Instant Message Payment Transaction Protocol (IMPT)

Doris Chia-ching Lin
CS544, Drexel University
dl963@drexel.edu

Contents

1. Service Description

The Instant Message Payment Transaction(IMPT) Protocol is a peer to peer application that allows two users to make a monetary transaction with a third-party payment service (PayPal, Venmo, Cash, etc) through an instant message application. It allows one user to send a payment and trigger transactions between both the user's payment service account when the other user accepts the payment online and has a corresponding payment service account. When the transaction is successfully made, both users get confirmation notification. If sending payment to a receiver doesn't have a corresponding account stored in the application or declined the payment, the conversation will be terminated immediately. [*Fig.1*]

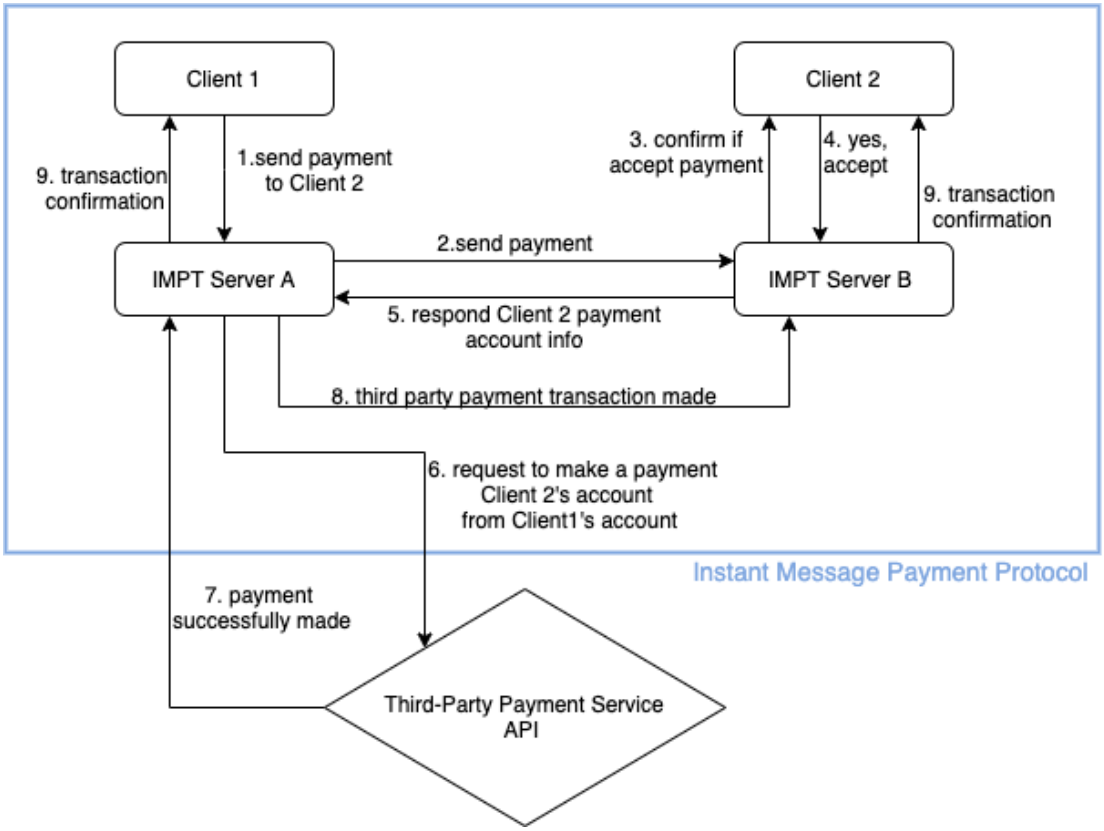


Fig.1 IMPT payment flow

The user can also make a payment request to the other user online with their chosen payment account. The receiving user can either initiate a payment transaction or decline the request to terminate the communication.

Users can store one or several third-party payment service account info in the application. When the user requests to stores account information, IMPT verifies the account info by sending

verification requests to third-party payment gateway. Only when the account is verified, the account info can be successfully stored. [*Fig.2*]

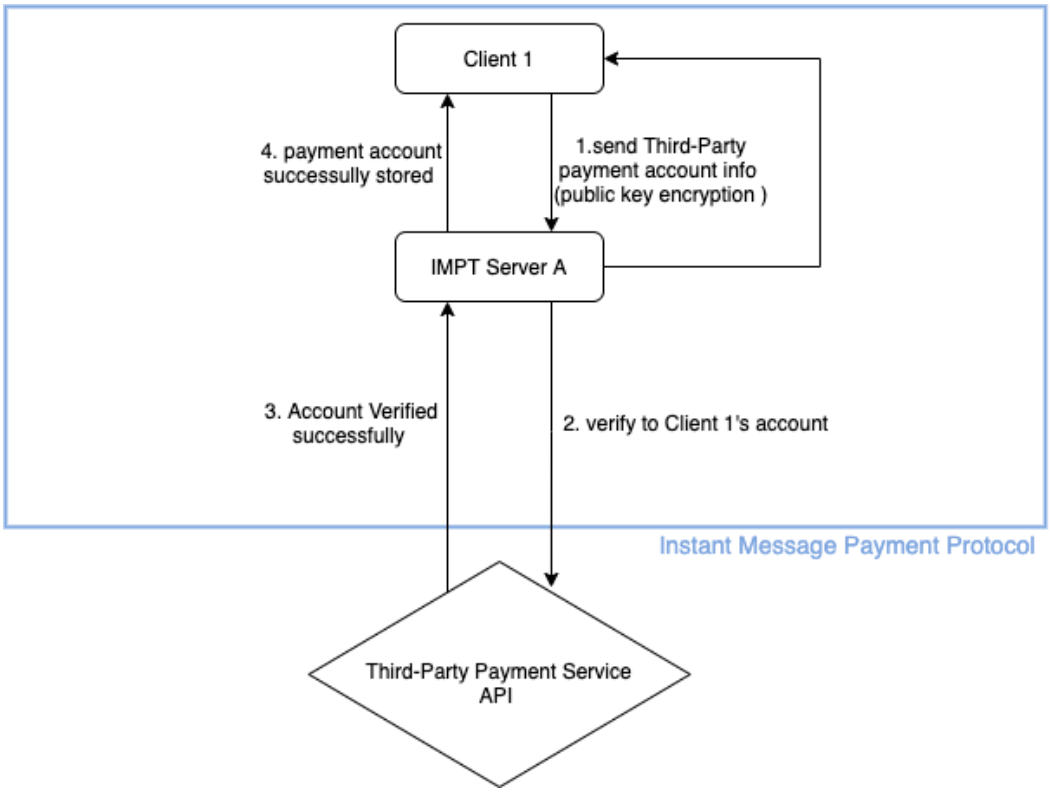


Fig.2 IMPT storage flow

Each third-party payment account only uses the user's default payment method(specific bank account, credit card account, etc) in the third-payment payment service and all the transaction activities can only be made and interact with users currently online.

IMPT also provides regular text message service to users. However, as the major feature of the protocol is a monetary transaction, this design draft will be more focused on the payment transaction connectivities.

2. Message Definition (PDUs)

2.1 Addressing

The IMPT protocol uses TCP/IP in the transport layer to support any reliable transport that has no boundaries like non-fixed-size messages. As Internet Assigned Number Authority (IANA) releasing assigned port numbers, TLC/5180 is available for specific service use. A connection to an IMPT server will then be to the server's IP address, on the respective port discussed above.

2.2 Flow Control

In the underlying TCP/IP layer, the flow control for IMPT is managed to ensure a reliable message transfer, network traffic moderation, and quality of service. It ensures that a sender is not overwhelming a receiver by sending packets faster than it can consume. The mechanism here is the node receiving data sends feedback to the node sending the data to let it now about its current condition. With a sliding widow protocol, TCP keeps it never has more bytes in flight than the window advertised by the receiver;

2.3 PDU Definitions

IMPT communication includes the following stages:

- *Handshake.* Agree on protocol version and conduct authentication.
- *Initialization*. make sure both users are online.
- *Internal Protocol Interaction.* The client sends action messages at will and receives responses from the server. The server acts based on client command. receives responses from the client, as well as notifies clients based on the result of actions.

All messages are constructed of a stream of bytes, either of a fixed size based on the message type or a custom size indicated in the message (which messages are of fixed size and which are of varying size is detailed in the rest of the section). the standard PDU is structured as:

```
sentBy > COMMAND [messageType] parameter1{<,parameter1>}
[<parameter2>{<,parameter2>}]
```

Implementation keys:

Pramamer	Definitions	Example
`sentBy`	Where the message is sent, either by a server or a client.	`C | S`
`[messageType]`	This is to define the message type. The `[BEGIN]` presents the initial message, `[RES]` presents the responding message, `[FINAL]` presents the final message to end the conversation.	`[BEGIN]| [RES]| [FINAL]`
`<errorMessage>`	Error message to be sent when error occurs.	`"Autheticat ion Error"`
`<paymentService>`	Assigned third-payment service company.	`"PayPal"`
`<paymentAmount>`	Payment amount to send or request.	`"50"`

<requestId>	IMPT generates a 10-digit unique ID number for each request for tracking purpose.	"2345657240"
<userIdToken>	Each user also obtains a random 16-character long unique token generated by the application during conversation for secure authentication and identity. When it is used in messages, it indicates the receiving user's token.	"rieoslfbdjlgfdem"
<version#>	IMPT version number	"IMPT 1.0"

Next, each phase is discussed with a detailed description of the IMPT message PDUs.

2.3.1 Handshake

This stage is designated for determining version and apply user authentication. To initialize the communication, the client checks the server with the INIT message:

INIT Message: the Init Message is to agree upon a version of IMPT to use.
=> C > INIT [BEGIN] <version#>

Server responses with its highest supported version:
=> S > INIT [RES] <version#>

Client response again to determine version:
=> C > INIT [FINAL] <version#>

2.3.2 Initialization

The handshake and initialization commands here is to establish an initial connection to confirm the other user is online. Each client has a unique ID token associated with their account within a limited time session.

Poke Message: the Poke Message is to establish connection.
=> C > POKE [messageType] <userIdToken>

If the other client is online, it responses
=> C > POKE [messageType] <userIdToken>

If no response after timeout restriction or error message received, the connection is terminated.

2.3.2 Internal Protocol Interaction

Once the connection is established, the client can start sending a payment request to the other user.

Client-to-Server messages:

PAYSND Message: the PAYSND message is used for a sending payment request to the other user. `<paymentService>` is the chosen third-party payment service and `<paymentAmount>` is desired payment amount. `<requestId>` is a unique id for tracking this request.

```
=> C > PAYSND [messageType] <userIdToken> <paymentService>
<paymentAmount> <requestId>
```

PAYRQST Message: the PAYRQST message is used for a requesting payment request to the other user.

```
=> C > PAYRQST [messageType] <userIdToken> <paymentService>
<paymentAmount> <requestId>
```

PAYACCEPT Message: the PAYACCEPT message is used to accept a sending/requesting payment request from the other user.

```
=> C > PAYACCEPT [messageType] <userIdToken> <requestId>
```

PAYDECLINE Message: the PAYDECLINE message is used to decline a sending/requesting payment request from the other user.

```
=> C > PAYDECLINE [messageType] <userIdToken> <requestId>
```

Server-to-Client messages:

TRANS_SUCCESS Message: the TRANS_SUCCESS message is used by server to notify client that the transaction is made successfully in third-party payment service account.

```
=> S > TRANS_SUCCESS [messageType] <userIdToken> <requestId>
```

Error messages:

ERR_INIT Message: ERR_INIT Message is used to response any error occurred during initial connection. For example, the other client ID token is invalid.

```
=> C|S > ERR_INIT [messageType] <userIdToken> <errorMessage>
```

ERR_PAY Message: ERR_PAY Message is used to response any error occurred during payment request. For example, the other client doesn't have a corresponding payment service account.

```
=> C|S > ERR_PAY [messageType] <userIdToken> <errorMessage>
<requestId>
```

ERR_TRANS Message ERR_TRANS Message is used to response any error occurred during third party payment service transaction. For example, the payment transaction fails.

```
=> C|S > ERR_TRANS [messageType] <userIdToken> <errorMessage>
<requestId>
```

Termination messages:
DISCONNECT Message: DISCONNECT Message is used to terminate the communication gracefully. Once sent by any of the sides, any pending actions / updates are disposed and the connection is closed.

```
=> C|S > DISCONNECT
```

2.4 Error Control

There are four potential error scenarios:

- In connection initialization, the receiving user is not active or the user account doesn't exist.
- The receiving user doesn't have a corresponding payment service account.
- Third-party payment service transaction fails
- A message is illegal at the current state of the communication.

All the error messages above are followed by terminating the connection for security purposes. It reduces an attacker to apply a fuzz attack on the server.

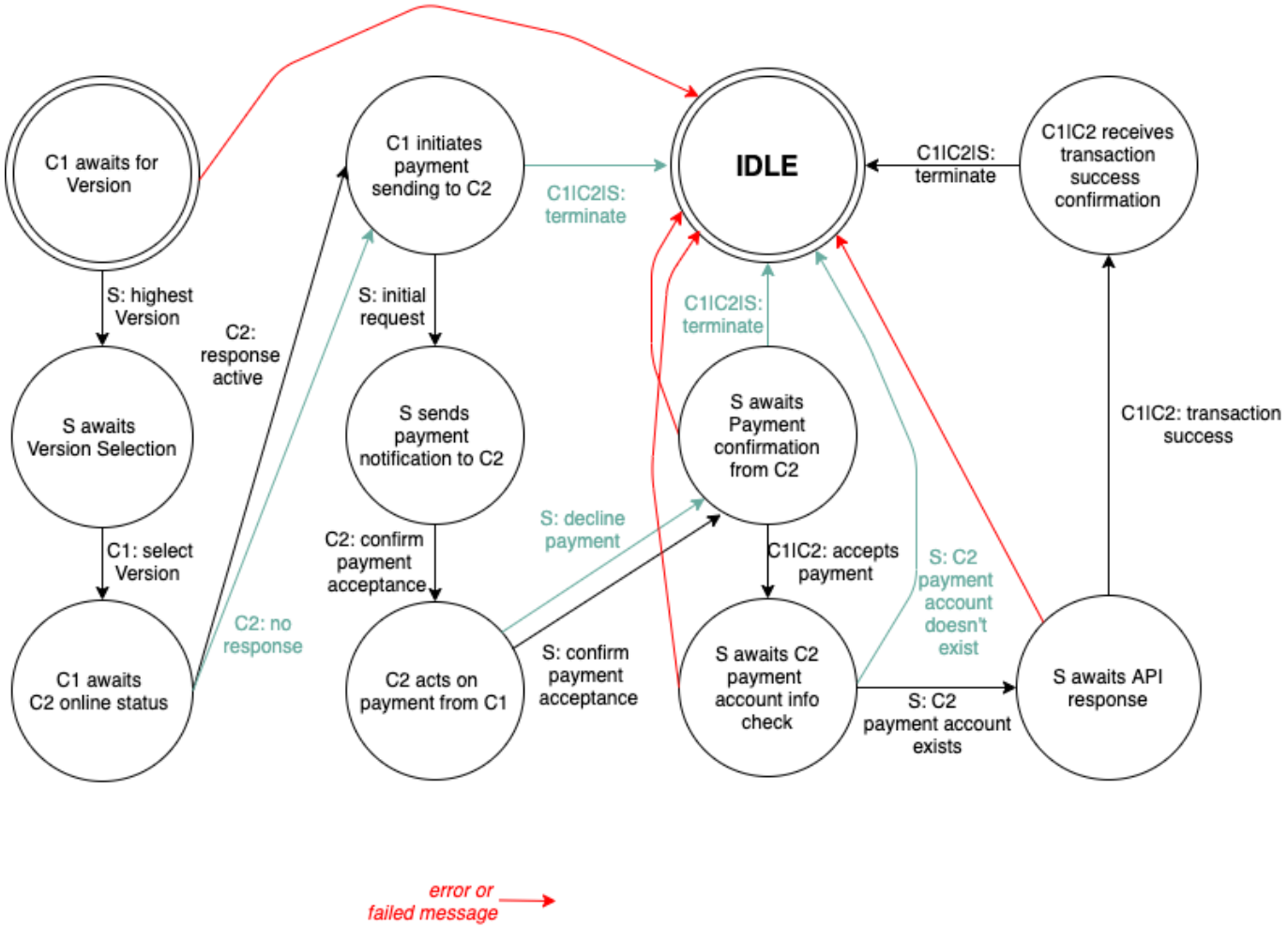
2.5 Quality of Service

The IMPT protocol offers numerous services to the cline to guarantee the quality is up-to-date through its use. This initial version of the protocol forms a basic baseline of QoS from TCP, including:

- **Bandwidth Reservation:** The ability to reserve a portion of bandwidth in a network or interface for a period of time, so that two devices can count on having that bandwidth for a particular operation.
- **Latency Management:** The ability to limit the latency in any data transfer between two devices to a known value.
- **Traffic Prioritization:** The ability to handle packets so that more important connections receive priority over less important one.
- **Traffic Shaping:** The ability to the use of buffers and limits that restrict traffic across a connection to be within a pre-determined maximum.
- **Network Congestion Avoidance:** The ability to monitoring particular connections in the network, and rerouting data when a particular part of the network is becoming congested.

3. DFA

A demonstration of a DFA for IMPT is shown below. During the main communication process, the server waits for a client action or the third-party payment service API response to proceed further step or terminate the process. All actions are asynchronously executed.



4. Extensibility

The initial version of IMPT in this documentation, v1, only allows one-to-one transactions between two users with a third-party payment service. In future versions, IMPT can introduce new methodologies to communication. By adding states to DFA and new messages, the potential extensions include the following but not limited to:

Transaction in group chat
The first version of IMPT does not support transactions in a group chat. In a future version, users in a group chat are allowed to assign another user in the given group chat to perform a payment transaction. The transaction is visible to other users within the group.

New payment methods
Currently, IMPT only integrates third-party payment services like PayPal or Venmo, not directly interacts with other payment gateways like banks or credit card companies. In an extended version, with other payment communication protocols being introduced, for example, secure electronic Transaction (SET), it is possible to store more different types of payment methods and make transactions among various institutions.

Offline communication notification
In general, IMPT does not allow offline communication for both messaging and transactions. In future versions, users are able to send messages and transaction requests to an offline user and an optional push-in notification can be sent to the device of an offline user. However, communication can be continued until both users are active online.

Of course, backward compatibility may be a goal of a future version. In that case, it is recommended that the only way to enter a new state is with a new message type.

5. Security implications

The security of the actual payment transaction is handled by the third-party payment service. The other aspects of security falls onto the cases below:

User Authentication
After a user enters his or her username, in order to increase security, the password with both letters and numbers, upper and lower case, special characters (such as $, %, or &), and no words found in the dictionary. It's also important to use long passwords of at least eight characters. Short, overly simple passwords is rejected and prevented from creating an account.

Initially when users log in, IMPT server supports secure authentication using Simple Authentication and Security Layer (SASL) framework and Secure Sockets Layer(SSL) to ensure

the authentication mechanism provides symmetric-encryption based private connection, trusted identification of communicating parties by public-key cryptography and robust connection reliability with a message integrity check. As for data in communication, SSL should be able to cover the security of data in communication.

Generic Error Messaging

Overall, IMPT uses general error message and strict communication termination rules. For example, when an authentication response is invalid for the given user, the error message thrown back to the client does not specify what went wrong, but simply notifies an authentication error has occurred. It doesn't contain more specific error information like "wrong password" or "wrong user-id" to avoid attackers brute force through the authentication stage. Upon authentication failure, the connection is terminated.

Moreover, when any illegal or unexpected message received at the current state of the protocol, each server and client throw a general error message to notify an illegal message has occurred and close connections. In addition, the timeout mechanism also gives servers and clients a limited time window to wait for response. These two implications reduce avoid the chance of fuzz attacks on the protocol.