
SPAM
BURGLAR

“We don't need to give a 5 minute soliloquy on what spam is and why it's bad because everyone knows that already.”

- Tom Ashley

# Why do we need a new method?

- Old methods:
  - Blacklist – A list of addresses not to accept email from
  - Checking for specific words – such as “click” or “Viagra”
  - Bulk Mail – mail not sent specifically to your address

# Why is our Bayesian Filter different?

- Cross-platform
- Works with most e-mail clients
- Can be trained to the individual' s preferences

# What we're going to talk about

- Algorithm
- Proxy
- Analyzers
- Demo
- Statistics

# Bayesian Filtering Overview

- Email messages are assigned probabilities indicating likelihood of being spam
- Spam probability of message depends on spam probabilities of individual tokens
- Spam probability of tokens depends on frequency of occurrence in previously-identified messages
- Algorithm is biased against classifying messages as spam
- Derived from Paul Graham's article, “A Plan For Spam” (www.paulgraham.com/spam.html)

# What comprises a token?

- All alphanumeric characters
- $ ! : - '
- Other characters are considered token delimiters
- Tokens in message headers have special classification: header_name*token_name
- e.g., if the subject header contains “Promotion!”, it will be stored as “subject*Promotion!”

# Counting and storage of tokens

- Tokens and frequencies are stored in two separate hashtable objects (spam and nonspam)
- For each token in a message, check appropriate hashtable for its frequency
- If token exists in table, increment its frequency. Otherwise, add it to table with frequency = 1.
- In each corpus, keep a count of total number of messages

# Spam probability of tokens

$$\frac{\left(\frac{b}{nbad}\right)}{\left(\left(\frac{b}{nbad}\right)+\left(2\frac{g}{ngood}\right)\right)}$$

- b = # of occurrences of token in spam table
- g = # of occurrences of token in nonspam table
- nbad = # of spam messages in total
- ngood = # of nonspam messages in total

# Spam probability of tokens

- Only assign a probability if 2g + b >= 5
- If token only exists in spam corpus, assign a probability of 0.99
- If token only exists in nonspam corpus, assign a probability of 0.01

# Spam identification of messages

- For each token in a message, obtain its spam probability from table.
- If token not in table, use probability of 0.4 as a default.
- Only use the 15 most “significant” tokens in message probability calculation.

# Spam identification of messages

$$\frac{\left(\prod_{i=1}^{15} P(T_i)\right)}{\left(\left(\prod_{i=1}^{15} P(T_i)\right)+\left(\prod_{i=1}^{15} (1-P(T_i))\right)\right)}$$

- Where P(T) is the spam probability of token T
- If the resulting message probability exceeds a threshhold, identify the message as spam

# Example calculation

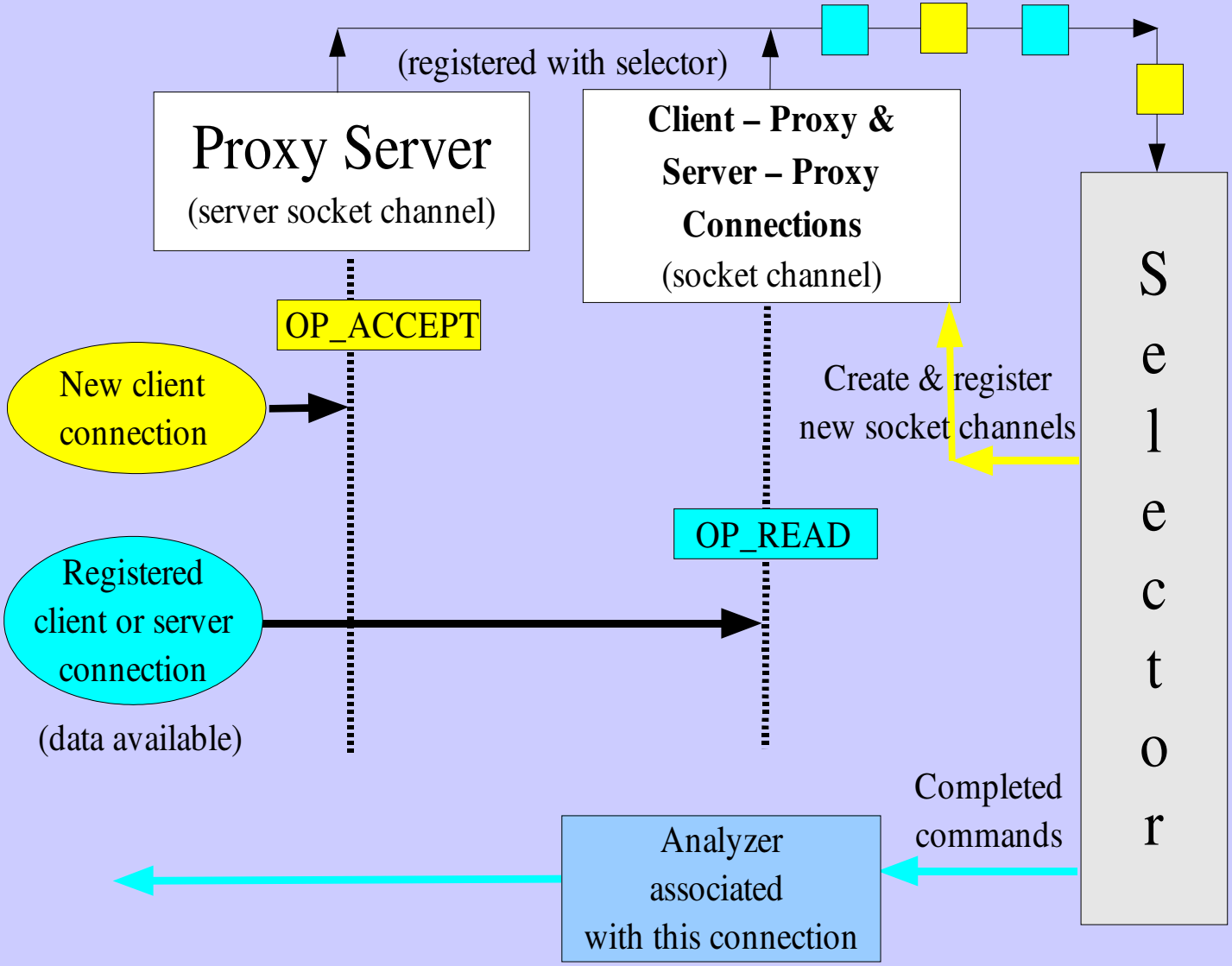
| Token | Probability |
|---|---|
| arial | 0.99 |
| subject*FREE | 0.99 |
| subject*V1AGRA | 0.99 |
| subject*Limited | 0.99 |
| promotion | 0.99 |
| offering | 0.99 |
| V1AGRA | 0.99 |
| FREE | 0.99 |
| limited | 0.99 |
| offer | 0.99 |
| continue | 0.01 |
| brigand | 0.01 |
| enervate | 0.01 |
| quite | 0.03 |
| prices | 0.93 |

Message Probability = 0.99999

# Proxy Server

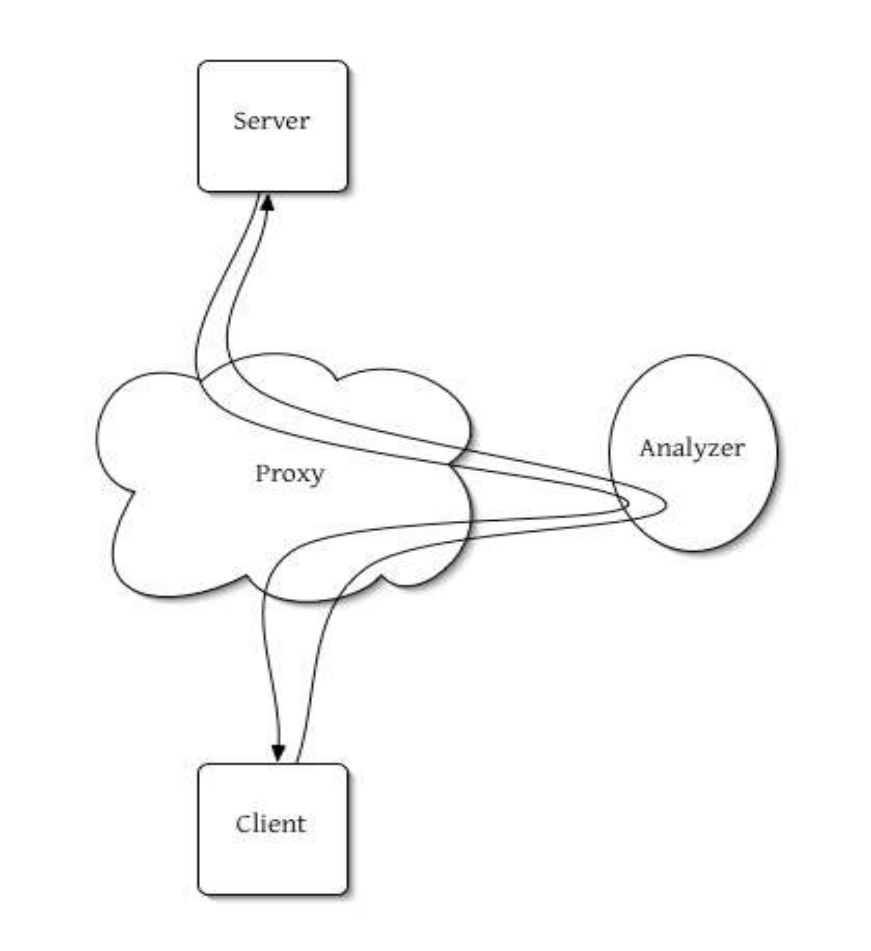
- Sits between client and mail server
  - Uses non-blocking IO
- Waits for client to connect
  - Accepts client connection, then establishes a connection from the proxy to the mail server
  - The client-proxy and server-proxy connections are registered with a **selector**
- Proxy server continues to wait for:
  - New client connections
  - Data to arrive on a connection registered with the selector

# How the Selector Works



# Analyzers

- Analyze communications intercepted by the proxy before allowing it to pass them on



# Analyzers

- Analyze communications intercepted by the proxy before allowing it to pass them on
- Build Email objects using data from the server and pass them on for evaluation as spam/nonspam
- Prevent the client and server from talking while working
- Store all processed messages for future training

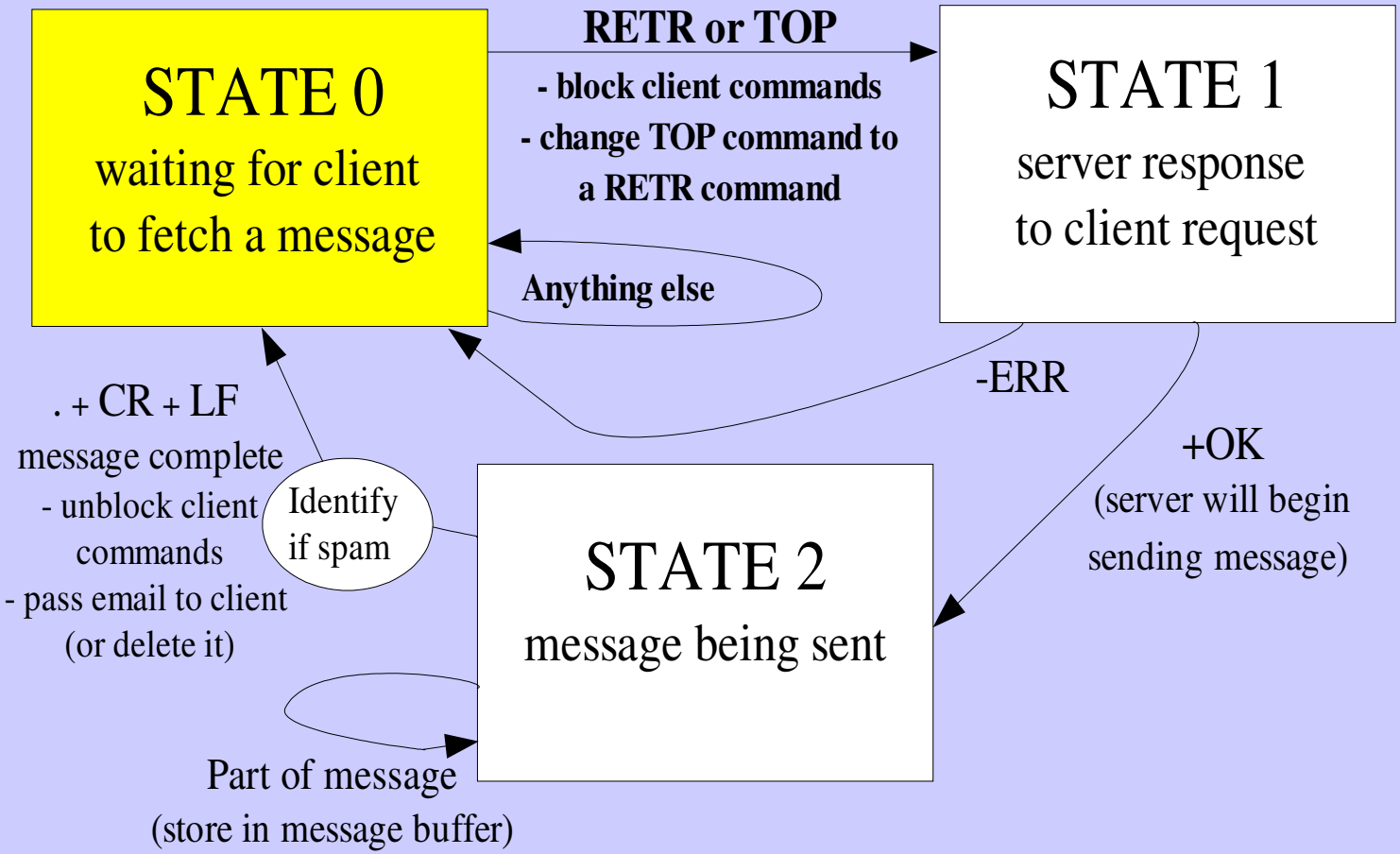
# Post Office Protocol (POP)

- Messages are generally downloaded to client rather than kept on the server
- POP accounts only have one mailbox (inbox)
- Only one message may be retrieved at a time
- Clients must determine if new mail has arrived
- Deleting a message immediately and permanently removes it from the server

# POP Analyzer

- Email messages are captured as they are sent from the server to the client
  - Messages only need to be retrieved once
- Once the spam status of a message has been determined, either:
  - Modify headers in transit by marking the subject line before passing the email on to the client
  - Send a delete request to the server for spam messages

# POP Analyzer Session

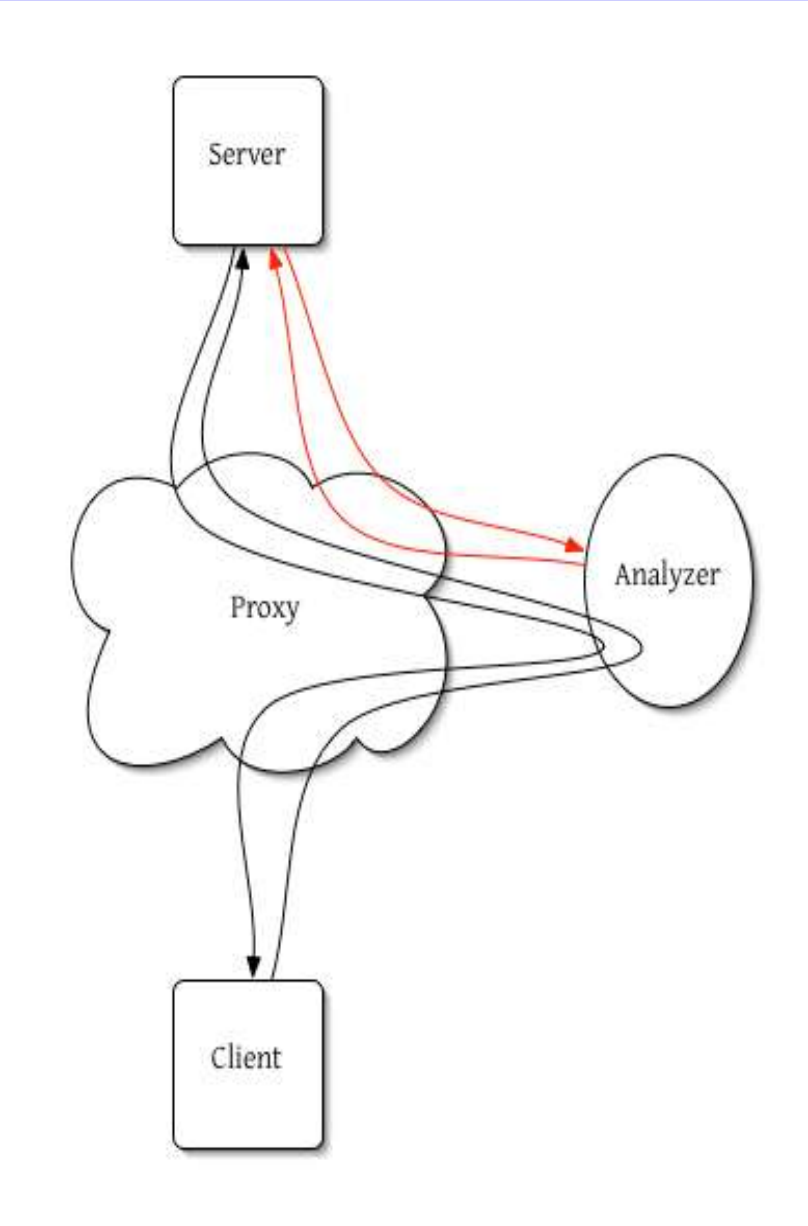


# Internet Mail Access Protocol (IMAP)

Important Features:

- Messages are kept on server instead of being downloaded to the client.
- Flags are used to keep track of message attributes.
  - The “\Recent” flag tells you if the message has arrived since the last time the mailbox was looked at.
  - The “\Flagged” and “\Deleted” flags are the ones that we use to indicate to the user whether a message is spam or not.

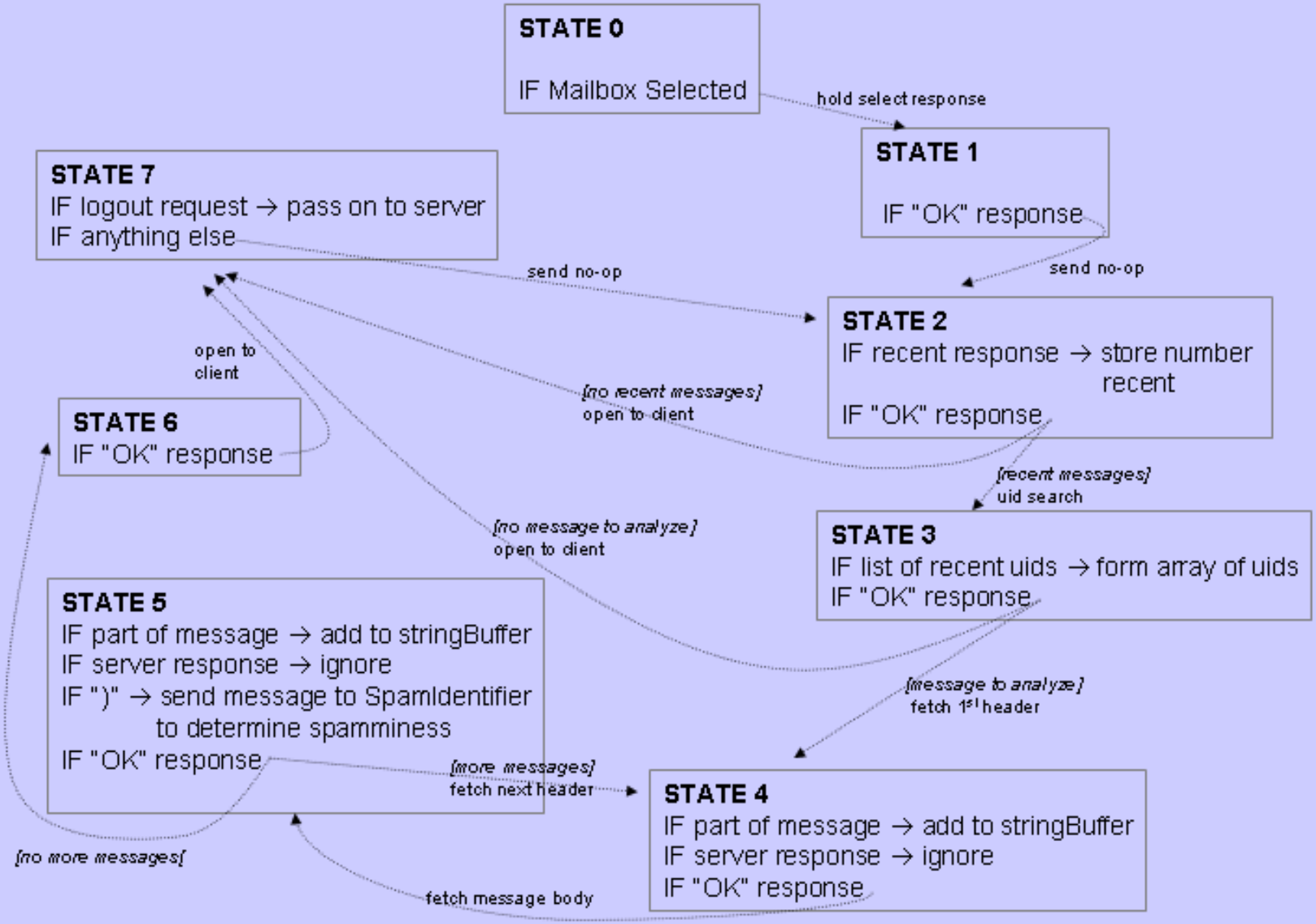
# IMAP Analyzer



# IMAP Analyzer

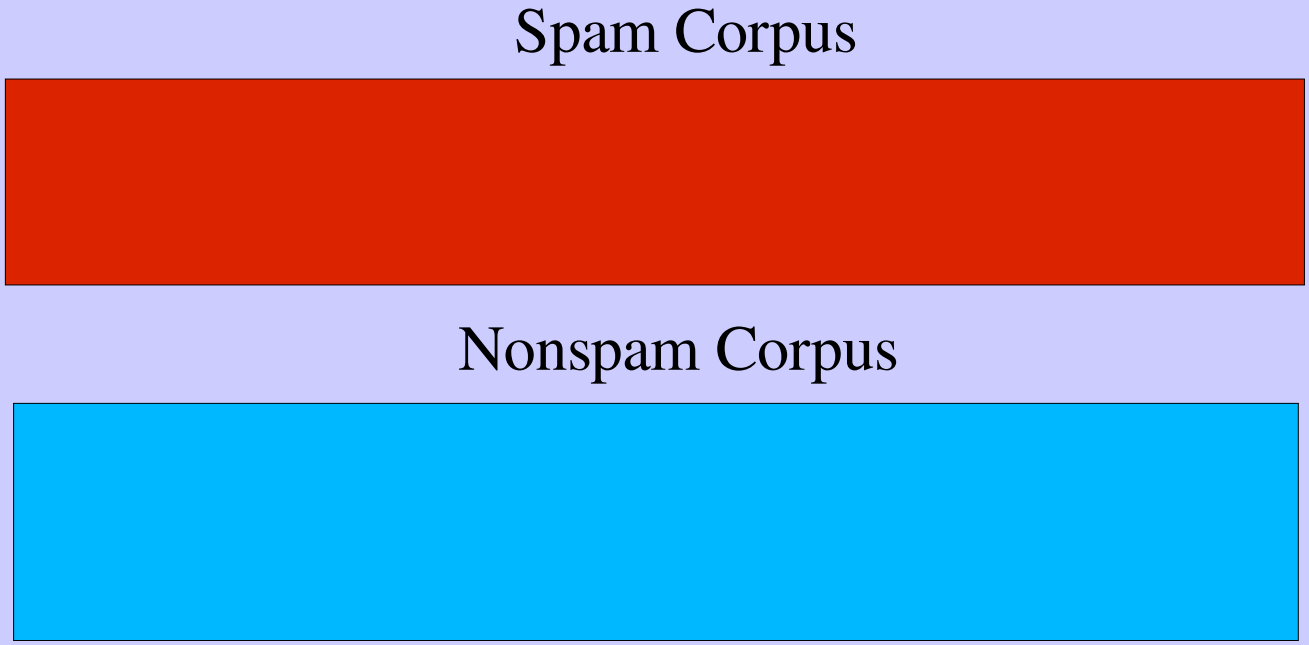
- Commands and responses between the server and client pass through unchanged
- Analyzer carries on its own conversation with the server, independent of the client
- All analysis and filtering of messages occurs before the client can see them
- Checking for new messages must happen every time the client sends a command

# IMAP Analyzer States



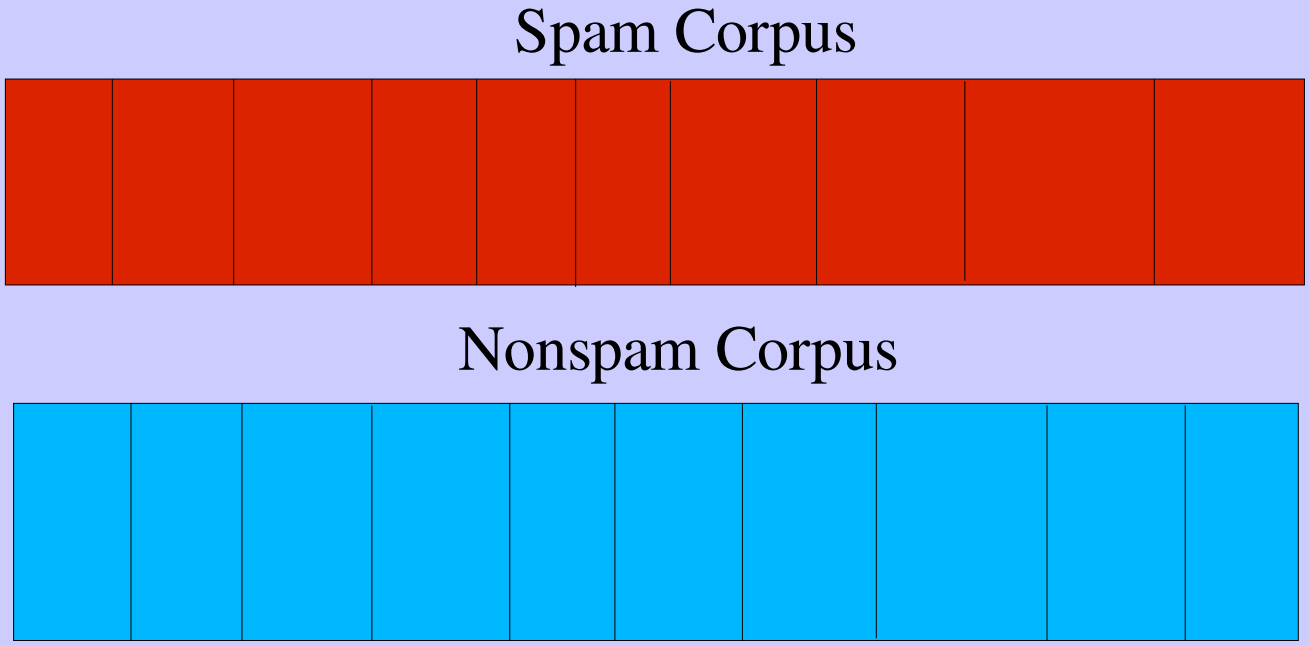
# 10-fold Cross Validation

- Take two corpora: one of spam emails, one of nonspam



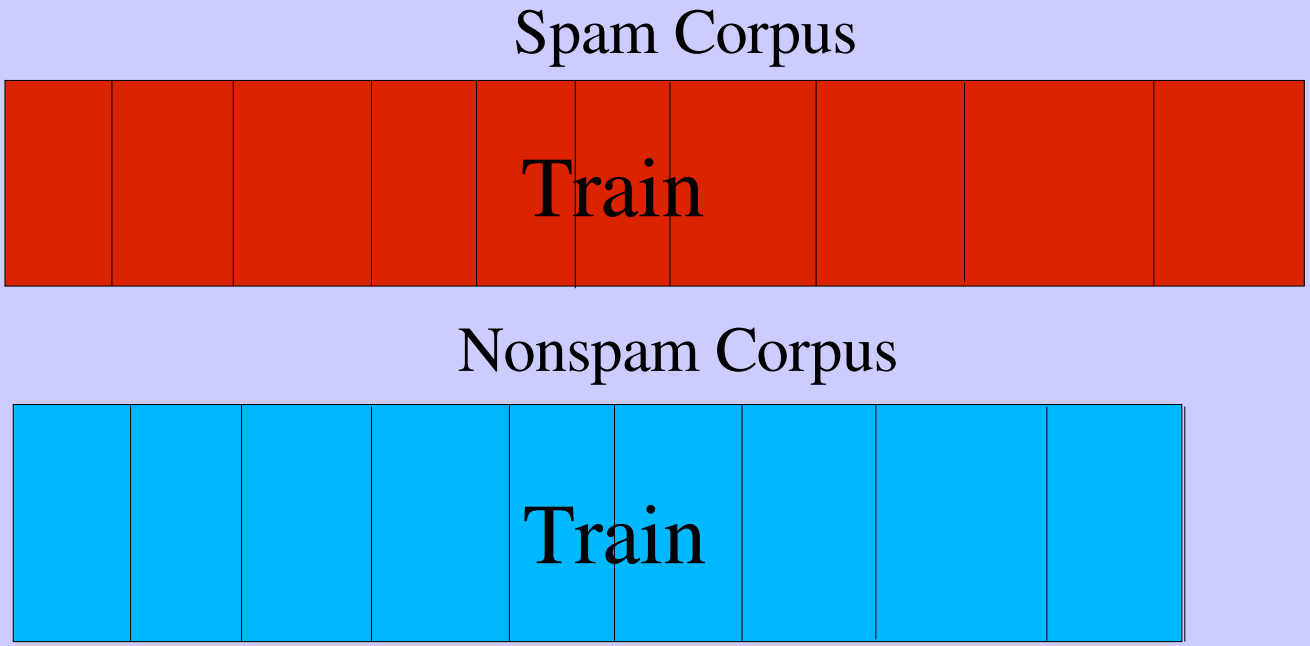
# 10-fold Cross Validation

- Randomly split them into 10 roughly equal parts



# 10-fold Cross Validation

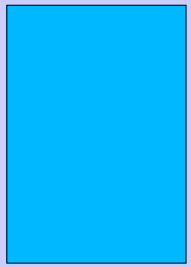
- Train on all of one corpus, and 90% or the other



# 10-fold Cross Validation

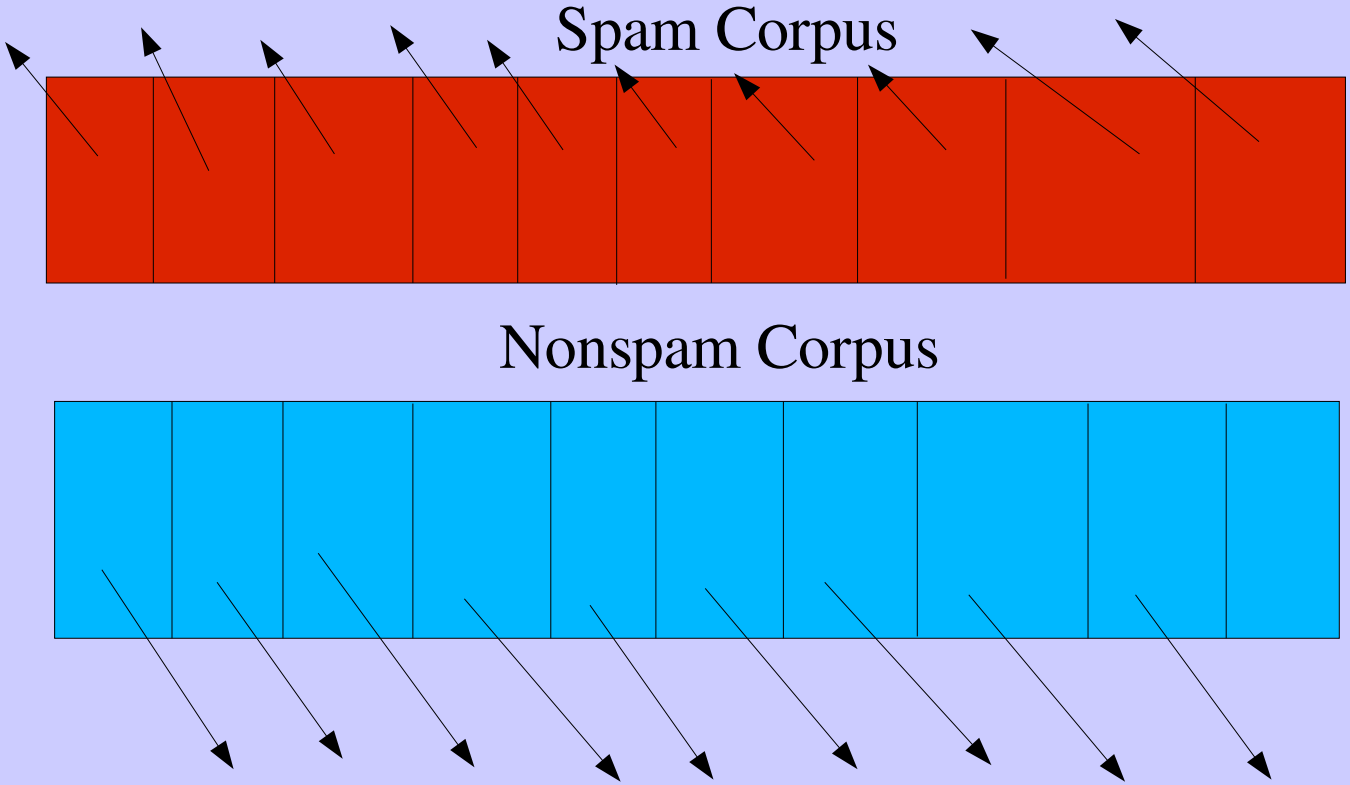
- Test on the remaining 10% of one corpus and record how many emails are marked correctly, and how many are marked incorrectly

Part 10 of the nonspam corpus



# 10-fold Cross Validation

- Repeat the process for each part of both corpora
- Compile your results



# Validation Results

| | Marked as Spam | Marked as Nonspam | Accuracy |
|---|---|---|---|
| Spam | 2317 | 37 | 98.4% |
| Nonspam | 0 | 3104 | 100.0% |

# About False Negatives

- User Misidentification
- Passed by one token/Too many unfamiliar tokens (Reduced with more extensive training)
- Spam Writer's tricks
  - innocent tokens randomly placed in message (sometimes hidden inside HTML tags)
  - short, nondescriptive messages with links to a site
  - Using images rather than text for advertisement
  - Adding characters to high spam probability words to make them unrecognizable

# Example

imbrue midnights porpoise exponentiation scurrilous pondering expectedly tents identically tansy adorable midspan policies poring scare talon ar temperance tales bolstered breaches meaty boss creek politic texas plight bohr adjusted megawatt brainstem adjudges mediated meteorology artemis excessive expandable activism mercilessly midband 10th pothole booboo bowl mighty husks humidifiers botulism metaphoric telephoners sec exams excitation acknowledgeable screaming immediacies bernardo aquarius tents polyploidy boners melodiously excepted adversaries examples horseshoe bottlenecks

[1][hgh1.gif]

excrete bragging bolstering crew portended scantiness accidentally

# Problems

– Poor implementations

– GUI matching functionality

– Speed/Performance

– Security

# If we had more time . . .

- Put training on a separate thread
- Implement alternate filtering algorithms
- More advanced error handling
- Optimize
  - sizes of string buffers, hashtables
- Implement more of Graham's ideas
  - token degradation, multiple tokens
  - follow links in email

# THANK YOU!

Paul Graham

RFC 1939 author J. Myers

RFC 2060 author M.Crispin

Java APIs

Dave Musicant

Mike Tie

Jen's Mom

ITS—Chris Doten