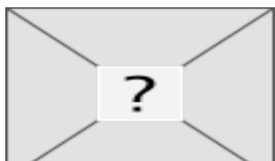
CCNx 1.0 Network Software

Computer Science Laboratory

Networking & Distributed Systems

March 2014



CCN 1.0

Cleaned API

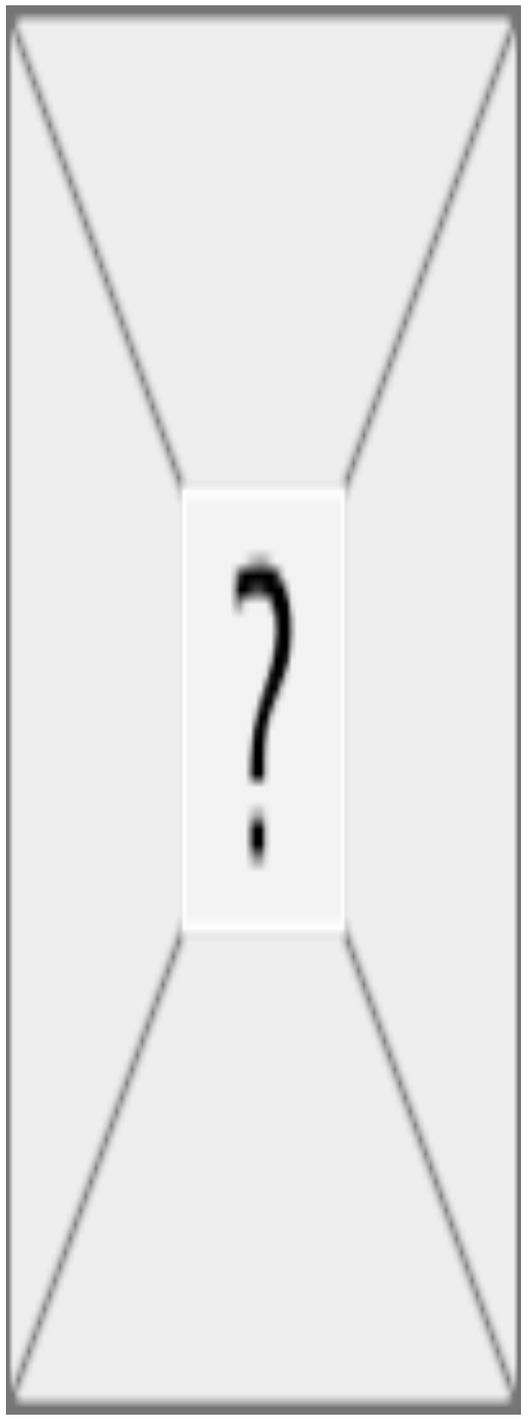
Lowered the learning curve
Separated concerns

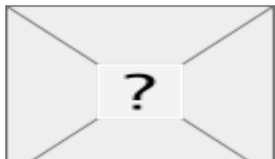
Cleaned Code

Improved maintainability
Increased modularity

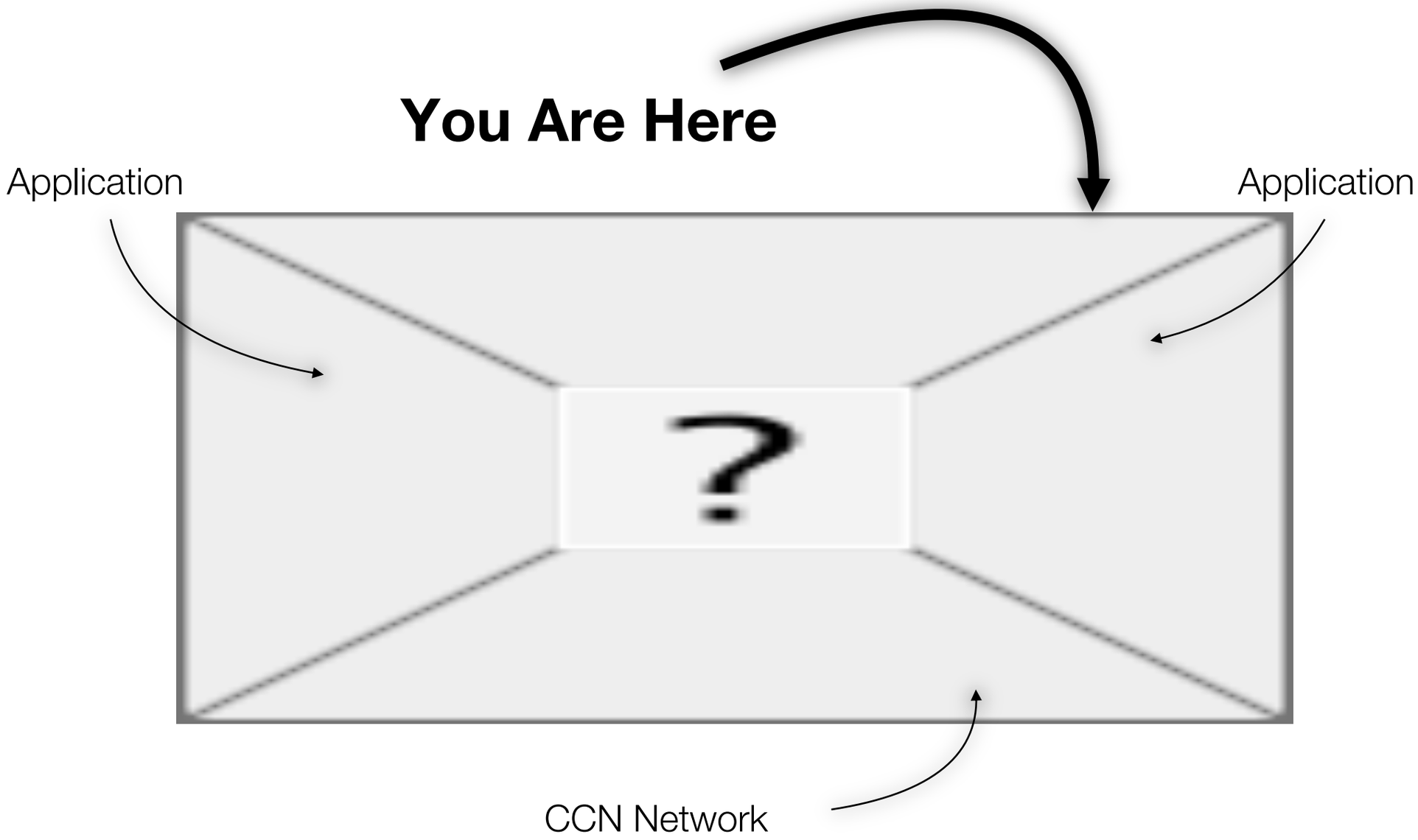
Cleaned Protocol

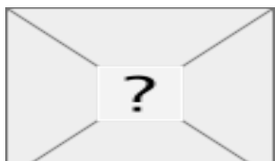
Defined minimum protocol
Specified auxiliary protocols





Overview





Requirements

Provide Structure and a Vocabulary

Design simple parts connected by clean interfaces.

Implement modularity and separate concerns.

Promote Stability and Enable Extensibility

Design for composition and substitution of components.

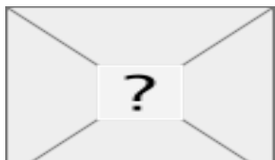
Implement the core functionality plus examples and proofs-of-concept.

Design for evolution.

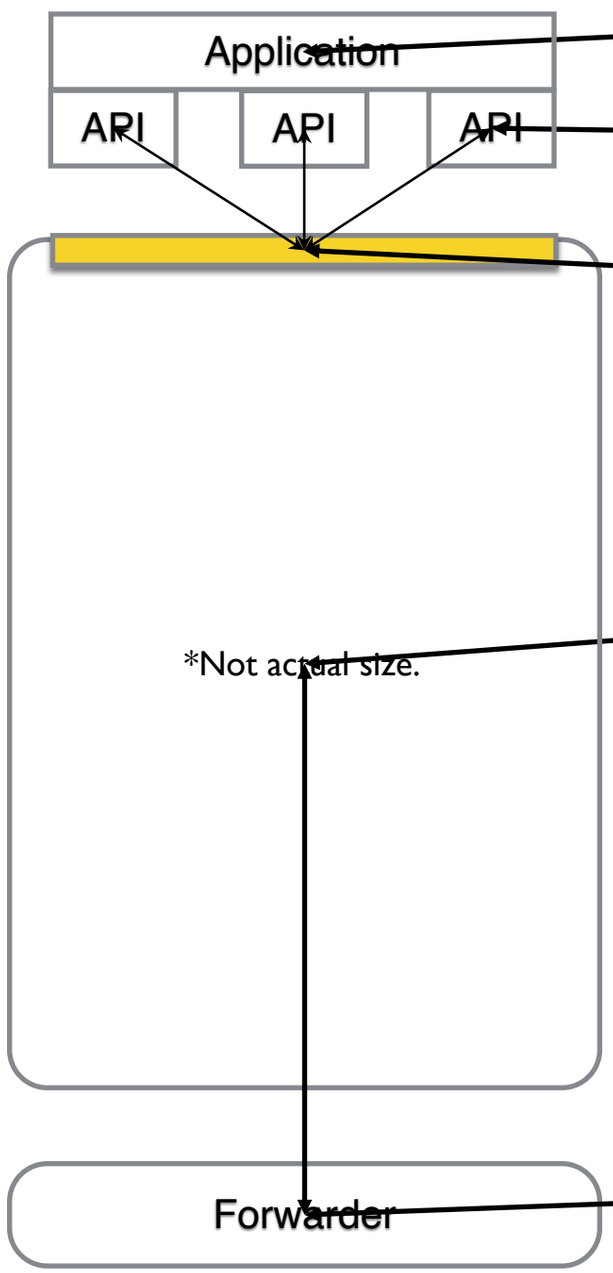
Enable Productivity

Design for inspection and debugging.

Implement for usability and “learnability.”



Vocabulary



A **program** that uses one or more CCN APIs to communicate

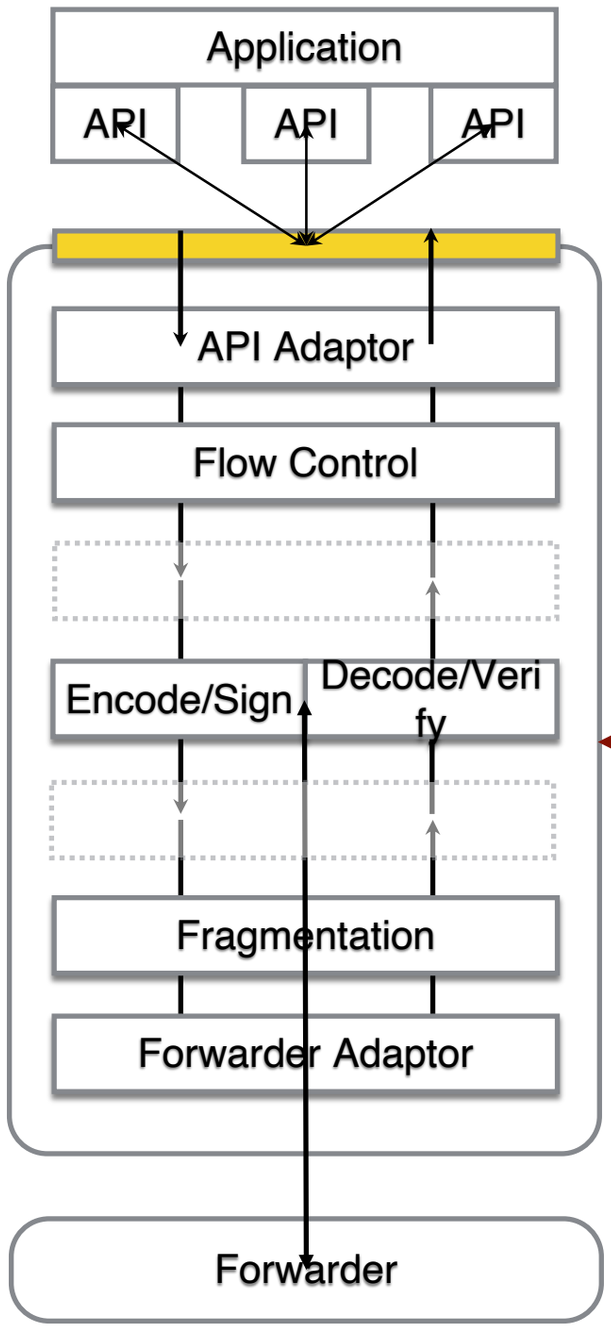
Application Programming Interfaces implement a particular kind of interaction with the CCN Transport

The **Transport Programming Interface** providing a uniform interface to the **Transport Framework**

Transport Framework containing one or more **Transport Stacks** conveying data between an API and the network

The CCN **Forwarder** directing traffic on the network

Transport Stack



Simple component model

Components are passive or active

Assembled and initialized at create-time

Separates concerns

- From the application
- From other, individuated components.
- Permits component reuse.

Implements CCN object protocols

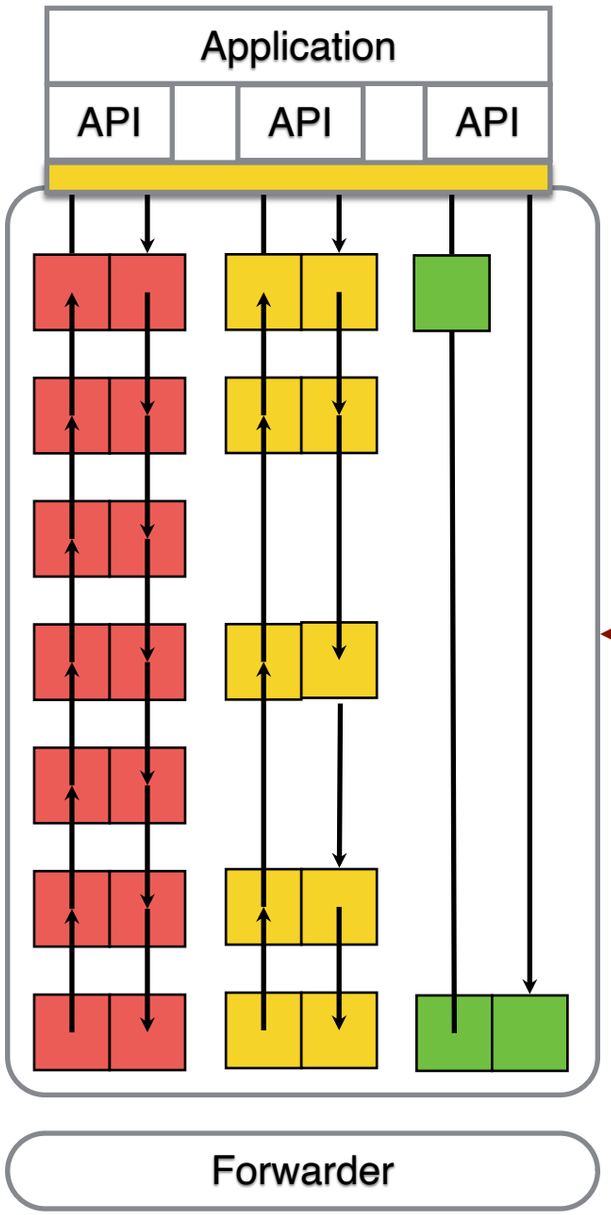
- Segmented objects
- Versioned objects
- CCNmq Message Queue protocol
- Key/Value protocol

Integrates external services

- External Caches
- Identity, Key and Certificate services
- Trust-model implementations

*Example only. Not all components are required for all stacks.

Transport Framework



Multiple Transport Stacks

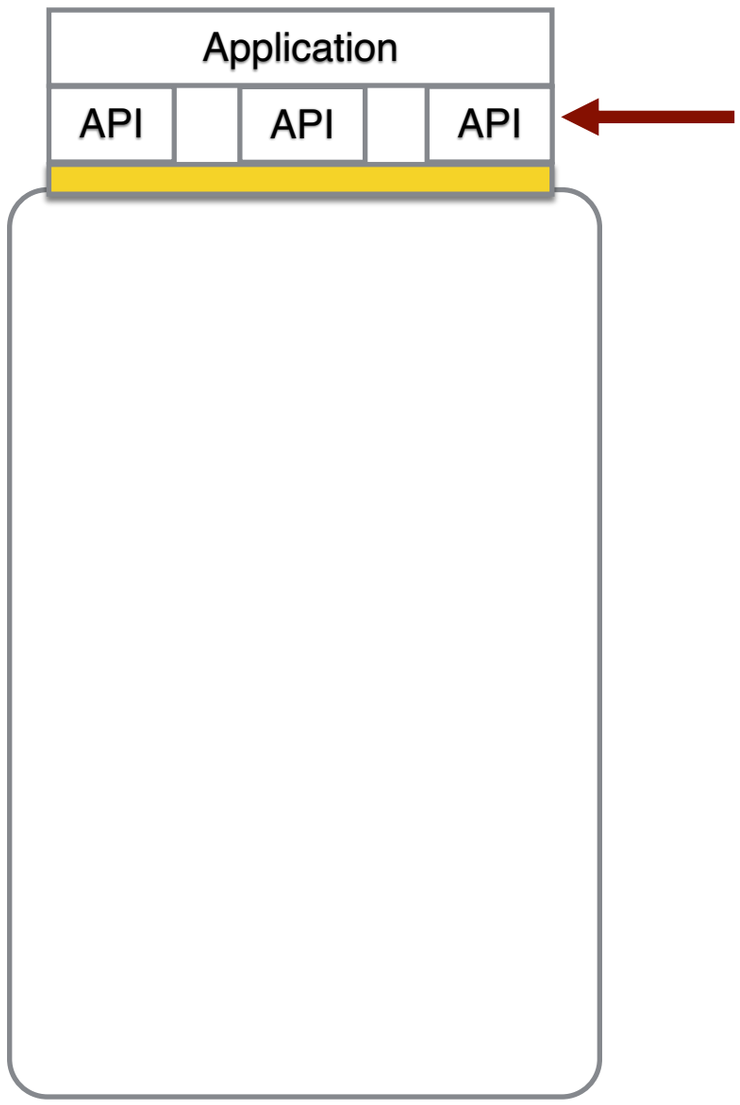
Maintains for each Stack:

- Runtime state
- Performance information

Maintains for the Framework:

- Runtime state
- Performance information
- Interfaces for extracting the data

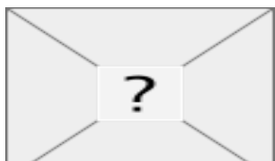
Application Programming Interfaces



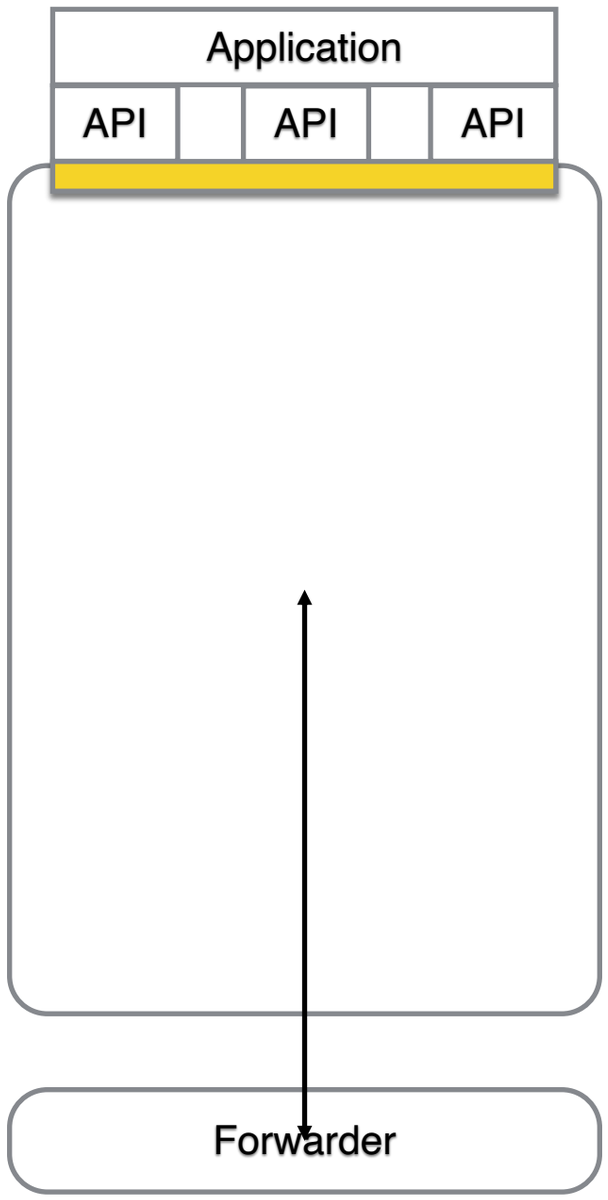
Many possible APIs

- Sockets
- CCN Repository
- Streaming
- Message queuing

Multiple simultaneous APIs



Transport Programming Interface



Programmatic boundary between the API and the Transport Framework

Message based communication

Passes messages to and from the API

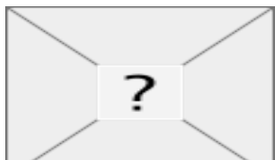
`Transport_Open`
: Opens and configure a Transport Stack

`Transport_Close`
: Closes a Transport Stack

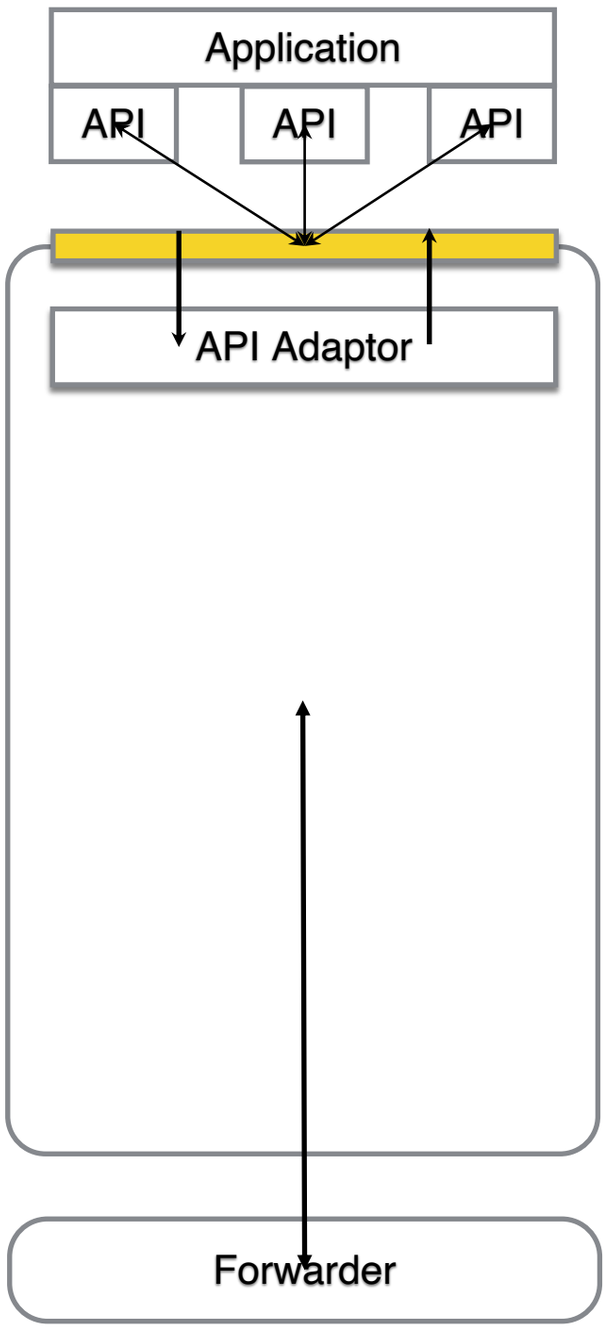
`Transport_Send`
: Send a message to a Transport Stack

`Transport_Receive`
: Receive a message from a Transport Stack

`Transport_Notify`
: Signal that an event occurred.



API Adaptor

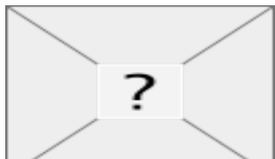


Communicates between the API

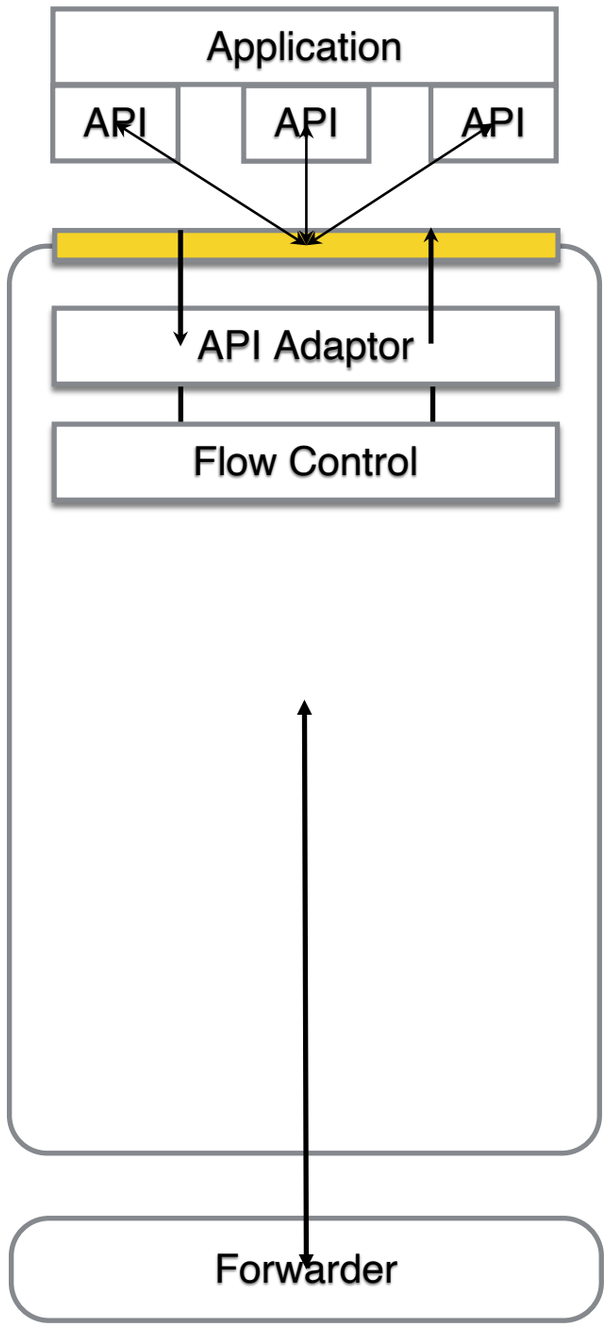
and a specific Transport Stack

and the Transport Framework

Required component



Flow Control



Traffic shaping and management

Interest retransmissions

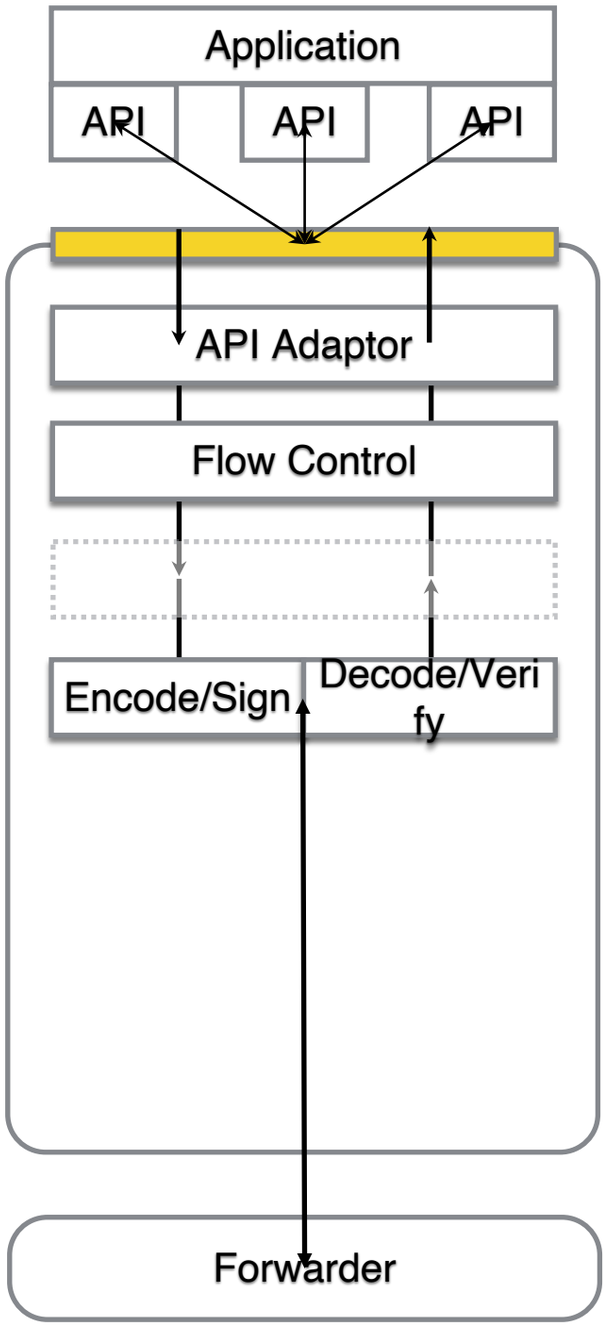
Interest pipelining

Content Object ordering

Link resolution

Optional component

Encoding and Signing
Decoding and Verification



Encode and sign outbound Content Objects

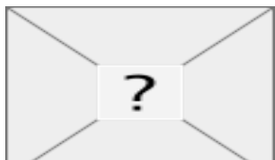
Decode and verify inbound Content Objects

Encode outbound Interests

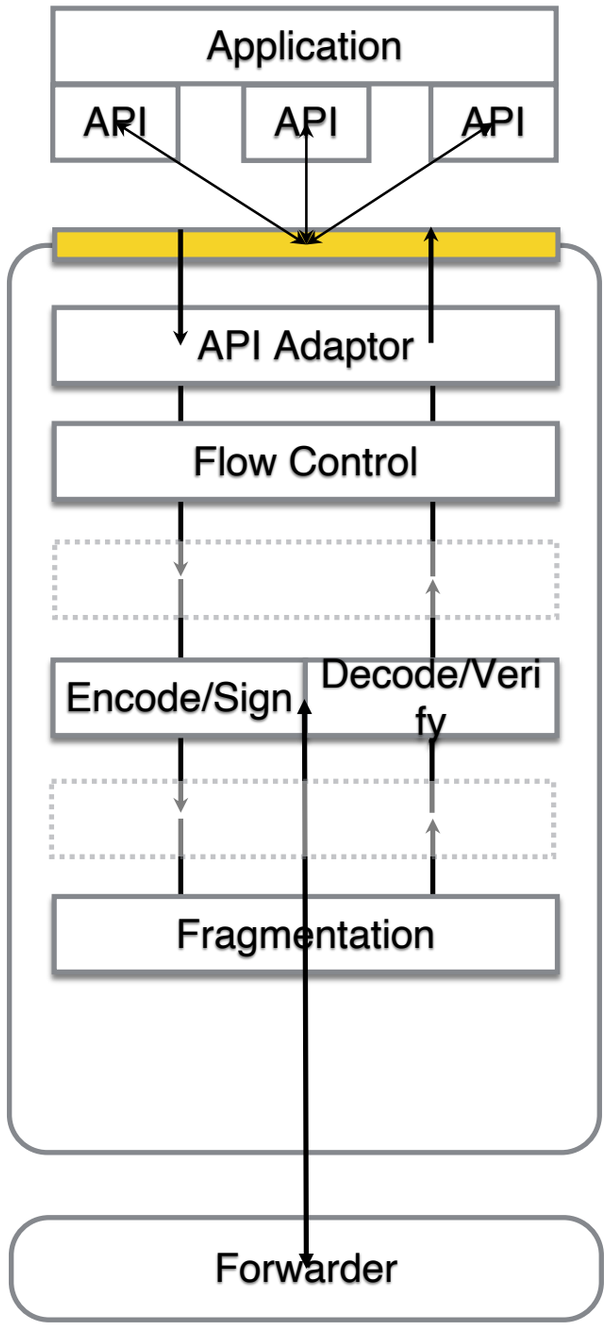
Decode inbound Interests

Interfaces to external key stores and key management systems

Optional component



Fragmentation

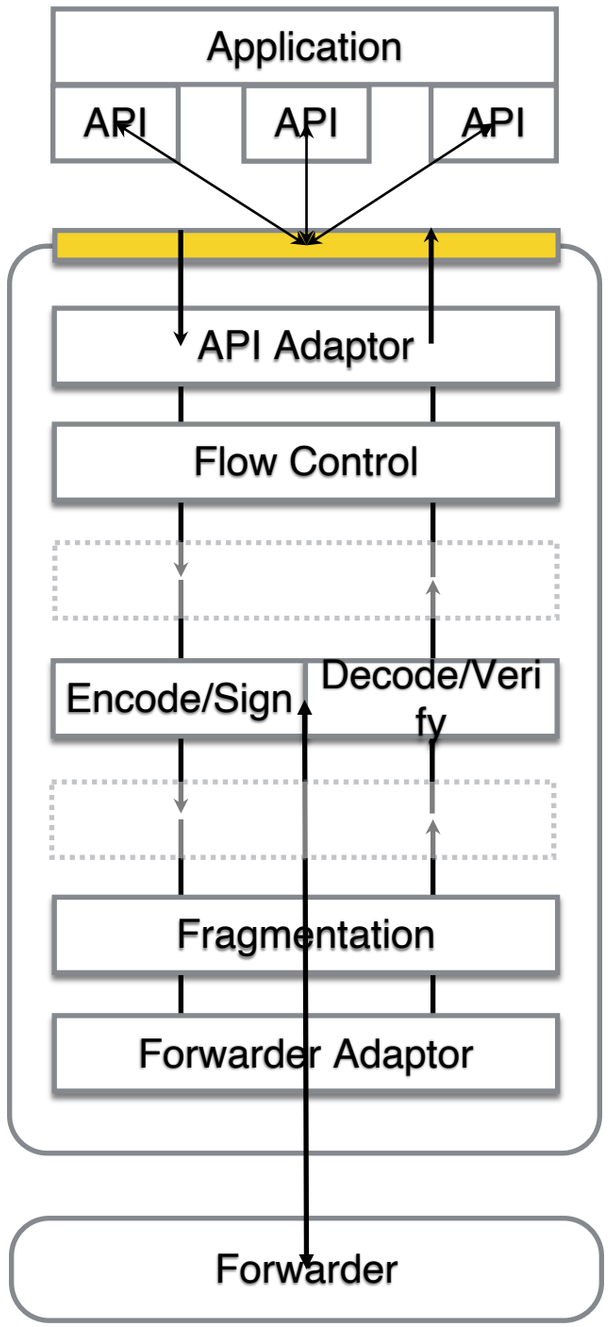


Outbound packet fragmentation

Inbound packet reassembly

Optional component

Forwarder Adaptor

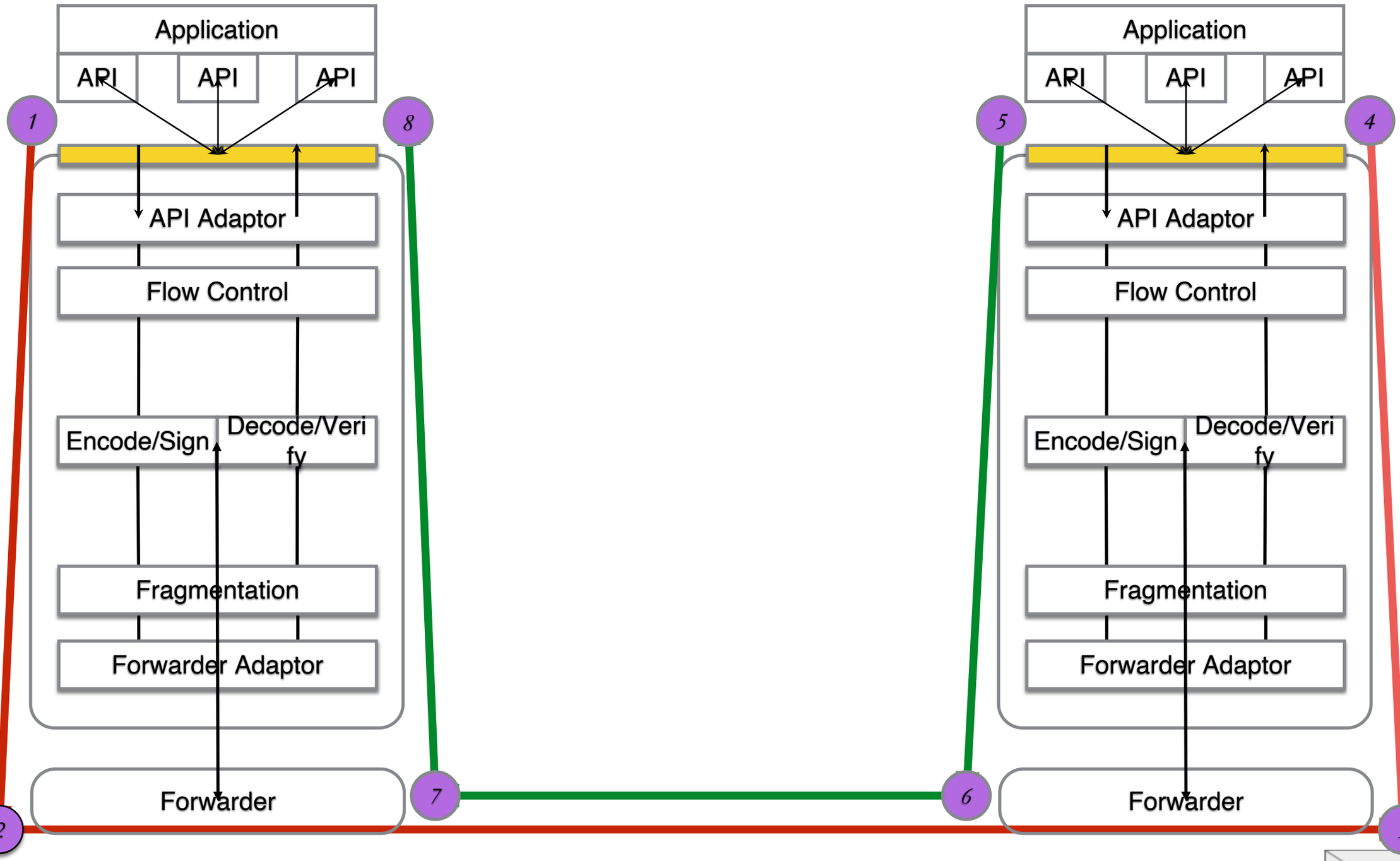


Communicates with a Forwarder.

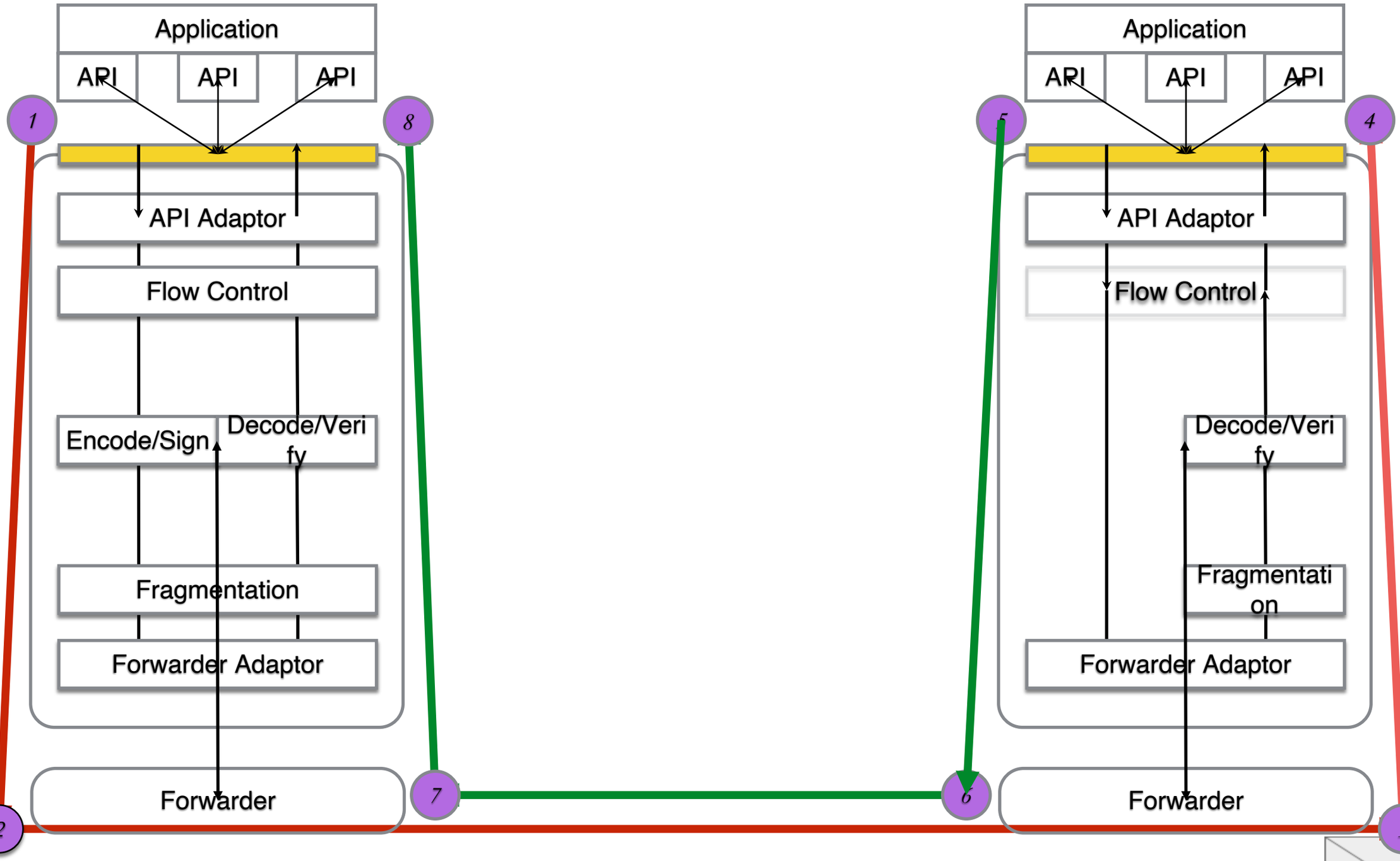
Performs the necessary operations to control the Forwarder on behalf of the Transport Stack.

Required component

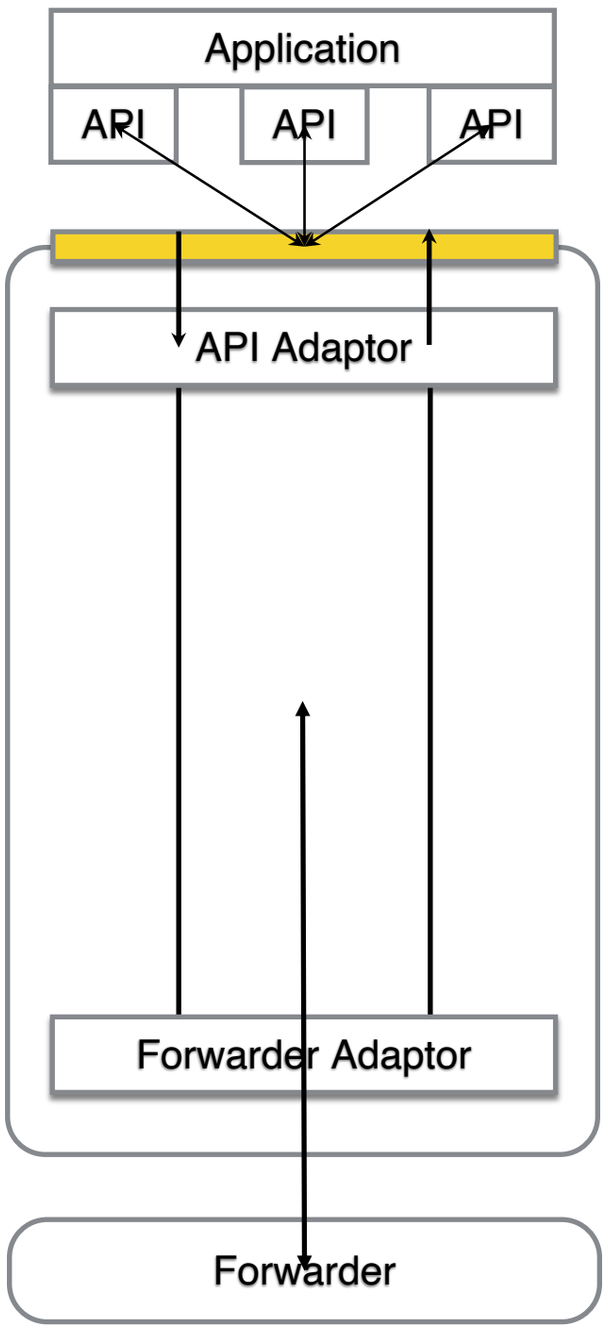
Example Path



Example Path



Simplest Stack



The API encodes Interests and Content objects.

The Stack simply transmits them to the forwarder

Software Components

Algorithm Library
General purpose facilities (buffers, lists, maps, etc)

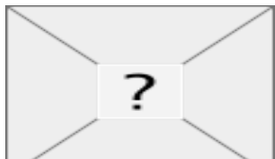
Transport *"Ready To Assemble"*
A Transport Stack implementation

Security Library
Cryptographic and other security related facilities

Development Tools
Runtime and at Development time

CCN Core
CCN constructs as C objects

Unit Testing Tools
TDD, developer and release aids



Module Dependencies