dtn
Internet-Draft
Intended status: Informational
Expires: 23 July 2026

X. Fan
H. Zhou
J. Qu
Beijing Jiaotong University

19 January 2026

Encapsulation of OpenFlow over Delay-Tolerant Networking (DTN) Using the Bundle Protocol
draft-fan-dtn-openflow-over-bp-00

Abstract

This document specifies a method for carrying OpenFlow messages over Delay-Tolerant Networking (DTN) using the Bundle Protocol (BP). The method encapsulates OpenFlow messages as BP payloads and defines the mapping between OpenFlow messages and Bundles, the payload format, and addressing and multiplexing considerations based on DTN Endpoint Identifiers (EIDs). This document further discusses conditions that may occur on intermittently connected or high-latency links, including fragmentation, duplicate delivery, out-of-order arrival, and expiration, and defines corresponding message handling rules. These rules enable the transmission of OpenFlow messages across DTN without modifying the semantics of the OpenFlow protocol.

Status of This Memo

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

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

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

This Internet-Draft will expire on 23 July 2026.

Copyright Notice

Table of Contents

1. Introduction

OpenFlow is widely used in Software-Defined Networking (SDN) to support communication between controllers and switches, enabling centralized policy distribution. DTN environments similarly require mechanisms for centralized policy dissemination. However, due to the absence of stable end-to-end paths in DTNs, OpenFlow control channels based on TCP/IP cannot operate reliably.

The Bundle Protocol (BP) supports constrained and disrupted network environments through a store-and-forward communication model. BP encapsulates application data into Bundles, employs Endpoint Identifiers (EIDs) for naming and addressing, and enables forwarding across heterogeneous networks via Convergence Layer Adapters (CLAs). These properties make BP suitable for carrying control-plane messages in DTNs.

This document describes how BP is used to encapsulate and transport OpenFlow messages in DTNs. It specifies the mapping between OpenFlow messages and Bundles, the Bundle payload format, and considerations related to EID-based addressing and multiplexing.

This document does not modify the OpenFlow protocol, does not define controller or switch control logic, and does not specify consistency, synchronization, or convergence mechanisms for the SDN control plane in disrupted or delay-tolerant environments. Security considerations are limited to issues introduced by carrying OpenFlow messages over BP and the use of existing BP security mechanisms.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used throughout this document:

Delay-Tolerant Networking (DTN) : A networking architecture designed to operate effectively over links characterized by long delays, intermittent connectivity, or the absence of continuous end-to-end paths.

Bundle Protocol (BP) : The protocol specified by the IETF DTN Working Group for store-and-forward transmission of application data in DTN environments. This document refers primarily to BP Version 7 as specified in [RFC9171], while noting that some deployments may still use BP Version 6.

Bundle : A protocol data unit defined by the Bundle Protocol, consisting of a primary block, zero or more extension blocks, and a payload block.

Endpoint Identifier (EID) : A globally unique identifier used by the Bundle Protocol to name a communication endpoint. EIDs are typically expressed using URI syntax.

Convergence Layer Adapter (CLA) : A protocol adaptation layer that maps Bundle Protocol operations onto specific underlying transport protocols (e.g., TCP, UDP, or LTP).

OpenFlow Controller : The control-plane entity responsible for managing forwarding behavior by communicating with OpenFlow switches using OpenFlow control messages.

OpenFlow Switch : A data-plane element that processes packets according to flow rules received from an OpenFlow controller.

3. Problem Statement and Use Cases

This section describes the problems addressed by this document and outlines representative use cases for carrying OpenFlow control messages in DTNs.

3.1. Problem Statement

DTNs differ from traditional IP networks in that connectivity may be intermittent and an end-to-end path may not exist at any given time. When OpenFlow messages are transmitted across such environments, delivery may be subject to significant delays, and non-ideal behaviors such as duplicate delivery or out-of-order arrival may occur.

For the purposes of this document, the following key issues are considered:

Delay tolerance: Control messages may be delivered only after extended delays, and the existence of a persistent transport session MUST NOT be assumed.

Disruption tolerance: Message delivery MUST tolerate temporary loss of connectivity and rely on store-and-forward mechanisms to complete forwarding.

Non-ideal delivery behavior: Conditions such as duplicate delivery, out-of-order arrival, and expiration may occur. Clear handling rules are therefore required at the OpenFlow message delivery interface.

3.2. Use Cases

The following use cases are illustrative:

Periodic or intermittent connectivity: Controllers and switches communicate over links that are available only at specific times. Control messages are queued and delivered when connectivity becomes available.

Long-delay paths: Control traffic traverses links with long propagation delays (e.g., satellite or deep-space links), where reliance on continuous end-to-end sessions is not feasible.

Disrupted multi-hop forwarding: Control messages are forwarded across multiple hops via relay nodes that provide temporary storage and forwarding capabilities, in order to tolerate disruptions or topology changes.

4. Architecture Overview

This section provides an overview of the system architecture described in this document and its major components. Figure 1 illustrates the logical architecture for encapsulating and transporting OpenFlow control messages in a DTN environment, showing the OpenFlow controller, DTN nodes, Bundle Protocol Agents, and the flow of messages among these components.

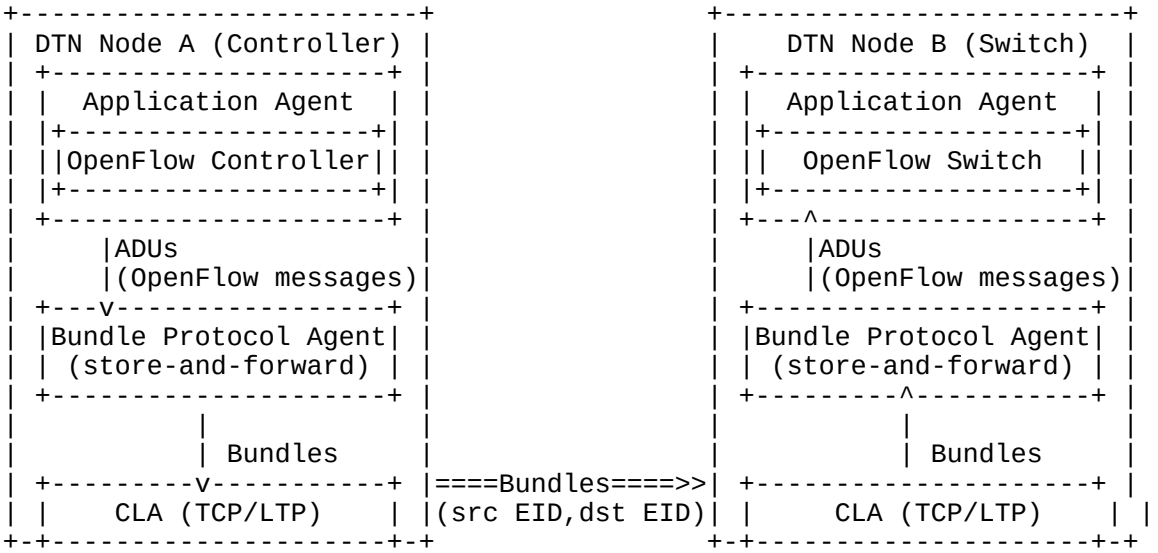

Figure 1: Architectural overview of OpenFlow control message carriage over a DTN using BP

In this architecture, the controller and the switch are deployed on DTN-capable nodes, referred to as DTN Node A (Controller) and DTN Node B (Switch), respectively. Within each DTN node, the OpenFlow entity operates as part of an application agent and generates or processes OpenFlow control messages. OpenFlow messages are treated as Application Data Units (ADUs) and are delivered to the local Bundle Protocol Agent.

The Bundle Protocol Agent encapsulates received ADUs into Bundles and forwards them within the DTN according to the store-and-forward mechanisms defined by the Bundle Protocol. Each Bundle is addressed using a source Endpoint Identifier (source EID) and a destination Endpoint Identifier (destination EID), thereby enabling message delivery in the absence of stable end-to-end paths.

Below the Bundle Protocol, Convergence Layer Adapters (CLAs) map Bundles onto specific underlying transport protocols (e.g., TCP or LTP) to accommodate different types of physical or logical links. Through these mechanisms, OpenFlow control messages can be transported across DTN nodes in network environments characterized by high delay or intermittent connectivity.

5. Encapsulation and Delivery Rules

This section specifies the method for encapsulating OpenFlow messages as BP payloads and describes delivery considerations in DTN environments, including message mapping, fragmentation, duplicate delivery, ordering, and Bundle lifetime.

5.1. Encapsulation Overview

BP is used to carry OpenFlow signaling data. During transmission over a link, a Bundle containing OpenFlow signaling is encapsulated by underlying link-layer, network-layer, and transport-layer protocols, as well as by the block structure defined by BP. Figure 2 illustrates an example encapsulation structure, which includes:

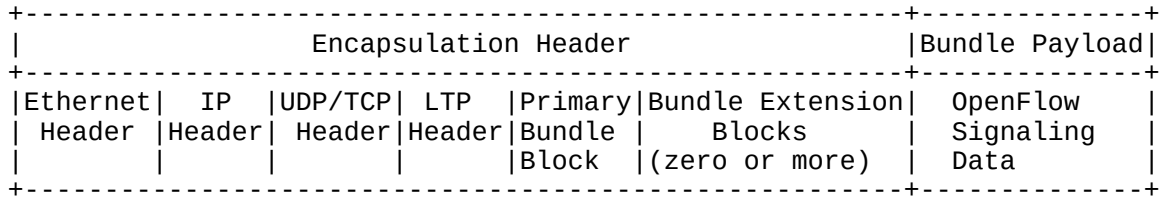

Figure 2: Illustrative example of OpenFlow control message encapsulation as a BP payload

* Ethernet Header
* IP Header
* UDP/TCP Header
* LTP Header
* Primary Bundle Block
* Bundle Extension Blocks (optional, zero or more)
* Bundle Payload

The Primary Bundle Block and any Bundle Extension Blocks together constitute the BP header and control information. The headers of the underlying carrier protocols (e.g., Ethernet/IP/UDP(TCP)/LTP), along with the BP Primary and Extension Blocks, are collectively referred to as the encapsulation header. OpenFlow signaling data is carried as the Bundle Payload.

Figure 2 is provided for illustrative purposes only and shows one possible encapsulation approach. BP may operate over different CLAs and underlying carrier protocols. This document does not mandate the use of TCP, UDP, LTP, or any specific link-layer combination.

Note: An implementation MAY realize this carriage using a tunnel-like approach; however, this document does not define a new tunneling protocol.

5.2. One-to-One Message Mapping

Each Bundle MUST carry exactly one OpenFlow message in its payload block.

The OpenFlow message MUST be encoded using the native wire format defined by the OpenFlow specification and carried directly as the Bundle Payload. BP entities and intermediate DTN forwarding nodes MUST treat the payload as opaque data.

Unless a future extension explicitly defines a message aggregation mechanism and such a mechanism is explicitly supported and configured by both communicating endpoints (or otherwise agreed upon), an implementation MUST NOT aggregate multiple OpenFlow messages into a single Bundle.

Except for the BP block structure itself, this specification does not define any additional application-layer encapsulation headers.

This specification does not require OpenFlow messages to be transformed, compressed, or re-encoded. If such processing is employed by a deployment, it MUST be explicitly configured or negotiated out of band, and the receiving endpoint MUST be able to reconstruct and preserve the semantics of the original OpenFlow wire format.

The objectives of this one-to-one mapping design include:

Minimizing parsing ambiguity; and Simplifying the handling of loss, duplication, and reordering in DTN environments.

5.3. Fragmentation and Reassembly

If an OpenFlow message cannot be forwarded as a single Bundle over the underlying carrier path (e.g., due to size limitations imposed by the convergence layer or the local BP implementation), a BP node MAY fragment the Bundle, provided that fragmentation is permitted under the applicable BP specification and supported by the implementation.

Specifically:

If the Bundle carries an explicit instruction equivalent to “do not fragment” (e.g., the relevant BPv7 Primary Block flag, or an equivalent operational policy in BPv6 implementations), fragmentation MUST NOT be performed.

If the Bundle is sent with an anonymous source (i.e., a null or anonymous source node identifier in BPv7, or an equivalent anonymous-source configuration in BPv6 deployments), fragmentation MUST NOT be performed.

When fragmentation is used, the BP agent MUST reassemble all fragments into the original ADU before delivering the OpenFlow message to the OpenFlow processing entity. An implementation MUST NOT expose partially reassembled OpenFlow messages to the OpenFlow processing entity.

Unless strictly necessary, implementations SHOULD avoid fragmenting OpenFlow messages at the application layer, as doing so may increase interoperability complexity and complicate ordering and duplicate-handling semantics.

Note: The impact of anonymous sources on duplicate suppression and idempotency is outside the scope of this subsection; related requirements are discussed in Section Section 5.5.2.

5.4. Delivery Semantics

DTN environments may be characterized by long delays, intermittent connectivity, and opportunistic contacts. Therefore, implementations SHOULD NOT assume interactive request/response timing relationships.

This carriage mechanism is intended to support multiple OpenFlow control message exchange scenarios, including but not limited to policy distribution, configuration updates, and telemetry or state exchange.

If message correlation is required, an application MAY use the OpenFlow Transaction Identifier (xid). If application-layer acknowledgment is required, a deployment MAY use a locally defined acknowledgment mechanism.

5.5. Duplicate Delivery

Due to BP forwarding, retransmission, and custody-transfer behaviors, a Bundle may be delivered more than once. The receiver MUST be able to tolerate duplicate OpenFlow messages. An implementation MUST NOT assume that an OpenFlow message is delivered only once in a DTN environment.

5.5.1. Duplicate Suppression

To support duplicate suppression, the receiver SHOULD maintain a bounded replay or duplicate cache. Cache keys SHOULD be derived from native BP fields that serve to identify the delivered Bundle in the applicable BP version.

Specifically, the cache key SHOULD consist of the source identifier and the creation timestamp:

In BPv6, the source identifier is the Source EID.

In BPv7, the source identifier is the Source Node ID.

If the delivered unit is a fragment, the cache key MUST additionally include the fragment offset and the total ADU length (or the payload length as defined by the applicable BP version), in order to avoid key collisions among different fragments.

Note: In both BPv6 and BPv7, the creation timestamp includes a sequence component that ensures uniqueness of Bundles generated by the same source. The sender MUST generate creation timestamps in accordance with the applicable BP specification in order to avoid producing Bundles that share the same source identifier and creation timestamp but carry different content within the applicable scope.

5.5.2. Anonymous Source

If a Bundle uses an anonymous source (i.e., a null or anonymous source identifier), the receiver MUST NOT rely on native BP Bundle identifiers for duplicate suppression, as such Bundles may not be uniquely identifiable. In this case, the deployment MUST take one of the following approaches:

* Disable BP-based duplicate suppression and instead rely on application-layer mechanisms (e.g., deployment-local message correlation information or other fields that uniquely identify messages); or

* Ensure that multiple applications of the same OpenFlow message are safe for the intended purpose (i.e., the operation is idempotent).

If neither condition can be satisfied, the receiver MUST reject duplicate messages or discard suspected duplicate deliveries according to the best available local policy.

5.6. Ordering Considerations

BP does not guarantee in-order delivery. Because OpenFlow semantics may be order-sensitive (e.g., applying rule deletion before rule addition may yield different outcomes than the reverse order), the receiver MUST NOT assume that arrival order reflects transmission order.

Senders SHOULD avoid transmitting OpenFlow update sequences that rely on strict ordering for correctness unless an explicit ordering control mechanism is used. In deployments that require ordering guarantees, endpoints MAY use one or more of the following mechanisms:

The OpenFlow Transaction Identifier (xid); and/or

An explicit monotonically increasing sequence number carried in the encapsulation header.

When ordering control mechanisms are used, the receiver MAY buffer out-of-order updates within a bounded time window and apply them in order. If a message cannot be safely applied due to missing dependencies, the receiver SHOULD discard or defer the message according to local policy and SHOULD record the condition to support operational visibility.

5.7. Bundle Lifetime and Expiration

Each Bundle carrying an OpenFlow message MUST be assigned a finite lifetime. This lifetime SHOULD reflect the operational validity window of the contained OpenFlow message. Senders SHOULD avoid assigning lifetimes that exceed the effective validity period of the corresponding control action.

The receiver MUST NOT apply the OpenFlow payload of any Bundle that is determined to be expired. If an expired Bundle is nevertheless delivered to the processing entity, that entity MUST discard the Bundle and MUST NOT apply its OpenFlow payload.

The receiver SHOULD record or account for control messages that arrive already expired in order to support operational visibility.

6. Endpoint Identification and Addressing

This section specifies endpoint identification and addressing conventions, including the association between OpenFlow entities and BP EIDs, bidirectional addressing configuration, and multiplexing and demultiplexing requirements.

6.1. OpenFlow Entities and DTN Endpoints

OpenFlow communication involves an OpenFlow controller and one or more OpenFlow switches. Each OpenFlow entity that participates in DTN-based carriage MUST be associated with a BP EID and MUST use that EID as the destination address for Bundles sent to that entity.

The mapping between OpenFlow identifiers (e.g., datapath identifiers) and BP EIDs is deployment-specific and MAY be established through configuration, directory services, or other management mechanisms.

6.2. EID Naming Considerations

To facilitate traffic identification, deployments SHOULD use a dedicated service name or sub-endpoint component (e.g., “/of”). For example, a controller MAY include such a service component in its EID, and switches MAY use the same service component in their respective EIDs.

This document does not mandate a specific URI scheme or naming syntax and instead relies on the BP EID formats supported by the implementation. The primary requirement is that controllers and switches use stable and unique EIDs.

Traffic identification is achieved through endpoint naming conventions and/or local configuration. A deployment MAY define local policy rules (e.g., routing, priority, or custody-transfer preferences) based on destination endpoint naming patterns and other BP metadata.

Deployments SHOULD ensure that DTN endpoints use singleton endpoints (i.e., unicast semantics). Non-singleton endpoints (e.g., anycast or multicast) SHOULD be used only when group delivery is explicitly intended and when safe group-delivery semantics have been defined. For deployments using DTN URI schemes, implementations SHOULD avoid selecting service components that would cause an EID to resolve to a non-singleton endpoint (e.g., service identifiers with group or collection semantics).

6.3. Bidirectional Addressing

Communication is typically bidirectional: messages from the controller to the switch carry configuration updates and control instructions, while messages from the switch to the controller carry telemetry, status reports, and optional control-plane notifications.

Accordingly, each endpoint MUST be configured with:

A local receiving EID; and A peer destination EID for transmitting outbound messages.

When BP status reports are used, an implementation MUST NOT interpret a BP delivery report as an indication that the corresponding OpenFlow action has been applied to the data path. Such reports only indicate that the message has been delivered to the receiving endpoint’ s BP application agent.

6.4. Multiplexing and Demultiplexing

A node that provides both DTN forwarding functionality and OpenFlow processing MUST demultiplex Bundles destined for its local EID and deliver their payloads to the local OpenFlow entity.

If a node hosts multiple OpenFlow entities, it MUST use distinct EIDs or other unambiguous local dispatch mechanisms to ensure that each payload is delivered to the intended entity.

7. Security Considerations

This document does not alter the security properties of OpenFlow or the Bundle Protocol. However, carrying control messages in a DTN store-and-forward environment may increase exposure (e.g., longer data residency times and larger replay windows). Deployments therefore need to apply appropriate integrity, authentication, and (where applicable) confidentiality protections.

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. Normative References

[OPENFLOW] Open Networking Foundation, "OpenFlow Switch Specification", <https://opennetworking.org/software-defined-standards/specifications/>.

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

[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/info/rfc8174>.

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

9.2. Informative References

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

[RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, DOI 10.17487/RFC5050, November 2007, <https://www.rfc-editor.org/rfc/rfc5050>.

Acknowledgments

TBC

Authors' Addresses

Xiaojing Fan
Beijing Jiaotong University
Email: 23111043@bjtu.edu.cn

Huachun Zhou
Beijing Jiaotong University
Email: hchzhou@bjtu.edu.cn

Jie Qu
Beijing Jiaotong University
Email: 24120101@bjtu.edu.cn