput of MBH part is going down and the overall service is impacted. Being able to detect the degradation of the microwave link bandwidth and redirect traffic over higher throughput links is very beneficial to mobile network operators.

3. Intended User and Administrator Experience

As MBH links are lowering the bandwidth, the user experience is impacted, as the data hungry applications are not served with usual quality of service and latency is increasing, due to dropped packets. MBH network administrators are not getting real time picture (usually today they see average link performance over 15 minutes period) and with users being highly mobile, they can't react to the challenges in the network. Administrators should be able to set intended policy on device, based on which device would start changing network

forwarding parameters based on which the current traffic would be routed via links with most throughput. With monitoring the link statistics, device can change forwarding and routing parameters in realtime based on the intended policy pushed on the device, without the need to interact with centralized management system, which would act based on sent link performance indicators. Such a network would improve end user experience, as well network administrators would be able to create better performing networks.

4. Analysis of Parameters and Information Involved

Numerous parameters are involved in monitoring MBH, from microwave link performance, to miscellaneous OSPF and MPLS parameters. MNO has to look at KPI that will relate to those that may impact revenue negatively, such as unavailability and MTTR. One thing to note here is that much emphasis is usually placed on availability, while most times not enough emphasis is placed on reliability. In defining Key Performance Indicators effectively, KPIs must align with BTS availability information for a mobile operator.

Microwave transmission

- o availability
- o capacity
- o delay
- o jitter

4.1. Parameters each device can decide for itself

All OSPF interfaces have a cost, which is a routing metric that is used in the link-state calculation. Routes with lower total path metrics are preferred to those with higher path metrics. OSPF assigns a default cost metric of 1 to reference bandwidth and default cost metric of 0 to the loopback interface (lo0). No bandwidth is associated with the loopback interface. So if reference bandwidth is set to 1Gbps, it means that all interfaces faster than 1Gbps have the same default cost metric of 1. If multiple equal-cost paths exist between a source and destination address, OSPF routes packets along each path alternately, in round-robin fashion.

Having the same default metric might not be a problem if all of the interfaces are running at the same speed. In MBH, microwave units will connect via ethernet to ethernet ports on routers and each link will have the same metric. That would not be a problem if all microwave links would have same performance, but links operate at

different speeds, and it is very probable that traffic is not routed over the fastest interface because OSPF equally routes packets across the different interfaces.

The autonomic agent agents collects operational statistics from ethernet ports to which microwave IDU is connected, as well from microwave ODU (local and remote) using SNMP. By collecting this statistics, optimal MBH OSPF agent can callculate links with best performance and change the metric value for each link accordingly.

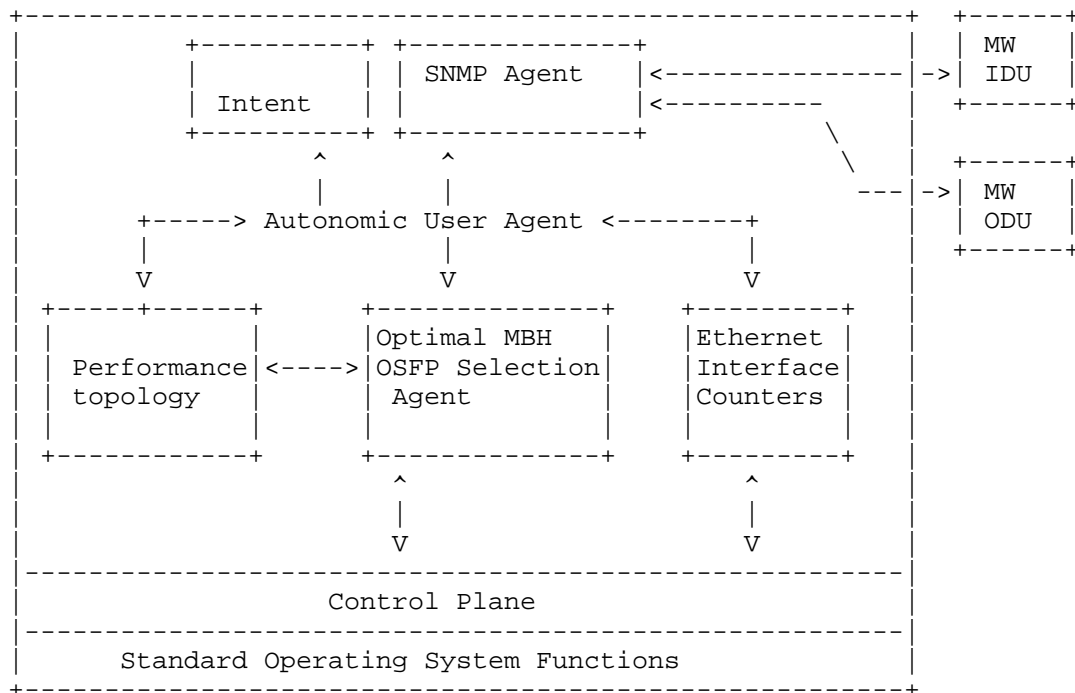


Figure 1

4.2. Information needed from policy intent

The section describes what information is needed to be provided by external entity, so that devices can operate autonomically. The policy would have to set:

- o reference bandwidth - example 1Gbps
- o low water mark threshold, at which point to change the metric

- o IP address or addresses of local IDU
- o IP address or addresses of local ODU
- o IP address or addresses of remote IDU
- o IP address or addresses of remote ODU
- o which ethernet port is connect to which microwave link

This list is not extensive and with more research it can be augmented with new parameters. The experience will show what parameters are important. There might be a need to put time restrictions between metric updates on the device or how often are statistics collected, as it is important not to negatively effect device forwarding capabilities.

5. Interaction with other devices

The device would interact with microwave IDU and ODU. It would interact with them via SNMP or some other protocol that allows to collect operating statistics of the microwave link. By collecting those statistics, it can compute the link performance. It is also possible to communicate with other autonomic device in MBH and to exchange information, so that devices can learn the whole topology in segment and performance of the microwave links in possible path.

5.1. Information needed from other devices

In Figure 2, a small example is shown how MBH router 1 is connected via microwave links to router MBH 2 and MBH 3. Microwave IDUs are connected via ethernet to MBH routers and each IDU has two ODUs connected. Microwave links usually have two beams in a link. Microwave IDUs send each incoming packet from MBH router 1 to each ODU connected to them, essentially copying packets over each beam in the microwave links. The terminating IDU C and D, on the other side, compare incoming packets from each ODU and drop the duplicate packets prior to forwarding the packet to their connected MBH routers. This mechanism allows collecting good operating statistics of the links, so autonomic agent on MBH routers can calculate end to end performance of each link, like latency, throughput, jitter, delay etc. This allows building performance topology on the MBH router by autonomic agent

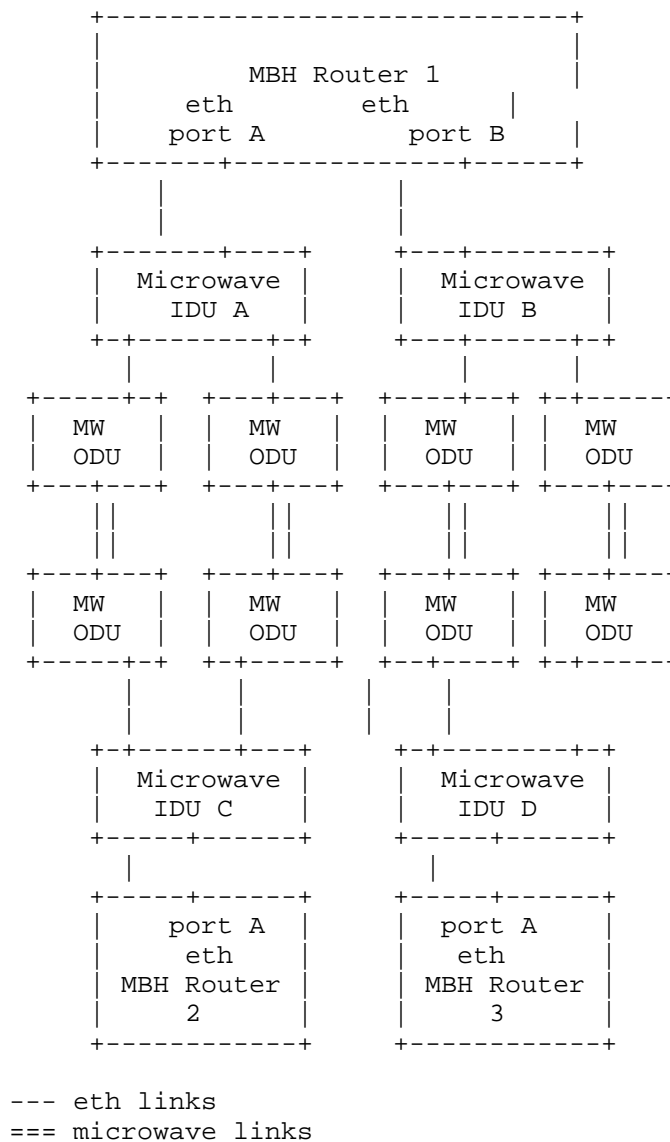


Figure 2

5.2. Monitoring, diagnostics and reporting

The autonomic user agent should provide feedback data to centralized management system, so that new improved intent policies can be created. Most of the data doesn't have to be provided in real time, except for cases when microwave link failure happens and causes loss

of data. This means that the autonomic agent didn't provide alternative path in time or all microwave links from the MBH router are down. Historical data, like what were the network conditions under which the autonomic agent enforcing the intent policies are very valuable, as well the performance topology from each device, as it allows to create whole performance view of the MBH.

6. Comparison with current solutions

There are some vendors (NSN, Ericsson) that are trying to create self organizing networks, but the intelligence is always centralized, which prevents distribution of the network intelligence and using it for autonomic use cases.

7. Security Considerations

As this stage, author of the draft didn't look into security considerations of the use case.

8. IANA Considerations

This document requests no action by IANA.

9. Acknowledgements

10. Change log [RFC Editor: Please remove]

11. References

[I-D.irtf-nmrg-an-gap-analysis]

Behringer, M., Carpenter, B., and S. Jiang, "Gap Analysis for Autonomic Networking", draft-irtf-nmrg-an-gap-analysis-00 (work in progress), April 2014.

[I-D.irtf-nmrg-autonomic-network-definitions]

Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking - Definitions and Design Goals", draft-irtf-nmrg-autonomic-network-definitions-00 (work in progress), December 2013.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

Author's Address

Dean Bogdanovic
Juniper Networks

Email: deanb@juniper.net

UCAN BOF
Internet-Draft
Intended status: Informational
Expires: November 20, 2014

B. Carpenter
Univ. of Auckland
May 19, 2014

Autonomic Networking Use Case for Home Networks
draft-carpenter-nmrg-homenet-an-use-case-01

Abstract

This document describes a use case for autonomic networking in home network scenarios. It is one of a series of use cases intended to illustrate requirements for autonomic networking.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 20, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Problem Statement	2
3. Intended User and Administrator Experience	3
4. Analysis of Parameters and Information Involved	3
4.1. Parameters each device can decide for itself	4
4.2. Information needed from policy intent	4
5. Interaction with other devices	5
5.1. Information needed from neighbor devices	5
5.2. Monitoring, diagnostics and reporting	6
6. Comparison with current solutions	6
7. Security Considerations	7
8. IANA Considerations	7
9. Acknowledgements	7
10. Change log [RFC Editor: Please remove]	7
11. References	7
Author's Address	9

1. Introduction

This document is one of a set of use cases being developed to clarify the requirements for discovery and negotiation protocols for autonomic networking (AN). The background to AN is described in [I-D.irtf-nmrg-autonomic-network-definitions] and [I-D.irtf-nmrg-an-gap-analysis]. A problem statement and outline requirements for the negotiation protocol are given in [I-D.jiang-config-negotiation-ps].

Note in draft: The format of this document may be modified as we agree on a common format for AN use cases. In particular, opinions may vary about how concrete vs how abstract a use case should be.

2. Problem Statement

The use case considered here is autonomic operation of home networks based on IPv6, in general accordance with [I-D.ietf-homenet-arch]. The model assumes that a typical homenet in the future will have multiple network segments, several routers, and a reasonably large number of hosts, but no expert human manager. For some aspects of homenet configuration, a protocol solution known as HNCP (homenet control protocol) has been designed and implemented [I-D.ietf-homenet-hncp]. A solution has also been described for bootstrapping trust in a homenet [I-D.behringer-homenet-trust-bootstrap].

Additional issues that impact homenet configuration are discussed in [I-D.winters-homenet-sper-interaction],

[I-D.pfister-homenet-prefix-assignment],
[I-D.mglt-homenet-naming-architecture-dhc-options] and
[I-D.stenberg-homenet-dnssd-hybrid-proxy-zeroconf].

The problem to be solved by AN is how to replace these and other partial solutions by a generic solution that sets all necessary parameters for the homenet to operate efficiently, reliably and securely, with minimal human intervention and without the need for traditional top-down configuration.

It should be noted that HNCP has a quite generic design, but so far has been described for a fairly narrow scope of application, and other aspects of homenet bootstrapping, discovery and configuration are currently handled by other methods. The present draft takes no position on these various solutions, since its goal is only to describe the use case.

This use case does not currently consider the challenges posed by multiple provisioning domains as defined by [I-D.ietf-mif-mpvd-arch].

3. Intended User and Administrator Experience

In a homenet, the basic assumption is that no human involved has technical knowledge beyond the ability to unwrap a product, connect a few cables, and follow simple instructions. Indeed, the parody "Have you tried turning it off and on again?" may apply literally. Therefore, the desired user experience is that everything just works, that there are no mandatory user actions, and that no specialist knowledge is needed. If any user choices are offered, there must be a reasonable default. When power failures or equipment failures occur, recovery to the best possible running state must be automatic. If any diagnostic messages are produced, they must be simple and clear, and of course provided in the user's own language. If any logs are recorded, it is to be expected that the normal user will never look at them and could not understand them.

4. Analysis of Parameters and Information Involved

Numerous parameters are involved in a homenet (some of them can of course be pre-configured with default values). They include:

- o Identity of a trust anchor to act as a local certification authority (CA) and registration authority (RA) for nodes inside the homenet.
- o Identity of border devices (equivalent to perimeter identification).

- o Firewall rules (for border devices and for host firewalls).
- o IP address prefix(es) for the whole homenet.
- o Individual prefix(es) for each subnet.
- o Choice of routing protocol.
- o Initial configuration of all routers.
- o Default router for each subnet.
- o Rules for address selection.
- o Local namespace information (delegated zone if any, etc.).
- o DNS server(s).

4.1. Parameters each device can decide for itself

This section identifies those of the above parameters that do not need external information in order for the devices concerned to set them to a reasonable value after bootstrap or after a network disruption. There are few of these:

- o Default firewall rules for hosts. Hosts should be shipped from the manufacturer with generally applicable default firewall rules.
- o Default rules for address selection should conform to [RFC6724].

4.2. Information needed from policy intent

This section identifies those parameters that need external information about policy intent in order for the devices concerned to set them to a non-default value. It's assumed that in a homenet, policy intent will likely be provided by the main homenet router, and may itself be a default setting in that router, since there is normally no human expert to set policy. Not all devices will need to know all of these intents.

- o Method of determining the trust anchor.
- o Whether firewall rules will be changed from their default settings.
- o Whether more than one GUA prefix will be deployed.
- o Whether a ULA prefix will be deployed.

- o Which routing protocol is preferred.
- o Whether DHCPv6 will be deployed.
- o Whether non-default rules for address selection will be deployed.
- o Whether IPv4 and DHCP will be deployed (IPv6 is assumed).

5. Interaction with other devices

5.1. Information needed from neighbor devices

This section identifies those of the above parameters that need external information from neighbor devices (such as other routers) in order for the devices concerned to set them. In many cases, two-way dialogue with neighbor devices is needed to set or optimise them.

- o Identity of a trust anchor.
- o Routers will need to discover their neighbors.
- o Routers will need to determine whether they are border devices.
- o Border routers will need to apply a default border firewall policy; interior routers will not be firewalls by default.
- o Hosts may need to acquire a non-default firewall policy.
- o Border routers will need to determine the IP address prefix(es) for the whole homenet.
- o One border router will need to generate the ULA prefix for the whole homenet.
- o Routers will need to discover the network topology and then to apply a prefix delegation method to deliver at least one prefix to each subnet.
- o With knowledge of its neighbors and after prefix delegation, each router will need to configure and launch the agreed routing protocol.
- o Hosts need to acquire a default router for each interface.
- o Hosts may need to acquire non-default rules for address selection.
- o The local namespace service must configure itself. Relevant devices will need to know at least the zone name delegated to the

homenet. This is a complex topic, so the reader is referred to drafts that already describe the needed functions:

[I-D.mglt-homenet-naming-architecture-dhc-options] and
[I-D.stenberg-homenet-dnssd-hybrid-proxy-zeroconf].

- o Hosts need to acquire DNS server address(es).

5.2. Monitoring, diagnostics and reporting

This section discusses what role devices should play in monitoring, fault diagnosis, and reporting.

- o In general, failure to successfully set reasonable values for any network parameter should be logged and notified to the user, in simple, non-technical words in the user's own language.
- o Similarly, hard failures should be logged and notified, even if the network has somehow routed around them.
- o Users are very unlikely to take an interest in warnings of any kind, so they are probably a waste of time.
- o Firewall incidents are typically logged in a proprietary fashion. It would be conceivable for all firewalls in a homenet to log incidents centrally but it seems unlikely that such a feature would ever be used by a typical home user.

6. Comparison with current solutions

This section briefly compares the above use case with current solutions. Today's typical single-router homenets do largely run without significant human intervention, relying on fixed DHCP setups for IPv4 and on out-of-the-box Router Advertisements for IPv6. This comparison is not very illuminating, since we are interested in complex homenets with multiple routers. A better comparison is with the emerging prototype homenet environment based on the various drafts cited in Section 2. The functionality described is very similar. The actual content of the messages would also be very similar to those in HNCP etc. However, using a generic autonomic discovery and negotiation protocol instead of a mixture of dedicated solutions has the advantage that additional parameters can be included in the autonomic solution without creating new mechanisms. This is the principal argument for a generic approach.

7. Security Considerations

Relevant security issues are discussed in [I-D.irtf-nmrg-autonomic-network-definitions], [I-D.jiang-config-negotiation-ps] and [I-D.ietf-homenet-arch]. The security specificity of a homenet is the need to establish a trust anchor in the absence of a human expert, which will allow remaining security features to configure themselves autonomically.

8. IANA Considerations

This document requests no action by IANA.

9. Acknowledgements

Valuable comments were received from Steven Barth, Michael Behringer, Sheng Jiang, Mark Townsley, and others.

This document was produced using the xml2rfc tool [RFC2629].

10. Change log [RFC Editor: Please remove]

draft-carpenter-nmrg-homenet-an-use-case-01: clarifications, more accurate characterisation of HNCP, 2014-05-19.

draft-carpenter-nmrg-homenet-an-use-case-00: original version, 2014-04-10.

11. References

- [I-D.behringer-homenet-trust-bootstrap]
Behringer, M., Pritikin, M., and S. Bjarnason,
"Bootstrapping Trust on a Homenet", draft-behringer-
homenet-trust-bootstrap-02 (work in progress), February
2014.
- [I-D.ietf-homenet-arch]
Chown, T., Arkko, J., Brandt, A., Troan, O., and J. Weil,
"IPv6 Home Networking Architecture Principles", draft-
ietf-homenet-arch-13 (work in progress), March 2014.
- [I-D.ietf-homenet-hncp]
Stenberg, M. and S. Barth, "Home Networking Control
Protocol", draft-ietf-homenet-hncp-00 (work in progress),
April 2014.

- [I-D.ietf-mif-mpvd-arch]
Anipko, D., "Multiple Provisioning Domain Architecture",
draft-ietf-mif-mpvd-arch-01 (work in progress), May 2014.
- [I-D.irtf-nmrg-an-gap-analysis]
Behringer, M., Carpenter, B., and S. Jiang, "Gap Analysis
for Autonomic Networking", draft-irtf-nmrg-an-gap-
analysis-00 (work in progress), April 2014.
- [I-D.irtf-nmrg-autonomic-network-definitions]
Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A.,
Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic
Networking - Definitions and Design Goals", draft-irtf-
nmrg-autonomic-network-definitions-00 (work in progress),
December 2013.
- [I-D.jiang-config-negotiation-ps]
Jiang, S., Yin, Y., and B. Carpenter, "Network
Configuration Negotiation Problem Statement and
Requirements", draft-jiang-config-negotiation-ps-02 (work
in progress), January 2014.
- [I-D.mglt-homenet-naming-architecture-dhc-options]
Migault, D., Cloetens, W., Griffiths, C., and R. Weber,
"DHCP Options for Homenet Naming Architecture", draft-
mglt-homenet-naming-architecture-dhc-options-01 (work in
progress), February 2014.
- [I-D.pfister-homenet-prefix-assignment]
Pfister, P., Paterson, B., and J. Arkko, "Prefix and
Address Assignment in a Home Network", draft-pfister-
homenet-prefix-assignment-01 (work in progress), May 2014.
- [I-D.stenberg-homenet-dnssd-hybrid-proxy-zeroconf]
Stenberg, M., "Auto-Configuration of a Network of Hybrid
Unicast/Multicast DNS-Based Service Discovery Proxy
Nodes", draft-stenberg-homenet-dnssd-hybrid-proxy-
zeroconf-00 (work in progress), February 2014.
- [I-D.winters-homenet-sper-interaction]
Winters, T., "Service Provider Edge Router Interaction",
draft-winters-homenet-sper-interaction-01 (work in
progress), February 2014.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
June 1999.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown,
"Default Address Selection for Internet Protocol Version 6
(IPv6)", RFC 6724, September 2012.

Author's Address

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Network Management Research Group
Internet-Draft
Intended status: Informational
Expires: November 2, 2015

S. Jiang
Huawei Technologies Co., Ltd
B. Carpenter
Univ. of Auckland
M. Behringer
Cisco Systems
May 1, 2015

General Gap Analysis for Autonomic Networking
draft-irtf-nmrg-an-gap-analysis-06

Abstract

This document is a product of the IRTF's Network Management Research Group. It provides a problem statement and general gap analysis for an IP-based Autonomic Network that is mainly based on distributed network devices. The document provides a background by reviewing the current status of autonomic aspects of IP networks and the extent to which current network management depends on centralisation and human administrators. Finally the document outlines the general features missing from current network abilities that are needed in the ideal Autonomic Network concept.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 2, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Automatic and Autonomic Aspects of Current IP Networks	3
3.1. IP Address Management and DNS	3
3.2. Routing	4
3.3. Configuration of Default Router in a Host	5
3.4. Hostname Lookup	5
3.5. User Authentication and Accounting	6
3.6. Security	6
3.7. State Synchronization	7
4. Current Non-Autonomic Behaviors	7
4.1. Building a New Network	7
4.2. Network Maintenance and Management	8
4.3. Security Setup	9
4.4. Troubleshooting and Recovery	9
5. Features Needed by Autonomic Networks	10
5.1. More Coordination among Devices or Network Partitions	11
5.2. Reusable Common Components	11
5.3. Secure Control Plane	12
5.4. Less Configuration	12
5.5. Forecasting and Dry Runs	13
5.6. Benefit from Knowledge	13
6. Security Considerations	14
7. IANA Considerations	14
8. Acknowledgements	14
9. Change log [RFC Editor: Please remove]	14
10. Informative References	15
Authors' Addresses	17

1. Introduction

The general goals and relevant definitions for Autonomic Networking are discussed in [I-D.irtf-nmrg-autonomic-network-definitions]. In summary, the fundamental goal of an Autonomic Network is self-management, including self-configuration, self-optimization, self-healing and self-protection. Whereas interior gateway routing protocols such as OSPF and IS-IS largely exhibit these properties, most other aspects of networking require top-down configuration,

often involving human administrators and a considerable degree of centralisation. In essence Autonomic Networking is putting all network configurations onto the same footing as routing, limiting manual or database-driven configuration to an essential minimum. It should be noted that this is highly unlikely to eliminate the need for human administrators, because many of their essential tasks will remain. The idea is to eliminate tedious and error-prone tasks, for example manual calculations, cross-checking between two different configuration files, or tedious data entry. Higher level operational tasks, and complex trouble-shooting, will remain to be done by humans.

This document represents the consensus of the IRTF's network management research group (NMRG). It first provides background by identifying examples of partial autonomic behavior in the Internet, and by describing important areas of non-autonomic behavior. Based on these observations, it then describes missing general mechanisms which would allow autonomic behaviours to be added throughout the Internet.

2. Terminology

The terminology defined in [I-D.irtf-nmrg-autonomic-network-definitions] is used in this document.

3. Automatic and Autonomic Aspects of Current IP Networks

This section discusses the history and current status of automatic or autonomic operations in various aspects of network configuration, in order to establish a baseline for the gap analysis. In particular, routing protocols already contain elements of autonomic processes, such as information exchange and state synchronization.

3.1. IP Address Management and DNS

For many years, there was no alternative to completely manual and static management of IP addresses and their prefixes. Once a site had received an IPv4 address assignment (usually a Class C /24 or Class B /16, and rarely a Class A /8) it was a matter of paper-and-pencil design of the subnet plan (if relevant) and the addressing plan itself. Subnet prefixes were manually configured into routers, and /32 addresses were assigned administratively to individual host computers, and configured manually by system administrators. Records were typically kept in a plain text file or a simple spreadsheet.

Clearly this method was clumsy and error-prone as soon as a site had more than a few tens of hosts, but it had to be used until DHCP

[RFC2131] became a viable solution during the second half of the 1990s. DHCP made it possible to avoid manual configuration of individual hosts (except, in many deployments, for a small number of servers configured with static addresses). Even so, prefixes had to be manually assigned to subnets and their routers, and DHCP servers had to be configured accordingly.

In terms of management, there is a linkage between IP address management and DNS management, because DNS mappings typically need to be appropriately synchronized with IP address assignments. At roughly the same time as DHCP came into widespread use, it became very laborious to manually maintain DNS source files in step with IP address assignments. Because of reverse DNS lookup, it also became necessary to synthesise DNS names even for hosts that only played the role of clients. Therefore, it became necessary to synchronise DHCP server tables with forward and reverse DNS. For this reason, Internet Protocol address management tools emerged, as discussed for the case of renumbering in [RFC7010]. These are, however, centralised solutions that do not exhibit autonomic properties as defined in [I-D.irtf-nmrg-autonomic-network-definitions].

A related issue is prefix delegation, especially in IPv6 when more than one prefix may be delegated to the same physical subnet. DHCPv6 Prefix Delegation [RFC3633] is a useful solution, but it requires specific configuration so cannot be considered autonomic. How this topic is to be handled in home networks is still in discussion [I-D.ietf-homenet-prefix-assignment]. Still further away is autonomic assignment and delegation of routeable IPv4 subnet prefixes.

An IPv6 network needs several aspects of host address assignments to be configured. The network might use stateless address autoconfiguration [RFC4862] or DHCPv6 [RFC3315] in stateless or stateful modes, and there are various alternative forms of Interface Identifier [RFC7136].

Another feature is the possibility of Dynamic DNS Update [RFC2136]. With appropriate security, this is an automatic approach, where no human intervention is required to create the DNS records for a host after it acquires a new address. However, there are coexistence issues with a traditional DNS setup, as described in [RFC7010].

3.2. Routing

Since a very early stage, it has been a goal that Internet routing should be self-healing when there is a failure of some kind in the routing system (i.e. a link or a router goes wrong). Also, the problem of finding optimal routes through a network was identified

many years ago as a problem in mathematical graph theory, for which well known algorithms were discovered (the Dijkstra and Bellman-Ford algorithms). Thus routing protocols became largely autonomic from the start, as it was clear that manual configuration of routing tables for a large network was impractical.

IGP routers do need some initial configuration data to start up the autonomic routing protocol. Also, BGP-4 routers need detailed static configuration of routing policy data.

3.3. Configuration of Default Router in a Host

Originally this was a manual operation. Since the deployment of DHCP, this has been automatic as far as most IPv4 hosts are concerned, but the DHCP server must be appropriately configured. In simple environments such as a home network, the DHCP server resides in the same box as the default router, so this configuration is also automatic. In more complex environments, where an independent DHCP server or a local DHCP relay is used, DHCP configuration is more complex and not automatic.

In IPv6 networks, the default router is provided by Router Advertisement messages [RFC4861] from the router itself, and all IPv6 hosts make use of it. The router may also provide more complex Route Information Options. The process is essentially autonomic as far as all IPv6 hosts are concerned, and DHCPv6 is not involved. However, there are still open issues when more than one prefix is in use on a subnet and more than one first-hop router may be available as a result (see for example [RFC6418]).

3.4. Hostname Lookup

Originally host names were looked up in a static table, often referred to as "hosts.txt" from its traditional file name. When the DNS was deployed during the 1980s, all hosts needed DNS resolver code, and needed to be configured with the IP addresses (not the names) of suitable DNS servers. Like the default router, these were originally manually configured. Today, they are provided automatically via DHCP or DHCPv6 [RFC3315]. For IPv6 end systems, there is also a way for them to be provided automatically via a Router Advertisement option. However, the DHCP or DHCPv6 server, or the IPv6 router, need to be configured with the appropriate DNS server addresses. Additionally, some networks deploy Multicast DNS [RFC6762] locally to provide additional automation of the name space.

3.5. User Authentication and Accounting

Originally, user authentication and accounting was mainly based on physical connectivity and the degree of trust that follows from direct connectivity. Network operators charged based on the set up of dedicated physical links with users. Automated user authentication was introduced by Point-to-Point Protocol [RFC1661], [RFC1994] and RADIUS protocol [RFC2865], [RFC2866] in the early 1990s. As long as a user completes online authentication through the RADIUS protocol, the accounting for that user starts on the corresponding AAA server automatically. This mechanism enables business models with charging based on traffic based or time based usage. However, the management of user authentication information remains manual by network administrators. It also becomes complex in the case of mobile users who roam between operators, since prior relationships between the operators are needed.

3.6. Security

Security has many aspects that need configuration and are therefore candidates to become autonomic. On the other hand, it is essential that a network's central policy should be applied strictly for all security configurations. As a result security has largely been based on centrally imposed configurations.

Many aspects of security depend on policy, for example password rules, privacy rules, firewall rulesets, intrusion detection and prevention settings, VPN configurations, and the choice of cryptographic algorithms. Policies are by definition human made and will therefore also persist in an autonomic environment. However, policies are becoming more high-level, abstracting addressing for example, and focusing on the user or application. The methods to manage, distribute and apply policy, and to monitor compliance and violations could be autonomic.

Today, many security mechanisms show some autonomic properties. For example user authentication via 802.1x allows automatic mapping of users after authentication into logical contexts (typically VLANs). While today configuration is still very important, the overall mechanism displays signs of self-adaption to changing situations.

BGP Flowspec [RFC5575] allows a partially autonomic threat defense mechanism, where threats are identified, the flow information is automatically distributed, and counter-actions can be applied. Today typically a human operator is still in the loop to check correctness, but over time such mechanisms can become more autonomic.

Negotiation capabilities, present in many security protocols, also display simple autonomic behaviours. In this case a security policy about algorithm strength can be configured into servers but will propagate automatically to clients.

3.7. State Synchronization

Another area where autonomic processes between peers are involved is state synchronization. In this case, several devices start out with inconsistent state and go through a peer-to-peer procedure after which their states are consistent. Many autonomic or automatic processes include some degree of implicit state synchronization. Network time synchronization [RFC5905] is a well-established explicit example, guaranteeing that a participating node's clock state is synchronized with reliable time servers within a defined margin of error, without any overall point of control of the synchronization process.

4. Current Non-Autonomic Behaviors

In current networks, many operations are still heavily dependent on human intelligence and decision, or on centralised top-down network management systems. These operations are the targets of Autonomic Networking technologies. The ultimate goal of Autonomic Networking is to replace human and automated operations by autonomic functions, so that the networks can run independently without depending on a human or NMS system for routine details, while maintaining central control where required. Of course, there would still be the absolute minimum of human input required, particularly during the network building stage, and during emergencies and difficult trouble-shooting.

This section analyzes the existing human and central dependencies in typical current networks and suggests cases where they could in principle be replaced by autonomic behaviors.

4.1. Building a New Network

Building a network requires the operator to analyze the requirements of the new network, design a deployment architecture and topology, decide device locations and capacities, set up hardware, design network services, choose and enable required protocols, configure each device and each protocol, set up central user authentication and accounting policies and databases, design and deploy security mechanisms, etc.

Overall, these jobs are quite complex work that cannot become fully autonomic in the foreseeable future. However, part of these jobs may

be able to become autonomic, such as detailed device and protocol configurations and database population. The initial network management policies/behaviors may also be transplanted from other networks and automatically localized.

4.2. Network Maintenance and Management

Network maintenance and management are very different for ISP networks and enterprise networks. ISP networks have to change much more frequently than enterprise networks, given the fact that ISP networks have to serve a large number of customers who have very diversified requirements. The current rigid model is that network administrators design a limited number of services for customers to order. New requirements of network services may not be able to be met quickly by human management. Given a real-time request, the response must be autonomic, in order to be flexible and quickly deployed. However, behind the interface, describing abstracted network information and user authorization management may have to depend on human intelligence from network administrators in the foreseeable future. User identification integration/consolidation among networks or network services is another challenge for Autonomic Network access. Currently, many end users have to manually manage their user accounts and authentication information when they switch among networks or network services.

Classical network maintenance and management mainly manages the configuration of network devices. Tools have been developed to enable remote management and make such management easier. However, the decision about each configuration detail depends either on human intelligence or rigid templates. One or the other of these is therefore the source of all network configuration errors - either the human was wrong, or the template was wrong (or both). This is also a barrier to increasing the utility of network resources, because the human managers cannot respond quickly enough to network events, such as traffic bursts, that were not foreseen in the template. For example, currently, a light load is often assumed in network design because there is no mechanism to properly handle a sudden traffic flood. It is therefore common to avoid performance collapses caused by traffic overload by configuring idle resources, with an overprovisioning ratio of at least 2 being normal [Xiao02].

There are grounds for concern that the introduction of new, more flexible, methods of network configuration, typified by software-defined networking (SDN), will only make the management problem more complex unless the details are managed automatically or autonomically. There is no doubt that SDN creates both the necessity and the opportunity for automation of configuration management, e.g.,

[Kim13]. This topic is discussed from a service provider viewpoint in [RFC7149].

Autonomic decision processes for configuration would enable dynamic management of network resources (by managing resource-relevant configuration). Self-adapting network configuration would adjust the network into the best possible situation, which also prevents configuration errors from having lasting impact.

4.3. Security Setup

Setting up security for a network generally requires very detailed human intervention, or relies entirely on default configurations that may be too strict or too risky for the particular situation of the network. While some aspects of security are intrinsically top-down in nature (e.g. broadcasting a specific security policy to all hosts), others could be self-managed within the network.

In an Autonomic Network, where nodes within a domain have a mutually verifiable domain identity, security processes could run entirely automatically. Nodes could identify each other securely, negotiating required security settings and even shared keys if needed. The location of trust anchors (certificate authority, registration authority), certificate revocation lists, policy server, etc., can be found by service discovery. Transactions such as a certificate revocation list download can be authenticated via a common trust anchor. Policy distribution can also be entirely automated, and secured via a common trust anchor.

These concepts lead to a network where the intrinsic security is automatic and applied by default, i.e., a "self-protecting" network. For further discussion, see [I-D.behringer-default-secure]

4.4. Troubleshooting and Recovery

Current networks suffer difficulties in locating the cause of network failures. Although network devices may issue many warnings while running, most of them are not sufficiently precise to be identified as errors. Some of them are early warnings that would not develop into real errors. Others are in effect random noise. During a major failure, many different devices will issue multiple warnings within a short time, causing overload for the NMS and the operators. However, for many scenarios, human experience is still vital to identify real issues and locate them. This situation may be improved by automatically associating warnings from multiple network devices together. Also, introducing automated learning techniques (comparing current warnings with historical relationships between warnings and

actual faults) could increase the possibility and success rate of Autonomic Network diagnoses and troubleshooting.

Depending on the network errors, some of them may always require human interventions, particularly for hardware failures. However, autonomic network management behavior may help to reduce the impact of errors, for example by switching traffic flows around. Today this is usually manual (except for classical routing updates). Fixing software failures and configuration errors currently depends on humans, and may even involve rolling back software versions and rebooting hardware. Such problems could be autonomically corrected if there were diagnostics and recovery functions defined in advance for them. This would fulfill the concept of self-healing.

Another possible autonomic function is predicting device failures or overloads before they occur. A device could predict its own failure and warn its neighbors; or a device could predict its neighbor's failure. In either case, an Autonomic Network could respond as if the failure had already occurred by routing around the problem and reporting the failure, with no disturbance to users. The criteria for predicting failure could be temperature, battery status, bit error rates, etc. The criteria for predicting overload could be increasing load factor, latency, jitter, congestion loss, etc.

5. Features Needed by Autonomic Networks

There are innumerable properties of network devices and end systems that today need to be configured either manually, by scripting, or by using a management protocol such as NETCONF [RFC6241]. In an Autonomic Network, all of these would need to either have satisfactory default values or be configured automatically. Some examples are parameters for tunnels of various kinds, flows (in an SDN context), quality of service, service function chaining, energy management, system identification and NTP configuration, but the list is endless.

The task of Autonomic Networking is to incrementally build up individual autonomic processes that could progressively be combined to respond to every type of network event. Building on the preceding background information, and on the reference model in [I-D.irtf-nmrg-autonomic-network-definitions], this section outlines the gaps and missing features in general terms, and in some cases mentions general design principles that should apply.

5.1. More Coordination among Devices or Network Partitions

Network services are dependent on a number of devices and parameters to be in place in a certain order. For example after a power failure a coordinated sequence of "return to normal" operations is desirable (e.g., switches and routers first, DNS servers second, etc.). Today, the correct sequence of events is either known only by a human administrator, or automated in a central script. In a truly Autonomic Network, elements should understand their dependencies, and be able to resolve them locally.

In order to make right or good decisions autonomically, the network devices need to know more information than just reachability (routing) information from the relevant or neighbor devices. There are dependencies between such information and configurations, which devices must be able to derive for themselves.

There are therefore increased requirements for horizontal information exchange in the networks. Particularly, three types of interaction among peer network devices are needed for autonomic decisions: discovery (to find neighbours and peers), synchronization (to agree on network status) and negotiation (when things need to be changed). Thus there is a need for reusable discovery, synchronization and negotiation mechanisms, which would support the discovery of many different types of device, the synchronization of many types of parameter and the negotiation of many different types of objective.

5.2. Reusable Common Components

Elements of autonomic functions already exist today, within many different protocols. However, all such functions have their own discovery, transport, messaging and security mechanisms as well as non-autonomic management interfaces. Each protocol has its own version of the above-mentioned functions to serve specific and narrow purposes. It is often difficult to extend an existing protocol to serve different purposes. Therefore, in order to provide the reusable discovery, synchronization and negotiation mechanism mentioned above, it is desirable to develop a set of reusable common protocol components for Autonomic Networking. These components should be:

- o Able to identify other devices, users and processes securely.
- o Able to automatically secure operations, based on the above identity scheme.
- o Able to manage any type of information and information flows.

- o Able to discover peer devices and services for various autonomic service agents (or autonomic functions).
- o Able to support closed-loop operations when needed to provide self-managing functions involving more than one device.
- o Separable from the specific autonomic service agents (or autonomic functions).
- o Reusable by other autonomic functions.

5.3. Secure Control Plane

The common components will in effect act as a control plane for autonomic operations. This control plane might be implemented in-band as functions of the target network, in an overlay network, or even out-of-band in a separate network. Autonomic operations will be capable of changing how the network operates and allocating resources without human intervention or knowledge, so it is essential that they are secure. Therefore the control plane must be designed to be secure against forged autonomic operations and man-in-the middle attacks, and as secure as reasonably possible against denial of service attacks. It must be decided whether the control plane needs to be resistant to unwanted monitoring, i.e., whether encryption is required.

5.4. Less Configuration

Many existing protocols have been defined to be as flexible as possible. Consequently, these protocols need numerous initial configurations to start operations. There are choices and options that are irrelevant in any particular case, some of which target corner cases. Furthermore, in protocols that have existed for years, some design considerations are no longer relevant, since the underlying hardware technologies have evolved meanwhile. To appreciate the scale of this problem, consider that more than 160 DHCP options have been defined for IPv4. Even sample router configuration files readily available on line contain more than 200 lines of commands. There is therefore considerable scope for simplifying the operational tools for configuration of common protocols, even if the underlying protocols themselves cannot be simplified.

From another perspective, the deep reason why human decisions are often needed mainly result from the lack of information. When a device can collect enough information horizontally from other devices, it should be able to decide many parameters by itself, instead of receiving them from top-down configuration.

It is desired that top-down management is reduced in Autonomic Networking. Ideally, only the abstract Intent is needed from the human administrators. Neither users nor administrators should need to create and maintain detailed policies and profiles; if they are needed, they should be built autonomically. The local parameters should be decided by distributed Autonomic Nodes themselves, either from historic knowledge, analytics of current conditions, closed logical decision loops, or a combination of all.

5.5. Forecasting and Dry Runs

In a conventional network, there is no mechanism for trying something out safely. That means that configuration changes have to be designed in the abstract and their probable effects have to be estimated theoretically. In principle, an alternative to this would be to test the changes on a complete and realistic network simulator. However, this is a practical impossibility for a large network which is constantly changing, even if an accurate simulation could be performed. There is therefore a risk that applying changes to a running network will cause a failure of somekind. An autonomic network could fill this gap by supporting a closed loop "dry run" mode in which each configuration change could be tested out dynamically in the control plane without actually affecting the data plane. If the results are satisfactory, the change could be made live on the running network. If there is a consistency problem such as over-commitment of resources or incompatibility with another configuration setting, the change could be rolled back dynamically with no impact on traffic or users.

5.6. Benefit from Knowledge

The more knowledge and experience we have, the better decisions we can take. It is the same for networks and network management. When one component in the network lacks knowledge that affects what it should do, and another component has that knowledge, we usually rely on a human operator or a centralised management tool to convey the knowledge.

Up to now, the only available network knowledge is usually the current network status inside a given device or relevant current status from other devices.

However, historic knowledge is very helpful to make correct decisions, in particular to reduce network oscillation or to manage network resources over time. Transplantable knowledge from other networks can be helpful to initially set up a new network or new network devices. Knowledge of relationships between network events

and network configuration may help a network to decide the best parameters according to real performance feedback.

In addition to such historic knowledge, powerful data analytics of current network conditions may also be a valuable source of knowledge that can be exploited directly by Autonomic Nodes.

6. Security Considerations

This document is focused on what is missing to allow autonomic network configuration, including of course security settings. Therefore, it does not itself create any new security issues. It is worth underlining that autonomic technology must be designed with strong security properties from the start, since a network with vulnerable autonomic functions would be at great risk.

7. IANA Considerations

This memo includes no request to IANA.

8. Acknowledgements

The authors would like to acknowledge the valuable comments made by participants in the IRTF Network Management Research Group. Reviews by Kevin Fall and Rene Struik were especially helpful.

This document was produced using the xml2rfc tool [RFC2629].

9. Change log [RFC Editor: Please remove]

draft-irtf-nmrg-an-gap-analysis-06: RFC Editor comments actioned, 2015-05-01.

draft-irtf-nmrg-an-gap-analysis-05: IESG Review comments actioned, 2015-03-23.

draft-irtf-nmrg-an-gap-analysis-04: Additional IRSG Review comments actioned, 2015-03-04.

draft-irtf-nmrg-an-gap-analysis-03: IRSG Review comments actioned, 2014-12-11.

draft-irtf-nmrg-an-gap-analysis-02: Review comments actioned, 2014-10-02.

draft-irtf-nmrg-an-gap-analysis-01: RG comments added and more content in "Approach toward Autonomy" section, 2014-08-30.

draft-irtf-nmrg-an-gap-analysis-00: RG comments added, 2014-04-02.

draft-jiang-nmrg-an-gap-analysis-00: original version, 2014-02-14.

10. Informative References

[I-D.behringer-default-secure]

Behringer, M., Pritikin, M., and S. Bjarnason, "Making The Internet Secure By Default", draft-behringer-default-secure-00 (work in progress), January 2014.

[I-D.ietf-homenet-prefix-assignment]

Pfister, P., Paterson, B., and J. Arkko, "Distributed Prefix Assignment Algorithm", draft-ietf-homenet-prefix-assignment-05 (work in progress), April 2015.

[I-D.irtf-nmrg-autonomic-network-definitions]

Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking - Definitions and Design Goals", draft-irtf-nmrg-autonomic-network-definitions-07 (work in progress), March 2015.

[Kim13]

Kim, H. and N. Feamster, "Improving Network Management with Software Defined Networking", IEEE Communications Magazine, Pages 114-119, February 2013.

[RFC1661]

Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC1994]

Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.

[RFC2131]

Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[RFC2136]

Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC2629]

Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC2865]

Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC2866]

Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, August 2009.
- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, November 2011.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.
- [RFC7010] Liu, B., Jiang, S., Carpenter, B., Venaas, S., and W. George, "IPv6 Site Renumbering Gap Analysis", RFC 7010, September 2013.
- [RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, February 2014.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, March 2014.
- [Xiao02] Xiao, X. and others, "A Practical Approach for Providing QoS in the Internet Backbone", IEEE Communications Magazine, Pages 56-59, December 2002.

Authors' Addresses

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: jiangsheng@huawei.com

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Michael H. Behringer
Cisco Systems
Building D, 45 Allee des Ormes
Mougins 06250
France

Email: mbehring@cisco.com

Internet Research Task Force
Internet-Draft
Intended status: Informational
Expires: September 24, 2015

M. Behringer
M. Pritikin
S. Bjarnason
A. Clemm
Cisco Systems
B. Carpenter
Univ. of Auckland
S. Jiang
Huawei Technologies Co., Ltd
L. Ciavaglia
Alcatel Lucent
March 23, 2015

Autonomic Networking - Definitions and Design Goals
draft-irtf-nmrg-autonomic-network-definitions-07.txt

Abstract

Autonomic systems were first described in 2001. The fundamental goal is self-management, including self-configuration, self-optimization, self-healing and self-protection. This is achieved by an autonomic function having minimal dependencies on human administrators or centralized management systems. It usually implies distribution across network elements.

This document defines common language, and outlines design goals and non-design goals for autonomic functions. A high level reference model illustrates how functional elements in an autonomic network interact.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 24, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction to Autonomic Networking	2
2. Definitions	4
3. Design Goals	5
3.1. Self-Management	5
3.2. Co-Existence with Traditional Management	6
3.3. By Default Secure	7
3.4. Decentralisation and Distribution	8
3.5. Simplification of Autonomic Node Northbound Interfaces	8
3.6. Abstraction	8
3.7. Autonomic Reporting	9
3.8. Common Autonomic Networking Infrastructure	9
3.9. Independence of Function and Layer	10
3.10. Full Life Cycle Support	10
4. Non Design Goals	10
4.1. Eliminate human operators	10
4.2. Eliminate emergency fixes	11
4.3. Eliminate central control	11
5. An Autonomic Reference Model	11
6. IANA Considerations	13
7. Security Considerations	13
8. Acknowledgements	13
9. Informative References	14
Authors' Addresses	15

1. Introduction to Autonomic Networking

Autonomic systems were first described in a manifesto by IBM in 2001 [Kephart]. The fundamental concept involves eliminating external systems from a system's control loops and closing of control loops within the autonomic system itself, with the goal of providing the

system with self-management capabilities, including self-configuration, self-optimization, self-healing and self-protection.

IP networking was initially designed with similar properties in mind. An IP network should be distributed and redundant to withstand outages in any part of the network. Routing protocols such as OSPF or ISIS exhibit properties of self-management, and can thus be considered autonomic in the definition of this document.

However, as IP networking evolved, the ever increasing intelligence of network elements was often not put into protocols to follow this paradigm, but external configuration systems. This configuration made network elements dependent on some process that manages them, either a human, or a network management system.

Autonomic functions can be defined in two ways:

- o On a node level: Nodes interact with each other to form feedback loops.
- o On a system level: Feedback loops include central elements as well.

System level autonomy is implicitly or explicitly the subject in many IETF working groups, where interactions with controllers or network management systems are discussed.

This work specifically focuses on node level autonomic functions. It focuses on intelligence of algorithms at the node level, to minimize dependency on human administrators and central management systems.

Some network deployments benefit from a fully autonomic approach, for example networks with a large number of relatively simple devices. Most of currently deployed networks however will require a mixed approach, where some functions are autonomic and others are centrally managed. Central management of networking functions clearly has advantages and will be chosen for many networking functions. This document does not discuss which functions should be centralised or follow an autonomic approach. Instead, it should help make the decision which is the best approach for a given situation.

Autonomic function cannot always discover all required information; for example, policy related information requires human input, because policy is by its nature derived and specified by humans. Where input from some central intelligence is required, it is provided in a highly abstract, network wide form.

Autonomic Computing in general and Autonomic Networking in particular have been the subject of academic study for many years. There is a large literature, including several useful overview papers (e.g., [Samaan], [Movahedi], and [Dobson]). In the present document we focus on concepts and definitions that seem sufficiently mature to become the basis for interoperable specifications in the near future. In particular, such specifications will need to co-exist with traditional methods of network configuration and management, rather than realising an exclusively autonomic system with all the properties that it would require.

There is an important difference between "automatic" and "autonomic". "Automatic" refers to a pre-defined process, such as a script. "Autonomic" is used in the context of self-management. It includes feedback loops between elements as well as northbound to central elements. See also the definitions in the next section. Generally, an automatic process works in a given environment, but has to be adapted if the environment changes. An autonomic process can adapt to changing environments.

This document provides the definitions and design goals for Autonomic Networking in the IETF and IRTF.

2. Definitions

We make the following definitions:

Autonomic: Self-managing (self-configuring, self-protecting, self-healing, self-optimizing); however, allowing high-level guidance by a central entity, through Intent (see below). An autonomic function adapts on its own to a changing environment.

Automatic: A process that occurs without human intervention, with step-by-step execution of rules. However it relies on humans defining the sequence of rules, so is not Autonomic in the full sense. For example, a start-up script is automatic but not autonomic. An automatic function may need manual adjustments if the environment changes.

Intent: An abstract, high level policy used to operate the network. Its scope is an autonomic domain, such as an enterprise network. It does not contain configuration or information for a specific node (see Section 3.2 on how Intent co-exists with alternative management paradigms). It may contain information pertaining to nodes with a specific role, for example an edge switch, or a node running a specific function. Intent is typically defined and provided by a central entity.

Autonomic Domain: A collection of autonomic nodes that instantiate the same Intent.

Autonomic Function: A feature or function which requires no configuration, and can derive all required information either through self-knowledge, discovery or through Intent.

Autonomic Service Agent: An agent implemented on an autonomic node which implements an autonomic function, either in part (in the case of a distributed function) or whole.

Autonomic Node: A node which employs exclusively autonomic functions. It requires (!) no configuration. (Note that configuration can be used to override an autonomic function. See Section 3.2 for more details.) An Autonomic Node may operate on any layer of the networking stack. Examples are routers, switches, personal computers, call managers, etc.

Autonomic Network: A network containing exclusively autonomic nodes. It may contain one or several autonomic domains.

3. Design Goals

This section explains the high level goals of Autonomic Networking, independent of any specific solutions.

3.1. Self-Management

The original design goals of autonomic systems as described in [Kephart] also apply to Autonomic Networks. The over-arching goal is self-management, which is comprised of several self-* properties. The most commonly cited are:

- o Self-configuration: Functions do not require to be configured, neither by an administrator nor a management system. They configure themselves, based on self-knowledge, discovery, and Intent. Discovery is the default way for an autonomic function to receive the information it needs to operate.
- o Self-healing: Autonomic functions adapt on their own to changes in the environment, and heal problems automatically.
- o Self-optimising: Autonomic functions automatically determine ways to optimise their behaviour against a set of well-defined goals.
- o Self-protection: Autonomic functions automatically secure themselves against potential attacks.

Almost any network can be described as "self-managing", as long as the definition of "self" is large enough. For example, a well-defined SDN system, including the controller elements, can be described over all as "autonomic", if the controller provides an interface to the administrator which has the same properties as mentioned above (high level, network-wide, etc).

For the work in the IETF and IRTF we define the "self" properties on the node level. It is the design goal to make functions on network nodes self-managing, in other words, minimally dependent on management systems or controllers, as well as human operators. Self-managing functions on a node might need to exchange information with other nodes in order to achieve this design goal.

As mentioned in the Introduction, closed-loop control is an important aspect of self-managing systems. This implies peer-to-peer dialogues between the parties that make up the closed loop. Such dialogues require two-way "discussion" or "negotiation" between each pair or groups of peers involved in the loop, so they cannot readily use typical top-down command-response protocols. Also, a discovery phase is unavoidable before such closed-loop control can take place. Multi-party protocols are also possible but can be significantly more complex.

3.2. Co-Existence with Traditional Management

For the foreseeable future, autonomic nodes and networks will be the exception; autonomic behaviour will initially be defined function by function. Therefore, co-existence with other network management paradigms has to be considered. Examples are management by command line, SNMP, SDN (with related APIs), NETCONF, etc.

Conflict resolution between autonomic default behaviour and Intent on one side, and other methods on the other is therefore required. This is achieved through prioritisation. Generally, autonomic mechanisms define a network wide behaviour, whereas the alternative methods are typically on a node by node basis. Node based management concepts take a higher priority over autonomic methods. This is in line with current examples of autonomic functions, for example routing: A (statically configured) route has priority over the routing algorithm. In short:

- o lowest priority: autonomic default behaviour
- o medium priority: autonomic Intent
- o highest priority: node specific network management concepts, such as command line, SNMP, SDN, NETCONF, etc. How these concepts are

prioritised between themselves is outside the scope of this document.

The above prioritisation essentially results in the actions of the human administrator always being able to over-rule autonomic behaviour. This is generally the expectation of network operators today, and remains therefore a design principle here. In critical systems, such as atomic power plants, sometimes the opposite philosophy is used: The expectation is that a well defined algorithm is more reliable than a human operator, especially in rare exception cases. Networking generally does not follow this philosophy yet. Warnings however should be issued if node specific overrides may conflict with autonomic behaviour.

In other fields, autonomic mechanisms disengage automatically if certain conditions occur: The auto-pilot in a plane switches off if the plane is outside a pre-defined envelope of flight parameters. The assumption is that the algorithms only work correctly if the input values are in expected ranges. Some opinions however suggest that exactly in exceptional conditions is the worst moment to switch off autonomic behaviour, since the pilots have no full understanding of the situation at this point, and may be under high levels of stress. For this reason we suggest here to NOT generally disable autonomic functions if they encounter unexpected conditions, because it is expected that this adds another level of unpredictability in networks, when the situation may already be hard to understand.

3.3. By Default Secure

All autonomic interactions should be by default secure. This requires that any member of an autonomic domain can assert its membership using a domain identity, for example a certificate issued by a domain certification authority. This domain identity is used for nodes to learn about their neighbouring nodes, to determine the boundaries of the domain, and to cryptographically secure interactions within the domain. Nodes from different domains can also mutually verify their identity and secure interactions as long as they have a mutually respected trust anchor.

A strong, cryptographically verifiable domain identity is a fundamental cornerstone in Autonomic Networking. It can be leveraged to secure all communications, and allows thus automatic security without traditional configuration, for example pre-shared keys.

Autonomic functions must be able to adapt their behaviour depending on the domain of the node they are interacting with.

3.4. Decentralisation and Distribution

The goal of Autonomic Networking is to minimise dependencies on central elements; therefore, de-centralisation and distribution are fundamental to the concept. If a problem can be solved in a distributed manner, it should not be centralised.

In certain cases it is today operationally preferable to keep a central repository of information, for example a user database on a AAA server. An autonomic network should be able to use such central systems, in order to be deployable. It is possible to distribute such databases as well, and such efforts should be at least considered. Depending on the case, distribution may not be simple replication, but involve more complex interactions and organisation.

3.5. Simplification of Autonomic Node Northbound Interfaces

Even in a decentralised solution, certain information flows with central entities are required. Examples are high level service definitions, as well as network status requests, audit information, logging and aggregated reporting.

Therefore, also nodes in an autonomic network require a northbound interface. However, the design goal is to maintain this interface as simple and high level as possible.

3.6. Abstraction

An administrator or autonomic management system interacts with an autonomic network on a high level of abstraction. Intent is defined at a level of abstraction that is much higher than that of typical configuration parameters, for example, "optimize my network for energy efficiency". Intent must not be used to convey low-level commands or concepts, since those are on a different abstraction level.

For example, the administrator should not be exposed to the version of the IP protocol running in the network.

Also on the reporting and feedback side an autonomic network abstracts information and provides high-level messages such as "the link between node x and y is down" (possibly with an identifier for the specific link, in case of multiple links).

3.7. Autonomic Reporting

An autonomic network, while minimizing the need for user intervention, still needs to provide users with visibility like in traditional networks. However, in an autonomic network, reporting should happen on a network wide basis. Information about the network should be collected and aggregated by the network itself, presented in consolidated fashion to the administrator.

The layers of abstraction that are provided via Intent need to be supported for reporting functions as well, in order to give users an indication about the effectiveness of their intent. For example, in order to assess how effective the network performs with regards to the Intent "optimize my network for energy efficiency", the network should provide aggregate information about the number of ports that were able to be shut down, and the corresponding energy savings, while validating current service levels are on aggregate still met.

Autonomic network events should concern the autonomic network as a whole, not individual systems in isolation. For example, the same failure symptom should not be reported from every system that observes it, but only once for the autonomic network as a whole. Ultimately, the autonomic network should support exception based management, in which only events that truly require user attention are actually notified. This requires capabilities that allow systems within the network to compare information and apply specific algorithms to determine what should be reported.

3.8. Common Autonomic Networking Infrastructure

[I-D.irtf-nmrg-an-gap-analysis] points out that there are already a number of autonomic functions available today. However, these are largely independent, and each has its own methods and protocols to communicate, discover, define and distribute policy, etc.

The goal of the work on Autonomic Networking in the IETF is therefore not just to create autonomic functions, but to define a common infrastructure that autonomic functions can use. This Autonomic Networking Infrastructure may contain common control and management functions such as messaging, service discovery, negotiation, Intent distribution, self-monitoring and diagnostics, etc. A common approach to define and manage Intent is also required.

Refer to the reference model below: All the components around the "autonomic service agents" should be common components, such that the autonomic service agents do not have to replicate common tasks individually.

3.9. Independence of Function and Layer

Autonomic functions may reside on any layer in the networking stack. For example, layer 2 switching today is already relatively autonomic in many environments, since most switches can be plugged together in many ways and will automatically build a simple layer 2 topology. Routing functions run on a higher layer and can be autonomic on layer 3. Even application layer functionality such as unified communications can be autonomic.

"Autonomic" in the context of this framework is a property of a function which is implemented on a node. Autonomic functions can be implemented on any node type, for example a switch, router, server, or call manager.

An Autonomic Network requires an overall control plane for autonomic nodes to communicate. As in general IP networking, IP is the spanning layer that binds all those elements together; autonomic functions in the context of this framework should therefore operate at the IP layer. This concerns neighbour discovery protocols and other Autonomic Control Plane functions.

3.10. Full Life Cycle Support

An autonomic function does not depend on external input to operate; it needs to understand its current situation and surrounding, and operate according to its current state. Therefore, an autonomic function must understand the full life cycle of the device it runs on, from first manufacturing testing through deployment, testing, troubleshooting, up to decommissioning.

The state of the life-cycle of an autonomic node is reflected in a state model. The behaviour of an autonomic function may be different for different deployment states.

4. Non Design Goals

This section identifies various items that are explicitly not design goals in the IETF/IRTF for autonomic networks, which are mentioned to avoid misunderstandings of the general intention. They address some commonly expressed concerns from network administrators and architects.

4.1. Eliminate human operators

Section 3.1 states that "It is the design goal to [...] minimally dependent on [...] human operators". It is however not a design goal to completely eliminate them. The problem targeted by Autonomic

Networking is the error-prone and hard to scale model of individual configuration of network elements, traditionally by manual commands but today mainly by scripting and/or configuration management databases. This does not, however, imply the elimination of skilled human operators, who will still be needed for oversight, policy management, diagnosis, reaction to help desk tickets, etc. The main impact on administrators should be less tedious detailed work and more high-level work. (They should become more like doctors than hospital orderlies.)

4.2. Eliminate emergency fixes

However good the autonomous mechanisms, sometimes there will be fault conditions etc. that they cannot deal with correctly. At this point skilled operator interventions will be needed to correct or work around the problem. Hopefully this can be done by high-level mechanisms (adapting the policy database in some way) but in some cases direct intervention at device level may be unavoidable. This is obviously the case for hardware failures, even if the autonomic network has bypassed the fault for the time being. Truck rolls will not be eliminated when faulty equipment needs to be replaced. However, this may be less urgent if the autonomic system automatically reconfigures to minimise the operational impact.

4.3. Eliminate central control

While it is a goal to simplify northbound interfaces (Section 3.5), it is not a goal to eliminate central control, but to allow it on a higher abstraction level. Senior management might fear loss of control of an autonomic network. In fact this is no more likely than with a traditional network; the emphasis on automatically applying general policy and security rules might even provide more central control.

5. An Autonomic Reference Model

An Autonomic Network consists of Autonomic Nodes. Those nodes communicate with each other through an Autonomic Control Plane which provides a robust and secure communications overlay. The Autonomic Control Plane is self-organizing and autonomic itself.

An Autonomic Node contains various elements, such as autonomic service agents which implement autonomic functions. Figure 1 shows a reference model of an autonomic node. The elements and their interaction are:

- o Autonomic Service Agents, which implement the autonomic behaviour of a specific service or function.

- o Self-knowledge: An autonomic node knows its own properties and capabilities
- o Network Knowledge (Discovery): An autonomic service agent may require various discovery functions in the network, such as service discovery.
- o Intent: Network wide high level policy. Autonomic Service Agents use an Intent interpretation engine to locally instantiate the global Intent. This may involve coordination with other Autonomic Nodes.
- o Feedback Loops: Control elements outside the node may interact with autonomic nodes through feedback loops.
- o An Autonomic User Agent, providing a front-end to external users (administrators and management applications) through which they can receive reports, and monitor the Autonomic Network.
- o Autonomic Control Plane: Allows the node to communicate with other autonomic nodes. Autonomic functions such as Intent distribution, feedback loops, discovery mechanisms, etc, use the Autonomic Control Plane. The Autonomic Control Plane can run inband, over a configured VPN, over a self-managing overlay network, as described in [I-D.behringer-autonomic-control-plane], or over a traditional out of band network. Security is a requirement for the Autonomic Control Plane, which can be bootstrapped by a mechanism as described in [I-D.pritikin-bootstrapping-keyinfrastructures].

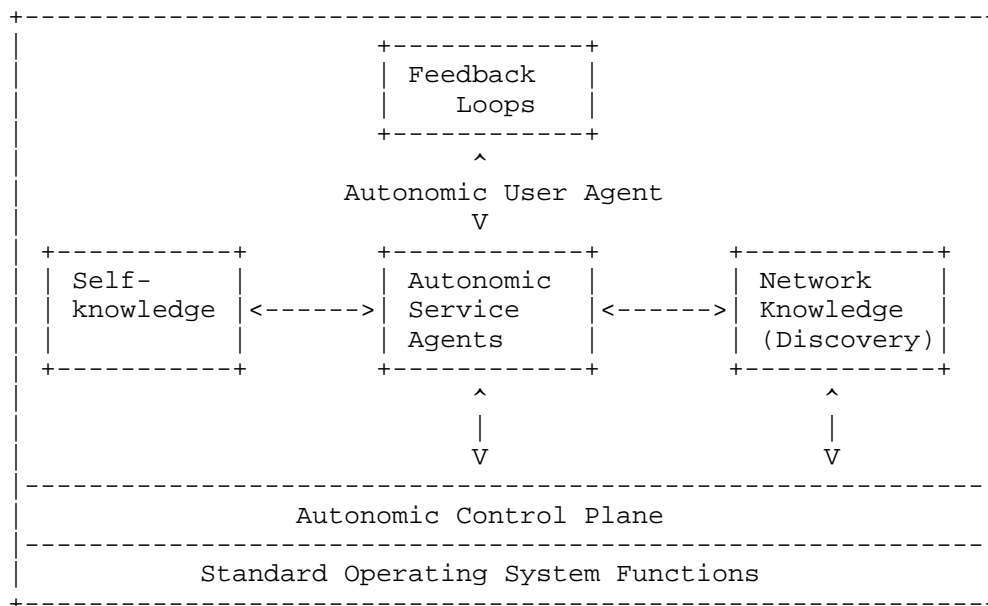


Figure 1: Reference Model for an Autonomic Node

6. IANA Considerations

This draft does not request any IANA action.

7. Security Considerations

This document provides definitions and design goals for Autonomic Networking. A full threat analysis will be required as part of the development of solutions, taking account of potential attacks from within the network as well as from outside.

8. Acknowledgements

Many parts of this work on Autonomic Networking are the result of a large team project at Cisco Systems. In alphabetical order: Ignas Bagdonas, Parag Bhide, Balaji BL, Toerless Eckert, Yves Hertoghs, Bruno Klauser.

We thank the following people for their input to this document: Dimitri Papadimitriou, Rene Struik, Kostas Pentikousis, Dave Oran, and Diego Lopez Garcia.

The ETSI working group AFI (<http://portal.etsi.org/afi>) defines a similar framework for Autonomic Networking in the "General Autonomic

Network Architecture" [GANA]. Many concepts explained in this document can be mapped to the GANA framework. The mapping is outside the scope of this document. Special thanks to Ranganai Chaparadza for his comments and help on this document.

9. Informative References

- [Dobson] Dobson et al., S., "A survey of autonomic communications", ACM Transactions on Autonomous and Adaptive Systems (TAAS) Volume 1 Issue 2, Pages 223-259 , December 2006.
- [GANA] ETSI GS AFI 002, , "Autonomic network engineering for the self-managing Future Internet (AFI): GANA Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management.", April 2013, <http://www.etsi.org/deliver/etsi_gs/AFI/001_099/002/01.01.01_60/gs_afi002v010101p.pdf>.
- [I-D.behringer-autonomic-control-plane] Behringer, M., Bjarnason, S., BL, B., and T. Eckert, "An Autonomic Control Plane", draft-behringer-autonomic-control-plane-00 (work in progress), June 2014.
- [I-D.irtf-nmrg-an-gap-analysis] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", draft-irtf-nmrg-an-gap-analysis-04 (work in progress), March 2015.
- [I-D.pritikin-bootstrapping-keyinfrastructures] Pritikin, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", draft-pritikin-bootstrapping-keyinfrastructures-01 (work in progress), September 2014.
- [Kephart] Kephart, J. and D. Chess, "The Vision of Autonomic Computing", IEEE Computer vol. 36, no. 1, pp. 41-50, January 2003, <<http://users.soe.ucsc.edu/~griss/agent-papers/ieee-autonomic.pdf>>.
- [Movahedi] Movahedi, Z., Ayari, M., Langar, R., and G. Pujolle, "A Survey of Autonomic Network Architectures and Evaluation Criteria", IEEE Communications Surveys & Tutorials Volume: 14 , Issue: 2 DOI: 10.1109/SURV.2011.042711.00078, Page(s): 464 - 490, 2012.

[Samaan] Samaan, N. and A. Karmouch, "Towards Autonomic Network Management: an Analysis of Current and Future Research Directions", IEEE Communications Surveys and Tutorials Volume: 11 , Issue: 3; DOI: 10.1109/SURV.2009.090303; Page(s): 22 - 36, 2009.

Authors' Addresses

Michael Behringer
Cisco Systems
Building D, 45 Allee des Ormes
Mougins 06250
France

Email: mbehring@cisco.com

Max Pritikin
Cisco Systems
5330 Airport Blvd
Boulder, CO 80301
USA

Email: pritikin@cisco.com

Steinthor Bjarnason
Cisco Systems
Mail Stop LYS01/5
Philip Pedersens vei 1
LYSAKER, AKERSHUS 1366
Norway

Email: sbjarnas@cisco.com

Alexander Clemm
Cisco Systems
170 West Tasman Drive
San Jose , California 95134-1706
USA

Email: alex@cisco.com

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: jiangsheng@huawei.com

Laurent Ciavaglia
Alcatel Lucent
Route de Villejust
Nozay 91620
France

Email: laurent.ciavaglia@alcatel-lucent.com

Network Management Research Group
Internet-Draft
Intended status: Informational
Expires: May 19, 2018

J. Nobre
University of Vale do Rio dos Sinos
L. Granville
Federal University of Rio Grande do Sul
A. Clemm
Huawei
A. Gonzalez Prieto
VMware
November 15, 2017

Autonomic Networking Use Case for Distributed Detection of SLA
Violations
draft-irtf-nmrg-autonomic-sla-violation-detection-13

Abstract

This document describes an experimental use case for autonomic networking concerning monitoring of Service Level Agreements (SLAs). The use case aims to detect violations of SLAs in a distributed fashion, striving to optimize and dynamically adapt the autonomic deployment of active measurement probes in a way that maximizes the likelihood of detecting service level violations with a given resource budget to perform active measurements, and is able to do so without any outside guidance or intervention.

This document is a product of the IRTF Network Management Research Group (NMRG). It is published for informational purposes.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 19, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Definitions and Acronyms	5
3. Current Approaches	6
4. Use Case Description	6
5. A Distributed Autonomic Solution	7
6. Intended User Experience	10
7. Implementation Considerations	10
7.1. Device Based Self-Knowledge and Decisions	11
7.2. Interaction with other devices	11
8. Comparison with current solutions	11
9. Related IETF Work	12
10. Acknowledgements	12
11. IANA Considerations	12
12. Security Considerations	13
13. Informative References	13
Authors' Addresses	14

1. Introduction

The Internet has been growing dramatically in terms of size, capacity, and accessibility in the last years. Communication requirements of distributed services and applications running on top of the Internet have become increasingly demanding. Some examples are real-time interactive video or financial trading. Providing such services involves stringent requirements in terms of acceptable latency, loss, or jitter.

Performance requirements lead to the articulation of Service Level Objectives (SLOs) which must be met. Those SLOs are part of Service Level Agreements (SLAs) that define a contract between the provider and the consumer of a service. SLOs, in effect, constitute a service

level guarantee that the consumer of the service can expect to receive (and often has to pay for). Likewise, the provider of a service needs to ensure that the service level guarantee and associated SLOs are met. Some examples of clauses that relate to service level objectives can be found in [RFC7297]).

Violations of SLOs can be associated with significant financial loss, which can be divided into two categories. For one, there is the loss that can be incurred by the user of a service when the agreed service levels are not provided. For example, a financial brokerage's stock orders might suffer losses when it is unable to execute stock transactions in a timely manner. An electronic retailer may lose customers when their online presence is perceived by customers as sluggish. An online gaming provider may not be able to provide fair access to online players, resulting in frustrated players who are lost as customers. In each case, the failure of a service provider to meet promised service level guarantees can have a substantial financial impact on users of the service. By the same token, there is the loss that is incurred by the provider of a service who is unable to meet promised service level objectives. Those losses can take several forms, such as penalties for not meeting the service and, in many cases more important, loss of revenue due to reduced customer satisfaction. Hence, service level objectives are a key concern for the service provider. In order to ensure that SLOs are not being violated, service levels need to be continuously monitored at the network infrastructure layer in order to know, for example, when mitigating actions need to be taken. To that end, service level measurements must take place.

Network measurements can be performed using active or passive measurement techniques. In passive measurements, production traffic is observed and no monitoring traffic is created by the measurement process itself. That is, network conditions are checked in a non intrusive way. In the context of IP Flow Information eXport (IPFIX), several documents were produced that define how to export data associated with flow records, i.e. data that is collected as part of passive measurement mechanisms, generally applied against flows of production traffic (e.g., [RFC7011]). In addition, it would be possible to collect real data traffic (not just summarized flow records) with time-stamped packets, possibly sampled (e.g., per [RFC5474], as a means of measuring and inferring service levels. Active measurements, on the other hand, are more intrusive to the network in the sense that it involves injecting synthetic test traffic into the network to measure network service levels, as opposed to simply observing production traffic. The IP Performance Metrics (IPPM) WG produced documents that describe active measurement mechanisms, such as: One-Way Active Measurement Protocol (OWAMP) [RFC4656], Two-Way Active Measurement Protocol (TWAMP) [RFC5357], and

Cisco Service Level Assurance Protocol (SLA) [RFC6812]. In addition, there are some mechanisms that do not cleanly fit into either active or passive categories, such as Performance and Diagnostic Metrics Destination Option (PDM) techniques [RFC8250].

Active measurement mechanisms offer a high level of control of what and how to measure. They do not require inspecting production traffic. Because of this, active measurements usually offer better accuracy and privacy than passive measurement mechanisms. Traffic encryption and regulations that limit the amount of payload inspection that can occur are non-issues. Furthermore, active measurement mechanisms are able to detect end-to-end network performance problems in a fine-grained way (e.g., simulating the traffic that must be handled considering specific Service Level Objectives - SLOs). As a result, active measurements are often preferred over passive measurement for SLA monitoring. Measurement probes must be hosted in network devices and measurement sessions must be activated to compute the current network metrics (e.g., considering those described in [RFC4148]). This activation should be dynamic in order to follow changes in network conditions, such as those related with routes being added or new customer demands.

While offering many advantages, active measurements are expensive in terms of network resource consumption. Active measurements generally involve measurement probes that generate synthetic test traffic that is directed at a responder. The responder needs to timestamp test traffic it receives and reflect it back to the originating measurement probe. The measurement probe subsequently processes the returned packets along with time stamping information in order to compute service levels. Accordingly, active measurements consume substantial CPU cycles as well as memory of network devices to generate and process test traffic. In addition, synthetic traffic increases network load. Active measurements thus compete for resources with other functions, including routing and switching.

The resources required and traffic generated by the active measurement sessions are to a large part a function of the number of measured network destinations. (In addition, the amount of traffic generated for each measurement plays a role, which in turn influences the accuracy of the measurement.) The more destinations are being measured, the larger the amount of resources consumed and traffic needed to perform the measurements. Thus, to have a better monitoring coverage it is necessary to deploy more sessions which consequently increases consumed resources. Otherwise, enabling the observation of just a small subset of all network flows can lead to an insufficient coverage.

Furthermore, while some end-to-end service levels can be determined by adding up the service levels observed across different path segments, the same is not true for all service levels. For example, the end-to-end delay or packet loss from a node A to a node C routed via a node B can often be computed simply by adding delays (or loss) from A to B, and B to C. This allows to decompose a large set of end-to-end measurements into a much smaller set of segment measurements. However, end-to-end jitter and (for example) Mean Opinion Scores cannot be decomposed as easily and, for higher accuracy, must be measured end-to-end.

Hence, the decision how to place measurement probes becomes an important management activity. The goal is to obtain maximum benefits of service level monitoring with a limited amount of measurement overhead. Specifically, the goal is to maximize the number of service level violations that are detected with a limited amount of resources.

The use case and the solution approach described in this document address an important practical issue. They are intended to provide a basis for further experimentation to lead into solutions for wider deployment. This document represents the consensus of the IRTF's Network Management Research Group (NMRG). It was discussed extensively and received three separate in-depth reviews.

2. Definitions and Acronyms

Active Measurements: Techniques to measure service levels that involve generating and observing synthetic test traffic

Passive Measurements: Techniques used to measure service levels based on observation of production traffic

AN: Autonomic Network; a network containing exclusively autonomic nodes, requiring no configuration and deriving all required information through self-knowledge, discovery, or intent.

Autonomic Service Agent (ASA): An agent implemented on an autonomic node that implements an autonomic function, either in part (in the case of a distributed function, as in the context of this document), or whole.

Measurement Session: A communications association between a Probe and a Responder used to send and reflect synthetic test traffic for active measurements

Probe: The source of synthetic test traffic in an active measurement

Responder: The destination for synthetic test traffic in an active measurement

SLA: Service Level Agreement

SLO: Service Level Objective

P2P: Peer-to-Peer

(Note: definitions of AN and ASA are borrowed from [RFC7575]).

3. Current Approaches

The current best practice in feasible deployments of active measurement solutions to distribute the available measurement sessions along the network consists in relying entirely on the human administrator expertise to infer which would be the best location to activate such sessions. This is done through several steps. First, it is necessary to collect traffic information in order to grasp the traffic matrix. Then, the administrator uses this information to infer which are the best destinations for measurement sessions. After that, the administrator activates sessions on the chosen subset of destinations considering the available resources. This practice, however, does not scale well because it is still labor intensive and error-prone for the administrator to determine which sessions should be activated given the set of critical flows that needs to be measured. Even worse, this practice completely fails in networks whose critical flows are too short in time and dynamic in terms of traversing network path, like in modern cloud environments. That is so because fast reactions are necessary to reconfigure the sessions and administrators are just not quick enough in computing and activating the new set of required sessions every time the network traffic pattern changes. Finally, the current active measurements practice usually covers only a fraction of the network flows that should be observed, which invariably leads to the damaging consequence of undetected SLA violations.

4. Use Case Description

The use case involves a service level provider who needs to monitor the network to detect service level violations using active service level measurements, and wants to be able to do so with minimal human intervention. The goal is to conduct the measurements in an effective manner maximizing the percentage of detected service level violations. The service level provider has a bounded resource budget with regards to measurements that can be performed, specifically, with regards to the number of measurements that can be conducted concurrently from any one network device, and possibly with regards

to the total amount of measurement traffic on the network. However, while at any one point in time the number of measurements conducted is limited, it is possible for a device to change which destinations to measure over time. This can be exploited to achieve a balance of eventually covering all possible destinations using a reasonable amount of "sampling" where measurement coverage of a destination cannot be continuous. The solution needs to be dynamic and be able to cope with network conditions which may change over time. The solution should also be embeddable inside network devices that control the deployment of active measurement mechanisms.

The goal is to conduct the measurements in a smart manner that ensures that the network is broadly covered and the likelihood of detecting service level violations is maximized. In order to maximize that likelihood, it is reasonable to focus measurement resources on destinations that are more likely to incur a violation, while spending less resources on destinations that are more likely to be in compliance. In order to do so, there are various aspects that can be exploited, including past measurements (destinations close to a service level threshold requiring more focus than destinations further from it), complementation with passive measurements such as flow data (to identify network destinations that are currently popular and critical), and observations from other parts of the network. In addition, measurements can be coordinated among different network devices to avoid hitting the same destination at the same time and to be able to share results that may be useful in future probe placement.

Clearly, static solutions will have severe limitations. At the same time, human administrators cannot be in the loop for continuous dynamic measurement probe reconfigurations. Accordingly, an automated or, ideally, autonomic solution is needed in which network measurements are automatically orchestrated and dynamically reconfigured from within the network. This can be accomplished using an autonomic solution that is distributed, using Autonomic Service Agents that are implemented on nodes in the network.

5. A Distributed Autonomic Solution

The use of Autonomic Networking (AN) [RFC7575] can help such detection through an efficient activation of measurement sessions. Such an approach, along with a detailed assessment confirming its viability, has been described [P2PBNM-Nobre-2012]. The problem to be solved by AN in the present use case is how to steer the process of Measurement Session activation by a complete solution that sets all necessary parameters for this activation to operate efficiently, reliably and securely, with no required human intervention other than setting overall policy.

When a node first comes online, it has no information about which measurements are more critical than others. In the absence of information about past measurements and information from measurement peers, it may start with an initial set of measurement sessions, possibly randomly seeding a set of starter measurements, perhaps taking a round robin approach for subsequent measurement rounds. However, as measurements are collected, a node will gain increasing information that it can utilize to refine its strategy of selecting measurement targets going forward. For one, it may take note of which targets returned measurement results very close to service level thresholds that may therefore require closer scrutiny compared to others. Second, it may utilize observations that are made by its measurement peers in order to conclude which measurement targets may be more critical than others, and in order to ensure that proper overall measurement coverage is obtained (so that not every node incidentally measure the same targets, while other targets are not measured at all).

We advocate for embedding Peer-to-Peer (P2P) technology in network devices in order to conduct the Measurement Session activation decisions using autonomic control loops. Specifically, we advocate for network devices to implement an autonomic function to monitor service levels for violations of service level objectives, determining which Measurement Sessions to set up at any given point in time based on current and past observations of the node, and of other peer nodes.

By performing these functions locally and autonomically on the device itself, which measurements to conduct can be modified quickly based on local observations while taking local resource availability into account. This allows a solution to be more robust and react more dynamically to rapidly changing service levels than a solution that has to rely on central coordination. However, in order to optimize decisions which measurements to conduct, a node will need to communicate with other nodes. This allows a node to take into account other nodes' observations in addition to its own in its decisions.

For example, remote destinations whose observed service levels are on the verge of violating stated objectives may require closer monitoring than remote destinations that are comfortably within a range of tolerance. It also allows nodes to coordinate their probing decisions to collectively achieve the best possible measurement coverage. As the amount of resources available for monitoring and for exchange of measurement data and coordination with other nodes are limited, a node may further be interested in identifying other nodes whose observations are most similar to and correlated with its own. This helps a node prioritize and guide with which other nodes

to primarily coordinate and exchange data with. All of this requires the use of a P2P overlay.

A P2P overlay is essential for several reasons:

- o It makes it possible for nodes (respectively Autonomic Service Agents that are deployed on those nodes) in the network to autonomically set up Measurement Sessions, without having to rely on central management system or controller to perform configuration operations associated with configuring measurement probes and responders.
- o It facilitates the exchange of data between different nodes to share measurement results so that each node can refine its measurement strategy based not just its own observations, but observations from its peers.
- o It allows nodes to coordinate their measurements to obtain the best possible test coverage and avoid measurements that have a very low likelihood of detecting service level violations.

The provisioning of the P2P overlay should be transparent for the network administrator. An Autonomic Control Plane such as defined in [I-D.anima-autonomic-control-plane] provides an ideal candidate for the P2P overlay to run on.

An autonomic solution for the distributed detection of SLA violations provide several benefits. First, efficiency: this solution should optimize the resource consumption and avoid resource starvation on the network devices. A device that is "self-aware" of its available resources will be able to adjust measurement activities rapidly as needed, without requiring a separate control loop involving resource monitoring by an external system. Secondly, placing logic where to conduct measurements in the node enables rapid control loops in which devices are able to react instantly to observations and adjust their measurement strategy. For example, a device could decide to adjust the amount of synthetic test traffic being sent during the measurement itself depending on results observed so far on this and on other concurrent measurement sessions. As a result, the solution could decrease the time necessary to detect SLA violations. Adaptivity features of an autonomic loop could capture faster the network dynamics than an human administrator and even a central controller. Finally, the solution could help to reduce the workload of human administrator, or, at least, to avoid their need to perform operational tasks.

In practice, these factors combine to maximize the likelihood of SLA violations being detected while operating within a given resource

budget, allowing to conduct a continuous measurement strategy that takes into account past measurement results, observations of other measures such as link utilization or flow data, sharing of measurement results between network devices, and coordinating future measurement activities among nodes. Combined this can result in efficient measurement decisions that achieve a golden balance between broad network coverage and honing in on service level "hot spots".

6. Intended User Experience

The autonomic solution should not require any human intervention in the distributed detection of SLA violations. By virtue of the solution being autonomic, human users will not have to plan which measurements to conduct in a network, often a very labor intensive task today that requires detailed analysis of traffic matrices and network topologies and is not prone to easy dynamic adjustment. Likewise, they will not have to configure measurement probes and responders.

There are some ways in which a human administrator may still interact with the solution. For one, the human administrator will of course be notified and obtain reports about service level violations that are observed. Second, a human administrator may set a policies regarding how closely to monitor the network for service level violations and how many resources to spend. For example, an administrator may set a resource budget that is assigned to network devices for measurement operations. With that given budget, the number of SLO violations that are detected will be maximized. Alternatively, an administrator may set a target for the percentage of SLO violations that must be detected, i.e. a target for the ratio between the number of detected SLO violations, and the number of total SLO violations that are actually occurring (some of which might go undetected). In that case, the solution will aim to minimize the resources spent (i.e. the amount of test traffic and Measurement Sessions) that are required to achieve that target.

7. Implementation Considerations

The active measurement model assumes that a typical infrastructure will have multiple network segments and Autonomous Systems (ASs), and a reasonably large number of routers. It also considers that multiple SLOs can be in place at a given time. Since interoperability in a heterogenous network is a goal, features found on different active measurement mechanisms (e.g. OWAMP, TWAMP, and IPSLA) and device programmability interfaces (such as Juniper's Junos API or Cisco's Embedded Event Manager) could be used for the implementation. The autonomic solution should include and/or reference specific algorithms, protocols, metrics and technologies

for the implementation of distributed detection of SLA violations as a whole.

Finally, it should be noted that there are multiple deployment scenarios, including deployment scenarios that involve physical devices hosting autonomic functions, or virtualized infrastructure hosting the same. Co-deployment in conjunction with Virtual Network Functions (VNF) is a possibility for further study.

7.1. Device Based Self-Knowledge and Decisions

Each device has self-knowledge about the local SLA monitoring. This could be in the form of historical measurement data and SLOs. Besides that, the devices would have algorithms that could decide which probes should be activated in a given time. The choice of which algorithm is better for a specific situation would be also autonomic.

7.2. Interaction with other devices

Network devices should share information about service level measurement results. This information can speed up the detection of SLA violations and increase the number of detected SLA violations. For example, if one device detects that a remote destination is in danger of violating an SLO, other devices may conduct additional measurements to the same destination or other destinations in its proximity. For any given network device, the exchange of data may be more important with some devices (for example, devices in the same network neighborhood, or devices that are "correlated" by some other means) than with others. The definition of network devices that exchange measurement data, i.e., management peers, creates a new topology. Different approaches could be used to define this topology (e.g., correlated peers [P2PBNM-Nobre-2012]). To bootstrap peer selection, each device should use its known endpoints neighbors (e.g., FIB and RIB tables) as the initial seed to get possible peers. It should be noted that a solution will benefit if topology information and network discovery functions are provided by the underlying autonomic framework. A solution will need to be able to discover measurement peers as well as measurement targets, specifically measurement targets that support active measurement responders and which will be able to respond to measurement requests and reflect measurement traffic as needed.

8. Comparison with current solutions

There is no standardized solution for distributed autonomic detection of SLA violations. Current solutions are restricted to ad hoc scripts running on a per node fashion to automate some

administrator's actions. There are some proposals for passive probe activation (e.g., DECON and CSAMP), but without the focus on autonomic features.

9. Related IETF Work

The following paragraphs discuss related IETF work and are provided for reference. This section is not exhaustive, rather it provides an overview of the various initiatives and how they relate to autonomic distributed detection of SLA violations.

1. [LMAP]: The Large-Scale Measurement of Broadband Performance Working Group aims at the standards for performance management. Since their mechanisms also consist in deploying measurement probes the autonomic solution could be relevant for LMAP specially considering SLA violation screening. Besides that, a solution to decrease the workload of human administrators in service providers is probably highly desirable.
2. [IPFIX]: IP Flow Information EXport (IPFIX) aims at the process of standardization of IP flows (i.e., netflows). IPFIX uses measurement probes (i.e., metering exporters) to gather flow data. In this context, the autonomic solution for the activation of active measurement probes could be possibly extended to address also passive measurement probes. Besides that, flow information could be used in the decision making of probe activation.
3. [ALTO]: The Application Layer Traffic Optimization Working Group aims to provide topological information at a higher abstraction layer, which can be based upon network policy, and with application-relevant service functions located in it. Their work could be leveraged for the definition of the topology regarding the network devices which exchange measurement data.

10. Acknowledgements

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from Mohamed Boucadair, Brian Carpenter, Hanlin Fang, Bruno Klauser, Diego Lopez, Vincent Roca, and Eric Voit. In addition, we thank Diego Lopez, Vincent Roca, and Brian Carpenter for their detailed reviews.

11. IANA Considerations

This memo includes no request to IANA.

12. Security Considerations

Security of the solution hinges on the security of the network underlay, i.e. the Autonomic Control Plane. If the Autonomic Control Plane were to be compromised, an attacker could undermine the effectiveness of measurement coordination by reporting fraudulent measurement results to peers. This would cause measurement probes to be deployed in an ineffective manner that would increase the likelihood that violations of service level objectives go undetected.

Likewise, security of the solution hinges on the security of the deployment mechanism for autonomic functions, in this case, the autonomic function that conducts the service level measurements. If an attacker were able to hijack an autonomic function, it could try to exhaust or exceed the resources that should be spent on autonomic measurements in order to deplete network resources, including network bandwidth due to higher-than-necessary volumes of synthetic test traffic generated by measurement probes. Again, it could also lead to reporting of misleading results, among other things resulting in non-optimal selection of measurement targets and in turn an increase in the likelihood that service level violations go undetected.

13. Informative References

[draft-anima-boot]

Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "draft-ietf-anima-bootstrapping-keyinfra", draft-ietf-anima-bootstrapping-keyinfra-08 (work in progress), October 2017.

[I-D.anima-autonomic-control-plane]

Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-12 (work in progress), October 2017.

[P2PBNM-Nobre-2012]

Nobre, J., Granville, L., Clemm, A., and A. Gonzalez Prieto, "Decentralized Detection of SLA Violations Using P2P Technology, 8th International Conference Network and Service Management (CNSM)", 2012, <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6379997>.

[RFC4148]

Stephan, E., "IP Performance Metrics (IPPM) Metrics Registry", BCP 108, RFC 4148, DOI 10.17487/RFC4148, August 2005, <<https://www.rfc-editor.org/info/rfc4148>>.

- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <<https://www.rfc-editor.org/info/rfc4656>>.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <<https://www.rfc-editor.org/info/rfc5357>>.
- [RFC5474] Duffield, N., Ed., Chiou, D., Claise, B., Greenberg, A., Grossglauser, M., and J. Rexford, "A Framework for Packet Selection and Reporting", RFC 5474, DOI 10.17487/RFC5474, March 2009, <<https://www.rfc-editor.org/info/rfc5474>>.
- [RFC6812] Chiba, M., Clemm, A., Medley, S., Salowey, J., Thombare, S., and E. Yedavalli, "Cisco Service-Level Assurance Protocol", RFC 6812, DOI 10.17487/RFC6812, January 2013, <<https://www.rfc-editor.org/info/rfc6812>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <<https://www.rfc-editor.org/info/rfc7297>>.
- [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>>.
- [RFC8250] Elkins, N., Hamilton, R., and M. Ackermann, "IPv6 Performance and Diagnostics Metrics (PDM) Destination Option", RFC 8250, October 2017.

Authors' Addresses

Jeferson Campos Nobre
University of Vale do Rio dos Sinos
Porto Alegre
Brazil

Email: jcnobre@unisinos.br

Lisandro Zambenedetti Granville
Federal University of Rio Grande do Sul
Porto Alegre
Brazil

Email: granville@inf.ufrgs.br

Alexander Clemm
Huawei
Santa Clara, California
USA

Email: ludwig@clemm.org

Alberto Gonzalez Prieto
VMware
Palo Alto, California
USA

Email: agonzalezpri@vmware.com

AN Use Case BOF
Internet-Draft
Intended status: Informational
Expires: October 30, 2014

S. Jiang, Ed.
Huawei Technologies Co., Ltd
B. Carpenter
Univ. of Auckland
Q. Sun
China Telecom
April 28, 2014

Autonomic Networking Use Case for Auto Address Management
draft-jiang-auto-addr-management-00

Abstract

This document describes a use case for autonomic address management in large-scale networks. It is one of a series of use cases intended to illustrate requirements for autonomic networking.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 30, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Problem Statement	2
3. Intended User and Administrator Experience	3
4. Analysis of Parameters and Information Involved	3
4.1. Parameters each device can decide for itself	4
4.2. Information needed from policy intent	5
5. Interaction with other devices	5
5.1. Information needed from other devices	5
5.2. Monitoring, diagnostics and reporting	6
6. Comparison with current solutions	6
7. Security Considerations	7
8. IANA Considerations	7
9. Acknowledgements	7
10. Change log [RFC Editor: Please remove]	7
11. References	7
Authors' Addresses	8

1. Introduction

This document is one of a set of use cases being developed to clarify the requirements for discovery and negotiation protocols for autonomic networking (AN). The background to AN is described in [I-D.irtf-nmrg-autonomic-network-definitions] and [I-D.irtf-nmrg-an-gap-analysis]. A problem statement and outline requirements for the negotiation protocol are given in [I-D.jiang-config-negotiation-ps].

This document is dedicated to how to make IP address management in large-scale networks as autonomic as possible, including operator (ISP) networks and large enterprise networks. Although this document is targeting pure IPv6 networks, autonomically sharing public IPv4 addresses among the Address Family Transition Routers (AFTRs) [RFC6333] or NAT64 [RFC6146] devices is also discussed.

Note in draft: This version is preliminary. In particular, opinions may vary about how concrete vs how abstract a use case should be.

2. Problem Statement

The autonomic networking use case considered here is autonomic IP address management in large-scale networks.

Although DHCPv6 Prefix Delegation [RFC3633] has supported automated delegation of IPv6 prefixes, the prefix management is still largely depending on human planning. In other words, there is no basic information or policies to support autonomic decisions on the prefix length that each router should request or be delegated, according to its role in the network. Roles could be locally defined or could be generic (edge router, interior router, etc.). Furthermore, the current IPv6 prefix management by humans is rigid and static after initial planning.

Additionally, the management of public IPv4 addresses on AFTRs or NAT64 devices is similarly rigid and static. The utilisation rate of addresses depends on the initial plan. Efficient utilisation of public IPv4 addresses is the most important requirement since they are a limited resource during the IPv4 exhaustion period.

The problem to be solved by AN is how to dynamically and autonomically manage IPv6 address space and public IPv4 addresses on AFTRs or NAT64 devices in large-scale networks, so that IP addresses can be used efficiently. The AN approach discussed in this document is based on the assumption that there was a generic discovery and negotiation protocol that enables direct negotiation between intelligent IP routers. [I-D.jiang-config-negotiation-protocol] is one of the attempts at such a protocol.

3. Intended User and Administrator Experience

The intended experience is, for the administrator(s) of a large-scale network, that the management of IPv6 address space can be run with minimum efforts, for both the network and network device initiation stage and during running time. In the most ideal scenario, the administrator(s) only have to configure a single IPv6 prefix for the whole network and the initial prefix length for each device role.

Where applicable, another intended experience is dynamically and autonomically sharing public IPv4 addresses on AFTRs or NAT64 devices without human intervention. The administrator only has to configure the total available IPv4 address range.

The actual address usage needs to be logged for the potential offline management operations including audit and security incident tracing.

4. Analysis of Parameters and Information Involved

For specific purposes of address management, a few parameters are involved on each device (some of them can be pre-configured before they are connected). They include:

- o Identity of this device. It can be verified by the certification authority (CA) that is maintained by the network administrator(s).
- o Identity of a trust anchor which is certification authority (CA) that is maintained by the network administrator(s).
- o Role of this device.
- o An IPv6 prefix length for this device.
- o An IPv6 prefix that is assigned to this device and its downstream devices.
- o A public IPv4 address pool if the device acts as an AFTR or NAT64 device.

A few parameters are involved in the network as a whole. They are:

- o Identity of a trust anchor which is a certification authority (CA) that is maintained by the network administrator(s).
- o Total IPv6 address space. It is one (or several) IPv6 prefix(es).
- o A public IPv4 address pool if the network provides IPv4 over IPv6 access or IPv4/IPv6 transition services.
- o The initial prefix length for each device role.

4.1. Parameters each device can decide for itself

This section identifies those of the above parameters that do not need external information in order for the devices concerned to set them to a reasonable value after bootstrap or after a network disruption. There are few of these:

- o Role of this device, this includes whether this device acts as an AFTR or NAT64 device.
- o Default IPv6 prefix length for this device.
- o Identity of this device.

The device may be shipped from the manufacture with pre-configured role and default prefix length.

4.2. Information needed from policy intent

This section identifies those parameters that need external information about policy intent in order for the devices concerned to set them to a non-default value.

- o Non-default value for the IPv6 prefix length for this device. This needs to be decided based on the role of this device.
- o The initial prefix length for each device role.
- o Identity of a trust anchor.
- o Whether to allow the device request more address space.
- o Whether to allow the device to request or share public IPv4 address.
- o The policy when to request more address space, for example, the address usage reaches a certain limit or percentage.

5. Interaction with other devices

5.1. Information needed from other devices

This section identifies those of the above parameters that need external information from neighbor devices (including the upstream devices). In many cases, two-way dialogue with neighbor devices is needed to set or optimise them.

- o Identity of a trust anchor.
- o The device will need to discover their neighbors, particularly, the upstream device, from which it can acquire IPv6 address space.
- o The initial prefix length for each device role, particularly for its own downstream devices.
- o The default value of the IPv6 prefix length may be overridden by a non-default value.
- o The device will need to request and acquire IPv6 prefix that is assigned to this device and its downstream devices.
- o The device may respond to prefix delegation request from its downstream devices.

- o The device may require to be assigned more IPv6 address space, if it used up its assigned IPv6 address space.
- o An AFTR or NAT64 device will need to request and acquire an initial public IPv4 address pool.
- o An AFTR or NAT64 device will need to discover its neighbors, from which it may acquire spare public IPv4 addresses.
- o An AFTR or NAT64 device may acquire spare public IPv4 addresses with their associated available period.

5.2. Monitoring, diagnostics and reporting

This section discusses what role devices should play in monitoring, fault diagnosis, and reporting.

- o The actual address assignments need to be logged for the potential offline management operations.
- o In general, the usage situation of address space should be reported to the network administrators, in an abstract way, for example, statistics or visualized report.
- o A forecast of address exhaustion should be reported.

6. Comparison with current solutions

This section briefly compares the above use case with current solutions. Currently, the address management is still largely depending on human planning. It is rigid and static after initial planning. The address requests will fail if the configured address space is used up.

Some functions, for autonomic and dynamic address management, may be achievable by extending the existing protocols, for example, extending DHCPv6-PD to request IPv6 address according to the device role. However, defining uniform device roles may not be a practical task. Some functions are not suitable to be achieved by any existing protocols, such as dynamically negotiating the sharing of public IPv4 addresses.

However, using a generic autonomic discovery and negotiation protocol instead of specific solutions has the advantage that additional parameters can be included in the autonomic solution without creating new mechanisms. This is the principal argument for a generic approach.

7. Security Considerations

Relevant security issues are discussed in [I-D.irtf-nmrg-autonomic-network-definitions], [I-D.jiang-config-negotiation-ps]. The security mechanism in this document is established on a Public Key Infrastructure (PKI) system [RFC3647] that is maintained by the network administrator(s).

8. IANA Considerations

This document requests no action by IANA.

9. Acknowledgements

Valuable comments were received from Michael Behringer and Chongfeng Xie.

This document was produced using the xml2rfc tool [RFC2629].

10. Change log [RFC Editor: Please remove]

draft-jiang-auto-addr-management-00: original version, 2014-04-28.

11. References

- [I-D.irtf-nmrg-an-gap-analysis]
Behringer, M., Carpenter, B., and S. Jiang, "Gap Analysis for Autonomic Networking", draft-irtf-nmrg-an-gap-analysis-00 (work in progress), April 2014.
- [I-D.irtf-nmrg-autonomic-network-definitions]
Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking - Definitions and Design Goals", draft-irtf-nmrg-autonomic-network-definitions-00 (work in progress), December 2013.
- [I-D.jiang-config-negotiation-protocol]
Jiang, S., Carpenter, B., Liu, B., and Y. Yin, "Configuration Negotiation Protocol for Network Devices", draft-jiang-config-negotiation-protocol-01 (work in progress), April 2014.
- [I-D.jiang-config-negotiation-ps]
Jiang, S., Yin, Y., and B. Carpenter, "Network Configuration Negotiation Problem Statement and Requirements", draft-jiang-config-negotiation-ps-02 (work in progress), January 2014.

- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC3647] Chokhani, S., Ford, W., Sabett, R., Merrill, C., and S. Wu, "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework", RFC 3647, November 2003.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.
- [RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

Authors' Addresses

Sheng Jiang (editor)
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: jiangsheng@huawei.com

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Qiong
China Telecom
No.118, Xizhimennei Street
Beijing 100035
P. R. China

Email: sunqiong@ctbri.com.cn

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 28, 2014

S. Jiang
Huawei Technologies Co., Ltd
B. Carpenter
Univ. of Auckland
B. Liu
Huawei Technologies Co., Ltd
June 26, 2014

Configuration Discovery and Negotiation Protocol for Network Devices
draft-jiang-config-negotiation-protocol-02

Abstract

This document defines a new protocol that enables intelligent devices to dynamically discover and negotiate their configuration with counterpart devices. This document only defines a general protocol as a negotiation platform while the negotiation objectives for specific scenarios are to be described in separate documents.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 28, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements Language and Terminology	4
3. CDNP Protocol Overview	4
3.1. IP Version Independent	5
3.2. Objective Oriented Discovery Mechanism	5
3.3. Neighbor Diverting Discovery Mechanism	5
3.4. Certificate-based Security Mechanism	6
3.4.1. Support for algorithm agility	7
3.4.2. Message validation on reception	7
3.4.3. TimeStamp checking	8
3.5. Negotiation Procedures	9
4. CDNP Constants	10
5. Device Identifier and Certificate Tag	10
6. Session Identifier	11
7. CDNP Messages	11
7.1. CDNP Message Format	11
7.2. Request Message	12
7.3. Negotiation Message	12
7.4. Negotiation-ending Message	13
7.5. Confirm-waiting Message	13
8. CDNP General Options	13
8.1. Format of CDNP Options	13
8.2. Divert Option	14
8.3. Accept Option	15
8.4. Decline Option	15
8.5. Waiting Time Option	16
8.6. Certificate Option	17
8.7. Signature Option	17
8.8. Locator Options	18
8.8.1. Locator IPv4 address option	19
8.8.2. Locator IPv6 address option	19
8.8.3. Locator FQDN option	19
9. Objective Options and Considerations	20
9.1. Organizing of CDNP Options	20
9.2. Vendor Specific Options	21
9.3. Experimental Options	21
10. Items for Future Work	21
11. Security Considerations	22
12. IANA Considerations	22
13. Acknowledgements	24
14. Change log [RFC Editor: Please remove]	24
15. References	24

15.1. Normative References	25
15.2. Informative References	25
Authors' Addresses	26

1. Introduction

The success of the Internet has made IP-based networks bigger and more complicated. Large-scale ISP networks have become more and more problematic for human based management. Also operational costs are growing quickly. Consequently, there are therefore increased requirements for autonomy in the networks. General aspects of autonomic networks are discussed in [I-D.irtf-nmrg-autonomic-network-definitions] and [I-D.irtf-nmrg-an-gap-analysis]. In order to fulfil autonomy, devices that are more intelligent need to be able to negotiate directly with each other. [I-D.jiang-config-negotiation-ps] describes the requirements and application scenarios for network device negotiation. It also describes a behavior model of a generic negotiation protocol. Prior to negotiation, devices must discover each other. The design of Configuration Discovery and Negotiation Protocol (CDNP) in this document is mainly based on this behavior model.

Although many negotiations may happen between distributed horizontal peers, the main target scenarios are still hierarchical networks, which is the major structure of current large-scale networks. Thus, where necessary, we assume that each network element has a hierarchical superior. Of course, the protocol itself is capable of being used in a small and/or flat network structure such as a small office or home network, too.

This document defines a generic discovery and negotiation protocol, named Configuration Discovery and Negotiation Protocol (CDNP), that can be used to control decision process among distributed devices or between networks. The newly defined CDNP in this document adapts a tight certificate-based mechanism, which needs a Public Key Infrastructure (PKI, [RFC5280]) system. The PKI may be managed by an operator or be autonomic. The document also introduces a new discovery mechanism, which is based on a neighbor learning process and is oriented towards negotiation objectives.

It is understood that in realistic deployments, not all devices will support CDNP. Such mixed scenarios are not discussed in this specification.

2. Requirements Language and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

- o Negotiation Objective: specific negotiation content, which needs to be decided in coordination with another network device. It is naturally based on a specific service or function or action. It could be a logical, numeric, or string value or a more complex data structure.
- o Negotiation Initiator: a device that spontaneously starts discovery or negotiation by sending a request message referring to a specific negotiation objective.
- o Negotiation Counterpart: a peer device with which the Negotiation Initiator negotiates a specific negotiation objective.
- o Device Identifier: a public key, which identifies the device in CDNP messages. It is assumed that its associated private key is maintained in the device only.
- o Device Certificate: A certificate for a single device, also the identifier of the device, further described in Section 5.
- o Device Certificate Tag: a tag, which is bound to the device identifier. It is used to present Device Certificate in short form.

3. CDNP Protocol Overview

The Configuration Discovery and Negotiation protocol is designed to be a generic platform, which is independent from the negotiation contents. It only takes care of the general intercommunication between negotiation counterparts. The negotiation contents vary, giving the various negotiation objectives and the different pairs of negotiating counterparts. CDNP runs over UDP.

The CDNP has been designed based on simple initiator/responder model, while multiple-party negotiations could be completed by indirect steps. The initiator requests a particular objective and the counterpart responds accordingly.

3.1. IP Version Independent

To be a generic platform, CDNP should be IP version independent. In other words, it should be able to run over IPv6 and IPv4. Its messages and general options are neutral with respect to the IP version.

However, some functions, such as multicasting or broadcasting on a link, might need to be IP version dependent. For these parts, the document defines support for both IP versions separately.

3.2. Objective Oriented Discovery Mechanism

Typically, one network device has multiple functions. It may be involved in different negotiation processes for different negotiation objectives. Therefore, the traditional topology-oriented device discovery mechanisms are not sufficient for CDNP. A new discovery mechanism is needed to find negotiation counterparts based on a specific negotiation objective. As a result, an objective-based discovery mechanism is described in this document.

For every new negotiation objective, the negotiation initiator needs to start a new discovery process in order to find the proper negotiation counterpart. Because a listening CDNP-enabled device has to know the requested negotiation objective to decide whether it is a proper negotiation counterpart and make a response, the discovery process needs to be tightly coupled with the request process. Therefore, in this document, the discovery process is merged into the request process. There is no need for an independent discovery message and process.

3.3. Neighbor Diverting Discovery Mechanism

We now discuss the general flow of Request, Negotiation, and Negotiation-Ending messages, and Accept, Decline and Divert options. Details of the options are given later.

Discovery starts as on-link operation. However, negotiation may continue either on-link or off-link. The Divert option can tell the negotiation initiator to contact an off-link counterpart.

Every Request message is sent by a negotiation initiator to the ALL_CDNP_NEIGHBOR multicast address (Section 4).

If the neighbor device is a proper negotiation counterpart, it MAY respond with a Negotiation message to start a negotiation process, or with a Negotiation-Ending message in the case of a clear Accept or Decline.

If the neighbor device is not a proper negotiation counterpart for the objective given in the Request message, but knows a proper negotiation counterpart, for example because it negotiated the same objective with that other negotiation counterpart before, it SHOULD respond with a Negotiation-Ending message with a Divert option pointed to the proper negotiation counterpart. If the neighbor device is not a proper negotiation counterpart, but does not know a proper negotiation counterpart, it SHOULD respond with a Negotiation-Ending message with a Divert option pointed to its hierarchical upstream device.

After a CDNP device successfully negotiated a specific objective with a negotiation counterpart, it SHOULD record this negotiation counterpart with this objective type locally. This record may be used for future negotiation or to pass to another neighbor as a Divert option. This learning mechanism should be able to support most network establishment scenarios.

3.4. Certificate-based Security Mechanism

A certification based security mechanism provides security properties for CDNP:

- o the identity of a CDNP message sender can be verified by a recipient.
- o the integrity of CDNP message can be checked by the recipient of the message.
- o anti-replay protection on the CDNP message recipient.

The authority of the CDNP message sender depends on a Public Key Infrastructure (PKI) system with a Certification Authority (CA), which should normally be run by the network operator. In the case of a network with no operator, such as a small office or home network, the PKI itself needs to be established by an autonomic process, which is out of scope for this specification.

A Request message MUST carry a Certificate option, defined in Section 8.6. The first Negotiation Message, responding to a Request message, SHOULD also carry a Certificate option. Using these messages, recipients build their certificate stores, indexed by the Device Certificate Tags included in every CDNP message. This process is described in more detail below.

Every message MUST carry a signature option, defined in Section 8.7.

For now, the authors do not think packet size is a problem. In this CDNP specification, there SHOULD NOT be multiple certificates in a single message. The current most used public keys are 1024/2048 bits, some may reach 4096. With overhead included, a single certificate is less than 500 bytes. Messages should be far shorter than the normal packet MTU within a modern network.

3.4.1. Support for algorithm agility

Hash functions are used to provide message integrity checks. In order to provide a means of addressing problems that may emerge in the future with existing hash algorithms, as recommended in [RFC4270], a mechanism for negotiating the use of more secure hashes in the future is provided.

In addition to hash algorithm agility, a mechanism for signature algorithm agility is also provided.

The support for algorithm agility in this document is mainly a unilateral notification mechanism from sender to recipient. If the recipient does not support the algorithm used by the sender, it cannot authenticate the message. Senders in a single administrative domain are not required to upgrade to a new algorithm simultaneously.

So far, the algorithm agility is supported by one-way notification, rather than negotiation mode. As defined in Section 8.7, the sender notifies the recipient what hash/signature algorithms it uses. If the responder doesn't know a new algorithm used by the sender, the negotiation request would fail. In order to establish a negotiation session, the sender MAY fall back to an older, less preferred algorithm. To avoid downgrade attacks it MUST NOT fall back to an algorithm considered weak.

3.4.2. Message validation on reception

When receiving a CDNP message, a recipient MUST discard the CDNP message if the Signature option is absent, or the Certificate option is in a Request Message.

For the Request message and the Response message with a Certification Option, the recipient MUST first check the authority of this sender following the rules defined in [RFC5280]. After successful authority validation, an implementation MUST add the sender's certification into the local trust certificate record indexed by the associated Device Certificate Tag, defined in Section 5.

The recipient MUST now authenticate the sender by verifying the Signature and checking a timestamp, as specified in Section 3.4.3.

The order of two procedures is left as an implementation decision. It is RECOMMENDED to check timestamp first, because signature verification is much more computationally expensive.

The signature field verification MUST show that the signature has been calculated as specified in Section 8.7. The public key used for signature validation is obtained from the certificate either carried by the message or found from a local trust certificate record by searching the message-carried Device Certificate Tag.

Only the messages that get through both the signature verifications and timestamp check are accepted and continue to be handled for their contained CDNP options. Messages that do not pass the above tests MUST be discarded as insecure messages.

3.4.3. TimeStamp checking

Recipients SHOULD be configured with an allowed timestamp Delta value, a "fuzz factor" for comparisons, and an allowed clock drift parameter. The recommended default value for the allowed Delta is 300 seconds (5 minutes); for fuzz factor 1 second; and for clock drift, 0.01 second.

The timestamp is defined in the Signature Option, Section 8.7. To facilitate timestamp checking, each recipient SHOULD store the following information for each sender:

- o The receive time of the last received and accepted CDNP message. This is called RDlast.
- o The time stamp in the last received and accepted CDNP message. This is called TSlast.

An accepted CDNP message is any successfully verified (for both timestamp check and signature verification) CDNP message from the given peer. It initiates the update of the above variables. Recipients MUST then check the Timestamp field as follows:

- o When a message is received from a new peer (i.e., one that is not stored in the cache), the received timestamp, TSnew, is checked, and the message is accepted if the timestamp is recent enough to the reception time of the packet, RDnew:

$$-\text{Delta} < (\text{RDnew} - \text{TSnew}) < +\text{Delta}$$

The RDnew and TSnew values SHOULD be stored in the cache as RDlast and TSlast.

- o When a message is received from a known peer (i.e., one that already has an entry in the cache), the timestamp is checked against the previously received CDNP message:

$$TS_{new} + fuzz > TS_{last} + (RD_{new} - RD_{last}) \times (1 - drift) - fuzz$$

If this inequality does not hold, the recipient SHOULD silently discard the message. If, on the other hand, the inequality holds, the recipient SHOULD process the message.

Moreover, if the above inequality holds and $TS_{new} > TS_{last}$, the recipient SHOULD update RD_{last} and TS_{last} . Otherwise, the recipient MUST NOT update RD_{last} or TS_{last} .

An implementation MAY use some mechanism such as a timestamp cache to strengthen resistance to replay attacks. When there is a very large number of nodes on the same link, or when a cache filling attack is in progress, it is possible that the cache holding the most recent timestamp per sender will become full. In this case, the node MUST remove some entries from the cache or refuse some new requested entries. The specific policy as to which entries are preferred over others is left as an implementation decision.

3.5. Negotiation Procedures

A negotiation initiator sends a negotiation request to discovered negotiation counterpart devices, which may be different according to different negotiation objectives. It may request relevant information from the negotiation counterpart so that it can decide its local configuration to give the most coordinated performance. It may request the negotiation counterpart to make a matching configuration in order to set up a successful communication with it. It may request certain simulation or forecast result by sending some dry run conditions. The details will be defined separately for each type of negotiation objective.

If the counterpart can immediately apply the requested configuration, it will give a positive (yes) answer. This will normally end the negotiation phase immediately. Otherwise it will give a negative (no) answer. Normally, this will not end the negotiation phase.

In the negative (no) case, the negotiation counterpart should be able to reply with a proposed alternative configuration that it can apply (typically, a configuration that uses fewer resources than requested by the negotiation initiator). This will start a bi-directional negotiation to reach a compromise between the two network devices.

The negotiation procedure is ended when one of the negotiation peers sends a Negotiation Ending message, which contains an accept or decline option and does not need a response from the negotiation peer.

A negotiation procedure concerns one objective and one counterpart. Both the initiator and the counterpart may take part in simultaneous negotiations with various other devices, or in simultaneous negotiations about different objectives. Thus, CDNP is expected to be used in a multi-threaded mode. Certain negotiation objectives may have restrictions on multi-threading, for example to avoid over-allocating resources.

4. CDNP Constants

o ALL_CDNP_NEIGHBOR (TBD1)

A link-local scope multicast address used by a CDNP-enabled router to discover CDNP-enabled neighbor (i.e., on-link) devices. All routers that support CDNP are members of this multicast group.

* IPv6 multicast address: TBD1

* IPv4 multicast address: TBD2

o CDNP Listen Port (TBD3)

A UDP port that every CDNP-enabled network device always listens to.

5. Device Identifier and Certificate Tag

A CDNP-enabled Device MUST generate a stable public/private key pair before it participates in CDNP. There MUST NOT be any way of accessing the private key via the network or an operator interface. The device then uses the public key as its identifier, which is cryptographic in nature. It is a CDNP unique identifier for a CDNP participant.

It then gets a certificate for this public key, signed by a Certificate Authority that is trusted by other network devices. The Certificate Authority SHOULD be managed by the network administrator, to avoid needing to trust a third party. The signed certificate would be used for authentication of the message sender. In a managed network, this certification process could be performed at a central location before the device is physically installed at its intended location. In an unmanaged network, this process must be autonomic, including the bootstrap phase.

A 128-bit Device Certificate Tag, which is generated by taking a cryptographic hash over the device certificate, is a short presentation for CDNP messages. It is the index key to find the device certificate in a recipient's local trusted certificate record.

The tag value is formed by taking a SHA-1 hash algorithm over the corresponding device certificate and taking the leftmost 128 bits of the hash result.

6. Session Identifier

A 24-bit opaque value used to distinguish multiple sessions between the same two devices. A new Session ID SHOULD be generated for every new Request message. All followup messages in the same negotiation procedure, which is initiated by the request message, SHOULD carry the same Session ID.

The Session ID SHOULD have a very low collision rate locally. It is RECOMMENDED to be generated by a pseudo-random algorithm using a seed which is unlikely to be used by any other device in the same network.

7. CDNP Messages

This document defines the following CDNP message format and types. Message types not listed here are reserved for future use. The numeric encoding for each message type is shown in parentheses.

7.1. CDNP Message Format

All CDNP messages share an identical fixed format header and a variable format area for options. Every Message carries the Device Certificate Tag of its sender and a Session ID. Options are presented serially in the options field, with no padding between the options. Options are byte-aligned.

The following diagram illustrates the format of CDNP messages:



MESSAGE_TYPE Identifies the CDNP message type. 8-bit.

Session ID Identifies this negotiation session, as defined in Section 6. 24-bit.

Device Certificate Tag
Present the Device Certificate, which identifies the negotiation deviceas, as defined in Section 5. The Device Certificate Tag is 128 bit, also defined in Section 5. It is used as index key to find the device certificate.

Options CDNP Options carried in this message. Options are defined in Section 8.

7.2. Request Message

REQUEST (1) A negotiation requesting node sends a REQUEST message to initiate a negotiation.

If the requesting node does not know any negotiation counterpart, it sends the REQUEST messages to the link-local ALL_CDNP_NEIGHBOR multicast address.

If the requesting node re-contacts a known negotiation counterpart, it sends the REQUEST message to the unicast address of the negotiation counterpart directly.

7.3. Negotiation Message

NEGOTIATION (2) A negotiation counterpart sends an NEGOTIATION message in response to a REQUEST message or a Negotiation message in a negotiation process which may need multiple steps.

7.4. Negotiation-ending Message

NEGOTIATION-ENDING (3)
A negotiation counterpart sends an NEGOTIATION-ENDING message to close the negotiation. It MUST contain one, but only one of accept/decline/divert option, defined in Section 8. It could be sent either by the requesting node or the responding node.

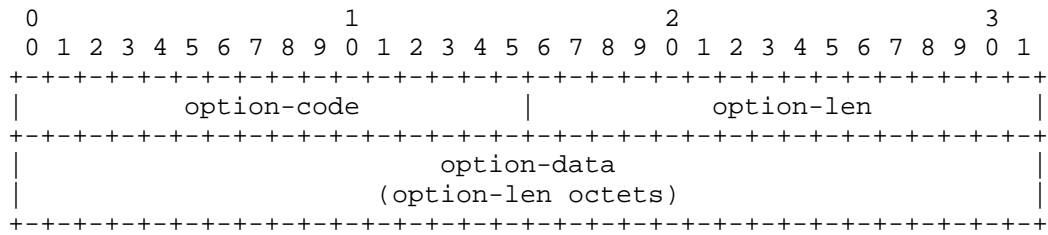
7.5. Confirm-waiting Message

CONFIRM-WAITING (4)
A responding node sends a CONFIRM-WAITING message to indicate the requesting node to wait for a further negotiation response. It might be that the local process needs more time or that the negotiation depends on another triggered negotiation. This message MUST NOT include any other options than the WAITING option defined in Section 8.5.

8. CDNP General Options

This section defines the CDNP general option for the negotiation protocol signalling. Option type 10~64 is reserved for CDNP general options defined in the future.

8.1. Format of CDNP Options



Option-code An unsigned integer identifying the specific option type carried in this option.

Option-len An unsigned integer giving the length of the option-data field in this option in octets.

Option-data The data for the option; the format of this data depends on the definition of the option.

CDNP options are scoped by using encapsulation. If an option contains other options, the outer Option-len includes the total size of the encapsulated options, and the latter apply only to the outer option.

8.2. Divert Option

The divert option is used to redirect a CDNP request to another node, which may be more appropriate for the intended negotiation. It may redirect to an entity that is known as a specific negotiation counterpart or a default gateway or a hierarchically upstream devices. The divert option MUST only be encapsulated in Negotiation-ending messages. If found elsewhere it SHOULD be silently ignored.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|               OPTION_DIVERT               |               option-len               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Locator Option (s) of Diversion Device(s)               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               .               |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Option-code OPTION_DIVERT (1).

Option-len The total length of diverted destination
 sub-option(s) in octets.

Locator Option (s) of Diverted Device
 Embedded Locator Option(s), defined in Section 8.8,
 that point to diverted destination device(s).

8.3. Accept Option

The accept option is used to indicate the negotiation counterpart that the proposed negotiation content is accepted.

The accept option MUST only be encapsulated in Negotiation-ending messages. If found elsewhere it SHOULD be silently ignored.

```

      1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|               OPTION_ACCEPT               |               option-len               |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Option-code OPTION_ACCEPT (2).

Option-len 0.

8.4. Decline Option

The decline option is used to indicate the negotiation counterpart the proposed negotiation content is declined and end the negotiation process.

The decline option MUST only be encapsulated in Negotiation-ending messages. If found elsewhere it SHOULD be silently ignored.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|               OPTION_DECLINE               |               option-len               |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Option-code OPTION_DECLINE (3).

Option-len 0.

Notes: there are scenarios where a negotiation counterpart wants to decline the proposed negotiation content and continue the negotiation process. For these scenarios, the negotiation counterpart SHOULD use a Response message, with either an objective option that contains at least one data field with all bits set to 1 to indicate a meaningless initial value, or a specific objective option that provides further conditions for convergence.

8.5. Waiting Time Option

The waiting time option is used to indicate that the negotiation counterpart needs to wait for a further negotiation response, since the processing might need more time than usual or it might depend on another triggered negotiation.

The waiting time option MUST only be encapsulated in Confirm-waiting messages. If found elsewhere it SHOULD be silently ignored.

The counterpart SHOULD send a Response message or another Confirm-waiting message before the current waiting time expires. If not, the initiator SHOULD abandon or restart the negotiation procedure, to avoid an indefinite wait.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|               OPTION_WAITING               |               option-len               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Time               |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

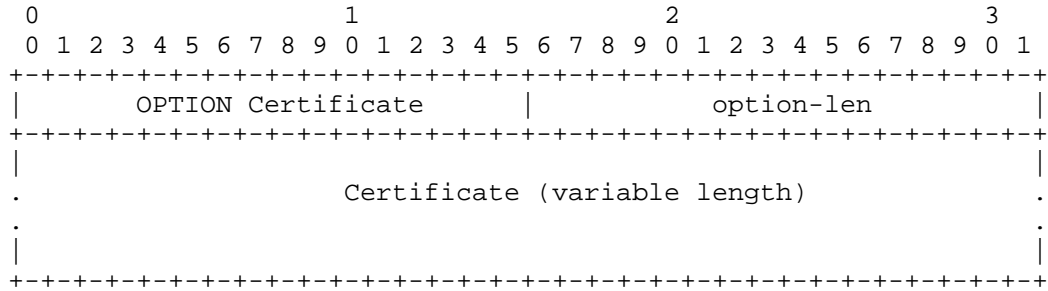
Option-code OPTION_WAITING (4).

Option-len 4, in octets.

Time The time is counted in millisecond as a unit.

8.6. Certificate Option

The Certificate option carries the certificate of the sender. The format of the Certificate option is as follows:



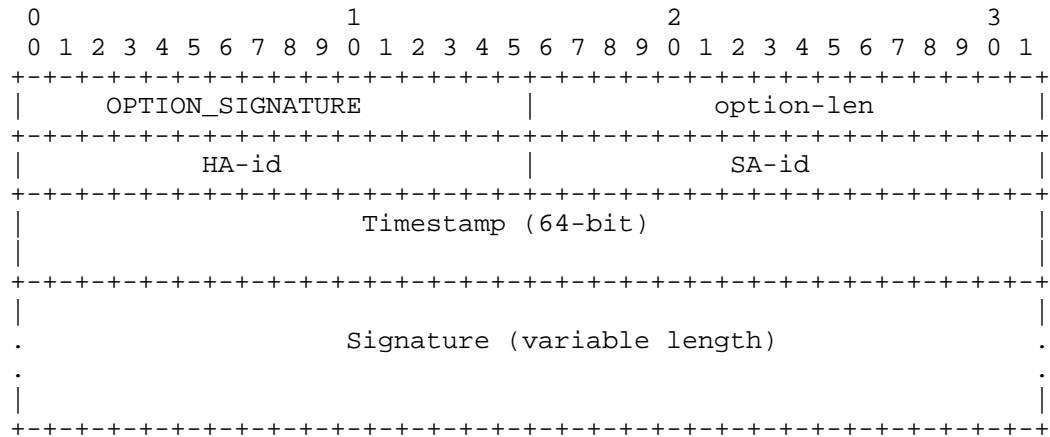
Option-code OPTION_CERT_PARAMETER (5)

Option-len Length of certificate in octets

Public key A variable-length field containing a certificate

8.7. Signature Option

The Signature option allows public key-based signatures to be attached to a CDNP message. The Signature option is REQUIRED in every CDNP message and could be any place within the CDNP message. It protects the entire CDNP header and options. A TimeStamp has been integrated in the Signature Option for anti-replay protection. The format of the Signature option is described as follows:



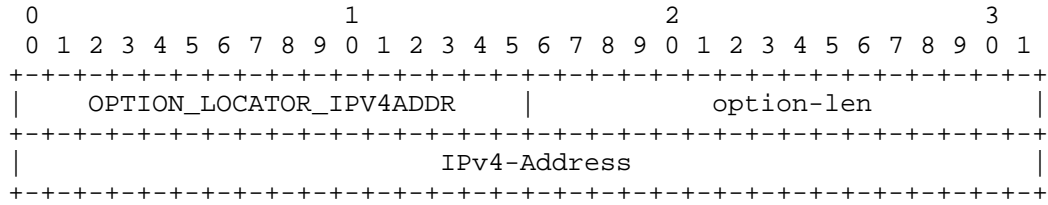
Option-code OPTION_SIGNATURE (6)

Option-len	12 + Length of Signature field in octets.
HA-id	Hash Algorithm id. The hash algorithm is used for computing the signature result. This design is adopted in order to provide hash algorithm agility. The value is from the Hash Algorithm for CDNP registry in IANA. The initial value assigned for SHA-1 is 0x0001.
SA-id	Signature Algorithm id. The signature algorithm is used for computing the signature result. This design is adopted in order to provide signature algorithm agility. The value is from the Signature Algorithm for CDNP registry in IANA. The initial value assigned for RSASSA-PKCS1-v1_5 is 0x0001.
Timestamp	The current time of day (NTP-format timestamp [RFC5905] in UTC (Coordinated Universal Time), a 64-bit unsigned fixed-point number, in seconds relative to 0h on 1 January 1900.). It can reduce the danger of replay attacks.
Signature	<p>A variable-length field containing a digital signature. The signature value is computed with the hash algorithm and the signature algorithm, as described in HA-id and SA-id. The signature constructed by using the sender's private key protects the following sequence of octets:</p> <ol style="list-style-type: none"> 1. The CDNP message header. 2. All CDNP options including the Signature option (fill the signature field with zeroes). <p>The signature field MUST be padded, with all 0, to the next 16 bit boundary if its size is not an even multiple of 8 bits. The padding length depends on the signature algorithm, which is indicated in the SA-id field.</p>

8.8. Locator Options

These locator options are used to present a device's or interface's reachability information. They are Locator IPv4 Address Option, Locator IPv6 Address Option and Locator FQDN (Fully Qualified Domain Name) Option.

8.8.1. Locator IPv4 address option

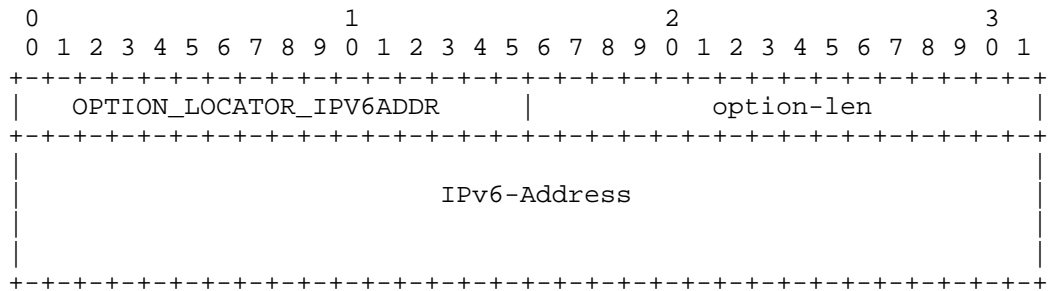


Option-code OPTION_LOCATOR_IPV4ADDR (7)

Option-len 4, in octets.

IPv4-Address The IPv4 address locator of the device/interface.

8.8.2. Locator IPv6 address option



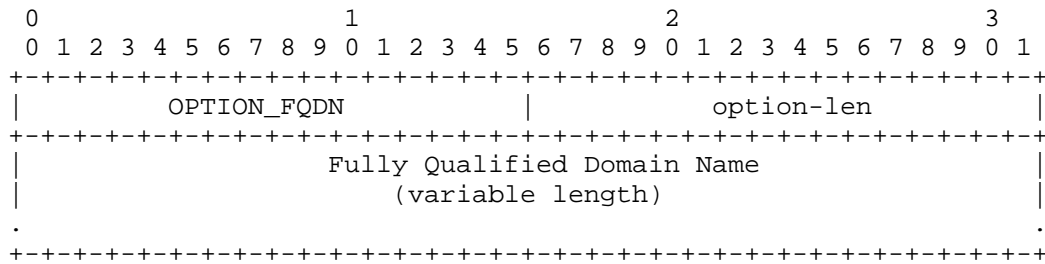
Option-code OPTION_LOCATOR_IPV6ADDR (8).

Option-len 16, in octets.

IPv6-Address The IPv6 address locator of the device/interface.

Note: link-local IPv6 address SHOULD be avoided when this option is used in the Divert option. It may create a connection problem.

8.8.3. Locator FQDN option



Option-code OPTION_FQDN (9).

Option-len Length of Fully Qualified Domain Name in octets.

Domain-Name The Fully Qualified Domain Name of the entity.

9. Objective Options and Considerations

The Objective options contains negotiation objectives, which are various according to different functions/services. They MUST be carried by Request or Negotiation Messages only. Objective options SHOULD be assigned an option type greater than 64 in the CDNP option table.

For most scenarios, there SHOULD be initial values in the negotiation requests. Consequently, the Objective options SHOULD always be completely presented in a Request message. If there is no initial value, the bits in the value field SHOULD all be set to 1 to indicate a meaningless value, unless this is inappropriate for the specific negotiation objective.

9.1. Organizing of CDNP Options

Naturally, a negotiation objective, which is based on a specific service or function or action, SHOULD be organized as a single CDNP option. It is NOT RECOMMENDED to organize multiple negotiation objectives into a single option.

A negotiation objective may have multiple parameters. Parameters can be categorized into two class: the obligatory ones presented as fixed fields; and the optional ones presented in TLV sub-options. It is NOT RECOMMENDED to split parameters in a single objective into multiple options, unless they have different response periods. An exception scenario may also be described by split objectives.

9.2. Vendor Specific Options

Option codes 128~159 have been reserved for vendor specific options. Multiple option codes have been assigned because a single vendor may use multiple options simultaneously. These vendor specific options are highly likely to have different meanings when used by different vendors. Therefore, they SHOULD NOT be used without an explicit human decision. They are not suitable for unmanaged networks such as home networks.

9.3. Experimental Options

Option code 176~191 have been reserved for experimental options. Multiple option codes have been assigned because a single experiment may use multiple options simultaneously. These experimental options are highly likely to have different meanings when used for different experiments. Therefore, they SHOULD NOT be used without an explicit human decision. They are not suitable for unmanaged networks such as home networks.

10. Items for Future Work

There are a few open design questions that are worthy of more work in the near future, as listed below:

- o UDP vs TCP: For now, this specification has chosen UDP as message transport mechanism. However, this is not closed yet. UDP is good for short conversations, fitting the divert scenarios well. However, it may have issues with large packets. TCP is good for stable and long sessions, with a little bit of time consumption during the session establishment stage. If messages exceed a reasonable MTU, a TCP mode may be necessary.
- o Message encryption: should CDNP messages be encrypted as well as signed, to protect against internal eavesdropping within the network?
- o TLS or DTLS vs built-in security mechanism. For now, this specification has chosen a PKI based build-in security mechanism. However, TLS or DTLS might be chosen as security infrastructure for simplification reasons.
- o Timeout for lost Negotiation Ending and other messages to be added.
- o CDNP currently requires every participant to have an NTP-synchronized clock. Is this OK for low-end devices?

- o Would use of MDNS have any impact on the Locator FQDN option?
- o Use case. A use case may help readers to understand the applicability of this specification. However, the authors have not yet decided whether to have a separate document or have it in this document. General uses cases for AN have been developed, but they are not specific enough for this purpose.
- o Rules about how data items are defined in a negotiation objective. Maybe a formal information model is needed.
- o We currently assume that there is only one counterpart for each discovery action. If this is false or one negotiation request receives multiple different responses, how does the initiator choose between them? Could it split them into multiple follow-up negotiations?
- o Alternatives to TLV format. It may be useful to provide a generic method of carrying negotiation objectives in a high-level format such as YANG or an XML schema. It may also be useful to provide a generic method of carrying existing configuration information such as DHCP(v6) or IPv6 RA messages. These features could be provided by encapsulating such messages in their own TLVs.

11. Security Considerations

Using certificate-based security mechanism and its verification mechanism in CDNP message exchanging provides the authentication and data integrity protection. The timestamp mechanism provides an anti-replay function.

Since CDNP is intended to be deployed in a single administrative domain recommended to operate its own CA, there is no need for a trusted third party.

12. IANA Considerations

Section 4 defines the following mltwmulticast addresses, which have been assigned by IANA for use by CDNP:

ALL_CDNP_NEIGHBOR multicast address (IPv6): (TBD1)

ALL_CDNP_NEIGHBOR multicast address (IPv4): (TBD2)

Section 4 defines the following UDP port, which have been assigned by IANA for use by CDNP:

CDNP Listen Port: (TBD3)

This document defined a new Configuration Discovery and Negotiation Protocol. The IANA is requested to create a new CDNP registry. The IANA is also requested to add two new registry tables to the newly-created CDNP registry. The two tables are the CDNP Messages table and CDNP Options table.

Initial values for these registries are given below. Future assignments are to be made through Standards Action or Specification Required [RFC5226]. Assignments for each registry consist of a type code value, a name and a document where the usage is defined.

CDNP Messages table. The values in this table are 16-bit unsigned integers. The following initial values are assigned in Section 7 in this document:

Type	Name	RFCs
0	Reserved	this document
1	Request Message	this document
2	Negotiation Message	this document
3	Negotiation-end Message	this document
4	Confirm-waiting Message	this document

CDNP Options table. The values in this table are 16-bit unsigned integers. The following initial values are assigned in Section 8 and Section 9 in this document:

Type	Name	RFCs
0	Reserved	this document
1	Divert Option	this document
2	Accept Option	this document
3	Decline Option	this document
4	Waiting Time Option	this document
5	Certificate Option	this document
6	Signature Option	this document
7	Device IPv4 Address Option	this document
8	Device IPv6 Address Option	this document
9	Device FQDN Option	this document
10~64	Reserved for future CDNP General Options	this document
128~159	Vendor Specific Options	this document
176~191	Experimental Options	this document

The IANA is also requested to create two new registry tables in the CDNP Parameters registry. The two tables are the Hash Algorithm for CDNP table and the Signature Algorithm for CDNP table.

Initial values for these registries are given below. Future assignments are to be made through Standards Action or Specification Required [RFC5226]. Assignments for each registry consist of a name, a value and a document where the algorithm is defined.

Hash Algorithm for CDNP. The values in this table are 16-bit unsigned integers. The following initial values are assigned for Hash Algorithm for CDNP in this document:

Name	Value	RFCs
Reserved	0x0000	this document
SHA-1	0x0001	this document
SHA-256	0x0002	this document

Signature Algorithm for CDNP. The values in this table are 16-bit unsigned integers. The following initial values are assigned for Signature Algorithm for CDNP in this document:

Name	Value	RFCs
Reserved	0x0000	this document
RSASSA-PKCS1-v1_5	0x0001	this document

13. Acknowledgements

Valuable comments were received from Zhenbin Li and Dacheng Zhang, and other participants in the xxx working group.

This document was produced using the xml2rfc tool [RFC2629].

14. Change log [RFC Editor: Please remove]

draft-jiang-config-negotiation-protocol-02: adapted scope to include discovery, multiple threads, mentioned YANG etc. encapsulation, 2013-06-26.

draft-jiang-config-negotiation-protocol-01: corrections and additions, 2014-04-21.

draft-jiang-config-negotiation-protocol-00: original version, 2013-10-19.

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

15.2. Informative References

- [I-D.irtf-nmrg-an-gap-analysis]
Behringer, M., Carpenter, B., and S. Jiang, "Gap Analysis for Autonomic Networking", draft-irtf-nmrg-an-gap-analysis-00 (work in progress), April 2014.
- [I-D.irtf-nmrg-autonomic-network-definitions]
Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking - Definitions and Design Goals", draft-irtf-nmrg-autonomic-network-definitions-00 (work in progress), December 2013.
- [I-D.jiang-config-negotiation-ps]
Jiang, S., Yin, Y., and B. Carpenter, "Network Configuration Negotiation Problem Statement and Requirements", draft-jiang-config-negotiation-ps-03 (work in progress), May 2014.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC4270] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", RFC 4270, November 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

Authors' Addresses

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: jiangsheng@huawei.com

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Bing Liu
Huawei Technologies Co., Ltd
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: leo.liubing@huawei.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: November 20, 2014

S. Jiang
Y. Yin
Huawei Technologies Co., Ltd
B. Carpenter
Univ. of Auckland
May 19, 2014

Network Configuration Negotiation Problem Statement and Requirements
draft-jiang-config-negotiation-ps-03

Abstract

This document describes a problem statement and general requirements for distributed autonomic configuration of multiple aspects of networks, in particular carrier networks. The basic model is that network elements need to negotiate configuration settings with each other to meet overall goals. The document describes a generic negotiation behavior model. The document also reviews whether existing management and configuration protocols may be suitable for autonomic networks.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 20, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements and Application Scenarios for Network Devices Negotiation	3
2.1. Negotiation between downstream and upstream network devices	4
2.2. Negotiation between peer network devices	5
2.3. Negotiation between networks	5
2.4. Information and status queries among devices	6
2.5. Unavoidable configuration	6
3. Existing protocols	6
4. A Behavior Model of a Generic Negotiation Protocol	8
5. Security Considerations	12
6. IANA Considerations	13
7. Acknowledgements	13
8. Change Log [RFC Editor please remove]	13
9. Informative References	13
Authors' Addresses	15

1. Introduction

The success of IP and the Internet has made the network model very complicated, and networks have become larger and larger. The network of a large ISP typically contains more than a hundred thousand network devices which play many roles. The initial setup configuration, dynamic management and maintenance, troubleshooting and recovery of these devices have become a huge outlay for network operators. Particularly, these devices are managed by many different staff requiring very detailed training and skills. The coordination of these staff is also difficult and often inefficient. There are therefore increased requirements for autonomy in the networks. [I-D.boucadair-network-automation-requirements] is one of the attempts to describe such requirements. It listed a "requirement for a protocol to convey configuration information towards the managed entities". However, this document is going further by requiring a configuration negotiation protocol rather than only unidirectional provisioning.

Autonomic operation means network devices could decide configurations by themselves. More background on autonomic networking is given in [I-D.irtf-nmrg-autonomic-network-definitions] and

[I-D.irtf-nmrg-an-gap-analysis]. There are already many existing internal implementations or algorithms for a network device to decide or compute its configuration according to its own status, often referred to as device intelligence. In one particular area, routing protocols, distributed autonomic configuration is a well established mechanism. The question is how to extend autonomy to cover all kinds of configuration, not just routing tables.

However, in order to make right or good decisions, the network devices need to know more information than just routes from the relevant or neighbor devices. There are dependencies between such information and configurations. Currently, most of these configurations currently require centralised manual coordination. The basic model for this document is that in an autonomic network, devices will need to negotiate directly with one another to provide this coordination.

Today, there is no generic negotiation protocol that can be used to control decision processes among distributed devices or between networks. Proprietary network management systems are widely used but tend to be hierarchical systems ultimately relying on a console operator and a central database. An autonomic system needs to be less hierarchical and with less dependence on an operator. This requires network elements to negotiate directly with each other, with an absolute minimum or zero configuration data at the installation stage.

This document analyzes the requirements for a generic negotiation protocol in view of various application use cases, then gives considerations for detailed technical requirements for designing such a protocol. Some existing protocols are also reviewed as part of the analysis. A protocol behavior model, which may be used to define such a negotiation protocol, is also described.

Note in draft: the requirements analysis will need to be reviewed and completed after a number of use cases for autonomic networking have been documented.

2. Requirements and Application Scenarios for Network Devices Negotiation

Routing protocols are a typical autonomic model based on distributed devices. But routing is mainly one-way information announcement (in both directions), rather than bi-directional negotiation. Its only focus is reachability. The future networks need to be able to manage many more dimensions of the network, such as power saving, load balancing, etc. The current routing protocols only show simple link status, as up or down. More information, such as latency,

congestion, capacity, and particularly available throughput, is very helpful to get better path selection and utilization rate.

A negotiation model with no human intervention is needed when the coordination of multiple devices can provide better overall network performance.

A negotiation model provides a possibility for forecasting. A "dry run" becomes possible before the concrete configuration takes place.

Another area is tunnel management, with automatic setup, maintenance, and removal. A related area is ad hoc routes, without encapsulation, to handle specific traffic flows (which might be regarded as a form of software defined networking).

Negotiation of security mechanisms, for example to determine the strongest possible protection for a given link, is another example.

When a new user or device comes online, it might be necessary to set up resources on multiple relevant devices, coordinated and matched to each other so that there is no wasted resource. Security settings might also be needed to allow for the new user/device.

Status information and traffic metrics need to be shared between nodes for dynamic adjustment of resources.

Troubleshooting should be as automatic as possible. Although it is far from trivial, there is a need to detect the "real" breakdown amongst many alerts, and then take action to reconfigure the relevant devices. Again, routing protocols have done this for many years, but in an autonomic network it is not just routing that needs to reconfigure itself after a failure.

2.1. Negotiation between downstream and upstream network devices

The typical scenario is that there is a new access gateway, which could be a wireless base station, WiFi hot spot, Data Center switch, VPN site switch, enterprise CE, home gateway, etc. When it is plugged into the network, bi-direction configuration/control is needed. The upstream network needs to configure the device, its delegated prefix(es), DNS server, etc. For this direction, DHCP might be suitable and sufficient. However, there is another direction: the connection of downstream devices also needs to trigger the upstream devices, for example the provider edge, to create a corresponding configuration, by setting up a new tunnel, service, authentication, etc.

Furthermore, after the communication between gateway and provider has been established, the devices would like to optimize their configurations interactively according to dynamic link status or performance measurements, power consumption, etc. For dynamic management and maintenance, there are many other network events that downstream network devices may need to report to upstream network devices and then initiate some configuration change on these upstream devices. Currently, these kinds of synchronizing operations require the involvement of human operators.

Similar requirements can also appear between other types of downstream and upstream network devices.

2.2. Negotiation between peer network devices

Within a large network, in many segments, there are network devices that are in equivalent positions. They have a peer rather than hierarchical relationship. There may be many horizontal traffic flows or tunnels between them. In order to make these connections efficient, their configurations (for example, quality of service parameters) have to match each other. Any change of a device's configuration may require synchronizing with its peer network devices.

However, in many cases the peer network devices may not be able to make the exact changes as requested. Instead, another slightly different change may be the best choice for optimal performance. In order to decide on this best choice, multiple rounds of information exchange between peers may be necessary. This should be done without requiring the involvement of human operators. To provide this ability, a mechanism for network devices to be able to negotiate with each other is needed.

2.3. Negotiation between networks

A network may announce some information about its internal capabilities to connected peer networks, so that the peer networks can react accordingly. BGP routing information is a simple example.

Beyond reachability, more information may enable better coordination among networks. Examples include traffic engineering among multiple connections between two networks, particularly when these connections are geographically distributed; dynamic capacity adjustment to match changing traffic from a peer network; dynamic establishment and adjustment of differentiated service classes to support Service Level Agreements; and so on.

2.4. Information and status queries among devices

In distributed routers, many data such as status indicators or traffic measurements are dynamically changing. These may be the triggers for subsequent negotiation. For example, assume there are two routers A and B sharing traffic load. Router A may request the traffic situation of router B, then start negotiation, such as requesting router B to handle all the traffic so that router A can enter power-saving mode. Another example is that a device may request its neighbor to send a forecast or dry-run result based on a given potential configuration change. Then, the initiating router can evaluate whether the potential configuration change would meet its original target.

2.5. Unavoidable configuration

Even with autonomic negotiation, some initial configuration data cannot be avoided in some devices. A design goal is to reduce this to an absolute minimum. This information may have to be pre-configured on the device before it has been deployed physically, and is typically static. A preliminary list of unavoidable configuration data is:

- o Authentic identity for each device. This may be a public key or a signed certification. This is necessary to protect the infrastructure against unauthorized replacement of equipment.
- o The role and function and capability of the device. The role and function may depend on the network planning. The capability is typically decided by the hardware.
- o On the network edge, the routers may need to be configured with the identity of each peer provider, and their entitlements to service.

Ideally, everything else (topology, link capacity, address prefixes, shared resources, customer authentication and authority, etc.) will be discovered or negotiated autonomously according to general policy for various negotiated objective.

3. Existing protocols

Routing protocols are mainly one-way information announcements. The receiver makes independent decisions based on the received information and there is no direct feedback information to the announcing peer. This remains true even though the protocol is used in both directions between peer routers; there is no negotiation, and each peer runs its route calculations independently.

Simple Network Management Protocol (SNMP) [RFC3416] uses a command/response model not well suited for peer negotiation. Network Configuration Protocol (NETCONF) [RFC6241] uses an RPC model that does allow positive or negative responses from the target system, but this is still not adequate for negotiation.

There are various existing protocols that have elementary negotiation abilities, such as Dynamic Host Configure Protocol for IPv6 (DHCPv6) [RFC3315], Neighbor Discovery (ND) [RFC4861], Port Control Protocol (PCP) [RFC6887], Remote Authentication Dial In User Service (RADIUS) [RFC2865], Diameter [RFC6733], etc. Most of them are configuration or management protocols. However, they either provide only a simple request/response model in a master/slave context or very limited negotiation abilities.

There are also signalling protocols with an element of negotiation. For example Resource ReSerVation Protocol (RSVP) [RFC2205] was designed for negotiating quality of service parameters along the path of a unicast or multicast flow. RSVP is a very specialised protocol aimed at end-to-end flows. However, it has some flexibility, having been extended for MPLS label distribution [RFC3209]. A more generic design is General Internet Signalling Transport (GIST) [RFC5971], but it is complex, tries to solve many problems, and is also aimed at per-flow signalling across many hops rather than at device-to-device signalling. However, we cannot yet exclude extended RSVP or GIST as a negotiation protocol.

It is worth noting that some of the above protocols have either an explicit information model describing their messages, or at least a flexible and extensible message format. A negotiation protocol will require such capabilities. One design consideration is whether to adopt an existing information model or to design a new one. Another consideration is whether to be able to carry some or all of the message formats used by the above protocols.

We now consider two protocols that are works in progress at the time of this writing. Firstly, RESTCONF [I-D.ietf-netconf-restconf] is a protocol intended to convey NETCONF information expressed in the YANG language via HTTP, including the ability to transit HTML intermediaries. While this is a powerful approach in the context of centralised configuration of a complex network, it is not well adapted to efficient interactive negotiation between peer devices, especially simple ones that are unlikely to include YANG processing already.

Secondly, we consider HomeNet Control Protocol (HNCP) [I-D.ietf-homenet-hncp]. This is defined as "a minimalist state

synchronization protocol for Homenet routers." Specific features are:

- o Every participating node has a unique node identifier.
- o "HNCP is designed to operate between directly connected neighbors on a shared link using link-local IPv6 addresses."
- o Currency of state is maintained by spontaneous link-local multicast messages.
- o HNCP discovers and tracks link-local neighbours.
- o HNCP messages are encoded as a sequence of TLV objects, sent over UDP.
- o Authentication depends on a signature TLV (assuming public keys are associated with node identifiers).
- o The functionality covered initially includes: site border discovery, prefix assignment, DNS namespace discovery, and routing protocol selection.

Clearly HNCP does not completely meet the needs of a general negotiation protocol, especially due to its limitation to link-local messages and its strict dependency on IPv6, but at the minimum it is a very interesting test case for this style of interaction between devices without needing a central authority.

4. A Behavior Model of a Generic Negotiation Protocol

This section describes a behavior model and some considerations for designing a generic negotiation protocol, which would act as a platform for different negotiation objectives.

- o A generic platform

The design of the network device protocol is desired to be a generic platform, which is independent from the negotiation contents. It should only take care of the general intercommunication between negotiation counterparts. The negotiation contents will vary according to the various negotiation objectives and the different pairs of negotiating counterparts.

- o Security infrastructure and trust relationship

Because this negotiation protocol may directly cause changes to device configurations and bring significant impacts to a running network, this protocol must be based on a restrictive security infrastructure. It should be carefully managed and monitored so that every device in this negotiation system behaves well and remains well protected.

On the other hand, a limited negotiation model might be deployed based on a limited trust relationship. For example, between two administrative domains, devices might also exchange limited information and negotiate some particular configurations based on a limited conventional or contractual trust relationship.

- o A uniform pattern for negotiation contents

The negotiation contents should be defined according to a uniform pattern. They could be carried either in TLV (Type, Length and Value) format or in payloads described by a flexible language, like XML. A protocol design should choose one of these two. The format must be extensible for unknown future requirements. As noted above, an existing information model and existing message format(s) should be considered.

- o A simple initiator/responder model

Multi-party negotiations are too complicated to be modeled and there may be too many dependencies among the parties to converge efficiently. A simple initiator/responder model is more feasible and could actually complete multiple-party negotiations by indirect steps. Naturally this process must be guaranteed to terminate and must contain tie-breaking rules.

- o Organizing of negotiation content

Naturally, the negotiation content should be organized according to the relevant function or service. The content from different functions or services should be kept independent from each other. They should not be combined into a single option or single session because these contents may be negotiated with different counterparts or may be different in response time.

- o Topology neighbor device discovery

Every network device that supports the negotiation protocol is a responder and always listens to a well-known (UDP?) port. A well-known link-local multicast address should be defined for discovery purposes. Upon receiving a discovery or request message, the recipient device should return a message in which it either indicates itself as a proper negotiation counterpart or diverts the initiator towards another more suitable device.

- o Self aware network device

Every network device should be pre-configured with its role and functions and be aware of its own capabilities. The roles may be only distinguished because of network behaviors, which may include forwarding behaviors, aggregation properties, topology location, bandwidth, tunnel or translation properties, etc. The role and functions may depend on the network planning. The capability is typically decided by the hardware or firmware. These parameters are the foundation of the negotiation behavior of a specific device.

- o Requests and responses in negotiation procedures

The initiator should be able to negotiate with its relevant negotiation counterpart devices, which may be different according to the negotiation objective. It may request relevant information from the negotiation counterpart so that it can decide its local configuration to give the most coordinated performance. It may request the negotiation counterpart to make a matching configuration in order to set up a successful communication with it. It may request certain simulation or forecast results by sending some dry run conditions.

Beyond the traditional yes/no answer, the responder should be able to reply with a suggested alternative if its answer is 'no'. This would start a bi-directional negotiation ending in a compromise between the two devices.

- o Convergence of negotiation procedures

The negotiation procedure should move towards convergent results. It means that when a responder makes a suggestion of a changed condition in a negative reply, it should be as close as possible to the original request or previous suggestion. The suggested value of the third or later negotiation steps should be chosen between the suggested values from the last two negotiation steps.

In any case there must be a mechanism to guarantee rapid convergence in a small number of steps.

- o Dependencies of negotiation

In order to decide a configuration on a device, the device may need information from neighbors. This can be reached through the above negotiation procedure. However, a given item in a neighbor may depend on other information from its own neighbors, which may need another negotiation procedure to obtain or decide. Therefore, there are dependencies among negotiation procedures. There need to be clear boundaries and convergence mechanisms for these negotiation dependencies. Also some mechanisms are needed to avoid loop dependencies.

- o End of negotiation

A single negotiation procedure also needs ending conditions if it does not converge. A limited number of rounds, for example three, should be set on the devices. It may be an implementation choice or a pre-configurable parameter. However, the protocol design needs to clearly specify this, so that the negotiation can be terminated properly. In some cases, a timeout might be needed to end a negotiation.

- o Failed negotiation

There must be a well-defined procedure for concluding that a negotiation cannot succeed, and if so deciding what happens next (deadlock resolution, tie-breaking, or revert to best-effort service).

- o Policy constraints

There must be provision for general policy rules to be applied by all devices in the network (e.g., security rules, prefix length, resource sharing rules). However, policy distribution might not use the negotiation protocol itself.

- o Management monitoring, alerts and intervention

Devices should be able to report to a monitoring system. Some events must be able to generate operator alerts and some provision

for emergency intervention must be possible (e.g. to freeze negotiation in a mis-behaving device). These features may not use the negotiation protocol itself.

5. Security Considerations

This document does not include a detailed threat analysis for autonomic configuration, but it is obvious that a successful attack on autonomic nodes would be extremely harmful, as such nodes might end up with a completely undesirable configuration. A concrete protocol proposal will therefore require a threat analysis, and some form of strong authentication and, if possible, built-in protection against denial of service attacks.

Separation of network devices and user devices may become very helpful in this kind of scenario.

Also, security configuration itself should become autonomic whenever possible. However, in the security area at least, operator override of autonomic configuration must be possible for emergency use.

As noted earlier, a cryptographically authenticated identity for each device is needed in an autonomic network. It is not safe to assume that a large network is physically secured against interference or that all personnel are trustworthy. Each autonomic device should be capable of proving its identity and authenticating its messages. One approach would be to use a private/public key pair and sufficiently strong cryptography. Each device would generate its own private key, which is never exported from the device. The device identity and public key would be recorded in a network-wide database. The alternative of using symmetric keys (shared secrets) is less attractive, since it creates a risk of key leakage as well as a key management problem when devices are installed or removed.

Generally speaking, no personal information is expected to be involved in the negotiation protocol, so there should be no direct impact on personal privacy. Nevertheless, traffic flow paths, VPNs, etc. may be negotiated, which could be of interest for traffic analysis. Also, carriers generally want to conceal details of their network topology and traffic density from outsiders. Therefore, since insider attacks cannot be prevented in a large carrier network, the security mechanism for the negotiation protocol needs to provide message confidentiality.

6. IANA Considerations

This draft does not request any IANA action.

7. Acknowledgements

The authors want to thank Zhenbin Li, Bing Liu for valuable comments.

This document was produced using the xml2rfc tool [RFC2629].

8. Change Log [RFC Editor please remove]

draft-jiang-negotiation-config-ps-03, text improvements, added RESTCONF and HNCP to existing protocols, 2014-05-19.

draft-jiang-negotiation-config-ps-02, text improvements, added extra existing protocols, 2014-01-19.

draft-jiang-negotiation-config-ps-01, add more requirements, and add more considerations for behavior model, 2013-10-11.

draft-jiang-negotiation-config-ps-00, original version, 2013-06-29.

9. Informative References

[I-D.boucadair-network-automation-requirements]

Boucadair, M., Jacquenet, C., and L. Contreras,
"Requirements for Automated (Configuration) Management",
draft-boucadair-network-automation-requirements-03 (work
in progress), February 2014.

[I-D.ietf-homenet-hncp]

Stenberg, M. and S. Barth, "Home Networking Control
Protocol", draft-ietf-homenet-hncp-00 (work in progress),
April 2014.

[I-D.ietf-netconf-restconf]

Bierman, A., Bjorklund, M., Watsen, K., and R. Fernando,
"RESTCONF Protocol", draft-ietf-netconf-restconf-00 (work
in progress), March 2014.

[I-D.irtf-nmrg-an-gap-analysis]

Behringer, M., Carpenter, B., and S. Jiang, "Gap Analysis
for Autonomic Networking", draft-irtf-nmrg-an-gap-
analysis-00 (work in progress), April 2014.

- [I-D.irtf-nmrg-autonomic-network-definitions]
Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A.,
Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic
Networking - Definitions and Design Goals", draft-irtf-
nmrg-autonomic-network-definitions-00 (work in progress),
December 2013.
- [RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S.
Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1
Functional Specification", RFC 2205, September 1997.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
June 1999.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson,
"Remote Authentication Dial In User Service (RADIUS)", RFC
2865, June 2000.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V.,
and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP
Tunnels", RFC 3209, December 2001.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C.,
and M. Carney, "Dynamic Host Configuration Protocol for
IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3416] Presuhn, R., "Version 2 of the Protocol Operations for the
Simple Network Management Protocol (SNMP)", STD 62, RFC
3416, December 2002.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman,
"Neighbor Discovery for IP version 6 (IPv6)", RFC 4861,
September 2007.
- [RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet
Signalling Transport", RFC 5971, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A.
Bierman, "Network Configuration Protocol (NETCONF)", RFC
6241, June 2011.
- [RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn,
"Diameter Base Protocol", RFC 6733, October 2012.
- [RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P.
Selkirk, "Port Control Protocol (PCP)", RFC 6887, April
2013.

Authors' Addresses

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: jiangsheng@huawei.com

Yuanbin Yin
Huawei Technologies Co., Ltd
Q15, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: yinyuanbin@huawei.com

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 30, 2015

M. Pritikin
M. Behringer
S. Bjarnason
Cisco
September 26, 2014

Bootstrapping Key Infrastructures
draft-pritikin-bootstrapping-keyinfrastructures-01

Abstract

This document specifies automated bootstrapping of an key infrastructure using vendor installed IEEE 802.1AR manufacturing installed certificates, in combination with a vendor based cloud service. Before being authenticated, a new device has only link-local connectivity, and does not require a routable address. When a vendor cloud service is provided devices can be forced to join only specific domains but for constrained environments we describe a variety of options that allow bootstrapping to proceed.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 30, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	4
2. Architectural Overview	4
3. Operational Overview	7
3.1. Instantiating the Domain Certification Authority	7
3.2. Instantiating the Registrar	7
3.3. Accepting New Entities	7
3.4. Operating the Network	8
4. Functional Overview	8
4.1. Behavior of a new entity	9
4.1.1. Proxy Discovery	10
4.1.2. Receiving and accepting the Domain Identity	11
4.1.3. Enrollment	12
4.1.4. After Enrollment	12
4.2. Behavior of a proxy	12
4.3. Behavior of the Registrar	13
4.3.1. Authenticating the Device	13
4.3.2. Accepting the Entity	13
4.3.3. Claiming the new entity	14
4.4. Behavior of the MASA Cloud Service	14
4.4.1. Issue Authorization Token and Log the event	14
4.4.2. Retrieve Audit Entries from Log	15
4.5. Leveraging the new key infrastructure / next steps	15
4.5.1. Network boundaries	15
5. Protocol Details	15
5.1. EAP-EST	16
5.2. Request bootstrap token	16
5.3. Request MASA authorization token	17
5.4. Request MASA authorization log	17
6. Reduced security operational modes	18
7. Security Considerations	18
8. References	19
8.1. Normative References	19
8.2. Informative References	19
Authors' Addresses	20

1. Introduction

To literally "pull yourself up by the bootstraps" is an impossible action. Similarly the secure establishment of a key infrastructure without external help is also an impossibility. Today it is accepted

that the initial connections between nodes are insecure, until key distribution is complete, or that domain-specific keying material is pre-provisioned on each new device in a costly and non-scalable manner. This document describes a zero-touch approach to bootstrapping an entity by securing the initial distribution of key material using third-party generic keying material, such as a manufacturer installed IEEE 802.1AR certificate [IDevID], and a corresponding third-party cloud service.

The two sides of an association being bootstrapped authenticate each other and then determine appropriate authorization. This process is described as four distinct steps between the existing domain and the new entity being added:

- o New entity authentication: "Who is this? What is its identity?"
- o New entity authorization: "Is it mine? Do I want it? What are the chances it has been compromised?"
- o Domain authentication: "What is this domain's claimed identity?"
- o Domain authorization: "Should I join it?"

A precise answer to these questions can not be obtained without leveraging an established key infrastructure(s). The domain's decisions are based on the new entity's authenticated identity, as established by verification of previously installed credentials such as a manufacturer installed IEEE 802.1AR certificate, and verified back-end information such as a configured list of purchased devices or communication with a trusted third-party. The new entity's decisions are made according to verified communication with a trusted third-party or in a strictly auditable fashion.

Optimal security is achieved with IEEE 802.1AR certificates on each new entity, accompanied by a third-party cloud service for verification. The concept also works with less requirements, but is then less secure. A domain can choose to accept lower levels of security when a trusted third-party is not available so that bootstrapping proceeds even at the risk of reduced security. Only the domain can make these decisions based on administrative input and known behavior of the new entity.

The result of bootstrapping is that a domain specific key infrastructure is deployed. Since IEEE 802.1AR PKI certificates are used for identifying the new entity and the public key of the domain identity is leveraged during communications with a cloud service, which is itself authenticated using HTTPS, bootstrapping of a domain specific Public Key Infrastructure (PKI) is fully described.

Sufficient agility to support bootstrapping alternative key infrastructures (such as symmetric key solutions) is considered although no such key infrastructure is described.

1.1. Terminology

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

The following terms are defined for clarity:

2. Architectural Overview

The logical elements of the bootstrapping framework are described in this section. Figure 1 provides a simplified overview of the components. Each component is logical and may be combined with other components as necessary.

Domain Identity: The domain identity is the 160-bit SHA-1 hash of the BIT STRING of the subjectPublicKey of the domain trust anchor that is stored by the Domain CA. This is consistent with the RFC5280 Certification Authority subject key identifier of the Domain CA's self signed root certificate. (A string value bound to the Domain CA's self signed root certificate subject and issuer fields is often colloqually used as a humanized identity value but during protocol discussions the more exact term as defined here is used).

Orchestrator: Although bootstrapping of an individual device is automated and requires zero administrative involvement (particularly on the New Entity) the orchestrator drives general operations of the domain. In simple deployments this might be a single administrator ordering a new device from the Factory and manually inputting a serial number from the bill-of-sale into a Registrar. In a more complex environment this might be an automated process that directs a hypervisor "Factory" to instantiate a new virtual machine.

Factory: This instantiates the New Entity. For physical devices this can be representative of third-party vendor manufacturing, ordering and shipping process(es) that results in a physical hardware device with an IEEE 802.1AR identity being drop shipped to a destination domain for physical installation. In a virtual machine environment this can be the virtual machine hypervisor control software that initiates a virtual machine instance, in which case the factory is a "virtual factory" and might be managed by the domain itself.

Factory CA: This Certification Authority is leveraged by the Factory to issue IEEE 802.1AR identities to each New Entity. For a virtual factory it may be reasonable to assume the domain certification authority is directly used but in a complex environment it is assumed the Factory does not have direct access to the Domain Certification Authority.

Registrar: A representative of the domain that is configured, perhaps autonomically, to decide whether a new device is allowed to join the domain. The administrator of the domain interfaces with a Registrar to control this process.

New Entity: A new device or virtual machine or software component that is not yet part of the domain.

Proxy: A domain entity that helps the New Entity join the domain. A Proxy facilitates communication for devices that find themselves

in an environment where they are not provided L3 connectivity until after they are validated as members of the domain.

MASA Cloud Service: A Manufacturer Authorized Signing Authority (MASA) cloud service on the global Internet. At a minimum the MASA provides a trusted repository for audit information concerning privacy protected bootstrapping events. As a service offering the MASA can incorporate many of the bootstrapping elements (such as the Registrar and the Domain CA) into the cloud service.

3. Operational Overview

This section describes how an operator interacts with a domain that supports the bootstrapping as described in this document.

3.1. Instantiating the Domain Certification Authority

This is a one time step by the domain administrator. This is an "off the shelf" CA with the exception that it is designed to work as an integrated part of the security solution. This precludes the use of 3rd party certification authority services that do not provide support for delegation of certificate issuance decisions to a domain managed Registration Authority.

3.2. Instantiating the Registrar

This is a one time step by the domain administrator. One or more devices in the domain are configured take on a Registrar function.

A device can be configured to act as a Registrar or a device can auto-select itself to take on this function, using a detection mechanism to resolve potential conflicts and setup communication with the Domain Certification Authority. An automated Registrar selection processes is not detailed here. [[EDNOTE: yet]]

3.3. Accepting New Entities

For each New Entity the Registrar is informed a priori the unique identifier (e.g. serial number). This can be supplied automatically from the Orchestrator [[EDNOTE: TBD]] or inputted manually by the administrator.

For each entity that will be accepted a Registrar maintains the Factory CA identity and the entity's unique identifier. The Factory CA identity could be implemented as the Factory CA root certificate keyIdentifier (the 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey). For user interface purposes the keyIdentifier

information can be mapped to a colloquial Factory name (Registrars can be shipped with the keyIdentifier of a significant number of third-party manufacturers).

Additional policy can be stored for future authorization decisions. For example an expected deployment time window or that a certain Proxy must be used.

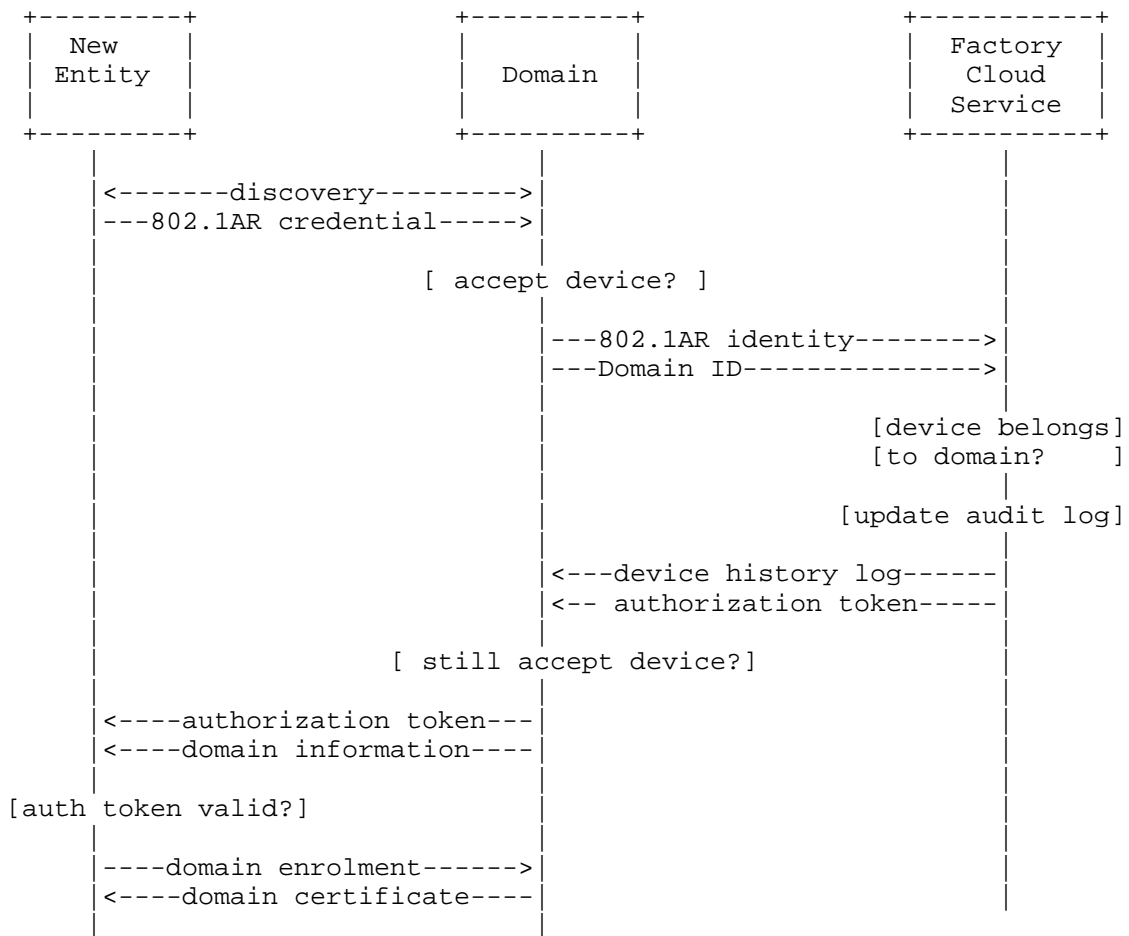
3.4. Operating the Network

Once devices are enrolled to the domain, the network operator can specify a policy, or otherwise configure the devices if required. This is outside scope for this document.

4. Functional Overview

Entities behave in an autonomic fashion. They discover each other and autonomically establish a key infrastructure delimiting the autonomic domain. See [I-D.behringer-autonomic-network-framework] for more information.

The overall flow is shown in Figure 2:



4.1. Behavior of a new entity

A New Entity that has not yet been bootstrapped attempts to find a local domain and join it. A number of methods are attempted for establishing communications with the domain in a specified order.

Client behavior is as follows:

1. Discover a communication channel to the "closest" Registrar by trying the following steps in this order:
 - A. Search for a Proxy on the local link using Neighbor Discovery. If multiple local proxies are discovered attempt

communications with each before widening the search to other options. If this fails:

- B. Obtain an IP address using DHCP, and search for a local registrar using DNS service discovery. If this fails:
 - C. Obtain an IP address using DHCP, and search for a pre-defined Factory provided global registrar using DNS.
2. Present IEEE 802.1AR credentials to the discovered Registrar (via a Proxy if necessary). Included is a generated nonce that is specific to this attempt.
 3. Verify the MASA cloud service generated authorization token as provided by the contacted Registrar. The nonce information previously provided is also checked, if it was not removed by the Registrar.
 4. If and only if step three is successful: Join Domain, by accepting the domain specific information from the registrar, and by enrolling a domain certificate from the registrar.
 5. The New Entity is now a member of the domain and will only repeat the discovery aspects of bootstrapping if it is returned to factory default settings.

[[EDNOTE: Step (1b and 1c) is similar to the vendor DNS mechanisms described in draft-kwatsen-netconf-zerotouch although the goal here is to contact a Registrar rather than a vendor supplied NMS]]

The following sections describe each of these steps in more detail.

4.1.1. Proxy Discovery

Existing protocols provide the appropriate functionality for both discovering the Proxy and facilitating communication through the Proxy:

IEEE 802.1X Where the New Entity can be cast as the "supplicant" and the Proxy is the "authenticator". The bootstrapping protocol messages are encapsulated as EAP methods. The "authenticator" reencapsulates the EAPOL frames and forwards them to the "Authentication Server", which provides Registrar functionalities.

PANA [RFC5191] [[EDNOTE: TBD]]

ND [RFC2461] / [RFC4861] [[EDNOTE: TBD]] NOTE: Neighbor Discovery protocols do not describe a mechanism for forwarding messages.

Each provides a method for the New Entity to discover and initiate communication with a local neighbor. In each protocol methods are available to support encapsulation of the bootstrapping protocol messages described elsewhere in this document. Other protocols for transporting bootstrapping messages can be added in future references.

All security associations established are between the new device and the Registrar regardless of proxy operations.

If multiple proxies are available the New Entity tries each until a successful bootstrapping occurs. The New Entity may prioritize proxies selection order as appropriate for the anticipated environment.

If Proxy discovery fails the New Entity moves on to discovering a Registrar directly.

4.1.2. Receiving and accepting the Domain Identity

The domain trust anchor is received by the New Entity during the bootstrapping protocol exchange.

EST [RFC7030] details a set of non-autonomic bootstrapping methods such as:

- o using the Implicit Trust Anchor database (not an autonomic solution because the URL must be securely distributed),
- o engaging a human user to authorize the CA certificate using out-of-band data (not an autonomic solution because the human user is involved),
- o and using a configured Explicit TA database (not an autonomic solution because the distribution of symmetric key material is not autonomic).

This document describes two additional autonomic methods:

MASA authorization token Authorization tokens are obtained by the Registrar from the MASA cloud service and presented to the New Entity for validation.

URL redirect If the New Entity discovers a well known global registrar using DNS then the EST protocol exchange is protected using an Implicit TA database, but also the MASA authorization is required. The global registrar MUST claim the device with the MASA server to ensure the logging information is consistent. The

global registrar forwards the New Entity to an alternate URI as described in EST [RFC7030].

If these methods fail the New Entity returns to discovery state and attempts bootstrapping with the next available discovered Registrar.

[[EDNOTE: move protocol discussion down into protocol section]] The domain trust anchor MUST be included in the TLS handshake Server Certificate "certificate_list" [RFC5246] or the client MUST request the EST Bootstrap Distribution of CA Certificates [RFC7030]. (This document defines an additional method for accepting the CA certificates).

4.1.3. Enrollment

As the final step of bootstrapping a Registrar helps to issue a domain specific credential to the New Entity. For simplicity in this document, a Registrar primarily facilitates issuing a credential by acting as an RFC5280 Registration Authority for the Domain Certification Authority.

Enrollment proceeds as described in Enrollment over Secure Transport (EST) [RFC7030]. The New Entity contacts the Registrar using EST as indicated:

- o The New Entity is authenticated using the IEEE 802.1AR credentials [[EDNOTE: or in the non-autonomic case using the the out of band secret]].
- o The EST section 4.1.3 CA Certificates Response is verified using the MASA authorization token provided domain identity.

4.1.4. After Enrollment

Functionality to provide generic "configuration" is supported. The parsing of this data and any subsequent use of the data, for example communications with a Network Management System is out of scope but is expected to occur after bootstrapping enrollment is complete.

See Section 4.5.

4.2. Behavior of a proxy

The role of the Proxy is to facilitate communications. The Proxy forwards messages between the New Entity and a Registrar. Where existing protocols as detailed in Section 4.1.1 already provide this functionality nothing additional is defined.

[[EDNOTE: If neighbor discovery protocols are used for Proxy discovery then a proxy forwarding protocol is to be defined here]]

4.3. Behavior of the Registrar

Once a registrar is established it listens for new entities and determines if they can join the domain. The registrar delivers any necessary authorization information to the new device and facilitates enrollment with the domain PKI.

Registrar behavior is as follows:

4.3.1. Authenticating the Device

The authentication methods detailed in EST [RFC7030] are:

- o the use of an IEEE 802.1AR IDevID credential,
- o or the use of a secret that is transmitted out of band between the New Entity and the Registrar (this use case is not autonomic).

4.3.2. Accepting the Entity

In a fully automated network all devices must be securely identified.

A Registrar accepts or declines a request to join the domain, based on the authenticated identity presented and other policy defined criteria such as Proxy identity. Automated acceptance criteria include:

- o allow any device of a specific type (as determined by the IEEE 802.1AR device identity),
- o allow any device from a specific Factory (as determined by the IEEE 802.1AR identity),
- o allow a specific device from a Factory (as determined by the IEEE 802.1AR identity)

In all cases a Registrar must use the globally available MASA cloud service to verify the device's history log does not include unexpected Registrars.

If a device is accepted into the domain, it is then invited to request a domain certificate through a certificate enrollment process. The result is a common trust anchor and device certificates for all autonomic devices in a domain. These certificates can subsequently be used to determine the boundaries of the homenet, to authenticate

other domain nodes, and to autonomically enable services on the homenet.

4.3.3. Claiming the new entity

During initial bootstrapping the New Entity provides a nonce specific to the particular bootstrapping attempt. The registrar should include this nonce when claiming the New Entity from the MASA cloud service. If a nonce is provided by the Registrar then claims from an unauthenticated Registrar are serviced by the MASA cloud resource.

The Registrar can claim a New Entity that is not online by forming the request using the entities unique identifier but not including a nonce in the claim request. MASA authorization tokens obtained in this way do not have a lifetime and they provide a permanent method for the domain to claim the device. Evidence of such a claim is provided in the audit log entries available to any future Registrar. Such claims reduce the ability for future domains to secure bootstrapping and therefore the Registrar MUST be authenticated by the MASA cloud service.

Claiming an entity establishes an audit log at the MASA server and provides the Registrar with proof, in the form of a MASA authorization token, that the log entry has been inserted. As indicated in Section 4.1.2 a New Entity will only proceed with bootstrapping if a validated MASA authorization token has been recieved. The New Entity therefore enforces that bootstrapping only occurs if the claim has been logged.

4.4. Behavior of the MASA Cloud Service

The cloud service is provided by the Factory provider. The URI of the cloud service is well known. The URI should be provided as an IEEE 802.1AR IDevID X.509 extension (a "MASA authorization token Distribution Point" extension).

The cloud service provides the following functionalities to Registrars:

4.4.1. Issue Authorization Token and Log the event

A Registrar POSTs a claim message optionally containing the bootstrap nonce to the MASA server.

If a nonce is provided the MASA cloud service responds to all requests. The MASA cloud service verifies the Registrar is representative of the domain and generates a privacy protected log entry before responding with the authorization token.

If a nonce is not provided the MASA cloud service MUST authenticate the Registrar as a valid customer. This prevents denial of service attacks. The specific level of authentication provided by the customer is not defined here. An MASA Practice Statement (MPS) similar to the Certification Authority CPS, as defined in RFC5280, is provided by the Factory such that Registrar's can determine the level of trust they have in the Factory.

4.4.2. Retrieve Audit Entries from Log

When determining if a New Entity should be accepted into a domain the Registrar retrieves a copy of the audit log from the MASA cloud service. This contains a list of privacy protected domain identities that have previously claimed the device. Included in the list is an indication of the time the entry was made and if the nonce was included.

4.5. Leveraging the new key infrastructure / next steps

As the devices have a common trust anchor, device identity can be securely established, making it possible to automatically deploy services across the domain in a secure manner.

Examples of services:

- o Device management.
- o Routing authentication.
- o Service discovery.

4.5.1. Network boundaries

When a device has joined the domain, it can validate the domain membership of other devices. This makes it possible to create trust boundaries where domain members have higher level of trusted than external devices. Using the autonomic User Interface, specific devices can be grouped into to sub domains and specific trust levels can be implemented between those.

5. Protocol Details

The bootstrapping protocol is an extension of EST [RFC7030].

[[EDNOTE: Insert figure here]]

EST provides a bootstrapping mechanism for new entities that are configured with the URI of the EST server or new entities that can

"engage a human user to authorize the CA certificate using out-of-band data such as a CA certificate". EST does not provide a completely automated method of bootstrapping the PKI. [[EDNOTE: This paragraph should be expanded to provide a detailed discussion of current EST functionalities, or do we assume the reader follows the normative reference?]].

The following additions provide for fully automated functionality. EST is extended by defining additional HTTP URIs and messages specific to bootstrapping. These are optionally supported by the EST server within the same .well-known URI tree as the existing EST URIs.

The "New Entity" is the EST client and the "Registrar" is the EST server.

5.1. EAP-EST

In order to support Proxy environments EAP-EST is defined.

[[EDNOTE: TBD. EST is TLS with some data. EAP-TLS and other similar protocols provide an example framework for filling out this section]]

5.2. Request bootstrap token

When the New Entity reaches the EST section 4.1.1 "Bootstrap Distribution of CA Certificates" state but wishes to proceed in a fully automated fashion it makes a request for a MASA authorization token from the Registrar.

This is done with an HTTPS POST using the operation path value of "/requestbootstraptoken".

The request format is a raw nonce value. [[EDNOTE: exact format TBD. There is an advantage to having the client sign the nonce (similar to a PKI Certification Signing Request) since this allows the MASA cloud service to confirm the actual device identity. It is not clear that there is a security benefit from this.]]

The Registrar validates the client identity as described in EST [RFC7030] section 3.3.2. The registrar performs authorization as detailed in Section 4.3.2. If authorization is successful the Registrar obtains a MASA authorization token from the MASA cloud service (see Section 5.3).

The received MASA authorization token is returned to the New Entity.

[[EDNOTE: update to CMS language]]

5.3. Request MASA authorization token

A registrar requests the MASA authorization token from the cloud service using this EST extension.

This is done with an HTTP POST using the operation path value of `"/requestMASAAuthorization"`.

The request format is an optional raw nonce value (as obtained from the bootstrap request) and the IEEE 802.1AR identity of the device as a serial number (the full certificate is not needed and no proof-of-possession information for the device identity is included). This information is encapsulated in a PKCS7 signed data structure that is signed by the Registrar. The entire certificate chain, up to and including the Domain CA, is included in the PKCS7.

The MASA cloud service checks the internal consistency of the PKCS7 but is unable to actually authenticate the domain identity information. The domain is not known to the MASA server in advance and a shared trust anchor is not implied. The MASA server verifies that the PKCS7 is signed by a Registrar (by checking for the `cmc-idRA` field in the Registrar certificate) certificate that was issued by the root certificate included in the PKCS7.

The domain ID is extracted from the root certificate and is used to generate the MASA authorization token and to update the audit log.

[[EDNOTE: update to CMS language]]

5.4. Request MASA authorization log

A registrar requests the MASA authorization log from the cloud service using this EST extension.

This is done with an HTTP GET using the operation path value of `"/requestMASALog"`.

The log data returned is a file consisting of each log entry. The data in each entry includes:

- o date/time of the entry
- o domain ID (this is just a hash of the public key information and is thus privacy protected)
- o nonce value

[[EDNOTE: exact format TBD]]

6. Reduced security operational modes

A common requirement of bootstrapping infrastructures is often that they support less secure operational modes. To support these operational modes the Registrar can choose to accept devices using less secure methods. For example:

1. The registrar may chose to accept all devices, or all devices of a particular type, at the administrator's discretion. This may occur when: Informing the Registrar of unique identifiers of new entities might be operationally difficult.
2. The registrar may chose to accept devices that claim a unique identity without the benefit of authenticating that claimed identity. This may occur when: The New Entity does not include an IEEE 802.1AR factory installed credential.
3. A representative of the Registrar (e.g. the Orchestrator) may request nonce-less authorization tokens from the MASA cloud service when network connectivity is available. These tokens can then be transmitted to the Registrar and stored until they are needed during bootstrapping operations. This may occur when: The target network is protected by an air gap and therefore can not contact the MASA cloud service during New Entity deployment.
4. The device may have an operational mode where it skips authorization token validation. For example if a physical button is depressed during the bootstrapping operation. This may occur when: A device Factory goes out of business or otherwise fails to provide a reliable MASA cloud service.
5. The device may not require the MASA cloud service authorization token. An entity that does not validate the domain identity is inherently dangerous as it may contain malware. This risk should be mitigated using attestation and measurement technologies. In order to support an unsecured imprint the New Entity MUST support remote attestation technologies such as is defined by the Trusted Computing Group. [[EDNOTE: How to include remote attestation into the bootstrapping protocol exchange is TBD]]. This may occur when: The device Factory does not provide a MASA cloud service.

7. Security Considerations

In order to support a variety of use cases, devices can be claimed by a registrar without proving possession of the device in question. This would result in a nonceless, and thus always valid, claim. The MASA cloud service is required to authenticate such Registrars but no programatic method is provided to ensure good behavior by the MASA

cloud service. Nonceless entries into the audit log therefore permanently reduce the value of a device because future Registrars, during future bootstrap attempts, must now be configured with policy to ignore previously (and potentially unknown) domains.

Future registrars are recommended to take the audit history of a device into account when deciding to join such devices into their network.

It is possible for an attacker to send an authorization request to the MASA cloud service directly after the real Registrar obtains an authorization log. If the attacker could also force the bootstrapping protocol to reset there is a theoretical opportunity for the attacker to use the authorization token to take control of the New Entity but then proceed to enrol with the target domain. To prevent this the MASA cloud service is rate limited to only generate authorization tokens at a rate of 1 per minute. The Registrar therefore has at least 1 minute to get the response back to the New Entity. [[EDNOTE: a better solution can likely be found. This text captures the issue for now.]] Also the Registrar can double check the log information after enrolling the New Entity.

The MASA cloud service could lock a claim and refuse to issue a new token. Or the MASA cloud service could go offline (for example if a vendor went out of business). This functionality provides benefits such as theft resistance, but it also implies an operational risk. This can be mitigated by Registrars that request nonce-less authorization tokens.

8. References

8.1. Normative References

- [IDevID] IEEE Standard, , "IEEE 802.1AR Secure Device Identifier", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC7030] Pritikin, M., Yee, P., and D. Harkins, "Enrollment over Secure Transport", RFC 7030, October 2013.

8.2. Informative References

[I-D.behringer-autonomic-network-framework]

Behringer, M., Pritikin, M., Bjarnason, S., and A. Clemm,
"A Framework for Autonomic Networking", draft-behringer-
autonomic-network-framework-01 (work in progress), October
2013.

Authors' Addresses

Max Pritikin
Cisco

Email: pritikin@cisco.com

Michael H. Behringer
Cisco

Email: mbehring@cisco.com

Steinthor Bjarnason
Cisco

Email: sbjarnas@cisco.com