# Suggested sync/async operations

- High-level (language-neutral) "API"s (details TBD)
  - Hub API
  - Application API
- Message send/reply examples
  - Send and forget
  - Asynchronous (response required)
  - Synchronous utility method

# Hub API (called by apps)

## Called by application sending a message:

Request with no reply required:

```
(msg-id =) notify(private-key, recipient-id, message)
(msg-id =) notifyAll(private-key, message)
```

Asynchronous request:

```
msg-id = request(private-key, recipient-id, message)
msg-id = requestAll(private-key, message)
```

Synchronous request (utility method):

```
response = call(private-key, recipient-id, message)
```

## Called by application responding to a message:

Used to send message replies:

```
void reply(private-key, msg-id, response, status)
```

# Application API (called by hub)

Receive message, requiring response or not:

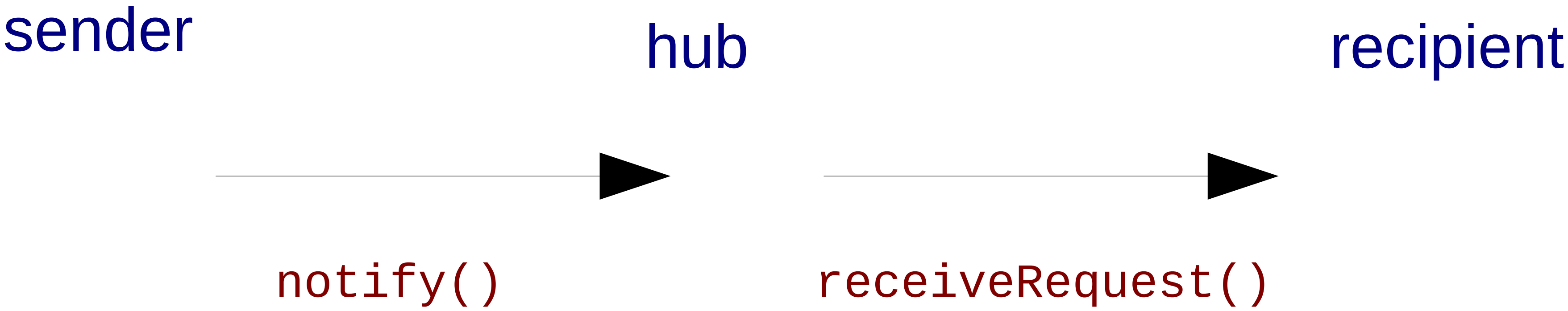
```
receiveRequest(sender-id, receiver-id,
               msg-id, message)
receiveNotification(sender-id, receiver-id,
                    msg-id, message)
```

(calls are the same, but client SHOULD call `hub.reply` after getting `receiveRequest`)

Receive response relating to earlier asynch request:

```
receiveResponse(responder-id, msg-id, response,
                status)
```

# Send and Forget



sender

hub

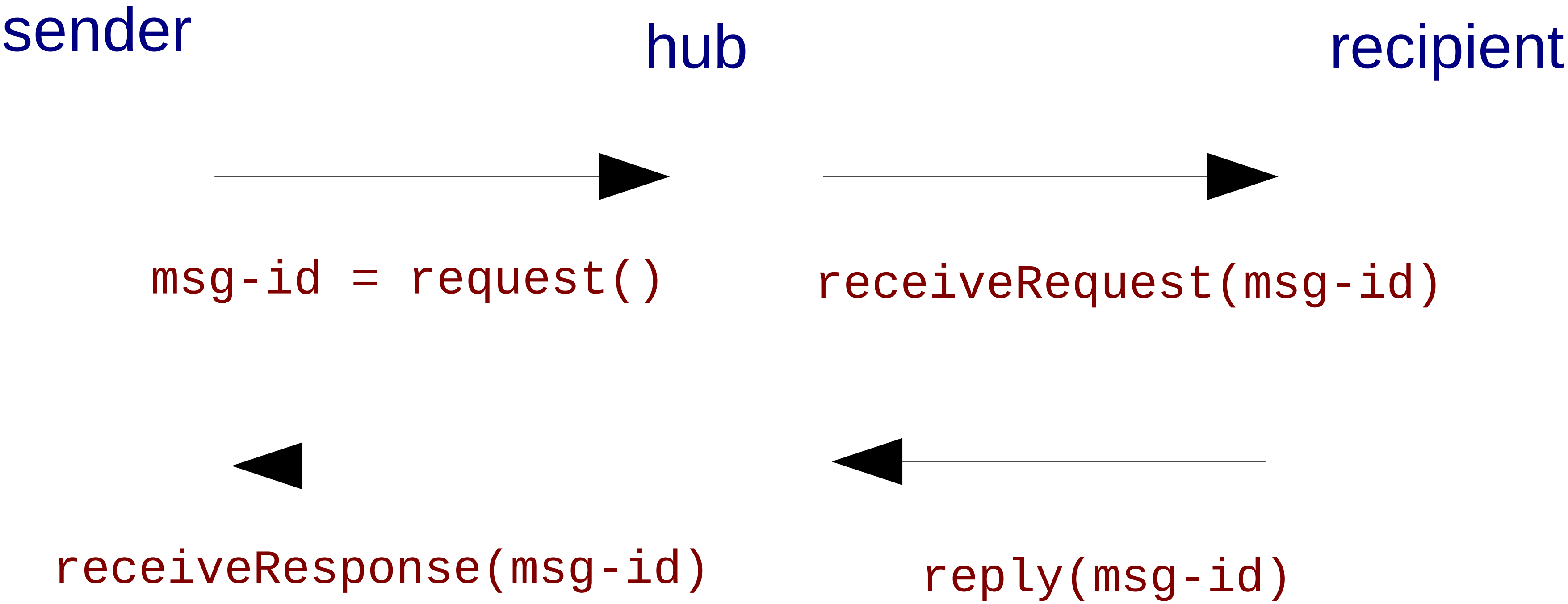
recipient

`notify()`

`receiveRequest()`

Also broadcast equivalent using `notifyAll()`

# Asynchronous (response required)



sender

hub

recipient

`msg-id = request()`

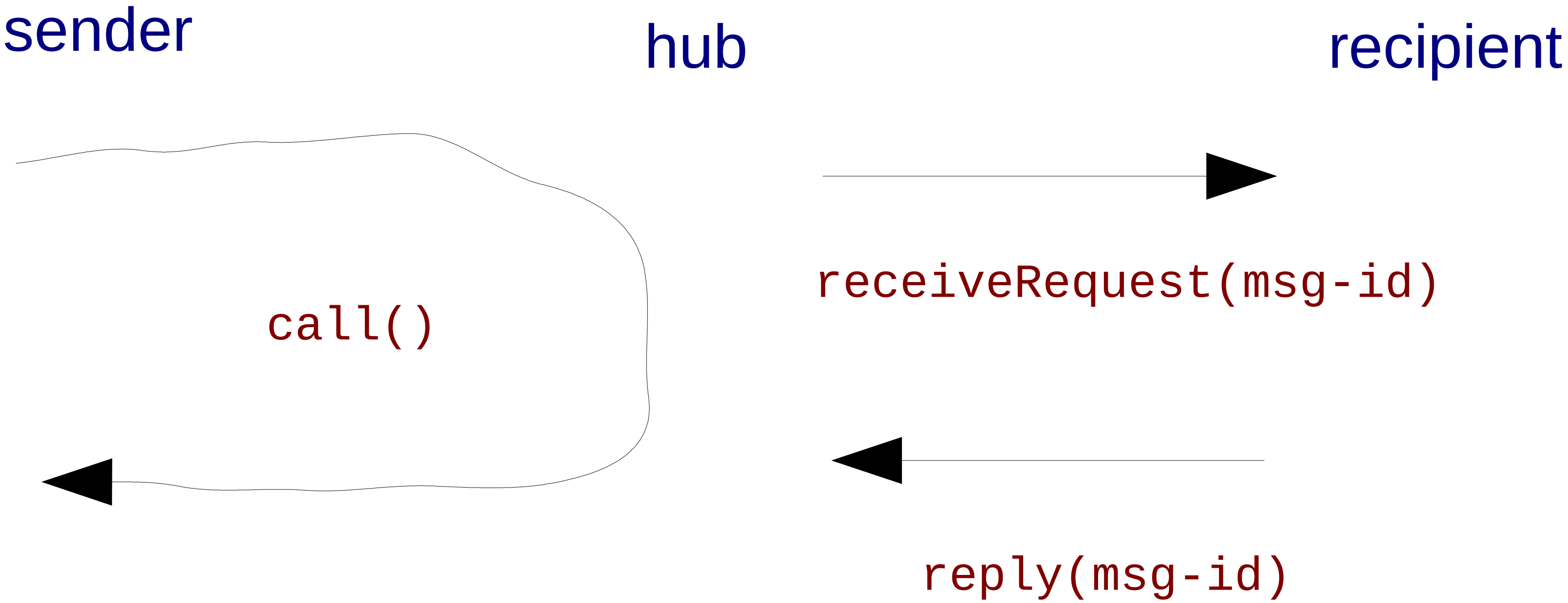
`receiveRequest(msg-id)`

`receiveResponse(msg-id)`

`reply(msg-id)`

Also broadcast equivalent using `requestAll()`

# Synchronous



sender

hub

recipient

`call()`

`receiveRequest(msg-id)`

`reply(msg-id)`

- No broadcast equivalent (use asynch if you need this)
- Identical to asynchronous case from recipient's point of view

# Summary

- Messaging based on asynchronous model
  - No unavoidable timeouts
  - Scarcely more difficult for recipient applications

    (call `hub.reply()` rather than return value from `receive()`)
- Provide synchronous utility method for sender
  - Makes it easy for, e.g. calling from scripting language
  - Less robust, but you can always use asynch
- Details up for grabs
  - Method names (and of course language bindings)
  - Hub methods vs. special messages
  - Different methods vs. same methods with different flags