Independent Submission
Internet-Draft
Intended status: Experimental
Expires: 15 March 2026

S. Li

11 September 2025

DART: Domain-Aware Routing Technology Protocol
draft-li-dart-protocol-00

Abstract

This document proposes a new addressing and routing protocol named Domain-Aware Routing Technology (DART). Built upon the existing IP network architecture, DART introduces a fully qualified domain name (FQDN)-centric addressing model to address the global exhaustion of IPv4 addresses and significantly enhance the scalability and evolvability of the Internet.

DART supports a name-based communication model and can be encapsulated over either IPv4 or IPv6 networks. This enables high compatibility with existing infrastructures and supports deployment and interoperability across heterogeneous protocol environments. Its addressing logic and architectural design exhibit the following characteristics:

* In terms of scalability
  DART localizes the IPv4 address space to individual Domain Names (DNS) domains, granting each domain an independent full address pool. This enables unlimited address reuse, greatly reduces global routing table growth, and promotes a more structured and aggregatable forwarding hierarchy.

* In terms of evolvability
  DART shifts the addressing paradigm from "IP-based location" to "name-based location", laying the foundation for a unified, name-driven networking model. It holds potential to redefine the semantics of network addressing and routing architecture.

DART is designed for incremental deployment. Network operators can derive a subdomain within the DNS system to serve as a DART domain, immediately gaining access to a dedicated 2^32-sized address space. Each DART domain may further delegate subdomains, forming a theoretically infinite, hierarchical, and scalable global addressing system.

Deployment requires only the installation of DART-capable gateways at the boundaries between DART domains and traditional networks, or between different DART domains, to enable protocol interoperability

and cross-domain forwarding. Internal network devices and endpoints within DART domains may continue operating under the existing IP stack without immediate modification. In particular, legacy IPv4-only devices located behind an edge DART gateway can seamlessly connect to the DART network via the *NAT-DART-4* mechanism. New devices may progressively evolve toward native DART support, enabling direct, end-to-end, name-based communication.

A working prototype of the DART protocol has been independently developed and deployed by the author. It is publicly accessible at http://dart-proto.cn

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

With the explosive growth of global Internet-connected devices, the traditional IPv4 [RFC791] address space has long been insufficient to meet the ever-increasing demand for connectivity. Although IPv6 offers a theoretically limitless address capacity, its deployment has been persistently hindered by various practical factors, including high upgrade costs, complex compatibility issues, and unclear operational incentives. As a result, most networks continue to rely on increasingly intricate systems of Network Address Translation (NAT [RFC3022]) and private address mappings to maintain functionality.

To address this dilemma, *Domain-Aware Routing Technology (DART)* emerges as a solution. DART is a communication protocol designed to provide *an effectively unlimited address space* while remaining *highly compatible with existing networks*. To achieve this goal, DART introduces a Fully Qualified Domain Name (FQDN [RFC1034][RFC1035])-based addressing model and is engineered to be encapsulated within either IPv4 or IPv6 [RFC8200] networks.

DART maintains full compatibility with IPv4 and can be deployed without requiring any modifications to existing endpoint devices. At the same time, DART natively supports IPv6, enabling operation over IPv6-only networks and facilitating seamless interoperability between IPv4 and IPv6 without the need for dual-stack architectures or complex address translation mechanisms.

By integrating DNS into the core of its addressing architecture, DART establishes a name-based routing model. Through virtual address mapping and lightweight encapsulation mechanisms, DART enables an abstract upgrade of the Internet's addressing semantics within the current network infrastructure. Whether deployed in home, enterprise, or carrier networks, DART can be integrated into the existing Internet at minimal cost, supporting incremental evolution and expanding the global addressable space without requiring architectural overhaul - thereby laying a sustainable technical foundation for the next-generation Internet [RFC1034][RFC1035][RFC2181].

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 BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

* DART
  Domain-Aware Routing Technology. A protocol that introduces a domain name-based addressing and encapsulation mechanism on top of the existing IP network, enabling scalable and highly compatible global communication.

* DART Domain
  A logical addressing domain defined by a DNS subdomain (e.g., dart.example.com). Each DART domain owns an independent IPv4 or IPv6 address space, which is used by devices within the domain.

* DART Address
  Essentially represented by a globally unique Fully Qualified Domain Name (FQDN), indicating the identity and location of a communication endpoint. During actual communication, it is mapped to an IP address allocated within the corresponding DART domain. The IP address itself is not globally unique.

* DART Encapsulation
  DART packets use DART addresses as both source and destination identifiers and are encapsulated in UDP datagrams for transmission over existing IP networks. The encapsulated payload supports standard IP packet structures.

* NAT-DART-4 Translation
  A gateway mechanism that enables legacy IPv4-only hosts to establish peer-to-peer communication via the DART network. The edge DART gateway performs transparent translation between DART packets and IPv4 packets, allowing seamless compatibility [RFC2663][RFC4787].

* DART Gateway
  A forwarding device deployed at the boundary of a DART domain, responsible for encapsulating and decapsulating DART packets, performing route selection, and interworking with legacy IPv4/IPv6 networks.

* Virtual Address Mapping
  When a local host queries domain names within a remote DART gateway’s subdomain, all such names resolve to the same external IP address of the remote DART gateway. In this case, IPv4-only hosts behind the local DART gateway are unable to distinguish between different hosts within the remote subdomain. To address this, the local DART gateway maps each remote domain name to a distinct virtual address, allowing internal IPv4-only hosts to differentiate between target hosts and maintain consistent mappings throughout the session. This mechanism is essential for enabling IPv4-only hosts to function as DART-capable endpoints.

* DART-capable Public Server
  A server deployed on the public Internet, primarily providing web or application services, usually associated with a stable domain name and operating continuously online.

* DART-capable End Device
  Devices intended for personal or enterprise use, such as PCs, mobile phones, and IoT devices.

3. Design Goals

Domain Aware Routing Technology (DART) aims to provide a sustainable and scalable addressing and forwarding architecture for the future Internet, addressing fundamental issues such as IPv4 address exhaustion, protocol fragmentation, and upgrade difficulties. The design emphasizes incremental deployment without disrupting existing network operations, guiding the global network ecosystem toward a unified, efficient, and sustainable model [RFC8504][RFC4864].

The core design goals include:

* Infinite Addressability
  DART utilizes Fully Qualified Domain Names (FQDNs) as principal identifiers, freeing addressing from fixed-length numeric schemes and inherently supporting unlimited scalability.

* Full IPv4 Compatibility
  The DART protocol operates over standard IPv4 networks without requiring any modifications to the IPv4 protocol stack, existing endpoints, or network devices, making it well-suited for deployment on widely deployed infrastructures.

* Native IPv6 Compatibility
  DART also supports encapsulation over IPv6 networks and can operate in IPv6-only environments, providing equivalent addressing and forwarding capabilities in pure IPv6 deployments.

* Seamless IPv4 and IPv6 Interoperability
  Through gateway mechanisms, DART enables transparent communication between IPv4-only and IPv6-only hosts without the need for dual-stack deployments or complex address translation schemes such as NAT64.

* Deep Integration with DNS
  DNS serves as the control plane core in DART, responsible for identity resolution and virtual address assignment, thereby supporting dynamic mapping, mobility, and service-level addressing.

* Support for Incremental Deployment
  DART can be deployed across residential, enterprise, and operator networks with strong backward compatibility and edge-introduction capabilities, allowing phased adoption without impacting existing services.

* Load Balancing and Mobility Support Outlook
  By returning multiple IP addresses via DNS, DART inherently supports load balancing. Although mobility mechanisms remain under active research, dynamic DNS updates and routing optimizations are expected to enable effective mobility support for moving nodes.

Together, these goals form the foundation of the DART protocol, enabling compatibility with existing network architectures while providing extensibility for future growth-not as a disruptive replacement, but as an evolutionary enhancement aligned with emerging network trends.

4. Protocol Overview

DART (Domain Aware Routing Technology) is a standalone network-layer protocol that enables packet forwarding based on Fully Qualified Domain Names (FQDNs) rather than fixed-length numeric addresses. Unlike IPv4 [RFC791] and IPv6 [RFC8200], which rely on prefix-based routing of 32-bit or 128-bit addresses, DART uses human-readable, hierarchical domain names as primary network identifiers.

DART is not an extension of IP, but an alternate network layer that operates encapsulated within IP packets. This structural encapsulation ensures compatibility with existing infrastructure while enabling DNS-driven routing. Typically, DART payloads are embedded in User Datagram Protocol (UDP [RFC768]) packets using a dedicated port (e.g., 0xDA27), allowing them to traverse NAT gateways and be transmitted over both IPv4 and IPv6 networks.

Routing decisions in DART networks are based on FQDN semantics, with DNS serving as the authoritative system for both name resolution and capability discovery (e.g., whether a target host supports DART). DART-capable nodes dynamically determine whether to initiate DART-based communication based on DNS responses.

DART gateways perform protocol translation for legacy IP-only hosts, enabling incremental adoption across heterogeneous networks. The design prioritizes evolutionary deployment without requiring changes to legacy endpoints, while offering a unified and extensible framework for next-generation networking.

This document treats DART as a first-class network protocol, capable of operating alongside traditional IP stacks. The following sections define its packet format, addressing model, routing mechanisms, and deployment strategies.

5. Packet Format

The DART protocol defines a lightweight packet header that carries identification information for the destination and source hosts, and guides packet forwarding across different DART domains. This header is inserted between the IP layer and the transport layer, positioning DART as a protocol operating at a "second network layer." It does not interfere with the existing IP header structure and, when necessary, provides virtual addressing support for IPv4-only hosts.

5.1. Header Structure

The DART packet header is structured as follows:

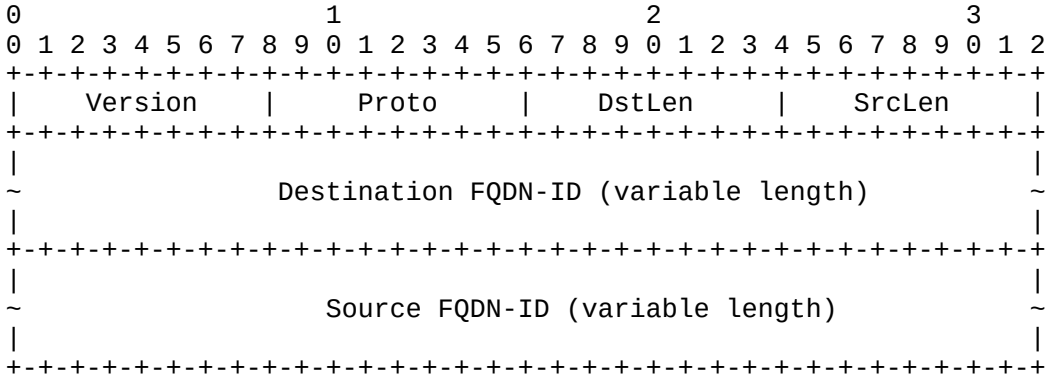

Figure 1: DART Packet Header Structure

* Version (8 bits)
  Indicates the version of the DART protocol. The current version is 1. Future versions may be used to extend the field structure or introduce new addressing mechanisms.

* Proto (8 bits)
  Specifies the upper-layer protocol type, using the same values as the IP protocol field (e.g., 6 for TCP, 17 for UDP, 1 for ICMP). When encapsulating a packet in DART, the Protocol field in the outer IP header should be adjusted accordingly (e.g., set to "UDP", with the UDP destination port set to 0xDA27), and the original protocol number should be copied into this field.

* DstLen (8 bits)
  Length of the Destination FQDN-ID, in bytes.

* SrcLen (8 bits)
  Length of the Source FQDN-ID, in bytes.

* Destination FQDN-ID (variable length)
  The Fully Qualified Domain Name of the destination host, e.g., dart-host.example.com. It serves as the primary routing identifier of the DART packet, and the receiver must be able to correctly parse its semantic structure.

* Source FQDN-ID (variable length)
  The Fully Qualified Domain Name of the source host. This field can be used for reverse routing of response paths, policy control, and other purposes.
  If the length of the Source FQDN-ID field is zero, it indicates that the packet is anonymous or unidirectional. In this case, the receiver should not expect a source identifier and should not

generate a response to this packet. This design is suitable for unidirectional notifications, broadcast or multicast packets, and privacy-protecting communication scenarios where the sender does not wish to reveal its identity.

5.2. Encoding and Alignment

* All fields are encoded in network byte order (big-endian);
* FQDN-ID fields are represented in UTF-8 encoding, not null-terminated and not aligned to fixed boundaries;
* During decapsulation, the positions of the FQDN fields must be determined precisely based on the values of DstLen and SrcLen.

5.3. Header Insertion Position

DART packets are not defined as native IP-layer protocols. Instead, they are encapsulated within the payload of UDP datagrams. A fixed UDP destination port (recommended: 55847 / 0xDA27) is used to distinguish DART packets and facilitate their processing. The encapsulation stack is illustrated as follows:

Layer	Remarks
Ethernet	
IP	v4/v6
UDP	DstPort=55847
DART	
Upper-layer Protocol	(e.g., TCP/UDP/ICMP)
Payload	

Table 1: Example of DART Header Insertion Position

Encapsulation Details:

* UDP Source Port: May be dynamically allocated (e.g., for session tracking);

* UDP Destination Port: Recommended to be statically set to 55847 (decimal), indicating a DART payload;

* UDP Length: Covers both the DART header and upper-layer protocol data;

* UDP Checksum: Optional, but recommended for integrity checking.

NAT Compatibility:

Since DART packets appear as regular UDP traffic on the wire, they inherit the following advantages:

* Full compatibility with consumer and enterprise-grade NATs;

* No reliance on IP options or Application Layer Gateways (ALGs);

* Can be managed by existing firewalls and access control policies based on UDP port;

* Enables NAT traversal techniques such as UDP hole punching, keep-alive, and reachability testing (While the

DART protocol aims to eliminate the necessity of NAT gateways, it remains compatible with them, though the use of overly complex NAT-related techniques is not encouraged).

5.4. Example

Assume that the client client.dart.cn initiates a DART connection to the server server.example.com. The DART header fields are as follows:

Field Name	Value	Description
Version	1	Protocol version
Proto	6	Upper-layer protocol (6 = TCP)
DstLen	18	Byte length of the destination FQDN
SrcLen	14	Byte length of the source FQDN
Dst FQDN-ID	"server.example.com"	Destination host name, encoded in UTF-8
Src FQDN-ID	"client.dart.cn"	Source host name, encoded in UTF-8

Table 2: DART Header Example

6. DART Addressing and Routing Model

DART replaces the reliance on IP addresses as globally unique host identifiers with an addressing architecture based on Fully Qualified Domain Names (FQDNs). It introduces a domain-based hierarchical routing model, enabling wide-area coverage, virtually unlimited addressability, and backward compatibility with IPv4-only hosts through the use of virtual addresses, DNS integration, and cooperation with DART gateways.

6.1. FQDN as Address

In the DART protocol, each host uses its Fully Qualified Domain Name (FQDN) as its logical address (FQDN-ID). This address is transmitted in clear text within packets and serves as the primary basis for forwarding decisions. This model offers the following advantages:

* The address space is theoretically unlimited, constrained only by the domain name system.

* It can intuitively reflect the network’s organizational structure and physical topology.

* It supports flexible address mapping, making scenarios such as load balancing and host mobility feasible-provided that DART gateways maintain consistent mappings.

* Effectively mitigates common problems of address conflicts and identity ambiguity typically encountered in NAT environments.

6.2. Inter-Domain Routing Model

The DART network adopts a hierarchical and distributed routing architecture based on domain names, with two core components: the DNS system and the DART Gateway:

* The DNS system constitutes the control plane of DART, responsible for domain name allocation, resolution, address mapping, and the dissemination and negotiation of control information. It determines the logical structure of communication paths.

* The DART Gateway is responsible for the data plane, handling data encapsulation and decapsulation, routing decisions, and protocol translation at domain boundaries.

This design follows the established principle of control plane and data plane separation in modern Internet architecture, enabling DART to achieve high scalability and flexible deployment. Inter-domain communication paths are resolved hierarchically through DNS, while actual packet forwarding is carried out by DART gateways at domain boundaries, thereby constructing a logically autonomous and structurally clear model of inter-domain connectivity.

It is worth noting that the host naming mechanism in DART networks is fully compatible with the existing DHCP+DNS infrastructure. In current local network practices, hosts typically submit their hostname during DHCP requests. The DHCP server assigns an IP address and can register the hostname-to-IP mapping in the DNS system using Dynamic DNS (DDNS) [RFC2136].

In future scenarios where hosts natively support the DART protocol stack, the DHCP server can centrally assign hostnames that conform to DART naming conventions, and DHCP clients can configure their local hostname accordingly. This ensures consistent and centrally managed naming within the network. Even in transitional phases where end hosts are not yet DART-aware, DHCP servers can enforce uniqueness by rejecting duplicate hostnames or appending disambiguating suffixes. This mechanism provides a practical and incremental deployment path for DART, allowing it to smoothly integrate with existing network environments.

6.2.1. Domain Structure and Gateway Interfaces

The DART network is built upon the hierarchical logic of the DNS naming system, in which each domain can be further divided into child domains, forming a multi-level logical naming tree. Correspondingly, DART Gateways serve as the connective backbone of this tree, responsible for cross-domain packet forwarding and control-plane signaling.

A typical DART Gateway includes the following interface structure:

* Parent domain interface (upstream): Used to connect to the parent DART domain. It typically handles access to the DNS control plane and the forwarding of upstream traffic. This interface may use various connection types, including direct physical links or indirect logical paths via intermediate networks (i.e., "third-party relay paths").

* Child domain interface (downstream): Used to connect to the child domains managed by the gateway. Each child domain is treated as a logical DART region. The downstream interface handles local DNS forwarding, virtual address allocation, DART encapsulation, and related tasks.

The domain relationship of a DART Gateway must conform to the following structural rules:

* Single parent domain: Each DART Gateway must belong to exactly one parent domain to maintain a hierarchical directed tree structure;

* Support for multiple child domains: A DART Gateway may manage one or more child domains, which together form its downstream logical namespace;

* Multiple interfaces to the same domain: A DART Gateway may connect to the same domain (either parent or child) via multiple interfaces to support redundancy, load balancing, path optimization, or lateral connectivity between sibling gateways;

* Intra-domain routing is delegated to the IP layer: For multiple interfaces connected to the same domain, routing decisions and path convergence are handled by standard IP-layer dynamic routing protocols (e.g., OSPF [RFC2328], BGP [RFC4271]). The DART layer does not interfere with intra-domain routing.

This interface model has the following properties:

* Scalable hierarchical extension
  Each child domain may recursively deploy its own DART Gateways and form deeper hierarchical structures;

* Explicit domain boundaries
  The DART Gateway acts as the sole ingress/egress point for domain-level communication, supporting policy enforcement and traffic management;

* Support for incremental deployment
  DART support can be gradually introduced at any layer without disrupting existing networks.

Through this structure, the DART network enables a flexible, multi-level architecture that provides a well-organized foundation for FQDN-based addressing and routing.

In practical deployments, it is possible for DART Gateway interconnections to span multiple hierarchical levels without a common parent domain (e.g., a capital node in a small country may connect to a neighboring border city in a large country instead of the distant capital). Such scenarios go beyond the expressive power of static hierarchical domain models. It is therefore recommended that industry experts consider adapting mature dynamic routing protocols from the IP layer to design a dynamic routing mechanism for the DART layer, enabling flexible routing across domain boundaries and hierarchy levels. This document does not delve into protocol design details, which are left for future research by specialists in the field.

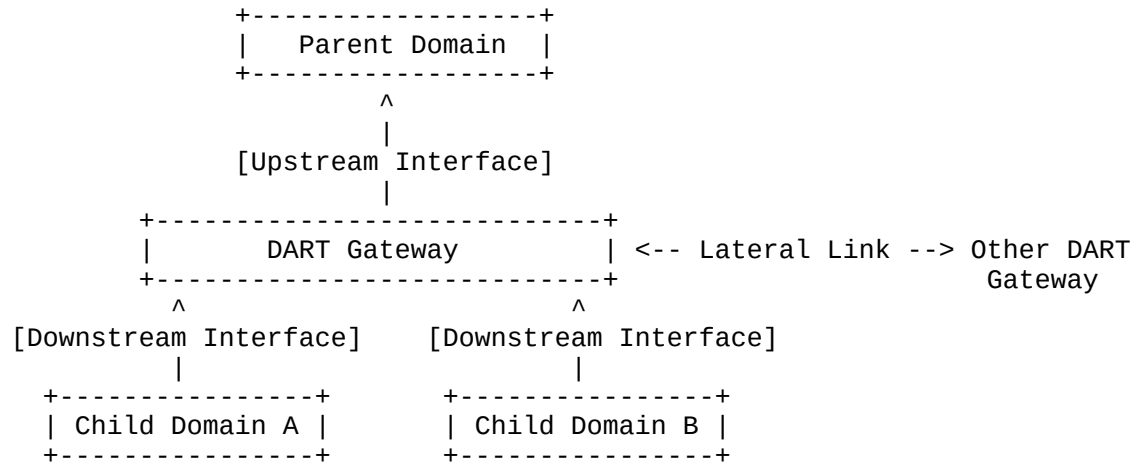

Figure 2: A Typical DART Inter-Domain Hierarchy

Notes:

* The parent domain is shown above, connected via the upstream interface;

* Two child domains are shown below, each connected via a downstream interface;

* The lateral link indicates peer-level connectivity, logically still part of the parent domain;

* Interface roles are annotated with bracketed labels;

* This structure reflects the DART model of "one parent, multiple children, and multi-interface support."

6.2.2. DART's Requirements and Suggestions for the DNS System

The DART protocol relies on FQDN-based addressing and uses the DNS system as its control plane. Domain-level routing, subdomain delegation, and host reachability all depend on DNS operations. As a result, DART places higher demands on DNS functionality, extensibility, and autonomy. The key requirements include:

* Dynamic Subdomain Delegation
  Each DART subdomain must register itself as an authoritative server under its parent domain. In many scenarios-especially for edge DART gateways that dynamically generate subdomains-there must be a mechanism to dynamically register or update NS and A/AAAA records within the DNS system.

* Precise FQDN Resolution
  As all packet forwarding decisions in a DART network are based on FQDN, DNS resolution must be complete, accurate, and consistent. The system must support fast TTL refresh, coherent caching strategies, and avoidance of partial or ambiguous resolution paths.

* Indication of DART-Ready Hosts
  To distinguish DART-ready hosts from IPv4-only ones, the DNS system may employ special markers, such as CNAME prefixes (e.g., dart-host.) or dedicated TXT records. These indicators help DART senders determine whether to encapsulate outgoing packets using the DART protocol.

* Enhanced DNS Security
  Since DART forwarding depends entirely on DNS results, the authenticity and integrity of DNS records are critical. Security measures such as DNSSEC are strongly recommended to prevent cache poisoning or redirection attacks that may compromise routing.

To fulfill these requirements, modern DNS software stacks can be extended via plugins, APIs, or integration with SDN controllers. Popular DNS platforms like BIND, PowerDNS, and CoreDNS offer sufficient programmability to serve as a foundation for early-stage DART trials.

In conclusion, the DNS system is the cornerstone of DART's control plane. Its completeness, performance, and adaptability will directly impact the scalability and manageability of DART networks. Future standardization work may consider extending the DNS protocol or defining new implementation guidelines to support DART's dynamic, secure, and FQDN-based addressing model.

6.2.3. Routing and Forwarding

DART is a parallel network layer protocol encapsulated above the IP layer. Due to the design principle that each DART domain owns an independent IPv4 address space, a DART gateway must maintain separate IP routing tables for each connected domain. These domains are isolated from each other at the IP layer and do not have direct inter-domain IP connectivity.

Therefore, a DART gateway manages two types of routing information simultaneously:

* DART-layer logical routing table
  Used to make forwarding decisions based on the destination FQDN contained in packets, determining which downstream interface or relay gateway to forward to;

* IP-layer routing table for each domain
  Used for forwarding IP packets within that domain, including communication between hosts and interaction with the DART gateway. Each IP routing table is valid only within its respective domain and does not support cross-domain addressing.

The IP layer routing mechanisms are mature and well-established and will not be elaborated here. The following focuses on DART-layer routing and forwarding principles:

* Domain name-based routing control
  DART gateways maintain logical routing tables organized by domain and forward packets according to the target Fully Qualified Domain Name (FQDN). DART employs a strictly hierarchical FQDN addressing system with globally unique addresses, abandoning prefix aggregation and traditional IP subnet nesting.

* Elimination of prefix holes and fragmentation issues
  Thanks to the natural separation afforded by domain name addressing, DART completely avoids prefix overlap, address penetration, and routing fragmentation problems. This significantly simplifies global routing table structures, enhances controllability and scalability, and effectively overcomes the scalability bottlenecks of IPv4 and BGP.

* DNS-assisted protocol identification and path selection

  - Special CNAME prefixes in DNS query results (e.g., dart-host. or dart-gateway.) indicate whether the target host or relay supports DART protocol;

  - A / AAAA records in DNS responses provide the next-hop IP address required by the IP layer, which may be the ultimate target host or a relay DART gateway.

  - Utilizing CNAME redirection to A records in DNS responses enables, under full compatibility with existing DNS protocols, the simultaneous return of whether the target host supports DART and its corresponding IP address, thereby improving resolution efficiency and protocol compatibility.

6.2.4. Hierarchical Address Encryption and Policy Visibility

To enhance privacy, DART allows the destination address field to be encrypted. This prevents intermediate nodes from observing the full semantic identity of the communication target. However, if a single-layer or fully opaque encryption scheme is applied, intermediate network devices-such as firewalls or DART-aware gateways-may lose visibility into the FQDN and thus be unable to enforce domain-based policies (see Section 8.6).

To reconcile the need for privacy with the operational requirements of policy enforcement, DART supports a Hierarchical Decryption mechanism. In this model, the destination FQDN is logically divided into multiple domain levels (e.g., www.example.com --> com, example, www), each of which is encrypted independently. Each layer of the network may decrypt only the domain segments it is authorized to process.

Example Structure

A destination domain such as www.example.com is encrypted as a sequence of encrypted domain labels:

Encrypted_Domain = Encrypt_Level3("www") + Encrypt_Level2("example") + Encrypt_Level1("com")

A national-level gateway may only decrypt the Level1 component (e.g., .com, .gov.cn) for regulatory filtering, while enterprise gateways may decrypt Level2 or deeper subdomains within their own namespace.

Design Guidelines

* DART headers SHOULD include structured, layered domain representations, encoded in a format that enables per-level processing (e.g., TLV or label stacks).

* Encryption MAY use symmetric or asymmetric cryptographic schemes, depending on the trust model and deployment scale.

* Each gateway or policy node holds only the decryption keys for its authorized domain layers, enabling partial semantic visibility.

* The encryption hierarchy aligns with domain delegation and administrative boundaries, not physical routing paths.

Benefits

This approach enables:

* End-host privacy: Prevents exposure of full FQDNs to unauthorized intermediate nodes.

* Scoped policy enforcement: Enables intermediate devices to make decisions based on locally visible domain levels.

* Granular trust models: Different administrative layers decrypt only the information they are entitled to process.

* Deployment flexibility: Can be combined with routing delegation and overlay mechanisms.

Comparison to Onion Routing

Although inspired by onion routing, DART’s hierarchical address encryption differs in that:

* Decryption layers are based on domain hierarchy, not hop count;

* Partial visibility is a feature, not a byproduct;

* Policy enforcement and routing remain cooperative, not adversarial.

This mechanism provides a scalable compromise between address confidentiality and operational control, supporting a wide range of deployment models including regulatory environments, enterprise segmentation, and cross-domain interoperability.

6.3. NAT-DART-4 and Address Virtualization

To ensure compatibility with IPv4-only hosts that do not support the DART protocol, DART introduces the NAT-DART-4 translation mechanism. At the core of this mechanism is the concept of a virtual address, sometimes also referred to as a pseudo address.

The virtual address serves as an abstraction layer that enables transparent communication between IPv4-only hosts and the DART network, allowing protocol interoperability without requiring changes to legacy systems.

6.3.1. Definition of Virtual Addresses

A virtual address is a pseudo address mapped to a remote FQDN, dynamically allocated by the local DART gateway. It is locally scoped and has no global significance. The introduction of virtual addresses serves two main purposes:

* To assign a unique virtual address to each remote FQDN acting as a communication target, allowing local IPv4-only hosts to distinguish between different remote hosts at the IP layer;

* To ensure that any assigned virtual address, when used as the destination address in an IP packet, can be properly routed to the local DART gateway according to standard IP routing rules.

In theory, any unallocated IPv4 address within the local DART subdomain can be used as a virtual address. However, to reduce conflict and improve deployment consistency, it is recommended to select addresses from a designated private address pool. The virtual address pool should meet the above routing requirement and maintain logical isolation from the public network.

Considering factors such as availability, standard compliance, and deployment safety, the address block 198.18.0.0/15, reserved by RFC 2544 for benchmarking purposes, is recommended as the primary address pool.

If additional address space is needed (e.g., in carrier-grade environments), a subnet from 100.64.0.0/10, reserved by RFC 6598 for Carrier-Grade NAT (CGN), may also be used as a virtual address pool-provided it does not overlap with actual CGN usage in the deployment environment.

6.3.2. Allocation and Reclamation of Virtual Addresses

Virtual addresses are dynamically allocated upon network access and reclaimed appropriately to prevent exhaustion of the virtual address pool. The process includes:

* For access requests initiated by local IPv4-only hosts, the DART domain’s DNS server dynamically allocates virtual addresses when responding to the corresponding DNS queries;

* For access initiated by remote hosts targeting local IPv4-only hosts, the DART gateway’s forwarding module shall allocate the corresponding virtual address prior to forwarding;

* The system must maintain mappings between FQDNs and virtual addresses along with ports. Due to the presence of NAT, the source port of packets passing through NAT may not remain fixed (e.g., to port 0xDA27), thus port mapping is a critical component for connection identification;

* Time-to-live (TTL) or least-recently-used (LRU) policies can be employed to age and evict mappings, ensuring the mapping table remains timely and space-efficient;

* Different combinations of remote domain names and ports shall map to distinct virtual addresses to enable the gateway to accurately distinguish and identify traffic destined for various target hosts.

6.3.3. NAT-DART-4 Workflow

Scenario 1: Local IPv4-only Host Initiates Access

1. The initiator is a local IPv4-only host;

2. The target resides in a remote DART domain;

3. The local host issues a DNS query, and the DNS response returns a virtual address allocated from the virtual address pool (not the gateway’s actual ingress address);

4. Since the virtual address is not assigned to a real host, according to the local host’s routing rules, packets destined to this virtual address are sent to the default gateway, i.e., the local DART gateway;

5. The gateway looks up the mapping for the virtual address and restores the real target FQDN;

6. The gateway inserts the DART header, encapsulates the packet, and forwards it to the target DART domain.

Scenario 2: Remote Host Initiates Access to Local IPv4-only Host

1. The initiator is a remote host, targeting a local IPv4-only host;

2. The remote host communicates using the real target FQDN;

3. Packets arrive at the local DART gateway, which looks up the mapping of the FQDN plus port to a virtual address; if no mapping exists, the gateway dynamically allocates one;

4. The allocated virtual address is used as the source address in the restored IP packet, while the destination address is the real IP of the local IPv4-only host;

5. The gateway forwards the packet to the local IPv4-only host;

6. Return packets sent by the local IPv4-only host are addressed to the virtual address; the gateway restores the original FQDN based on the mapping and forwards the packets back to the remote host.

Notes

* The virtual address mechanism enables local IPv4-only hosts to distinguish different remote hosts, supporting bidirectional communication;

* The DART gateway manages mappings between virtual addresses and FQDNs;

* This mechanism is compatible with NAT and existing network architectures, ensuring transparent and efficient communication.

6.3.4. Oversized Packet Handling

During the encapsulation of IP packets, NAT-DART-4 inserts a UDP header and a DART header, which may cause the resulting packet to exceed the MTU.

To ensure reliable delivery, the following mechanisms are proposed for handling oversized packets. They are presented in the order of recommendation priority.

6.3.4.1. MSS Clamping (Recommended)

In TCP connections, the DART gateway or edge host may adjust the MSS (Maximum Segment Size) during the TCP handshake to reserve enough space for the DART header.

Advantages:

* Proactively avoids MTU violations.

* Requires no runtime signaling.

* Supported by most TCP stacks and firewall/NAT equipment.

Drawbacks:

* Applicable only to TCP traffic.

* Requires inspection or modification of TCP handshake packets.

Use Cases:

* Ideal for hosts or gateways initiating TCP sessions.

* Common in access networks or CPE devices that implement TCP MSS rewriting.

6.3.4.2. Path MTU Discovery (PMTUD)

The network can rely on standard Path MTU Discovery mechanisms, in which intermediate routers generate ICMP "Packet Too Big" messages to inform the sender.

Advantages:

* Protocol-agnostic (works for TCP, UDP, etc.).

* Standards-compliant and widely supported.

Drawbacks:

* ICMP filtering may prevent delivery of "Packet Too Big" messages.

* Initial packet loss occurs before adjustment.

* Not reliable across all firewalls or carrier-grade NATs.

Use Cases:

* Suitable for end-to-end communication over networks with proper ICMP support.

* Best used in environments with known ICMP reliability.

6.3.4.3. MTU Increase by Network Provisioning

Network operators may provision larger MTU values on key network segments (e.g., 1600 bytes) to accommodate DART headers and ensure encapsulated packets fit.

Advantages:

* Transparent to endpoints.

* Works with all protocols and traffic types.

Drawbacks:

* Requires administrative changes to network infrastructure.

* May not be feasible in all environments (e.g., across the public internet).

Use Cases:

* Effective in controlled enterprise or data center environments.

* Suitable for backbone links or intra-domain DART overlay networks.

6.3.4.4. Packet Fragmentation (Least Preferred)

DART gateways may fragment oversized packets during forwarding, using either IP-layer fragmentation or application-layer splitting.

Advantages:

* Ensures delivery even when no other mitigation is available.

Drawbacks:

* Fragmentation introduces processing overhead and reliability issues.

* IPv6 discourages in-network fragmentation.

* May interfere with middleboxes or degrade performance.

Use Cases:

* Emergency fallback when other methods fail.

* Use with caution in highly heterogeneous or unmanaged networks.

Summary

MSS Clamping is the most effective and widely compatible method for avoiding DART-related MTU issues, particularly for TCP traffic. PMTUD serves as a secondary method but depends on ICMP reliability. Raising MTU provides the best experience but requires network-wide coordination. Fragmentation should be used only as a last resort due to performance and compatibility concerns.

6.4. NAT44

For IP packets originating from a subdomain and destined for a public IPv4-only host, the DART gateway may optionally perform NAT44 encapsulation, treating the packet as an exception path for forwarding.

Due to the technical constraints of NAT44, the subdomain (i.e., the downstream domain of the DART gateway) is required to use private IPv4 address space for internal address assignment.

NAT44 is a mature and widely deployed legacy technology. Although it is not part of the DART protocol specification, it is fully compatible with DART. In practice, enabling NAT44 on DART gateways-especially those located at the network edge-can facilitate access to public IPv4-only destinations and promote a smoother and more incremental transition toward DART deployment.

6.5. Routing Autonomy

The DART protocol preserves full end-to-end semantics. The source host can determine whether the destination supports DART by examining the CNAME prefix in the DNS response, without relying on intermediate hops or centralized path computation. Path selection is driven by the DNS control plane and the distributed routing tables maintained by DART gateways, forming a decentralized routing control loop.

DART employs a hierarchical domain-based architecture, where each DART domain (i.e., DNS domain) operates as an independent routing autonomy. Within each domain, the DART gateway maintains and updates the routing table autonomously. This structure supports incremental deployment, hierarchical scalability, and high controllability.

Although DART does not rely on a centralized controller for routing management, it does not preclude the optional use of centralized components in specific scenarios-for example, to optimize inter-domain paths, distribute policy updates, or provide global visualization capabilities. The design and deployment of such mechanisms are left to implementers and operators and are outside the scope of this document.

6.6. Implicit Source Routing Capability

The DART protocol uses fully qualified domain names (FQDNs) with semantic structure as address identifiers, allowing the source host to indirectly guide the forwarding path by selecting different destination domain names without explicitly specifying each hop. This mechanism forms a capability similar to "implicit source routing," granting the endpoint partial control over path selection.

Within the DART network, gateways make forwarding decisions based on local inter-domain routing tables and DNS query results. Due to the hierarchical naming nature of DNS, different target FQDNs-even if pointing to the same physical host-may logically belong to different parent domains or path structures, resulting in variations in forwarding paths. Provided these FQDNs and their resolution records are pre-configured in DNS and DART routing policies, the source host can realize logical path guidance by choosing among multiple FQDNs.

For example, the same host may be accessed via host1.city-a.example.dart or host1.city-b.example.dart, which respectively steer traffic through different subdomain gateways, enabling path diversity. This design does not rely on explicit routing headers or centralized controllers but establishes a logically guided system with predictable and controllable path behavior through the interplay of DNS and distributed DART gateways.

Essentially, this mechanism embodies a semantics-driven path selection model that combines the flexibility of source routing with the autonomy of a distributed architecture, akin to "implicit loose source routing." Path control depends on the naming structure and deployment policies, lacking on-demand dynamic path specification but offering substantial policy flexibility.

It is important to note that the DART architecture generally adheres to a "single parent domain principle," where each DART gateway logically belongs to only one parent domain, maintaining a tree-like hierarchical structure. However, in certain practical scenarios, the local DART gateway may need to support multiple parent domain interfaces to enable multi-path access. In such cases, this locally breaks the "single parent domain" rule by allowing the gateway to connect to multiple parent domains simultaneously, thereby providing multiple logical path entries for the end host.

While this extension increases gateway topology complexity, it significantly enhances path redundancy and flexibility. Therefore, the DART architecture should accommodate such multi-parent domain access needs while preserving the overall clarity of the hierarchical structure. Relevant details and specification refinements are left for further discussion in future work.

Through the above mechanism, DART not only improves path controllability and resilience but also empowers endpoints with richer policy options without requiring protocol extensions, demonstrating the notable advantages of a naming-based network architecture.

7. NAT Traversal and Compatibility

To ensure compatibility with existing network environments, DART encapsulates packets over UDP, providing limited adaptation to mainstream NAT behaviors-especially outbound connection mapping mechanisms. This approach lowers deployment barriers and facilitates smooth transition. However, DART’s design philosophy does not simply accommodate the existence of NAT; instead, it aims to fundamentally eliminate the long-term necessity of NAT. Therefore, DART does not pursue aggressive traversal techniques such as UDP hole punching to forcibly penetrate all types of NAT. Rather, DART adopts a moderate compatibility stance without excessive compromise, applying appropriate upgrade pressure on certain NAT gateways and guiding network infrastructure evolution toward the DART architecture.

7.1. NAT Background

Network Address Translation (NAT [RFC4787], [RFC5128]) has been widely deployed in enterprise and home networks to alleviate the IPv4 address exhaustion problem. By modifying packet headers and maintaining connection mappings, NAT allows multiple internal hosts to share a single public IPv4 address.

However, NAT breaks the original end-to-end communication model of the Internet, especially introducing additional complexity and protocol incompatibility issues for applications requiring inbound connections or using non-standard protocols. Therefore, NAT is both a technical compromise and an architectural trade-off.

7.2. Compatibility Encapsulation: UDP

To facilitate deployment in existing network environments during the transition phase, DART packets are encapsulated within UDP datagrams. This approach allows DART traffic to traverse NAT gateways, firewalls, and other intermediate devices without requiring modifications to existing network equipment.

DART currently uses UDP destination port 0xDA27 (decimal 55847) for encapsulated communication.

Encapsulation over UDP provides the following compatibility advantages:

* Supports outbound NAT traversal: most NAT gateways permit internal devices to initiate outbound UDP connections and dynamically establish return paths;

* Compatible with firewall policies: UDP traffic is generally allowed in the outbound direction;

* Avoids interference from intermediate devices: DART packets are treated as ordinary UDP traffic, reducing the risk of filtering due to unknown protocols.

7.3. Compatibility with Existing Infrastructure

Because DART packets are encapsulated within standard UDP datagrams, from the perspective of NAT gateways, firewalls, and routers, their behavior is indistinguishable from ordinary UDP traffic. Therefore, DART can operate over the current IPv4 network without requiring modifications to existing devices or network architectures, even when the network includes multiple layers of nested NAT.

* Naturally, under this mode, DART hosts remain subject to NAT behavior limitations. For example, hosts can only initiate connections actively; to accept incoming connections, IP and port mappings must still be configured on NAT gateways.

* This characteristic allows DART devices to be deployed relatively freely within existing networks. Even if upstream devices have not yet adopted the DART protocol, underlying hosts can connect first, thus enabling a "transparent, incremental" evolutionary deployment path.

7.4. Design Position on NAT Traversal Techniques

Traditional UDP applications often use NAT traversal techniques such as UDP hole punching, STUN, and TURN to establish peer-to-peer connections. However, these approaches are not aligned with the design goals of the DART protocol.

The core objective of DART is to localize the scope of IP addresses, enabling virtually unlimited address space reuse and thus providing an abundant supply of addresses. This reduces the necessity and relevance of traditional NAT gateways.

Therefore, implementers of the DART protocol should not incorporate NAT traversal techniques (e.g., UDP hole punching) within the protocol. Although such techniques remain practical in conventional networks, they are considered outdated within the DART architecture. The focus of research and implementation should be on promoting native DART network deployment and minimizing dependency on NAT, rather than continuing to provide compatibility adaptations for NAT.

By reducing intermediate-layer processing and simplifying the protocol stack, DART implementers can more easily achieve efficient and controllable end-to-end communication, truly restoring the original Internet principle of direct connectivity.

8. Deployment and Evolution Strategy

One of the design goals of the DART protocol is to introduce virtually unlimited address space and domain name-based addressing capabilities without disrupting the operation of the existing IPv4 network, while maintaining compatibility with current IPv4 access. To this end, DART offers a gradual deployment path that adapts to today’s highly complex and heterogeneous Internet environment.

8.1. Technical Requirements and Deployment Prerequisites

The DART protocol leverages the DNS system as its control plane. Addressing, domain name resolution, and host registration within DART rely heavily on DNS coordination. Therefore, before deploying DART gateways and DART-ready hosts, the following technical requirements must be met:

DNS System Capabilities

* Authoritative Domain Delegation: Each DART subdomain must be registered with the DNS system through authoritative delegation. Upon deploying a DART gateway, operators need to delegate the corresponding subdomain in the parent DNS servers to map domain names to the DART gateway.

* Bidirectional FQDN-to-Address Mapping: The DNS system must support correct mapping from host FQDNs to DART addresses (A records or virtual addresses), as well as support reverse PTR queries to enhance manageability.

* Dynamic Registration and Automation: To facilitate dynamic scaling and elastic deployment of the DART network, DNS systems should provide interfaces for automatic registration (e.g., DNS UPDATE, API, or protocol extensions) allowing hosts and gateways to register their domain information upon connection.

DART Gateway Requirements

* DNS Control Plane Integration: DART gateways should actively register their subdomains, maintain virtual address mapping tables, and periodically synchronize resolution states with DNS.

* Independent Management: Gateways must independently manage their subdomain address and naming spaces.

* Full DART Encapsulation/Decapsulation: Gateways shall implement complete DART encapsulation and decapsulation logic, including support for NAT-DART-4 (optional) and DART packet forwarding.

* Parent Domain Uplink: Network interfaces must be properly configured with parent domain information to ensure correct inter-domain routing.

DART-Ready Host Requirements

* DART Packet Support: Hosts must be capable of generating and parsing DART packets, either through protocol stack upgrades, user-space daemons, or APIs.

* DNS Query Capability: Hosts should be able to actively query DNS for DART address information and establish connections accordingly.

* FQDN Registration: To receive incoming connections, hosts must be able to register their FQDNs within their respective DART subdomain DNS.

Subsequent sections will describe typical deployment models for national-level DART gateways, ISP internal DART gateways, edge gateways (home and enterprise), and hosts. Prior to device deployment, ensure DNS system capabilities meet these requirements and establish unified domain and address management coordination within the organization.

8.2. Deployment of DART-Ready Hosts

8.2.1. Capability Requirements for DART-capable Hosts

DART-capable Host refers to a network node that implements the DART protocol stack, including both public servers and end devices. Such a host MUST be able to generate and parse DART packets.

Deployment of DART capabilities can be achieved through, but is not limited to, the following approaches:

* upgrading the operating system kernel or network protocol stack;

* installing a user-space daemon;

* integrating the DART protocol stack via an SDK or API.

A DART-capable host MUST be able to communicate directly using Fully Qualified Domain Names (FQDNs), without relying on fixed IP addresses, thereby enabling elastic deployment, automatic addressing, and cross-domain communication.

Due to the compatibility properties of the DART protocol, a DART-capable host can coexist with existing IP protocols. For example:

* An IPv4-only host, once upgraded to support DART, MUST retain its IPv4 communication capability and continue to access existing Internet services without requiring the presence of a DART gateway.

* Similarly, an IPv6-only host upgraded with DART capability MUST retain its IPv6 communication capability.

This property of "forward enhancement and backward compatibility" ensures that host devices MAY be upgraded independently of changes to the underlying network infrastructure, thereby supporting flexible and incremental deployment of DART.

8.2.2. DART-capable Public Servers

The deployment of a DART network relies on the support of public servers. As providers of fundamental Internet services, upgrading public servers to DART-capable hosts is a prerequisite for enabling inter-domain communication and efficient utilization of address space.

In the following cases, public servers MUST provide native DART protocol support:

* When a large region (e.g., a country) upgrades to an independent DART subdomain, the DART gateways connecting this subdomain to the global Internet CANNOT enable NAT44 (which requires private internal addressing) or NAT-DART-4 (which requires an excessively large pool of pseudo addresses). In such cases, servers on the global Internet MUST support DART for hosts in the subdomain to reach them.

* Because the range of access sources is highly diverse, public servers SHOULD NOT rely on protocol conversion software (e.g., DartWinDivert, essentially a host-based NAT-DART-4 implementation).

* In these circumstances, only native DART support on servers MUST be used.

Although the number of public servers is large, the diversity of operating systems they run is limited. Therefore, upgrading them is considered manageable. Prioritizing DART support in mainstream server operating systems and cloud service platforms MAY provide a stable foundation for expanding the DART ecosystem. Upgrading public servers WILL improve overall DART compatibility and interoperability.

After upgrading to DART readiness, a public server MUST update its DNS zone to signal this capability. Specifically, the authoritative DNS server for the domain SHOULD insert a CNAME resource record with the prefix "dart-host." before the final A record is returned.

For example, when a client queries www.example.com, and the host has been upgraded to DART readiness, the DNS response would be structured as follows:

```
www.example.com.            CNAME    dart-host.www.example.com.
dart-host.www.example.com.  A        A.B.C.D
```

This mechanism ensures compatibility with existing DNS standards, while providing differentiated behavior for DART-capable and non-DART-capable clients in a single query:

* A non-DART-capable client will process the A record and obtain the correct IPv4 address (A.B.C.D).

* A DART-capable client will additionally detect the CNAME record prefixed with "dart-host." and thereby recognize that the queried host supports DART.

8.2.3. DART-capable End Devices

End devices typically access the network through a Customer Premises Equipment (CPE). If the CPE is DART-ready, it must implement DHCP-DNS coordination to meet the control-plane requirements of the DART protocol. When NAT-DART-4 is enabled, all IPv4-only end devices connected beneath it can appear as DART-ready terminals and support bidirectional communication. If NAT44 is also enabled, access to non-upgraded public servers is preserved.

From an architectural perspective, upgrading the CPE is the ideal choice. Technically, if the device has sufficient performance and supports software upgrades, software upgrading should be prioritized; otherwise, hardware replacement is required, which incurs higher costs. If necessary, NAT-DART-4 can also be enabled at a slightly higher network level to bypass the need for upgrading the CPE.

As a result, the upgrade of end devices is NOT an immediate requirement. Such upgrades MAY be introduced incrementally after other parts of the network are largely deployed.

8.3. Deployment of DART Gateways

Each deployment of a DART gateway that establishes a new subdomain is allocated a complete IPv4 address space (i.e., 2^32 addresses) within that domain. Therefore, only a limited number of DART gateways are required to effectively mitigate the problem of IPv4 address exhaustion.

8.3.1. National-level DART Gateway

A national-level DART gateway is deployed at the Internet entry/exit points of a country (e.g., international link exits) and functions as the backbone node for inter-domain communication.

Its functions and constraints are as follows:

* A national-level DART gateway MUST perform forwarding of DART packets and inter-domain address resolution.

* A national-level DART gateway MUST NOT use NAT44 (which requires private internal addresses) or NAT-DART-4 (which requires an excessively large pseudo address pool).

* After deployment, the internal network MAY implement local IPv4 address reuse and address planning.

Deployment conditions:

* Public servers SHOULD have completed DART upgrade

* There must be a suitably positioned DART gateway that has enabled NAT-DART-4 to provide address translation services for accessing IPv4-only terminals.

Behavior after deployment:

* End devices within the domain, regardless of upgrade status, MAY access the DART network.

* Hosts inside the domain CAN NOT directly access servers outside the domain that have not been upgraded.

* Access to non-upgraded external hosts MAY be provided via HTTP or SOCKS proxies; however, this method MUST ONLY be used as a temporary solution.

8.3.2. DART Gateways within the Operator Network

To address IPv4 address exhaustion, many network operators have widely deployed CGNAT (Carrier-Grade NAT) mechanisms within their internal networks. Typical CGNAT topologies often involve two layers of NAT mapping: the Customer Premises Equipment (CPE) at home performs the first NAT, while the operator’s CGNAT gateway performs the second. In some smaller operators, community broadband, rural networks, campus networks, or public Wi-Fi networks, there may even be more layers of nested NAT.

Although multi-level NAT alleviates address shortages, it introduces numerous negative impacts such as blocked inbound connections, frequent failures of UDP traversal techniques, difficulty in tracking user behavior, and severe degradation of P2P performance. The DART protocol offers a more natural and scalable alternative in such deeply nested NAT environments.

Operators can upgrade existing CGNAT gateways to DART gateways to obtain a full IPv4 address space and significantly simplify the network architecture. Possible Deployment Paths:

There are two main deployment paths for operators when introducing DART:

1. Upgrading only the top-level NAT gateway to a DART gateway

    * The operator MAY immediately obtain the full IPv4 address space (2^3^2) and downgrade lower-level NAT gateways to ordinary IP routers.

    * This approach involves the shortest deployment path and minimal device modifications; however, because the lower-level networks previously used private addresses, the operator MUST reassign addresses to all downstream end devices, which may incur address reallocation costs and operational overhead.

    * This path SHOULD assume that public servers and CPE gateways have already been upgraded to DART-capable.

2. Full upgrade to a hierarchical DART gateway structure

    * The operator MAY replace all NAT gateways at multiple layers with DART gateways, maintain each layer as an independent subdomain, and continue using the existing address allocation logic.

    * Since the DART protocol eliminates the distinction between "private" and "public" addresses, each subdomain MAY have the full address space, without concerns for address conflicts or the need to retain traditional private address concepts.

    * This approach facilitates hierarchical management and incremental deployment, requires a limited number of DART gateways, does not significantly increase overall deployment costs, and provides scalability for future connections to the DART core domain.

* During the upgrade transition, DART gateways MAY continue to enable NAT44 to maintain the existing multi-level NAT architecture, and MUST disable NAT44 once public servers and CPEs are upgraded.

8.3.3. Home Gateways

At the network edge, home gateways can adopt one of three configurations.

8.3.3.1. Bridging as a Layer 2 Device

After the upstream operator network has been upgraded to support DART, converting a home gateway into a Layer 2 bridge shifts the burden of NAT44 and NAT-DART-4 to the operator-side DART gateways. This setup may lead to multiple households sharing the same Ethernet broadcast domain, posing risks to network performance and security. Therefore, this option has significant deployment drawbacks and is generally not recommended.

8.3.3.2. Operating as a Layer 3 Router (with NAT disabled)

After the ISP completes the DART network upgrade, home gateways can operate purely as IP routers, with their address ranges uniformly assigned by the ISP. If they continue to serve as DHCP servers, they must first obtain the subnet assigned by the ISP, which may limit user autonomy over home gateway configuration and increase management and maintenance complexity.

At this stage, NAT44/NAT-DART-4 must be performed on the ISP’s internal DART gateways. The advantage is that CPEs do not require large-scale upgrades. However, care must be taken not to place the DART gateway too high in the topology; otherwise, the number of external hosts accessed by downstream devices may exceed the capacity of the pseudo-address pool.

8.3.3.3. Upgrading to a DART Gateway

In this configuration, the home gateway is upgraded to support the DART protocol stack and independently maintains a DART subdomain. It connects upstream to the operator’s DART network via its parent domain interface. Internally, it may enable NAT-DART-4 to allocate virtual addresses for legacy IPv4-only devices, ensuring seamless compatibility.

This solution offers several advantages:

* Autonomy and flexibility: The home gateway manages its local address space and FQDN mappings independently, allowing users to configure policies without relying on the operator;

* Legacy device compatibility: IPv4-only devices can connect without any modification;

* Support for early deployment: Even if the operator’s network has not yet been upgraded, the gateway can still enable traditional NAT44, allowing the home network to transition to DART ahead of its upstream infrastructure;

* Strong scalability: As an independent subdomain, the gateway can host local services and smart devices, supporting both local and cross-domain communication;

* Reduced upstream load: DART encapsulation and address translation are handled locally, offloading processing pressure from the operator’s DART gateways.

Considerations: - The device must support the DART protocol stack, which may require hardware or software upgrades; - The upstream interface must be configured with the parent domain information provided by the operator to enable correct inter-domain routing.

Security Protection and Firewall Capabilities

After a home gateway is upgraded to support the DART protocol, internal hosts will be assigned publicly routable addresses, making access control mechanisms essential. As the critical node connecting the home network to the external world, the gateway should incorporate basic security features to protect household devices from external attacks and unauthorized access.

* The firewall should automatically manage traffic entering and leaving the home network, blocking potential threats without requiring complex user configuration;

* Simple access control options should be provided to help users restrict specific devices or services from connecting to the network;

* Full compatibility with the DART protocol must be ensured to maintain the security of internal devices in DART-enabled environments;

* To enhance the security of home and small networks, the gateway should, by default, block all unsolicited inbound connection attempts from external sources. It is recommended to support a UPnP-like dynamic port mapping mechanism that allows internal hosts to securely request the exposure of specific ports through an authenticated process. This strikes a balance between security and flexibility, simplifying configuration for non-technical users while improving overall network usability and protection.

Users are advised to always keep security features enabled and to install official firmware updates in a timely manner to maintain network stability and protection.

Summary

This configuration offers full backward compatibility while enabling independent, progressive migration to the DART architecture. It balances flexibility, control, and long-term scalability, and is the most strategically valuable deployment path.

8.3.4. Enterprise Gateways

Enterprise networks typically maintain their own internal subnets, address management policies, and security configurations. Within the context of DART deployment, enterprise gateways can adopt similar architectural strategies as home gateways, but with greater flexibility and scalability in address management.

A recommended approach is to upgrade enterprise gateways to support the DART protocol, enabling them to manage an independent DART subdomain and connect upstream to the carrier’s DART domain. This solution offers the following advantages:

* Internal Control and Autonomy: Enterprises can independently manage their DART subdomain, maintain local FQDN mappings, and define routing policies;

* Compatibility with Legacy Devices: IPv4-only endpoints within the enterprise can continue operating without modification via the NAT-DART-4 mechanism;

* Support for Gradual Deployment: Enterprises may adopt DART ahead of their upstream provider. During the transition period, the enterprise DART gateway can still enable NAT44 to maintain compatibility with the legacy IPv4 Internet;

* Improved Scalability and Manageability: As an independent subdomain, the enterprise network can more effectively expand services, enforce fine-grained access controls, and enable cross-domain communication.

Considerations: - DART protocol support must be integrated into the enterprise gateway’s software or hardware platform; - The upstream interface should be configured with parent domain information provided by the carrier to ensure proper inter-domain routing; - Given the typically higher security requirements of enterprise networks, enabling DART introduces public reachability for internal hosts, which may expand the attack surface. It is recommended to integrate firewall functionality within the enterprise DART gateway to strengthen access controls and implement robust security policies.

In summary, upgrading enterprise gateways into DART-compliant nodes facilitates seamless integration with the DART ecosystem, while preserving network autonomy, supporting legacy infrastructure, and enabling a smooth transition toward full DART adoption.

8.4. Incremental Deployment and Backward Compatibility

The DART protocol is designed to support compatibility and progressive evolution, with the following characteristics [RFC5555]:

* It can automatically detect whether the remote peer is a DART-capable device without requiring additional signaling negotiation;

* All DART-capable nodes inherently support dual-stack operation and are fully compatible with traditional IP;

* Through the NAT-DART-4 mechanism, IPv4-only hosts can be proxied by a local DART gateway and appear externally as DART-capable nodes, enabling full bidirectional communication within the DART network;

* Edge DART gateways can coexist with traditional NAT44 setups, allowing IPv4-only hosts in the local network to access legacy servers on the public Internet without modification;

* Crucially, IPv4-only devices under an edge DART gateway can remain unmodified indefinitely while still maintaining full communication capabilities over the DART network. This ensures long-term compatibility for non-upgradable endpoints.

From a deployment strategy perspective, the DART network prefers that public servers are upgraded first to DART-ready hosts and CPEs are upgraded to DART-ready gateways (each proceeding independently,

without mutual dependency). Based on this, the upgrade sequence for other network infrastructure and end devices can be flexibly determined by users or operators, without strict dependency constraints. DART does not require full-scale upgrades of end devices; instead, it leverages the capabilities of edge gateways to allow legacy devices to continue operating, ensuring a smooth transition for users.

Thanks to these features, DART supports incremental, on-demand deployment. Even partial upgrades do not break existing connectivity or disrupt Internet services. The upgrade process is therefore inherently smooth, allowing both users and operators to adopt DART at their own pace and without requiring a wholesale migration of their network environment.

8.5. Long-Term Evolution Path

From a long-term perspective, the DART protocol offers the following potential directions for evolution and technical advancement:

* Progressive replacement of public IPv4 dependency
  DART establishes a name-centric addressing architecture, enabling end-to-end communication without the need for globally assigned public IP addresses or centralized address allocation.

* Full compatibility with IPv6
  DART can coexist and interoperate with IPv6-only networks, supporting dual-stack deployment and integrated forwarding in multi-protocol environments.

* Support for global unified addressing
  DART provides a consistent abstraction layer that facilitates seamless communication between IPv4 and IPv6 networks, paving the way for a globally unified, name-based interconnection model.

* Significant potential for global routing optimization

  - Theoretically unbounded DART address space eliminates the fragmentation and inefficiencies caused by IPv4 scarcity and micro-prefix allocations.

  - Within each DNS domain, address resources are abundant, allowing for highly aggregated allocations without the need for hierarchical compression.

- In typical deployment scenarios, based on current DNS domain granularity and expected density, both global DART routing tables and intra-domain IP routing tables can be maintained at a manageable scale-expected to stay under 10^3 entries.

* Transformative changes in network architecture
  DART inherently promotes the creation of new autonomous subdomains, leading to a distributed and hierarchical address management structure.

* Gradual phasing out of traditional NAT
  Conventional NAT gateways will be replaced by DART gateways, which provide address translation and domain isolation in a more flexible and efficient manner.

* Enhanced network security capabilities
  By leveraging FQDN-based identity and access control, DART enables stronger communication authentication and fine-grained policy enforcement.

* Balance between authentication and privacy
  DART has the potential to offer user privacy protection through dynamic naming, proxy-based forwarding, and encryption mechanisms-while maintaining reliable identity validation.

* Support for innovative applications and service models
  The flexible name mapping and dynamic routing of DART could drive the development of novel use cases such as name-based content delivery, collaborative edge computing, trusted IoT communication frameworks, and intelligent network management with QoS guarantees and routing optimization.

This evolution path provides a practical and forward-compatible roadmap for DART to gradually replace traditional IP architecture and support the foundation of the next-generation Internet infrastructure.

8.6. Native FQDN-Based Policy Enforcement

8.6.1. Background

In traditional IP-based networking, policy enforcement devices such as firewalls, routers, or gateways rely primarily on IP addresses to match and apply rules. This model presents the following limitations:

* A single service domain may resolve to different IP addresses over time (e.g., CDNs, multi-site deployments).

* Domain names are typically visible only at the application layer, making it difficult for network-layer devices to inspect or control access based on FQDN.

* The widespread adoption of encrypted DNS (e.g., DoH, DoT) and TLS 1.3 with Encrypted Client Hello (ECH) further undermines traditional DPI and SNI-based filtering.

* Network operators and security systems require the ability to define access policies based on semantic identifiers such as FQDNs, rather than ephemeral IP addresses.

The DART protocol addresses this challenge by natively integrating the destination Fully Qualified Domain Name (FQDN) into its addressing and routing model, enabling in-network devices to perform FQDN-aware policy enforcement without reliance on external inspection.

8.6.2. Protocol-Level Support

Each DART packet includes an explicit, structured destination FQDN field (see Section 5), which is accessible at the network layer and actively involved in routing and forwarding decisions. This design provides:

* Protocol-native domain visibility for in-path devices, eliminating the need for DNS interception or TLS header inspection.

* Structured semantic identifiers for destination hosts, enabling direct matching and tagging.

* Reliable domain-based control, independent of IP address volatility or encryption-related obfuscation.

* First-packet visibility, allowing immediate policy application without requiring prior DNS resolution.

8.6.3. Use Cases

The FQDN-aware nature of DART packets enables a wide range of policy applications:

* Domain-Based Access Control Gateways or firewalls can allow or block traffic based on explicit domain rules, without relying on third-party DNS logs or IP blacklists. For example, enterprise networks may block *.gambling.com and *.malware.site while permitting *.edu.cn or approved business domains.

* Granular Traffic Steering Traffic can be routed or prioritized based on the destination domain. For example, video content from cdn.example.com may be directed to a high-bandwidth path, while update traffic to update.device.example.com may be assigned lower priority.

* Fine-Grained Access Policy Enterprises can enforce access rules at the subdomain level-for instance, allowing access to vpn.internal.example.com and docs.portal.example.com, while restricting mail.personal.example.com or video.external.example.com.

* Domain-Based Auditing and Billing DART gateways can classify and count traffic by destination domain, enabling security auditing, usage analytics, or differentiated billing by service category.

* Regulatory Compliance and Governance Government or carrier-grade DART gateways can implement domain-based regulatory policies in real time, without relying on unstable IP blacklists or higher-layer protocol signatures.

* Security Policy Reinforcement Traffic to known malicious domains (e.g., botnet C2 servers, phishing domains) can be identified and blocked natively-e.g., *.botnet.example, stealer.login.xyz-based on FQDN intelligence feeds.

8.6.4. Deployment Guidelines

To leverage FQDN-based policy enforcement, DART-aware devices SHOULD support:

* FQDN-based rule engines, including:

* Exact match: www.example.com

* Wildcard match: *.example.com

* Regex or domain suffix match: .*.edu.cn

* Structured FQDN visibility in DART packet headers, to facilitate efficient lookup and classification.

* Partial address visibility in encrypted modes: When address encryption is enabled, domain-aware policy devices MAY be configured to inspect only locally visible (e.g., intra-domain) FQDN segments.

* Dynamic policy updates, enabling rapid adaptation to new blacklists, whitelists, or domain-based QoS rules.

8.6.5. Comparison with Legacy Systems

Feature	Traditional IP Networks	DART-Based Networks
Policy Matching Basis	IP Address	FQDN
DNS Visibility	Passive/sniffed	Native to protocol
TLS Impact (e.g., TLS 1.3 + ECH)	May block visibility	Unaffected
Blocking Precision	Low (IP instability)	High (domain-specific)
Management Overhead	High (IP tracking/ updating)	Low (stable domain semantics)

Table 3: Comparison with Legacy Systems

8.6.6. Summary

By treating domain names as first-class routing entities, DART enables native FQDN-based policy enforcement at every stage of the forwarding path. This enhances the security, controllability, and regulatory compliance of the network while reducing the reliance on complex DPI or opaque side-channel analysis. Implementers are encouraged to leverage this capability to improve operational visibility and policy expressiveness in both public and private networks.

9. IANA Considerations

This section specifies the numeric resources required by the DART protocol during deployment and formally requests or reserves them for IANA registration and allocation.

9.1. UDP Port Assignment

The DART protocol relies on UDP as its transport encapsulation format. Therefore, an officially assigned UDP port number is required to identify DART packets [RFC8126].

* Name: DART
* Transport Protocol: UDP
* Suggested Port Number: 0xDA27 (decimal 55847)
* Usage Description: DART packets are encapsulated in UDP to support NAT traversal and compatibility with existing IP networks. All DART-capable gateways and hosts listen on this port.

Note: UDP port 55847 is currently used for experimental purposes. Once the protocol proceeds through the IETF standardization process, an official Well-Known Port (WKP) under 1024 will be formally requested from IANA.

9.2. DNS CNAME Prefix Conventions (Non-IANA)

To indicate whether a host supports the DART protocol, special CNAME prefixes in DNS responses are used. This draft proposes the following naming conventions:

* dart-host.: Indicates that the target host is DART-capable and can receive DART-encapsulated packets directly.
* dart-gateway.: Indicates that the target host resides in another DART subdomain; packets should be forwarded to the local DART gateway. The associated A record provides the gateway’s IP address.

These prefixes do not require formal IANA reservation but are recommended for documentation and community consensus to avoid conflicts with other protocols.

9.3. IP Protocol Number (Optional/Future Use)

Currently, DART packets are encapsulated within UDP and do not require a new IP protocol number. However, if in the future the community seeks to transport DART directly at the IP layer (e.g., as a new L3 protocol), a new IP protocol number may be requested from IANA.

* Current Status: No new IP protocol number requested.

* Future Extensibility: Reserved as an option for future community-defined extensions.

10. Security Considerations

The DART protocol introduces a name-based addressing model and message encapsulation format. Its overall security depends on the integrity of the DNS system, the behavior of DART gateways, and the processing of DART packets. This section outlines potential security threats and preliminary mitigation strategies.

10.1. Risks Related to DNS

DART relies on DNS responses (particularly CNAME and A records) for address resolution and capability indication. This dependency exposes it to DNS-based attacks:

* DNS spoofing / cache poisoning: An attacker may forge or manipulate DNS responses, tricking clients into sending DART messages to malicious gateways or hosts.

Mitigations:

* Strongly recommend the use of DNSSEC to validate DNS response integrity [RFC4033][RFC4034][RFC4035].

* DART clients should cache DNS results with appropriate TTLs to reduce exposure.

* DART gateways may maintain ACLs or whitelists to restrict valid destination domains.

10.2. Spoofing or Abuse of Virtual Addresses

DART assigns virtual IP addresses to IPv4-only hosts to enable DART-based communication. Attackers might attempt to spoof or predict these mappings, potentially causing [RFC2827]:

* Virtual address collisions,

* Misrouting of traffic to unintended hosts,

* DoS attacks leveraging forged virtual addresses.

Mitigations:

* DART gateways must dynamically assign and manage virtual addresses, ensuring mappings are host-specific and not globally visible.

* Virtual addresses should be drawn from a private, controlled range (e.g., 198.18.0.0/15) to avoid conflicts with real-world addresses.

* Randomized address assignment can help reduce predictability.

Note: Since virtual address mapping is DNS-based, successful spoofing attempts would require DNS manipulation or MitM capabilities-equivalent to classic DNS attacks. Thus, deploying DNSSEC or encrypted DNS (e.g., DoT/DoH) is strongly recommended.

10.3. Injection of Forged DART Packets

Attackers may forge DART-encapsulated UDP packets with false headers in an attempt to manipulate gateway behavior or inject invalid traffic.

Mitigations:

* DART gateways should validate source addresses and maintain ingress filtering.

* If deployed in untrusted environments, consider encapsulating DART traffic using IPsec [RFC4301], QUIC, or similar secure transport layers.

* Gateways may apply basic rate-limiting and session-based admission control to mitigate abuse.

10.4. Inter-Domain Route Spoofing

Because DART relies on domain-based interconnection paths, adversaries may register deceptive subdomains or craft malicious DNS entries to redirect traffic.

Mitigations:

* Parent domain administrators should audit subdomain configurations.

* DNS zone administration must follow traditional best practices with access controls.

* DART deployments may benefit from introducing domain registration policies or a trust framework to validate subdomain authenticity.

10.5. Privacy Exposure of Private Network Hosts

In some environments, private network hosts using DART may include identifiable information (e.g., FQDNs) in outbound messages, potentially exposing user identities in sensitive contexts.

Mitigations:

* DART gateways may be configured to obfuscate or replace source identity fields, acting as privacy-preserving proxies.

* Encrypted DNS (DoT/DoH) and encrypted transports (e.g., TLS, QUIC) are encouraged for protecting name resolution and message confidentiality.

* Gateways may implement firewalls or access controls based on domain, port, and identity attributes.

* Future work may explore anonymization or pseudonymization of DART headers to reduce traceability.

Design Philosophy Note:

DART aims to enable global end-to-end reachability, which inherently requires addressability and identity visibility. Unlike traditional NAT models that rely on obscurity for "implicit security," DART adopts a transparent model with structured namespaces and policy-driven access control.

To be reachable is to be observable; to be connected is to be identifiable.

Security in DART is achieved through encryption, domain-level boundaries, and controllable access-not through concealment. The protocol encourages a shift from "unreachability as security" to "policy-enforced security" as a forward-looking paradigm.

11. References

11.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/rfc/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/rfc/rfc1035>.

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

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/rfc/rfc2136>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/rfc/rfc2181>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/rfc/rfc2328>.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, DOI 10.17487/RFC2663, August 1999, <https://www.rfc-editor.org/rfc/rfc2663>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <https://www.rfc-editor.org/rfc/rfc2827>.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <https://www.rfc-editor.org/rfc/rfc3022>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/rfc/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/rfc/rfc4035>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/rfc/rfc4271>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/rfc/rfc4301>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <https://www.rfc-editor.org/rfc/rfc4787>.

[RFC5555] Soliman, H., Ed., "Mobile IPv6 Support for Dual Stack Hosts and Routers", RFC 5555, DOI 10.17487/RFC5555, June 2009, <https://www.rfc-editor.org/rfc/rfc5555>.

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/rfc/rfc768>.

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/rfc/rfc791>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/rfc/rfc8200>.

11.2. Informative References

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/rfc/rfc4033>.

[RFC4864] Van de Velde, G., Hain, T., Droms, R., Carpenter, B., and E. Klein, "Local Network Protection for IPv6", RFC 4864, DOI 10.17487/RFC4864, May 2007, <https://www.rfc-editor.org/rfc/rfc4864>.

[RFC5128] Srisuresh, P., Ford, B., and D. Kegel, "State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)", RFC 5128, DOI 10.17487/RFC5128, March 2008, <https://www.rfc-editor.org/rfc/rfc5128>.

[RFC8504] Chown, T., Loughney, J., and T. Winters, "IPv6 Node Requirements", BCP 220, RFC 8504, DOI 10.17487/RFC8504, January 2019, <https://www.rfc-editor.org/rfc/rfc8504>.

Appendix A. Example Deployment Topology

A.1. Network Architecture

Assume the following network environment:

* Top-level domain:
  dart.example, serving as the parent domain of two DART gateways. The parent domain interface IPs of gw1 and gw2 are 10.0.0.1 and 10.0.0.2, respectively.

* Subdomains:

  - sub1.dart.example (Subdomain 1), connected to DART gateway gw1. The interface IP of gw1 in this subdomain is 10.1.0.1.

  - sub2.dart.example (Subdomain 2), connected to DART gateway gw2. The interface IP of gw2 in this subdomain is 10.2.0.1.

The connection topology is as follows:

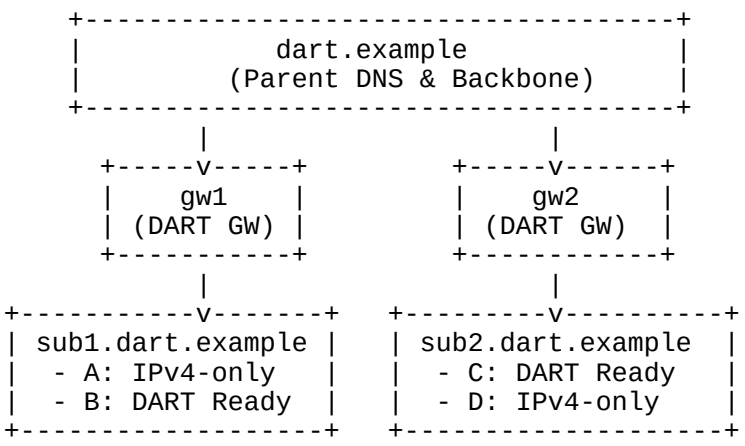

Figure 3: Example Deployment Topology

* A: 10.1.0.11, IPv4-only

* B: 10.1.0.12, DART Ready

* C: 10.2.0.11, DART Ready

* D: 10.2.0.12, IPv4-only

Note: Since DART logically isolates IP address scopes across domains, the three domains above (one parent and two subdomains) may reuse the same IP subnets without collision. Different subnets are used here solely to avoid reader confusion.

In this deployment, gw1 and gw2 each integrate a DHCP server and a DNS server. These services must tightly integrate with the forwarding plane. If implemented as standalone servers, effective information sharing with the DART gateway is essential.

A.2. Procedures

Initialization (DHCP Phase)

* Hosts A/B/C/D obtain IP addresses from their local DART gateway via DHCP.

* DART-ready hosts (B and C) include a special DHCP Option to indicate DART capability.

* DART-ready hosts also use the domain name provided by the DHCP server to construct their own FQDNs.

* IPv4-only hosts (A and D) are unaware of DART and join the network using legacy DHCP processes.

* During this phase, the DHCP server:

  - Allocates IPs, default gateways, and DNS servers;

  - Records whether the host supports DART;

  - Constructs FQDNs for IPv4-only hosts using their hostnames.

DNS Resolution

Each gateway has an embedded DNS server with the following behavior:

* For queries received via the parent domain interface:

  - Always respond with the gateway’s parent domain IP, regardless of the actual destination. This forces the sender to route packets via the gateway.

* For queries received via the subdomain interface:

  - If querying an in-domain host: respond with the actual IP of the target.

  - If querying a host in another domain:

    o If the querier is DART-ready: respond with the gateway’s subdomain interface IP, so the DART packet is sent via the local interface.

    o If the querier is IPv4-only: allocate a virtual IP, bind it to the queried FQDN, and respond with this virtual IP. The IPv4-only host will send the packet to the virtual address, which routes to the gateway.

Forwarding

* For DART packets arriving at the parent domain interface:

  - If the destination is a DART-ready host (B/C), forward via the corresponding subdomain interface.

* For DART packets arriving at the subdomain interface (e.g., sent by B or C):

  - Forward via the parent domain interface.

* In both cases:

  - DART header remains unchanged;

  - IP header source and destination are rewritten appropriately.

NAT-DART-4 Encapsulation (Sender Side)

Example: Host A (IPv4-only) sending to a host in sub2 (C or D):

1. A queries C.sub2.dart.example from its configured DNS server (10.1.0.1).

2. The DNS server determines that C is external to the subdomain and:

   * Allocates virtual address 198.18.1.34 and binds it to C.sub2.dart.example;

   * Returns:
     C.sub2.dart.example. CNAME dart-gateway.sub1.dart.example.
     dart-gateway.sub1.dart.example. A 198.18.1.34

   Although A is IPv4-only and ignores the CNAME, this prefix signals to operators that this is a DART-based mapping and 198.18._._ is a virtual IP.

3. A constructs an IP packet with destination 198.18.1.34 and sends it.

4. The packet is routed to its default gateway gw1 (as the destination is non-local).

5. At gw1, the following actions occur:

   * Detects virtual IP as destination;

   * Inserts DART header and UDP encapsulation;

   * Fills in destination FQDN (C.sub2.dart.example) from DNS mapping;

   * Determines source FQDN (A.sub1.dart.example) from DHCP records;

   * Resolves C.sub2.dart.example via parent domain DNS to obtain 10.2.0.1;

* Sets source IP as 10.0.0.1 (gw1’s parent interface);
* Sends encapsulated packet from parent interface.

6. The packet is routed via IP backbone and arrives at gw2’s parent interface.
7. gw2:
    * Recognizes C as a DART-ready host;
    * Rewrites the IP header (destination: 10.2.0.11, source: 10.2.0.1);
    * Forwards the DART packet via its subdomain interface to C.

NAT-DART-4 Decapsulation (Receiver Side)

Example: Host D (IPv4-only) receives a DART packet via gw2:

1. gw2 receives a DART packet from parent interface, destined for D.sub2.dart.example.
2. Determines D is an IPv4-only host:
    * Allocates a virtual source address;
    * Maps sender’s FQDN to this virtual IP;
    * Strips DART and UDP headers;
    * Constructs a native IP packet:
        - Destination: D’s IP 10.2.0.12
        - Source: virtual address
    * Sends via subdomain interface.
3. D receives the packet and replies to the virtual source IP.
4. gw2 re-encapsulates the reply using the same virtual-IP-to-FQDN mapping.
5. The response is routed back to the original sender through the DART infrastructure.

A.3. Case Walkthrough

Let host A (A.sub1.dart.example, 10.1.0.11, IPv4-only) send a message to host C (C.sub2.dart.example, 10.2.0.11, DART Ready):

1. A queries C.sub2.dart.example via DNS.

2. DNS server returns:

   C.sub2.dart.example. CNAME dart-gateway.sub1.dart.example.
   dart-gateway.sub1.dart.example. A 198.18.1.34

3. A sends packet to 198.18.1.34, which is routed to gw1.

4. gw1 detects virtual destination:

   * Builds DART header with source FQDN A.sub1.dart.example and destination FQDN C.sub2.dart.example;

   * Looks up IP 10.2.0.1 for destination;

   * Sends encapsulated DART packet from 10.0.0.1.

5. Packet is delivered to gw2.

6. gw2 finds that destination C is DART-ready:

   * Forwards DART packet to 10.2.0.11 from 10.2.0.1.

A.4. Summary

This example illustrates a multi-subdomain, mixed-host environment and highlights:

* DNS-driven forwarding behavior;

* Role of virtual addresses in enabling IPv4-only interoperability;

* Encapsulation and decapsulation by DART gateways;

* Cross-domain communication between legacy and DART-ready nodes.

Appendix B. Acknowledgements

The author, Li Shi, is solely responsible for the design and implementation of the DART protocol at the current stage.

During the drafting of this document, the author received valuable support from modern AI-assisted tools, including ChatGPT, DeepSeek, Tongyilingma and GitHub Copilot. These tools played a helpful role in shaping the structure, clarifying concepts, and accelerating document generation.

The author also thanks the IETF community and its working groups for providing an open platform for the development and discussion of new protocols.

Constructive feedback from future reviewers is sincerely appreciated.

Appendix C. Implementation and Availability

A working prototype of the DART protocol has been independently developed and deployed by the author. It is publicly accessible at http://dart-proto.cn

This live system demonstrates the core features of the DART protocol, including DNS-based FQDN redirection, virtual address mapping, DART encapsulation and routing, and NAT-DART-4 conversion.

The source code of the implementation is fully open-source and available on GitHub at the following repositories:

* DART Gateway (Linux-based)
  https://github.com/rancho-dart/dart_forwarder

* DART-ready client for Windows (based on WinDivert)
  https://github.com/rancho-dart/DartWinDivert

The author encourages implementers, researchers, and operators to experiment with the live system, review the source code, and provide feedback to improve the protocol.

Author’s Address

Shi Li
Shanghai
China
Email: lishi.china@qq.com