Auto-Complete

Project Overview

Create an auto-complete client-server system. The server should accept a partially completed word and return complete words which share the common prefix. The server should allow clients to add words, remove words, and retrieve words.

Protocol Details

Connectivity Check

Check the connectivity between the client and the server. The client should send this upon startup. The server should respond with the same format and transaction ID supplied by the client. Client operation should not proceed until the client and server successfully complete a connectivity check. The client program should permit the user to perform additional connectivity checks at any time.

The connectivity check is a client program requirement. The server is not required to enforce the connectivity check before interacting with a client.

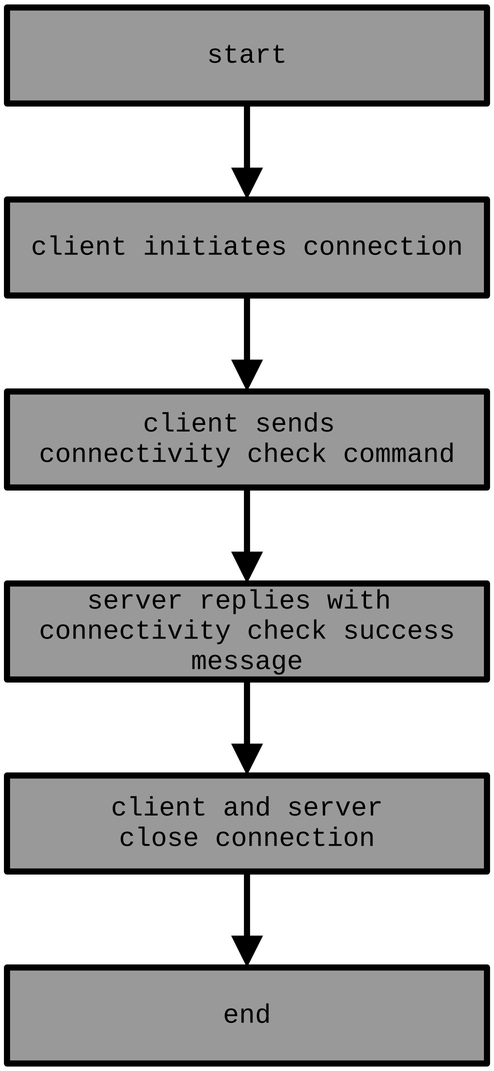
Note: The intent of the transaction ID is to facilitate matching log entries on the client and server. The transaction ID does not uniquely identify the client nor provide any type of authentication mechanism. A reasonable effort should be made by the client to choose a unique transaction ID by using a pseudo-random number, but the protocol does allow transaction ID collisions.

Client request/server response:

0	1	2	3	4	5	6	7
opcode	reserved		t_id				rsvd

Mnemonic	**Size (# bytes)**	**Description**
opcode	1	0x00: connectivity check
reserved	2	reserved field, discard this data
t_id	4	pseudo-random number to identify this transaction
rsvd	1	reserved field, discard this data

Connectivity Check Flow:


start
client initiates connection
client sends connectivity check command
server replies with connectivity check success message
client and server close connection
end

Add Words

Add words to the auto-complete server. The number of words to add must be greater than 0. A variable number of words will immediately follow the add words header.

Client request:

0	1	2	3	4	5	6	7	...	...	...	...	...
opcode	n_words		t_id				rsvd	str_len		str_data		

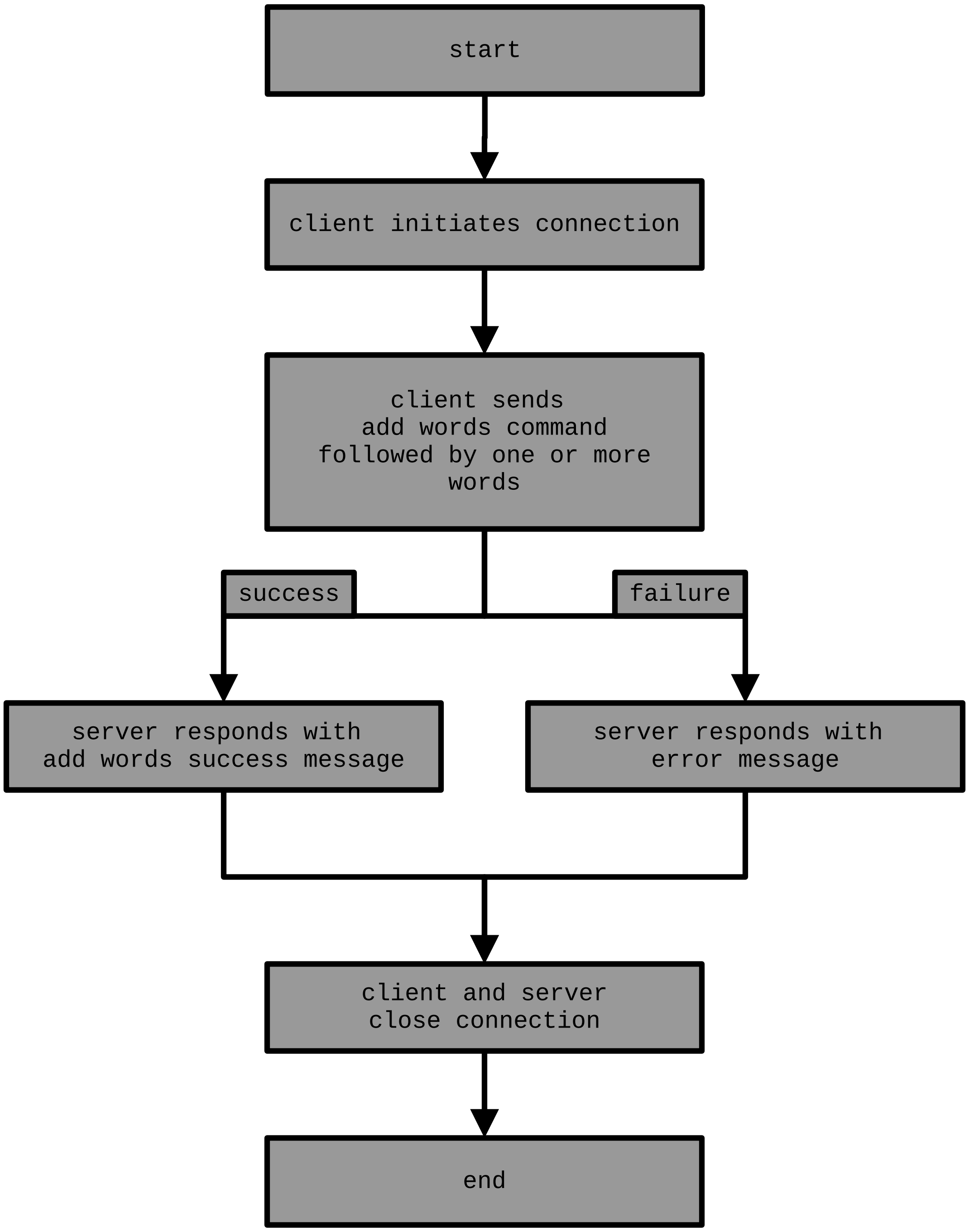
Mnemonic	Size (# bytes)	Description
opcode	1	0x01: add words
n_words	2	number of words to add
t_id	4	pseudo-random number to identify this transaction
rsvd	1	reserved field, discard this data
str_len	2	length of the word to add
str_data	[1, 0xffff]	word string data

Server response:

0	1	2	3	4	5	6	7
opcode	reserved		t_id				rsvd

Mnemonic	Size (# bytes)	Description
opcode	1	0x01: add words
reserved	2	reserved field, discard this data
t_id	4	pseudo-random number to identify this transaction
rsvd	1	reserved field, discard this data

Add Words Flow:


start
client initiates connection
client sends add words command followed by one or more words
success
failure
server responds with add words success message
server responds with error message
client and server close connection
end

Get Words

Get auto-complete suggestions from the server. The client will provide the desired maximum number of results, a minimum word length, a maximum word length, and the desired order of the search results. Immediately following the request header, the client will send exactly one string representing the search prefix.

The server will respond with the search result header, a variable number of search result strings, and leave the connection open. The server should respond with the success message even if 0 words will be returned. The error message should only be returned if the server receives a malformed request or encounters an internal error.

The client will present the results and prompt the user for a selection. The client program will send the selected word back to the server, or a word of size 0 if the client did not choose any of the suggestions. The server should wait up to 15 seconds for a response.

If the user made a selection from the search results, the popularities should be updated. If no selection was made, no popularities should be updated. Regardless of the client-specified request order, the server will use the selection to update the popularities of all words which share the search prefix. Each word which was not selected will have it’s popularity adjusted according to the formula:

```
     popularity *= 1.0 – GAMMA
```

where GAMMA is a compile-time constant representing the learning rate.

The word which was selected will have it’s popularity updated according to the formula:

```
     popularity *= 1.0 + GAMMA
```

In all cases the updated popularity must be within the range (0, 1].

Client request:

0	1	2	3	4	5	6	7	...	...	...	...	...
opcode	n_words		min_len		max_len		order	str_len		str_data		

Mnemonic	**Size (# bytes)**	**Description**
opcode	1	0x02: get words
n_words	2	maximum number of search results to return
min_len	2	minimum word length to return in search results
max_len	2	maximum word length to return in search results
order	1	0x00: alphabetical 0x01: reverse-alphabetical 0x02: popularity
str_len	2	length of the search prefix string
str_data	[0, 0xffff]	prefix string data

Server response:

0	1	2	3	4	5	6	7	...	...	...	...	...
opcode	n_words		reserved					str_len		str_data		

Mnemonic	**Size (# bytes)**	**Description**
opcode	1	0x02: get words
n_words	2	number of search results in the range [0, number of words in client request]
reserved	5	reserved field, discard this data
str_len	2	length of the search result
str_data	[1, 0xffff]	Search result string data

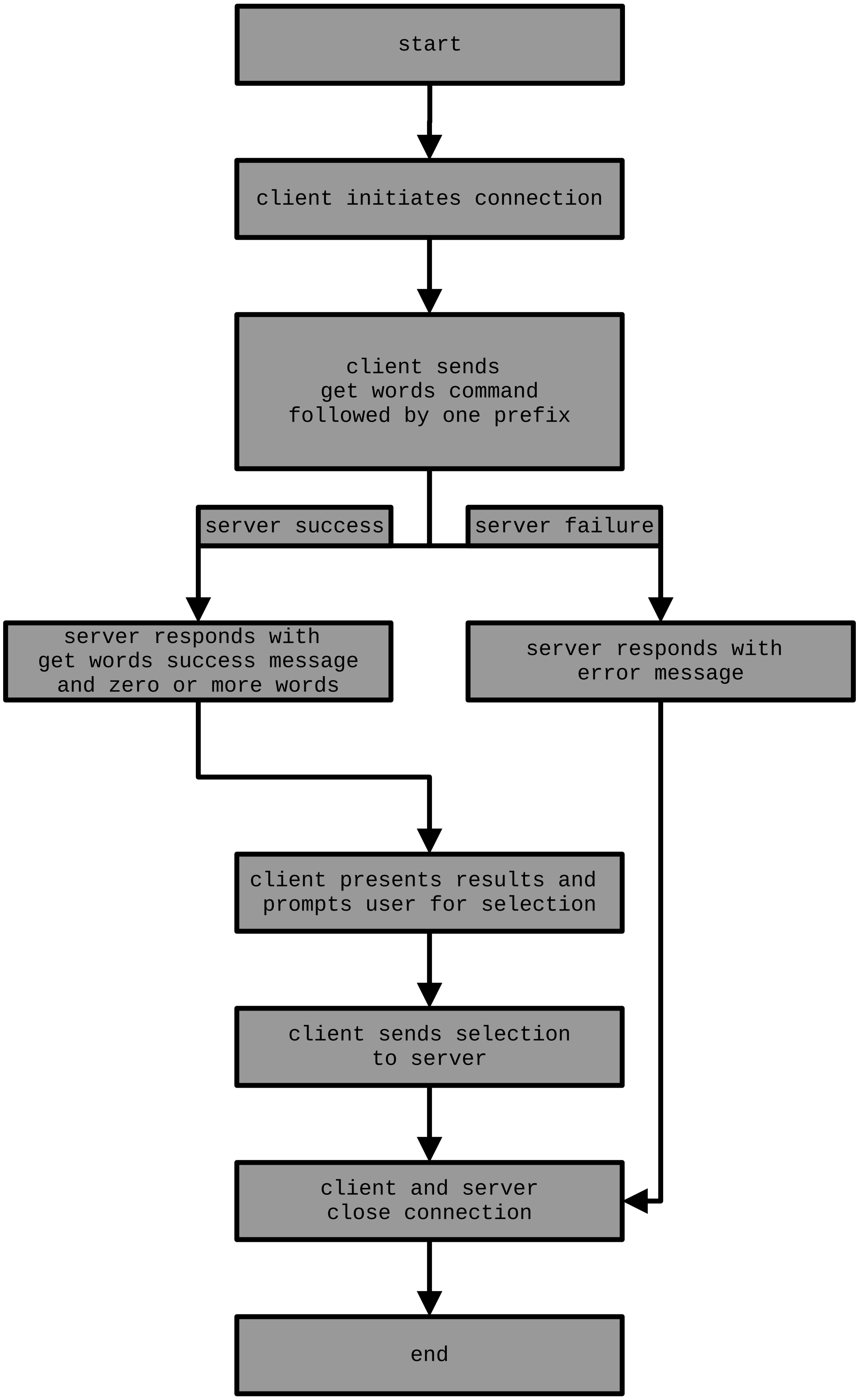
Client response:

The word data of the selection will be returned to the server. If none of the results were selected by the client, the client should communicate this by sending a str_len of 0.

...	...	...	...	...
str_len		str_data		

Mnemonic	**Size (# bytes)**	**Description**
str_len	2	word string length
str_data	[0, 0xffff]	word string data

Get Words Flow:


start
client initiates connection
client sends get words command followed by one prefix
server success
server failure
server responds with get words success message and zero or more words
server responds with error message
client presents results and prompts user for selection
client sends selection to server
client and server close connection
end

Remove Words By Value:

Remove words from the auto-complete server by explicitly specifying the words in their entirety. The client will send a variable number of words immediately following the remove words header. The server will respond with the remove words success message or the error message. The operation is considered successful if at the end of the operation all words specified by the client do not exist in the database. If the client specifies a word which does not exist in the database, this is not failure criteria. A failure message is only returned if a specified word remains in the database, or if an internal server error occurs.

Client request:

0	1	2	3	4	5	6	7	...	...	...	...	...
opcode	n_words		t_id				rsvd	str_len		str_data		

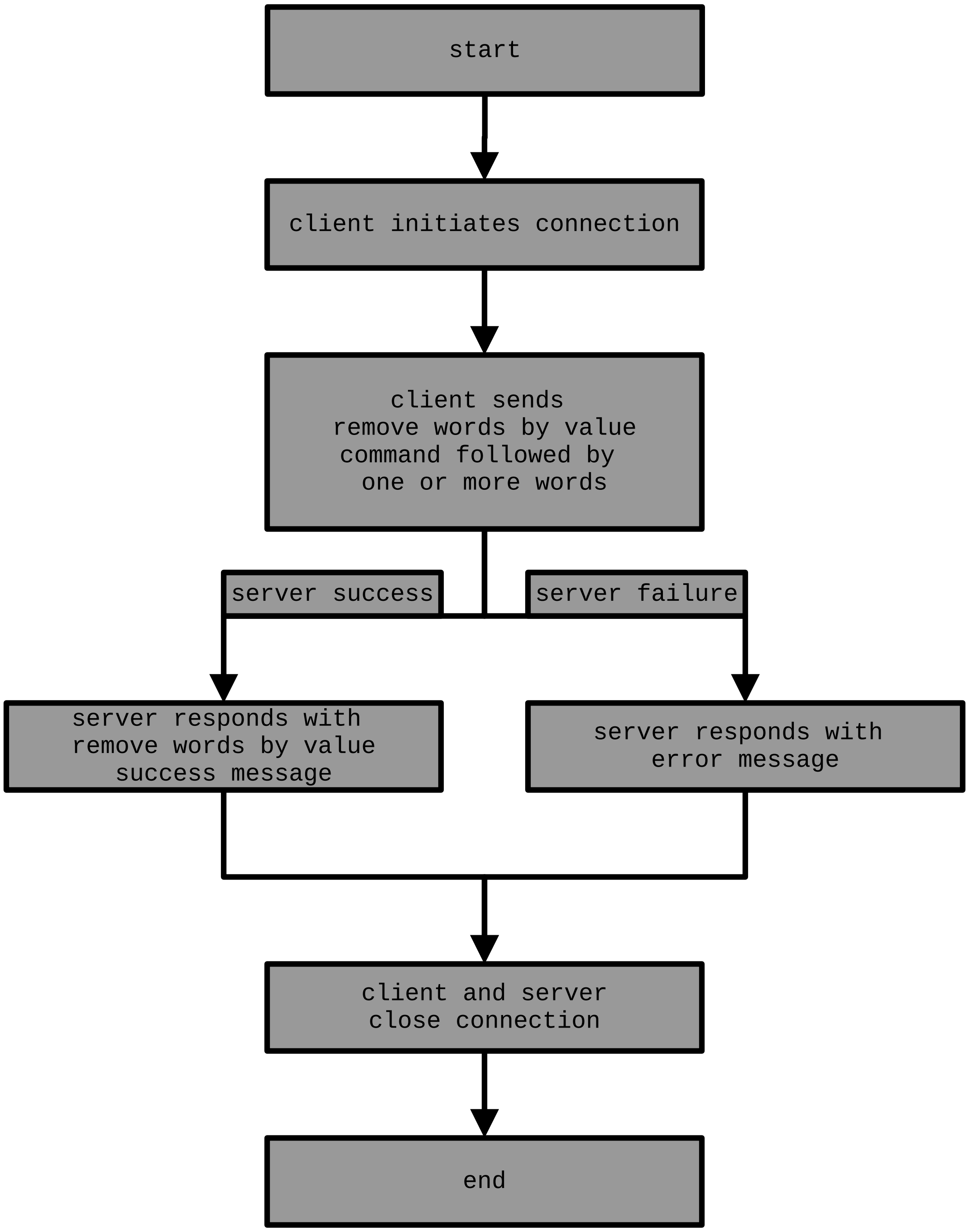
Mnemonic	Size (# bytes)	Description
opcode	1	0x03: remove words by value
n_words	2	number of words to delete
t_id	4	pseudo-random number to identify this transaction
rsvd	1	reserved field, discard this data
str_len	2	word string length
str_data	[1, 0xffff]	word string data

Server response:

0	1	2	3	4	5	6	7
opcode	reserved		t_id				rsvd

Mnemonic	Size (# bytes)	Description
opcode	1	0x03: remove words by value
reserved	2	reserved field, discard this data
t_id	4	transaction ID supplied by the client
rsvd	1	reserved field, discard this data

Remove Words By Value Flow:


start
client initiates connection
client sends remove words by value command followed by one or more words
server success
server failure
server responds with remove words by value success message
server responds with error message
client and server close connection
end

Remove Words By Prefix:

Remove words from the server using the search criteria provided by the get words request. After sending the remove words by prefix header, the client immediately sends the get words request corresponding with the search criteria of the words to be removed. The server responds in the same manner as get words with the exception that the words returned are the words which have been removed from the server. The operation is considered successful if all words which match the specified prefix criteria have been removed from the server. By this definition, if zero words in the server database match the search criteria, the operation is still successful.

Client request:

0	1	2	3	4	5	6	7
opcode	reserved		t_id				rsvd

8	9	a	b	c	d	e	f	...	...	...	...	...
opcode	n_words		min_len		max_len		order	str_len		str_data		

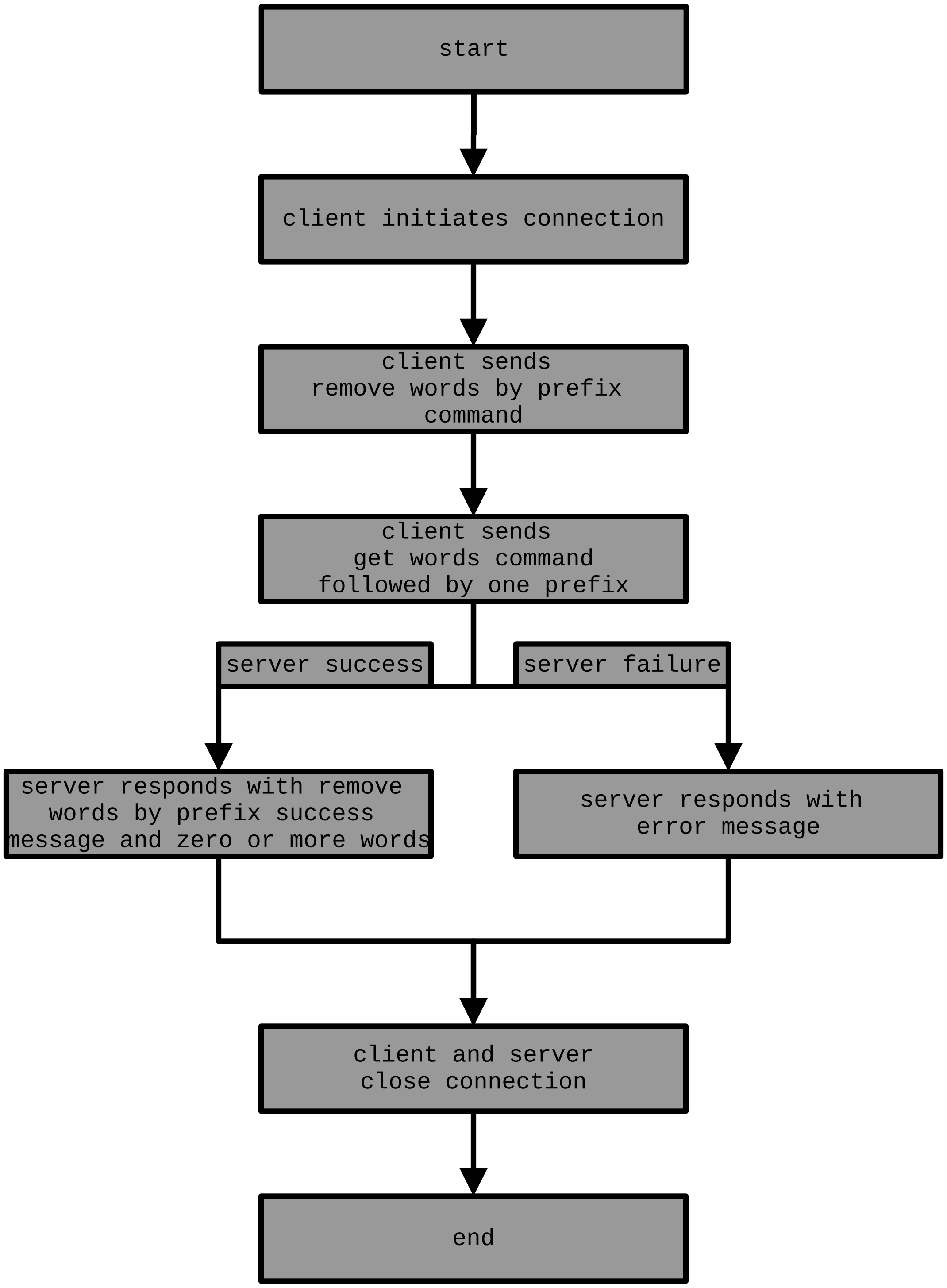
Mnemonic	**Size (# bytes)**	**Description**
opcode	1	0x04: remove words by prefix
reserved	2	reserved field, discard this data
t_id	4	pseudo-random number to identify this transaction
rsvd	1	reserved field, discard this data
opcode	1	0x02: get words
n_words	2	maximum number of search results to remove
min_len	2	minimum word length to remove
max_len	2	maximum word length to remove
order	1	0x00: alphabetical 0x01: reverse-alphabetical 0x02: popularity
str_len	2	length of the search prefix string
str_data	[0, 0xffff]	prefix string data

Server response:

0	1	2	3	4	5	6	7	...	...	...	...	...
opcode	n_words		t_id					str_len		str_data		

Mnemonic	**Size**	**Description**
opcode	1	0x04: remove words by prefix
n_words	2	number of removed words in the range [0, number of words in client request]
t_id	4	pseudo-random number to identify this transaction
Rsvd	1	reserved field, discard this data
str_len	2	length of the removed word
str_data	[1, 0xffff]	removed word string data

Remove Words By Prefix Flow:


start
client initiates connection
client sends remove words by prefix command
client sends get words command followed by one prefix
server success
server failure
server responds with remove words by prefix success message and zero or more words
server responds with error message
client and server close connection
end

Error

An error is indicated by the following message. This message should be passed by the server to the client as a response to any request where the operation failed, when a malformed request is received, or if an internal error occurs. If the error message is in response to a message which has a transaction ID field, the transaction ID should be specified. Otherwise, the transaction ID should be set to 0.

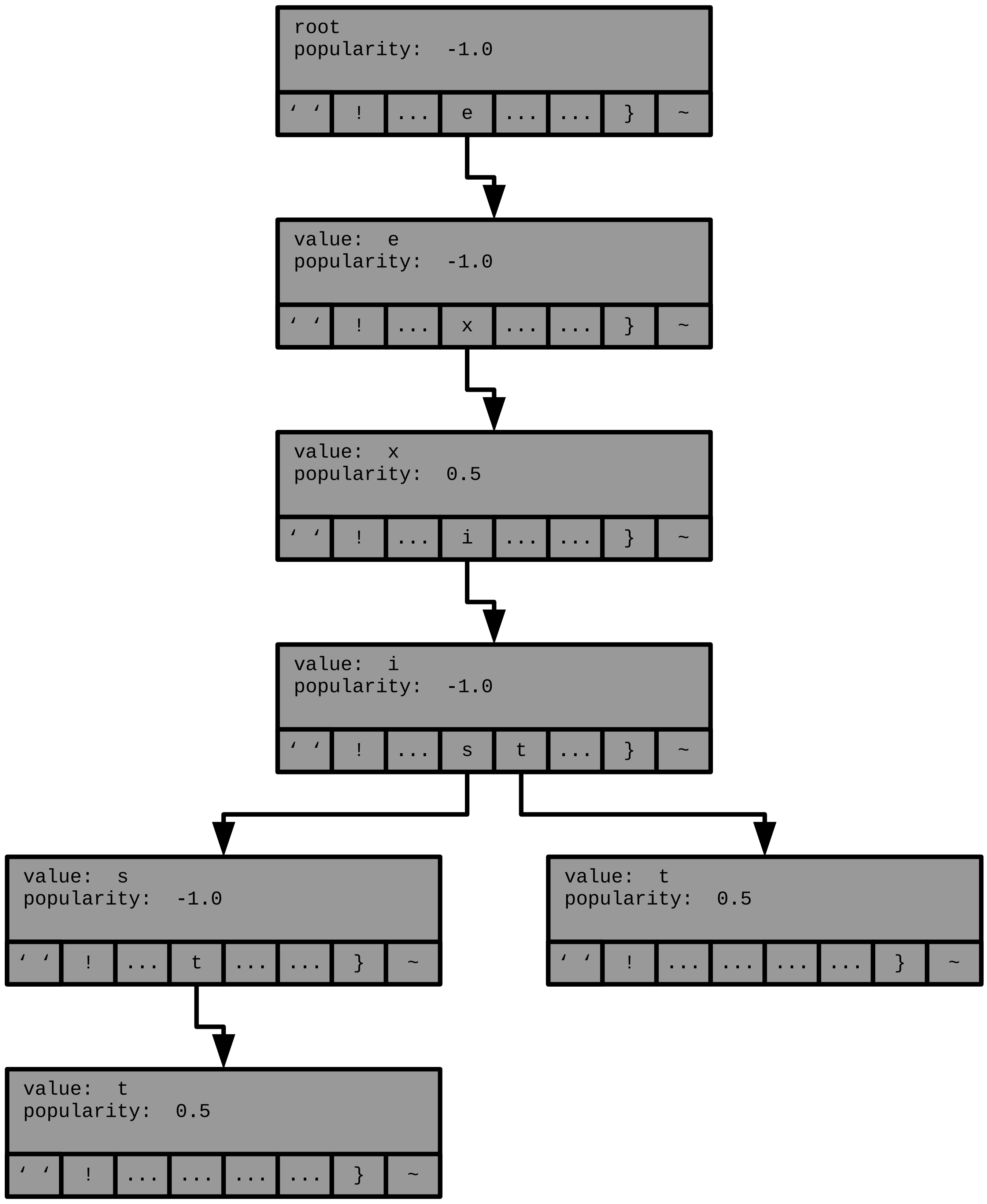
0	1	2	3	4	5	6	7
opcode	reserved		t_id				rsvd

Mnemonic	**Size**	**Description**
opcode	1	0xff: error
reserved	2	reserved field, discard this data
t_id	4	transaction ID (if required by original message)
rsvd	1	reserved field, discard this data

Datastructure

A trie/prefix-tree (https://en.wikipedia.org/wiki/Trie) is ideally suited for the auto-complete search task. Each word within the trie will be unique. The popularity of the word is encoded at the terminating node of the word. The initial value of the popularity should be 0.5. A relatively small value for the learning rate GAMMA should be selected. A value of 0.001 would be a reasonable choice. The server should support all ASCII characters in the range [0x20, 0x7e].

The trie can be constructed from a root node and 0 or more child nodes. A popularity greater than 0 indicates the node is a termination point of a word along the path from the root node to the current node. A popularity less than 0 indicates the node is not a termination point for a word. The three words “ex”, “exit”, and “exist” are represented in the trie below.


root
popularity: -1.0
‘ ‘ ! ... e ... ... } ~
value: e
popularity: -1.0
‘ ‘ ! ... x ... ... } ~
value: x
popularity: 0.5
‘ ‘ ! ... i ... ... } ~
value: i
popularity: -1.0
‘ ‘ ! ... s t ... } ~
value: s
popularity: -1.0
‘ ‘ ! ... t ... ... } ~
value: t
popularity: 0.5
‘ ‘ ! ... ... } ~
value: t
popularity: 0.5
‘ ‘ ! ... ... } ~

Additional Requirements

- Server:
  - must be written in C
  - must accept a port number to listen on as a command line parameter
  - must maintain the state of the word database across program restarts/system crashes
  - must support at least 10 concurrent connections
  - must use TCP
- Client:
  - must be written in Python
  - must be interactive
  - must provide menu items for each of the protocol commands
  - must support adding words from a file (newline separated)
  - must use TCP
- Prepare a README file which explains at a minimum:
  - how to build/compile the project
  - how to start the server with a new database
  - how to start the server with an existing database
  - how to use the client program
- All stale connections should time out after a reasonable period has elapsed. Pick a value that would work well with real-world network latency (2-6 seconds would be reasonable). Unless otherwise specified, apply this timeout to every step in the protocol where a party will be waiting for a response.

Areas of Emphasis

- Struct packing/unpacking
- Trie data structure
- Depth first search
- Stack/queue data structures
- Sorting algorithms
- Comparison of floating point types
- Serialization of floating point types

Example Client/Server Interaction

The following is an example client/server interaction from the client’s perspective. This is an example user interface, not the required implementation.

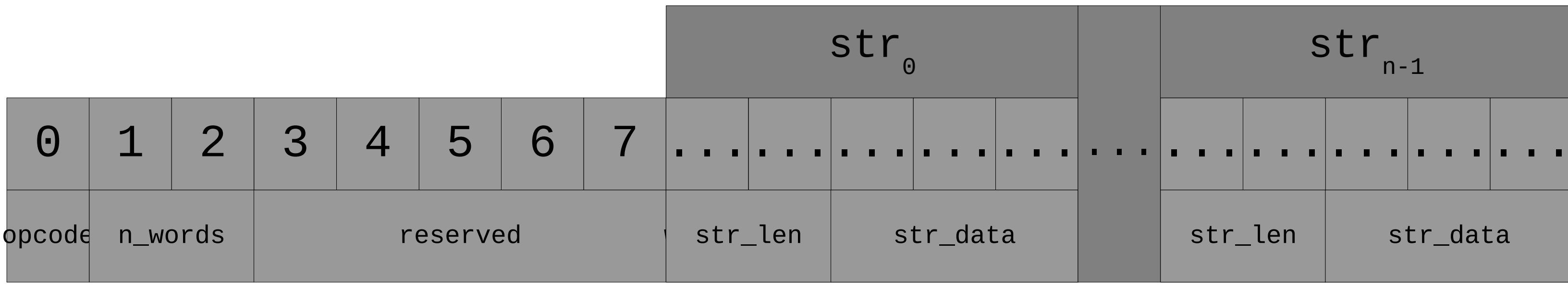
```
./autocomplete-client localhost 1234

autocomplete-client
remote host is listening on specified port
remote host appears to be an autocomplete server

################################################################################
what would you like to do?

0 - connectivity check
1 - add words
2 - get words
3 - remove words by value
4 - remove words by prefix
5 - add words from file

> 1

provide a comma separated list of words

> ex, exit, exist, existential, extraneous

operation successful

################################################################################
what would you like to do?

0 - connectivity check
1 - add words
2 - get words
3 - remove words by value
4 - remove words by prefix
5 - add words from file

> 2

provide the search criteria

prefix: ex
max results: 10
min length: 3
max length: 10
order (0-alphabetical, 1-reverse-alphabetical, 2-popularity): 1

operation successful
number of results: 3
words:
    extraneous
    exit
    exist
```

```
select a word from the list above (ENTER for none)

> exist

operation successful

################################################################################
what would you like to do?

0 - connectivity check
1 - add words
2 - get words
3 - remove words by value
4 - remove words by prefix
5 - add words from file

> 4

********************************************************************************
********************************************************************************
** WARNING                                                                    **
** you are about to remove words by prefix                                    **
** press CTRL + D to abort                                                    **
********************************************************************************
********************************************************************************

provide the search criteria

prefix: exi
max results: 10
min length: 3
max length: 100
order (0-alphabetical, 1-reverse-alphabetical, 2-popularity): 1

operation successful
number of results: 3
words:
    exit
    existential
    exist

################################################################################
what would you like to do?

0 - connectivity check
1 - add words
2 - get words
3 - remove words by value
4 - remove words by prefix
5 - add words from file

> 2

provide the search criteria

prefix:
max results: 10
min length: 0
max length: 100
```

```
order (0-alphabetical, 1-reverse-alphabetical, 2-popularity): 0

operation successful
number of results: 2
words:
    ex
    extraneous

select a word from the list above (ENTER for none)

>

operation successful

################################################################################
what would you like to do?

0 - connectivity check
1 - add words
2 - get words
3 - remove words by value
4 - remove words by prefix
5 - add words from file

>
```

Transferring Multiple Strings

If a command is followed by a variable number of strings (ex. get words or add words), each str_data should have it's own str_len.


str0
strn-1
0
1
2
3
4
5
6
7
...
opcode
n_words
reserved
str_len
str_data
str_len
str_data