RSPAMD

spam filtering system

Всеволод Стахов
https://rspamd.com

HighLoad++

Часть I: Знакомство

Зачем нужен rspamd?

SpamAssassin → rspamd в “Рамблер-Почте”

```
CPU: 99.4% user,  0.0% nice,  0.4% system,  0.2% interrupt,  0.0% idle
Mem: 2696M Active, 2354M Inact, 631M Wired, 21M Cache, 828M Buf, 2225M Free
Swap: 2060M Total, 2060M Free

  PID USERNAME  THR PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
43293 spamd       1  62    0 61776K 39416K RUN     1   0:14 15.38% perl5.8.8
43291 spamd       1  64    0 68300K 45852K RUN     1   3:33 14.60% perl5.8.8
43321 spamd       1  63    0 61096K 39072K select  0   0:10 14.36% perl5.8.8
43290 spamd       1  55    0 68636K 46372K select  0   3:31 12.50% perl5.8.8
43292 spamd       1  55    0 66268K 43348K select  0   0:29 12.16% perl5.8.8
43320 spamd       1  52    0 58812K 36908K select  0   0:09  9.96% perl5.8.8
```

```
CPU:  3.4% user,  0.0% nice,  0.0% system,  0.4% interrupt, 96.2% idle
Mem: 2713M Active, 2355M Inact, 633M Wired, 21M Cache, 828M Buf, 2204M Free
Swap: 2060M Total, 2060M Free

  PID USERNAME  THR PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
42785 nobody      1   4    0   275M   238M kqread  0   0:22  9.18% rspamd
```

Какой бывает спам

Актуальные виды

Какой бывает спам

Актуальные виды

- Мошеннические письма (scam):

Какой бывает спам

Актуальные виды

- Мошеннические письма (scam):
  - “Нигерийские” письма

Какой бывает спам

Актуальные виды

- Мошеннические письма (scam):
  - “Нигерийские” письма
  - Фишинг (банки, соцсети)

Какой бывает спам

Актуальные виды

- Мошеннические письма (scam):
  - “Нигерийские” письма
  - Фишинг (банки, соцсети)
- Реклама:

Какой бывает спам

Актуальные виды

- Мошеннические письма (scam):
  - “Нигерийские” письма
  - Фишинг (банки, соцсети)
- Реклама:
  - Традиционная текстовая

Какой бывает спам

Актуальные виды

- Мошеннические письма (scam):
  - “Нигерийские” письма
  - Фишинг (банки, соцсети)
- Реклама:
  - Традиционная текстовая
  - Картинки, сложный html и другие методы обфусикации

Какой бывает спам

Актуальные виды

- Мошеннические письма (scam):
  - “Нигерийские” письма
  - Фишинг (банки, соцсети)
- Реклама:
  - Традиционная текстовая
  - Картинки, сложный html и другие методы обфусикации
  - Подделки под сообщения соцсетей

From Mail Administrator <notifyusage00@inbox.ru>

Subject **HIGHSECURE.RU Data Usage Alert**

To Me <vsevolod@highsecure.ru>

Dear Vsevolod,

Mailbox Storage

98% Full

Your HIGHSECURE.RU Mailbox Has Exceeded Its Quota Limit And You May Not Be Able To Send Or Receive New Mails Until You Re-Validate.

To Re-Validate, Please Click: RE-VALIDATE For More Storage Data to Be Added To Your Mailbox

Thank You,

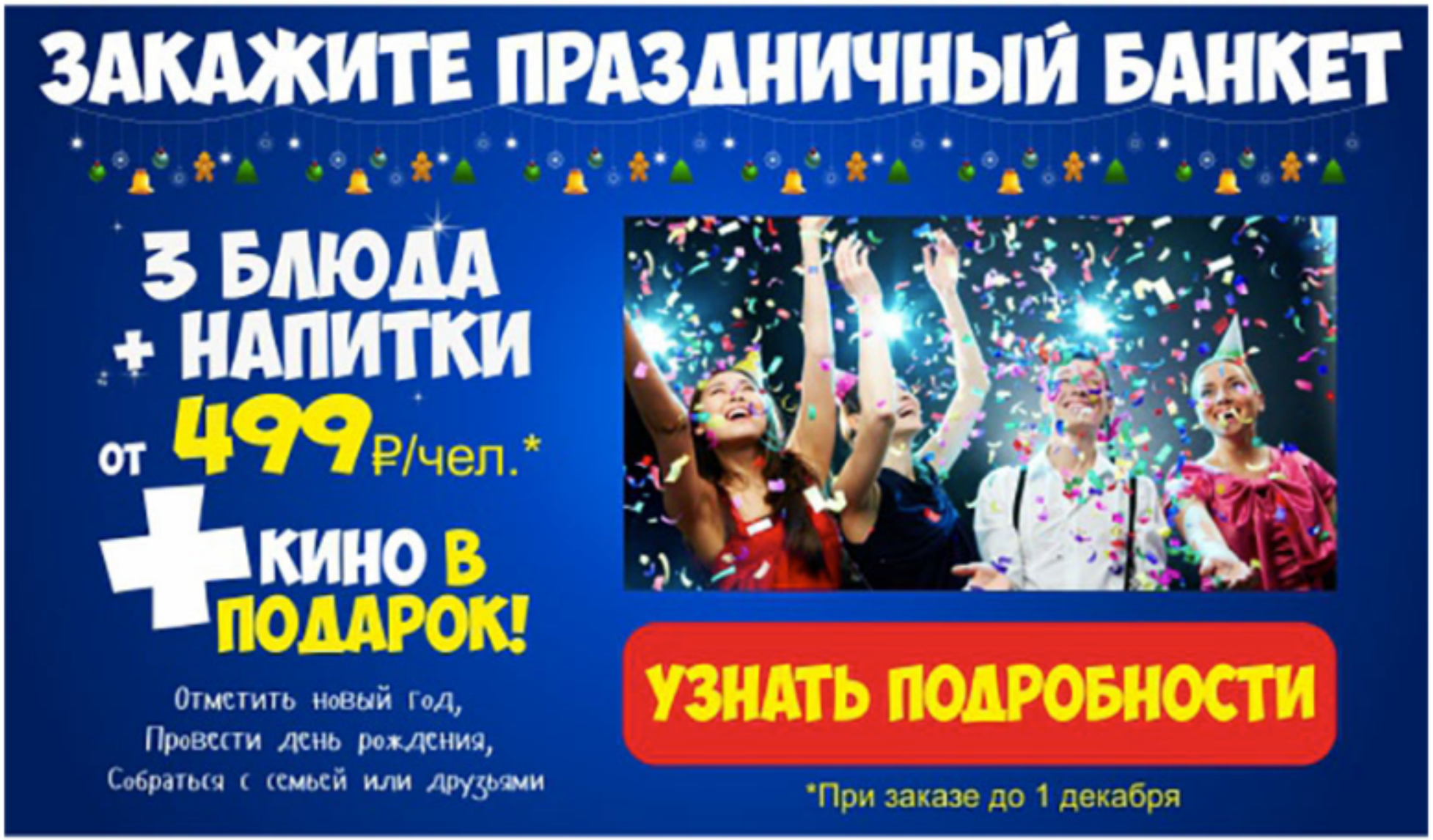
HIGHSECURE.RU Mail Team

Scam: типичный пример

From Организатор <orgolhujfv@giobioka.co.ua>
Subject **Праздник от 499 руб. на гостя**
To nkarpova@gsib.ru

Reply | Reply All | Forward | Redirect | Archive

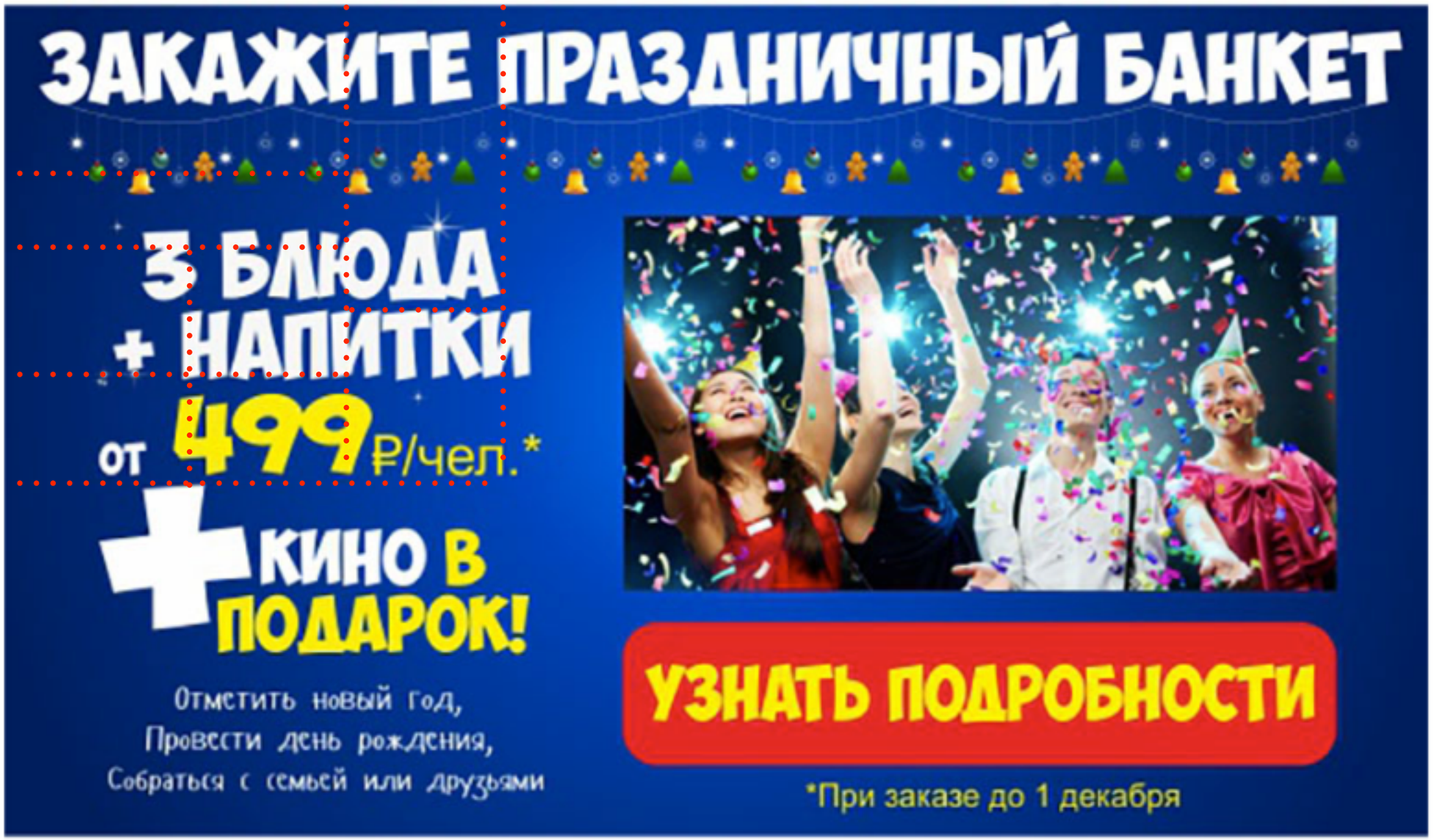
ТОЛЬКО ДЛЯ: МОСКВЫ, КАЗАНИ, ОРЕНБУРГА, ЕКАТЕРИНБУРГА, ИЖЕВСКА



Реклама: примеры

From Организатор <orgolhujfv@giobioka.co.ua>
Subject **Праздник от 499 руб. на гостя**
To nkarpova@gsib.ru

Reply | Reply All | Forward | Redirect | Archive



Реклама: примеры

Методики борьбы

Принципы

Методики борьбы

Принципы

- **Политики** (особенно эффективны против Scam’a)
  - DNS списки (ip адреса, URL)
  - SPF, DKIM, DMARC
  - Репутация отправителя

Методики борьбы

Принципы

- **Политики** (особенно эффективны против Scam’a)
  - DNS списки (ip адреса, URL)
  - SPF, DKIM, DMARC
  - Репутация отправителя
- **Контент фильтрация** (поиск заданных паттернов в письмах)

Методики борьбы

Принципы

- **Политики** (особенно эффективны против Scam’a)
  - DNS списки (ip адреса, URL)
  - SPF, DKIM, DMARC
  - Репутация отправителя
- **Контент фильтрация** (поиск заданных паттернов в письмах)
- **Статистический анализ** (например, bayes)

Методики борьбы

Трудности

Методики борьбы

Трудности

- **Политики**:
  - Требуют сетевых запросов, которые могут долго выполняться
  - Ложные срабатывания

Методики борьбы

Трудности

- **Политики**:
  - Требуют сетевых запросов, которые могут долго выполняться
  - Ложные срабатывания
- **Контент фильтрация:**
  - Затратна в плане CPU
  - Быстро устаревает

Методики борьбы

Трудности

- **Политики**:
  - Требуют сетевых запросов, которые могут долго выполняться
  - Ложные срабатывания
- **Контент фильтрация:**
  - Затратна в плане CPU
  - Быстро устаревает
- **Статистический анализ:**
  - Неточен
  - Сложно сканировать в многопользовательских системах

Пример работы rspamd

Особенности rspamd

Особенности rspamd

- Нормализация текста (snowball stemmer)

Особенности rspamd

- Нормализация текста (snowball stemmer)
- Конвертирование всех писем в **utf-8**

Особенности rspamd

- Нормализация текста (snowball stemmer)
- Конвертирование всех писем в **utf-8**
- Поддержка настроек пользователей

Особенности rspamd

- Нормализация текста (snowball stemmer)
- Конвертирование всех писем в **utf-8**
- Поддержка настроек пользователей
- Возможность создания нескольких метрик для писем

Особенности rspamd

- Нормализация текста (snowball stemmer)
- Конвертирование всех писем в **utf-8**
- Поддержка настроек пользователей
- Возможность создания нескольких метрик для писем
- Web интерфейс для управления

Особенности rspamd

- Нормализация текста (snowball stemmer)
- Конвертирование всех писем в **utf-8**
- Поддержка настроек пользователей
- Возможность создания нескольких метрик для писем
- Web интерфейс для управления
- Совместимость с правилами SpamAssassin’a

Особенности rspamd

- Нормализация текста (snowball stemmer)
- Конвертирование всех писем в **utf-8**
- Поддержка настроек пользователей
- Возможность создания нескольких метрик для писем
- Web интерфейс для управления
- Совместимость с правилами SpamAssassin’a
- OpenSource (BSD 2-clause)

Методы фильтрации

Политики

Методы фильтрации

Политики

- Поддержка SPF, DKIM, DMARC

Методы фильтрации

Политики

- Поддержка SPF, DKIM, DMARC
- Плагин DNSBL - работа с DNS блок-листами

Методы фильтрации

Политики

- Поддержка SPF, DKIM, DMARC
- Плагин DNSBL - работа с DNS блок-листами
- Плагин SURBL - работа с DNS листами URL

Методы фильтрации

Политики

- Поддержка SPF, DKIM, DMARC
- Плагин DNSBL - работа с DNS блок-листами
- Плагин SURBL - работа с DNS листами URL
- Плагин репутации (IP, подсети, автономной сети и страны)

Методы фильтрации

Политики

- Поддержка SPF, DKIM, DMARC
- Плагин DNSBL - работа с DNS блок-листами
- Плагин SURBL - работа с DNS листами URL
- Плагин репутации (IP, подсети, автономной сети и страны)
- Универсальный плагин для обработки списков

Методы фильтрации

Анализ контента

Методы фильтрации

Анализ контента

- Регулярные выражения

Методы фильтрации

Анализ контента

- Регулярные выражения
- Быстрый поиск по шаблону (trie)

Методы фильтрации

Анализ контента

- Регулярные выражения
- Быстрый поиск по шаблону (trie)
- Совместимость с существующими правилами SpamAssassin

Методы фильтрации

Анализ контента

- Регулярные выражения
- Быстрый поиск по шаблону (trie)
- Совместимость с существующими правилами SpamAssassin
- Специализированные плагины (maillist, one received, chartable)

Методы фильтрации

Статистические методы

Методы фильтрации

Статистические методы

- Bayes фильтр с sqlite/mmap бэкэндами

Методы фильтрации

Статистические методы

- Bayes фильтр с sqlite/mmap бэкэндами
- Нечеткие хеши

Интеграция rspamd

milter

Интеграция rspamd

milter

- **rmilter** + postfix/sendmail:

Интеграция rspamd

milter

- **rmilter** + postfix/sendmail:
  - поддерживает все возможности rspamd (например, выборочный грейлистинг)

Интеграция rspamd

milter

- **rmilter** + postfix/sendmail:
  - поддерживает все возможности rspamd (например, выборочный грейлистинг)
  - умеет разделять входящие и исходящие потоки почты

Интеграция rspamd

milter

- **rmilter** + postfix/sendmail:
  - поддерживает все возможности rspamd (например, выборочный грейлистинг)
  - умеет разделять входящие и исходящие потоки почты
  - умеет хранить информацию об ответах на собственные письма

Интеграция rspamd

milter

- **rmilter** + postfix/sendmail:
  - поддерживает все возможности rspamd (например, выборочный грейлистинг)
  - умеет разделять входящие и исходящие потоки почты
  - умеет хранить информацию об ответах на собственные письма
  - дополнительные проверки: clamav, dcc

Интеграция rspamd

milter

- **rmilter** + postfix/sendmail:
  - поддерживает все возможности rspamd (например, выборочный грейлистинг)
  - умеет разделять входящие и исходящие потоки почты
  - умеет хранить информацию об ответах на собственные письма
  - дополнительные проверки: clamav, dcc
  - умеет подписывать исходящие письма (DKIM)

Интеграция rspamd

Другие способы

Интеграция rspamd

Другие способы

- **exim** - встроенная поддержка (с версии 4.86)

Интеграция rspamd

Другие способы

- **exim** - встроенная поддержка (с версии 4.86)
- **haraka** - встроенная поддержка

Интеграция rspamd

Другие способы

- **exim** - встроенная поддержка (с версии 4.86)
- **haraka** - встроенная поддержка
- любые MTA - **LDA** режим, когда клиент rspamd проверяет письмо и вызывает через pipe LDA для доставки письма (не работает грейлистинг)

Часть II: Архитектура

Rspamd в двух словах

Rspamd в двух словах

- Написан на **plain C**

Rspamd в двух словах

- Написан на **plain C**
- Использует **неблокирующую** (событийную) модель обработки данных

Rspamd в двух словах

- Написан на **plain C**
- Использует **неблокирующую** (событийную) модель обработки данных
- Плагины и правила написанны на языке **Lua**

Rspamd в двух словах

Приоритеты разработки

Rspamd в двух словах

Приоритеты разработки

- Производительность и масштабируемость

Rspamd в двух словах

Приоритеты разработки

- Производительность и масштабируемость
- Качество фильтрации и расширяемость

Rspamd в двух словах

Приоритеты разработки

- Производительность и масштабируемость
- Качество фильтрации и расширяемость
- Портабельность и интеграция с MTA

Rspamd в двух словах

Приоритеты разработки

- Производительность и масштабируемость
- Качество фильтрации и расширяемость
- Портабельность и интеграция с MTA
- Безопасность

Rspamd в двух словах

Приоритеты разработки

- Производительность и масштабируемость
- Качество фильтрации и расширяемость
- Портабельность и интеграция с MTA
- Безопасность
- Простота, работа “из коробки” и документация

Событийная модель

Никогда не блокируется*

*почти никогда

Событийная модель

Никогда не блокируется*

- Плюсы:

  ✅ Может обрабатывать сетевые запросы независимо

  ✅ Посылает все сетевые запросы одновременно

  ✅ Обрабатывает сразу много сообщений и хорошо сканируется

*почти никогда

Событийная модель

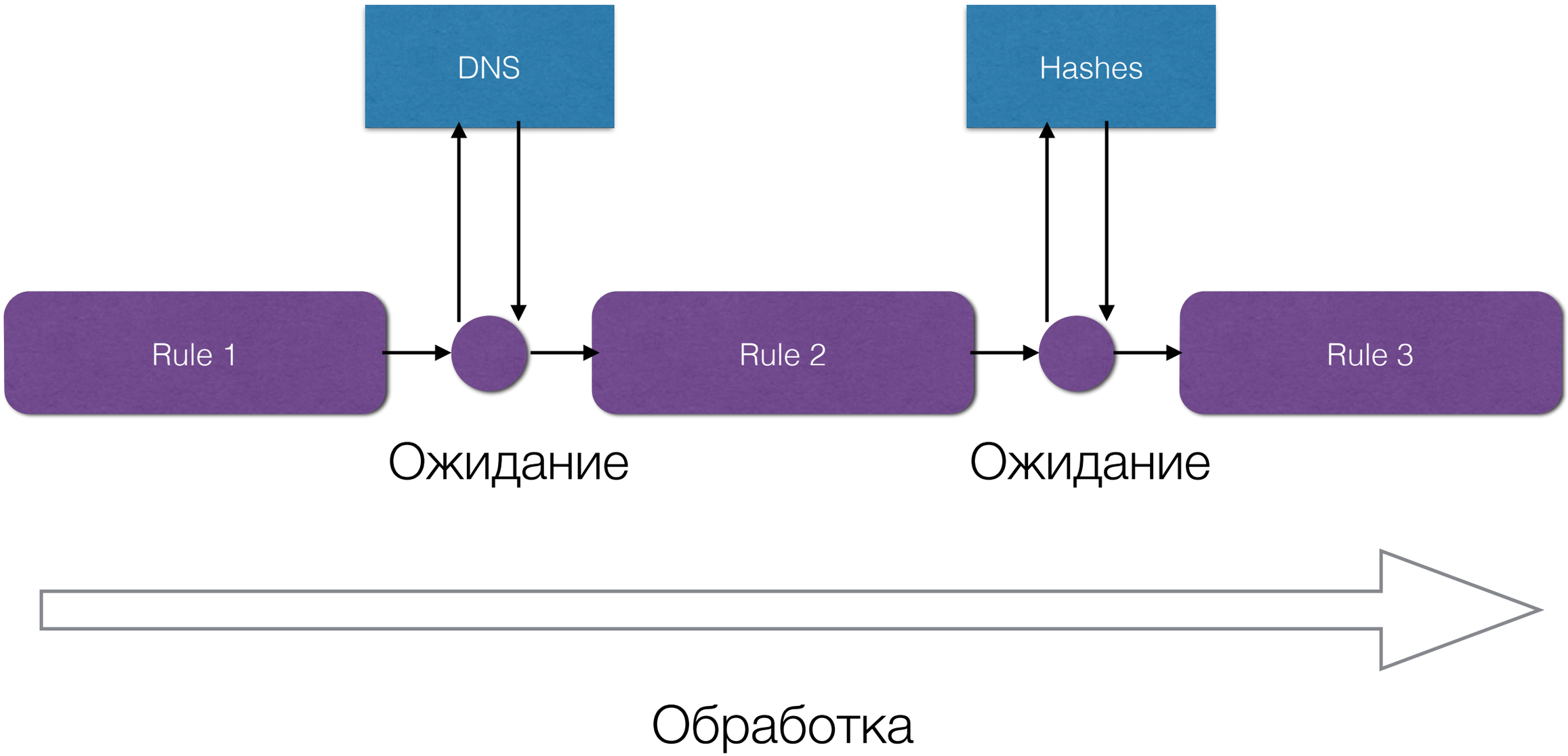
Никогда не блокируется*

- Плюсы:
  - ✅ Может обрабатывать сетевые запросы независимо
  - ✅ Посылает все сетевые запросы одновременно
  - ✅ Обрабатывает сразу много сообщений и хорошо сканируется
- Cons:
  - 🔥 Очень сложная работа с callback’ами
  - ⛔ Сложно ограничить потребление памяти

*почти никогда

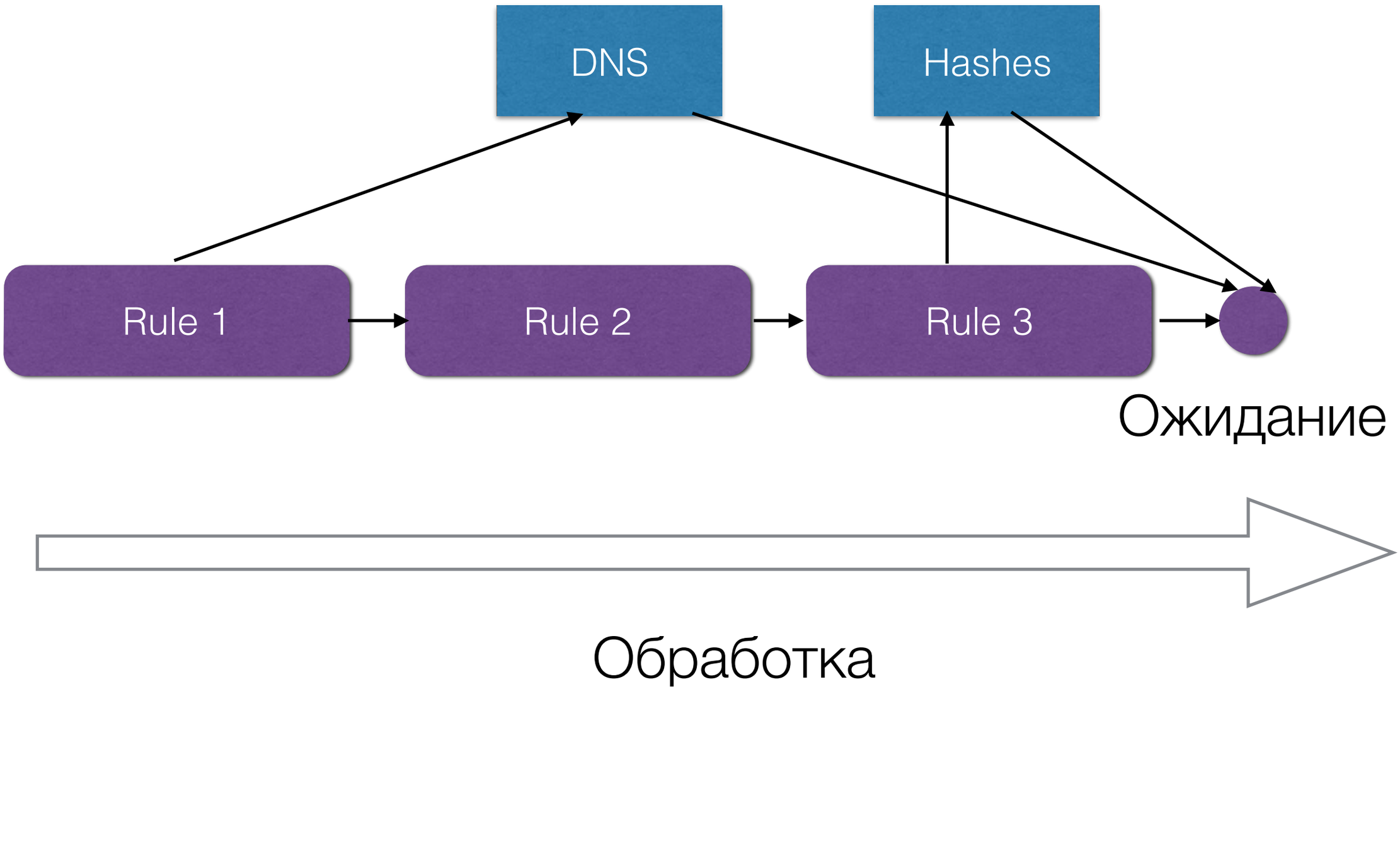
Последовательная обработка

Традиционный подход



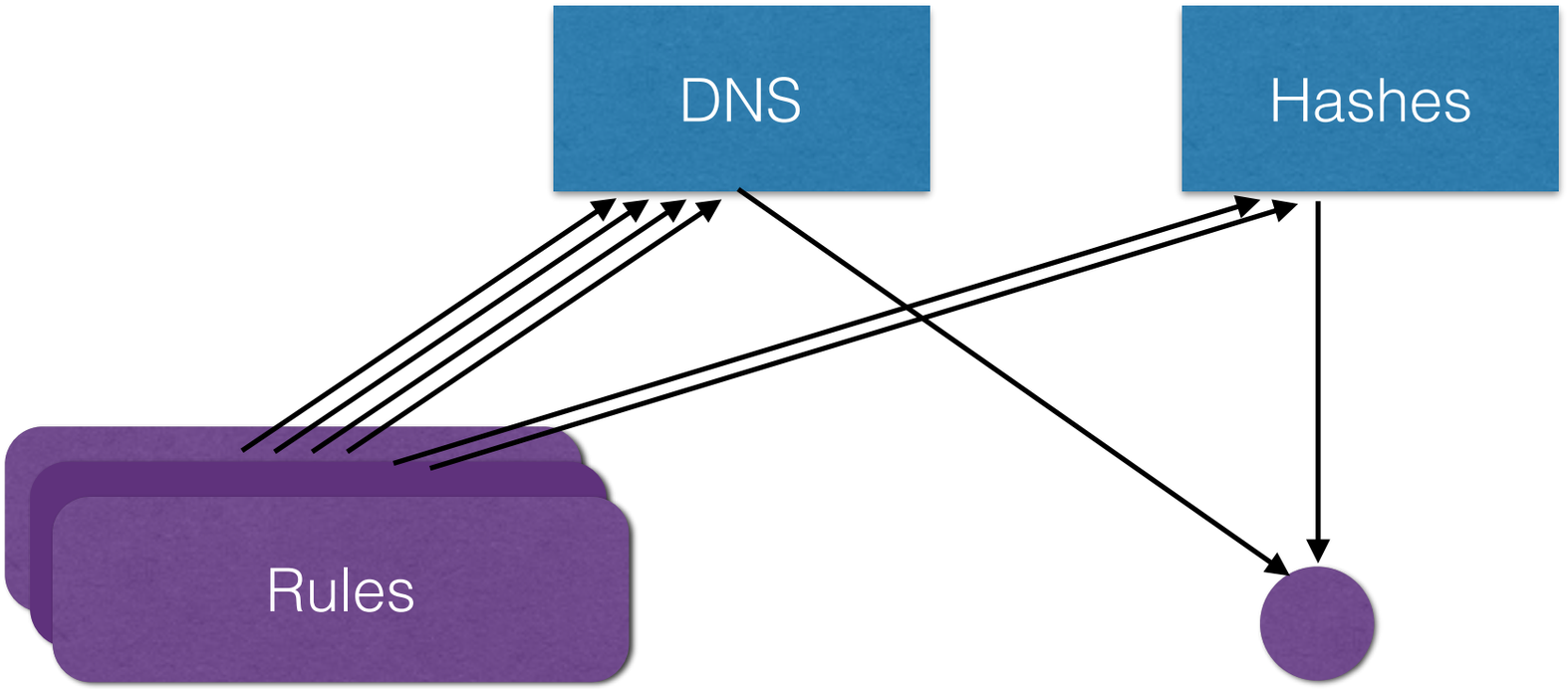
Событийная модель

Подход rspamd



Событийная модель

Что происходит на самом деле



Ожидание

Событийная модель

Несколько наблюдений

- Rspamd может генерировать множество параллельных сетевых запросов:

time: 5540.8ms real, 2427.4ms virtual, dns req: 120543

```
 =E6=E5=EB=F2=EE=E2=E0=F2=FB=EC =C2 =E2=EE=E4=F3. =F5=EE=EB=EE=E4=ED=EE =CF=^M
=E0=E2=EB=EE=E2=ED=E0 =E2=E0=F1. =F3=E6=E5 =F7=F2=EE=E1 =C2=E0=EC =F1=EB=EE=^M
=E2=EE=EC, =F2=E8=F5=E8=EC=E8, =E2=E8=E4 ] =ED</title>=09^M
</head>      =09^M
<body>^M
<font lang=3D"ru"></font>^M
<p>=CF=F0=E8=E2=E5=F2<br />     =09^M
</p>^M
<title>^M
<p>^M
http://www.whojgvj.com<br />http://www.pywyquqpxh.com<br />http://www.nce=^M
euvhmgz.com<br />http://www.jxyxbkfeeh.com<br />http://www.pniwriksw.com<=^M
br />http://www.gvlvqiobom.com<br />http://www.bcyvdnjao.com<br />http://=^M
www.mcbcbbgygx.com<br />http://www.cdbgjfbya.com<br />http://www.qgcclclo=^M
mo.com<br />http://www.xtyrean.com<br />http://www.vtwuvr.com<br />http:/=^M
/www.qevptvaz.com<br />http://www.isgizbyvuv.com<br />http://www.otomeg.c=^M
om<br />http://www.zkvcqgty.com<br />http://www.alcjvwfsty.com<br />http:=^M
//www.cfbsydfo.com<br />http://www.nysiolowh.com<br />http://www.pdxvurkr=^M
c.com<br />http://www.udbykqwa.com<br />http://www.spoarodgf.com<br />htt=^M
p://www.ibrbxnsyk.com<br />http://www.aaqylafn.com<br />http://www.vmtddz=^M
fx.com<br />http://www.naydsziqg.com<br />http://www.reabvivech.com<br />=^M
http://www.ukgvjf.com<br />http://www.ulltntrh.com<br />http://www.skspya=^M
zi.com<br />http://www.mvvxnxs.com<br />http://www.ovxukyassi.com<br />ht=^M
tp://www.vsqmdqvtsw.com<br />http://www.cexzgzki.com<br />http://www.ivhz=^M
gxgw.com<br />http://www.rasdszmis.com<br />http://www.xaaejtow.com<br />=^M
http://www.stjnfeor.com<br />http://www.rhpxtwlymp.com<br />http://www.ok=^M
msudrfl.com<br />http://www.adyuawo.com<br />http://www.mbpmqrjxo.com<br =^M
/>http://www.afwchnjbt.com<br />http://www.xymsrgvb.com<br />http://www.y=^M
ujeoqvtsw.com<br />http://www.gaogtegv.com<br />http://www.kslkyurnbc.com=^M
<br />http://www.lxjgqzv.com<br />http://www.xjattr.com<br />http://www.g=^M
iehcgm.com<br />http://www.squasyq.com<br />http://www.oalbkbofcf.com<br =^M
~/Downloads/longtime.eml
```

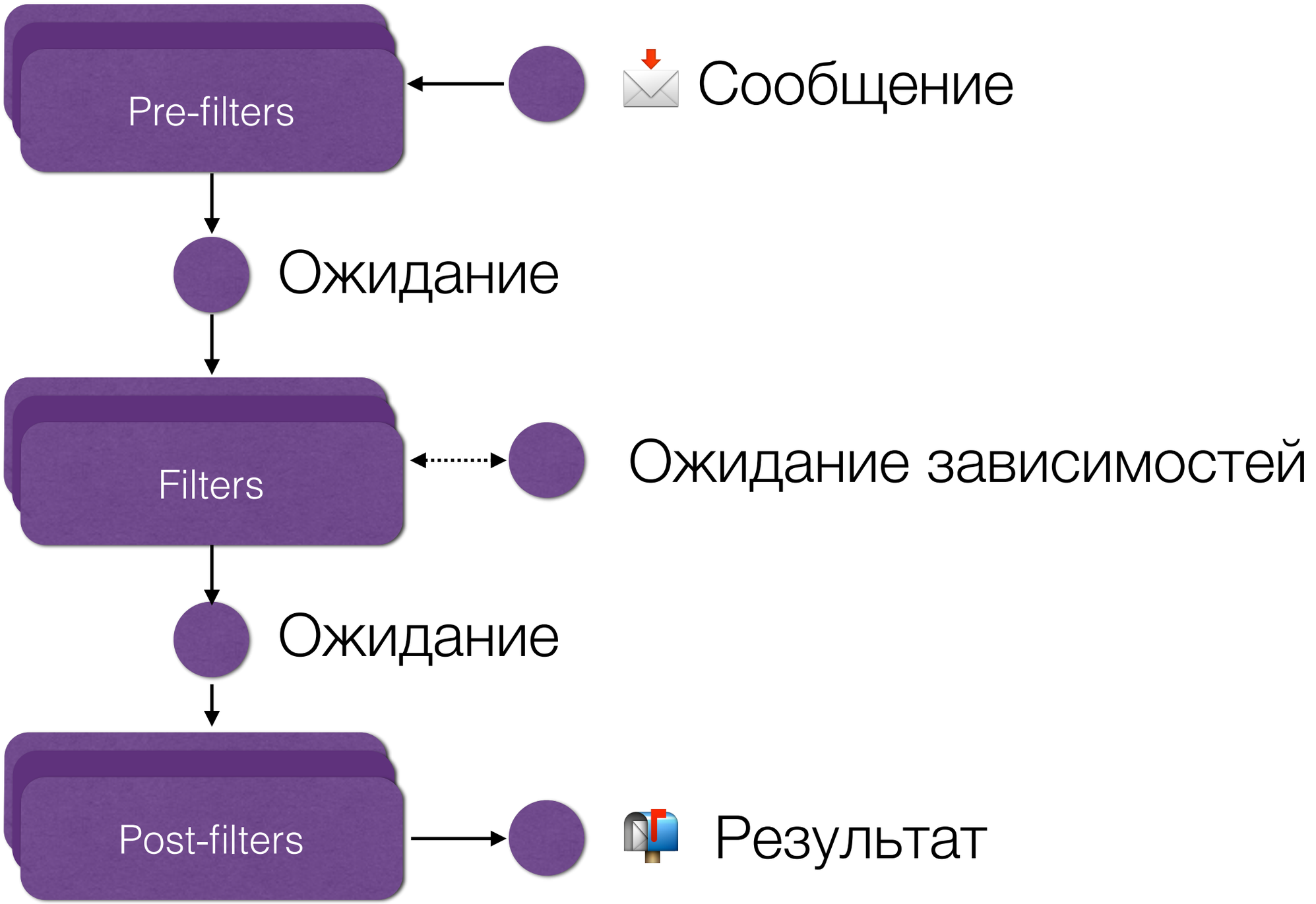
Событийная модель

Несколько наблюдений

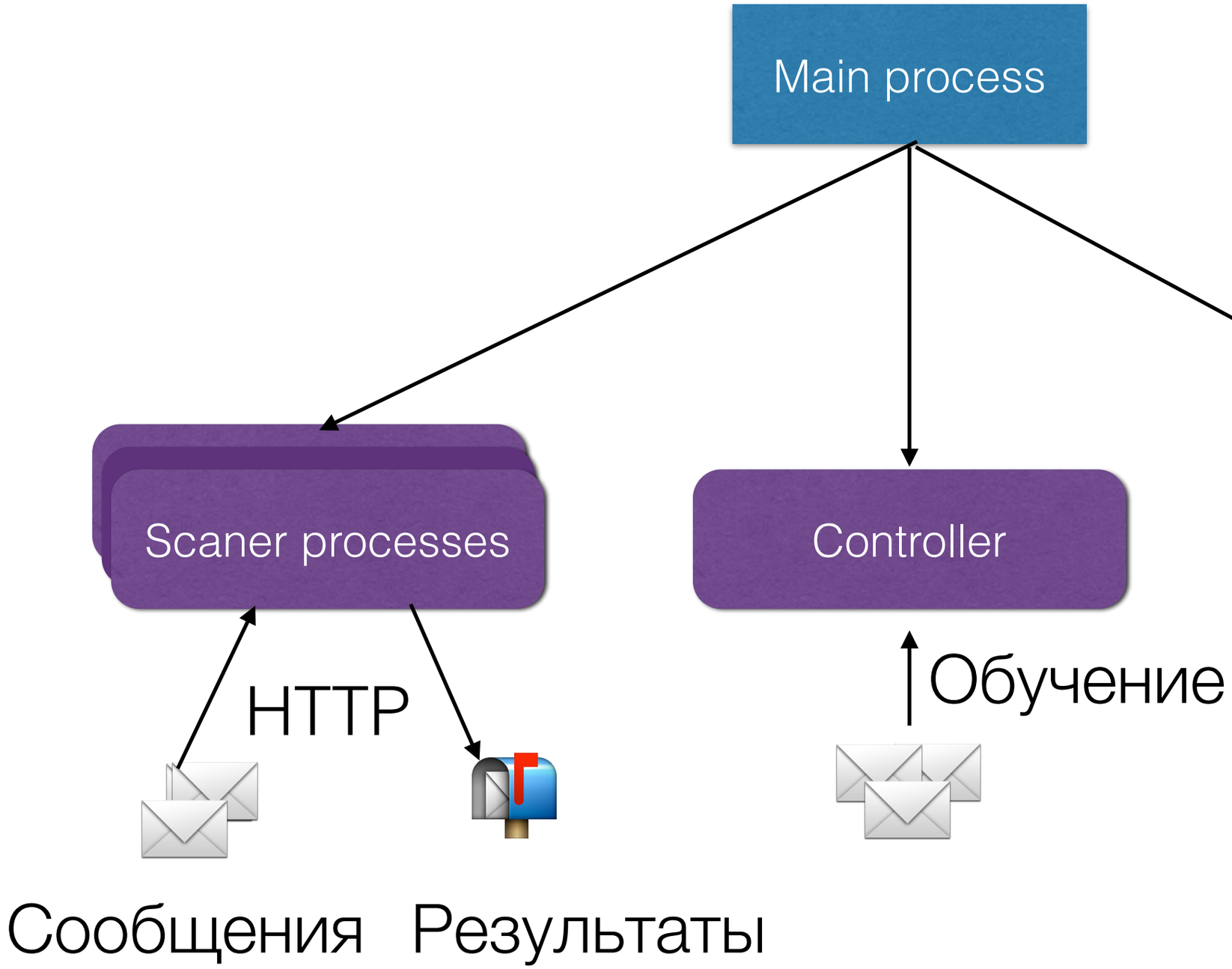
- Rspamd может генерировать множество параллельных сетевых запросов:
  `time: 5540.8ms real, 2427.4ms virtual, dns req: 120543`

- Для коротких сообщений (которые составляют 99% от всех сообщений) ожидание занимает в сотни раз больше времени, чем обработка:
  `time: 996.140ms real, 22.000ms virtual,`

Обработка сообщения в rspamd



Процессы rspamd



Статистика в rspamd

Статистика в rspamd

- Использует цепи Маркова из статистических токенов

Статистика в rspamd

- Использует цепи Маркова из статистических токенов
- Использует метаданные в письме (mime структуру, вложения, заголовки)

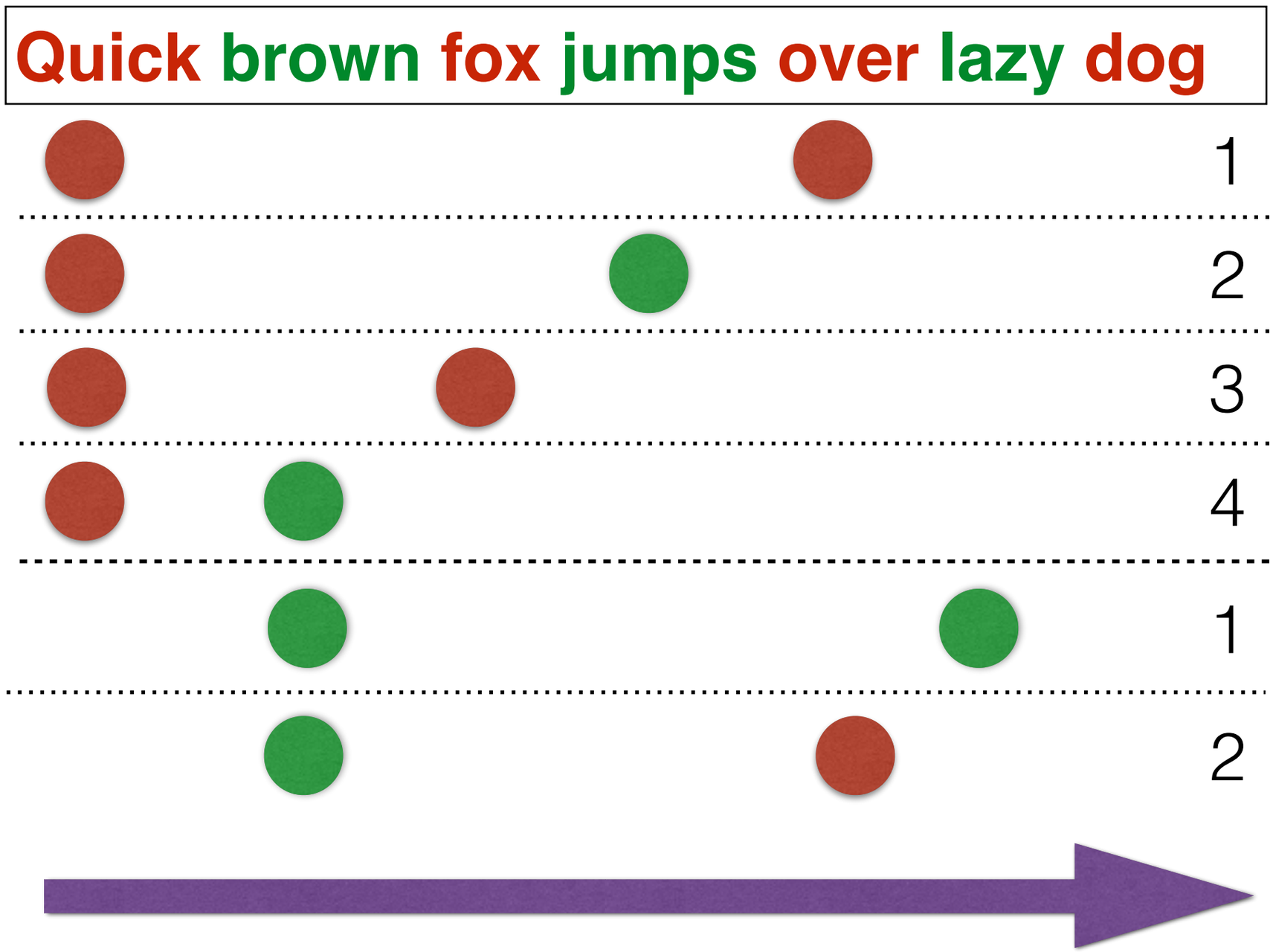
Статистика в rspamd

- Использует цепи Маркова из статистических токенов
- Использует метаданные в письме (mime структуру, вложения, заголовки)
- Статистические данные хранятся в sqlite3

Статистика в rspamd

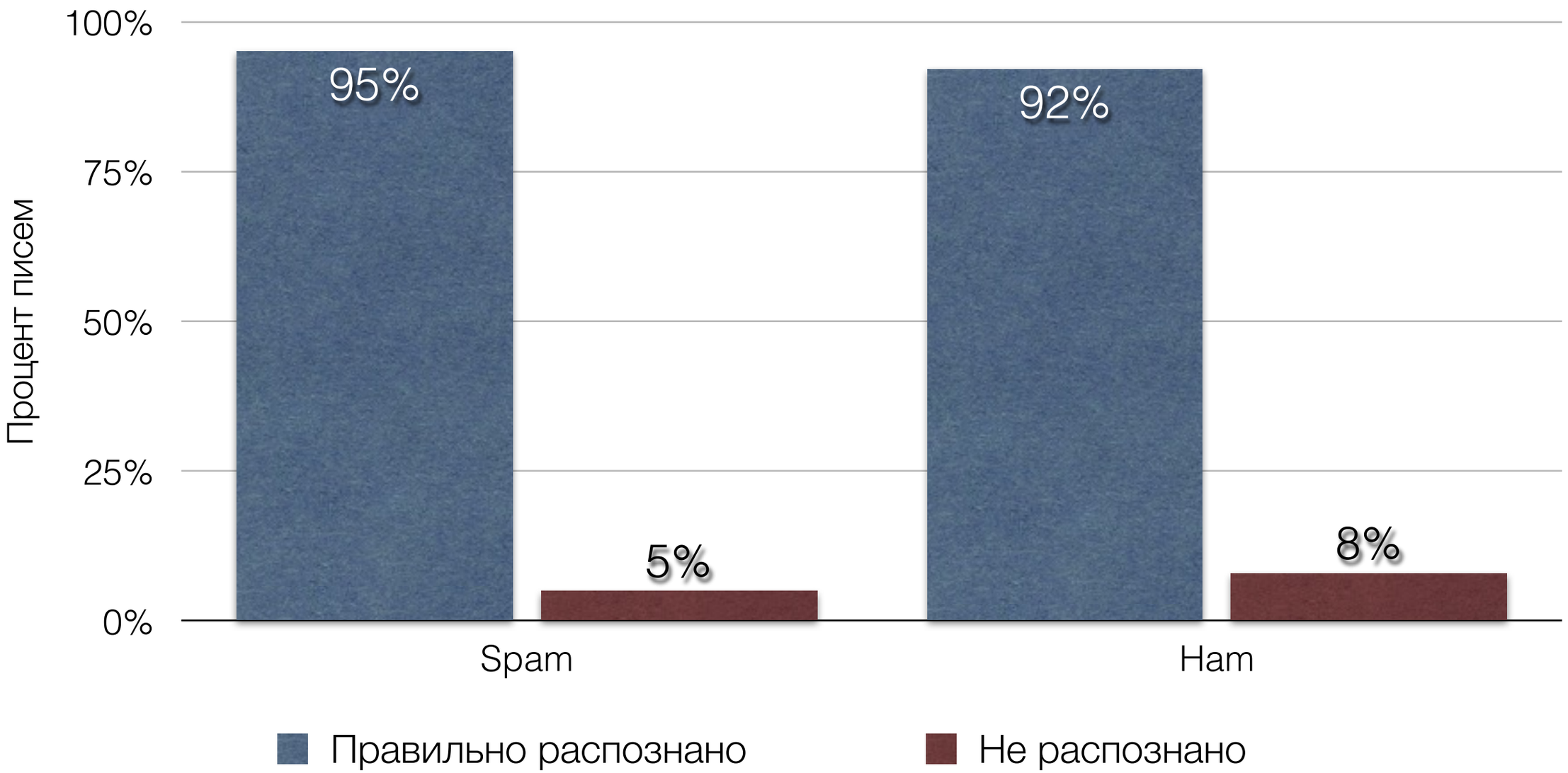
- Использует цепи Маркова из статистических токенов
- Использует метаданные в письме (mime структуру, вложения, заголовки)
- Статистические данные хранятся в sqlite3
- Синхронизация: обучение всех серверов вместе (есть защита от повторного обучения)

Структура токенов



Пример работы статистики

Сложный случай - спам в картинках



Нечеткие хеши

Общие данные

Нечеткие хеши

Общие данные

- Используются для поиска **похожих** писем

Нечеткие хеши

Общие данные

- Используются для поиска **похожих** писем
- Сочетают обычные хеши для вложений и алгоритм шинглов для текста

Нечеткие хеши

Общие данные

- Используются для поиска **похожих** писем
- Сочетают обычные хеши для вложений и алгоритм шинглов для текста
- Хранилище хешей - sqlite3

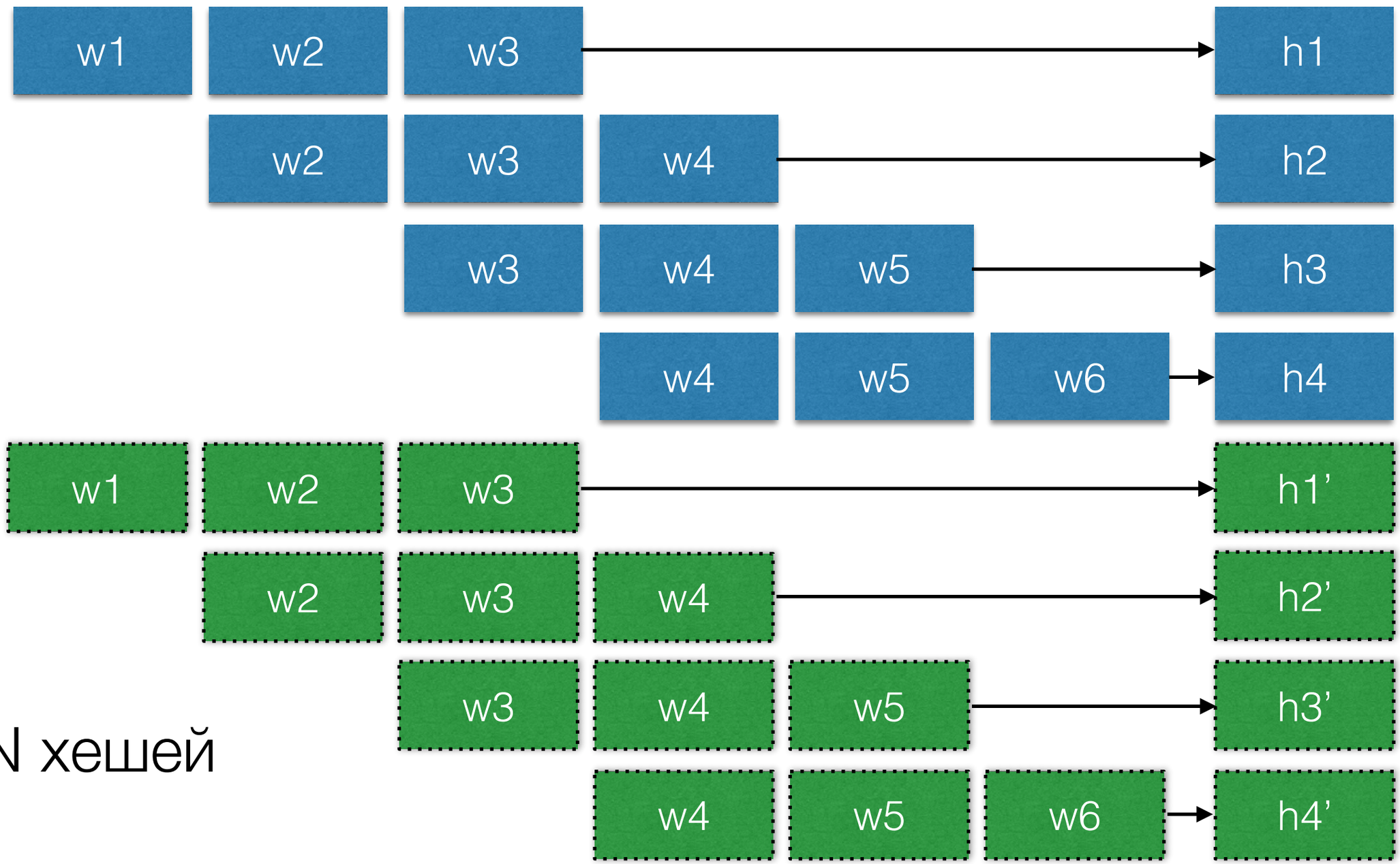
Нечеткие хеши

Общие данные

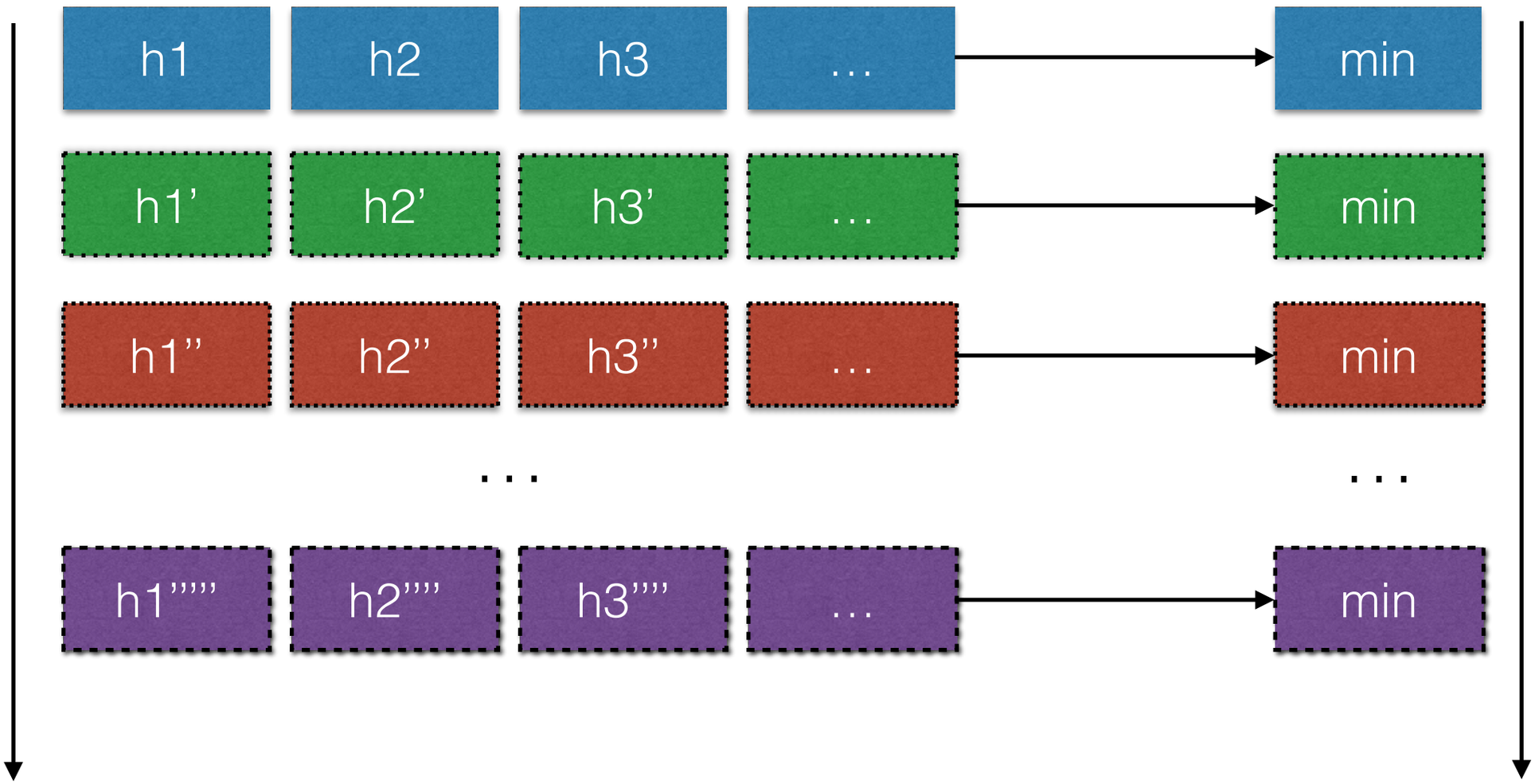
- Используются для поиска **похожих** писем
- Сочетают обычные хеши для вложений и алгоритм шинглов для текста
- Хранилище хешей - sqlite3
- Синхронизация: обучение всех серверов вместе (есть защита от повторного обучения)

Алгоритм шинглов

Quick brown fox jumps over lazy dog



Алгоритм шинглов



N наборов хешей

N шинглов

Алгоритм шинглов

Алгоритм шинглов

- Однозначный вероятностный алгоритм

Алгоритм шинглов

- Однозначный вероятностный алгоритм
- В качестве хеш-функции - **siphash** с разными ключами

Алгоритм шинглов

- Однозначный вероятностный алгоритм
- В качестве хеш-функции - **siphash** с разными ключами
- Ключи генерируются при помощи криптографической хеш-функции

Алгоритм шинглов

- Однозначный вероятностный алгоритм
- В качестве хеш-функции - **siphash** с разными ключами
- Ключи генерируются при помощи криптографической хеш-функции
- Размер окна - 3 слова, число хешей - 32 (больше - медленно)

Часть III:
Производительность

Общие принципы

Общие принципы

- Думать о производительности. **Всегда.**

Общие принципы

- Думать о производительности. **Всегда.**
- Если что-то можно не делать, то **не делать** это

Общие принципы

- Думать о производительности. **Всегда.**
- Если что-то можно не делать, то **не делать** это
- Если есть более оптимальное **частное** решение задачи, то использовать его (ассемблерные вставки, конечные автоматы)

Общие принципы

- Думать о производительности. **Всегда.**
- Если что-то можно не делать, то **не делать** это
- Если есть более оптимальное **частное** решение задачи, то использовать его (ассемблерные вставки, конечные автоматы)
- Если можно сделать **примерно**, но быстро, или очень точно, но долго, то делать быстро

Оптимизация правил

Глобальная оптимизация

Оптимизация правил

Глобальная оптимизация

- **Останавливать** проверки, если письмо уже набрало порог очков

Оптимизация правил

Глобальная оптимизация

- **Останавливать** проверки, если письмо уже набрало порог очков
- Обрабатывать правила с **отрицательным** весом в первую очередь (для избежания ошибок второго рода)

Оптимизация правил

Глобальная оптимизация

- **Останавливать** проверки, если письмо уже набрало порог очков

- Обрабатывать правила с **отрицательным** весом в первую очередь (для избежания ошибок второго рода)

- Выполнять правила, которые можно выполнить **быстрее**, в первую очередь:
  - Учитывается среднее время выполнения, вес и частота срабатывания
  - Жадный алгоритм сортировки

Оптимизация правил

Локальные оптимизации

Оптимизация правил

Локальные оптимизации

- Каждое правило оптимизируется при помощи абстрактного синтаксического дерева (**AST**): дало прирост в 3-4 раза для больших писем

Оптимизация правил

Локальные оптимизации

- Каждое правило оптимизируется при помощи абстрактного синтаксического дерева (**AST**): дало прирост в 3-4 раза для больших писем
- Элементы в дереве упорядочиваются по аналогичному “жадному” алгоритму

Оптимизация правил

Локальные оптимизации

- Каждое правило оптимизируется при помощи абстрактного синтаксического дерева (**AST**): дало прирост в 3-4 раза для больших писем

- Элементы в дереве упорядочиваются по аналогичному “жадному” алгоритму

- Регулярные выражения компилируются при помощи **PCRE JIT** (+100-150% скорости)

Оптимизация правил

Локальные оптимизации

- Каждое правило оптимизируется при помощи абстрактного синтаксического дерева (**AST**): дало прирост в 3-4 раза для больших писем

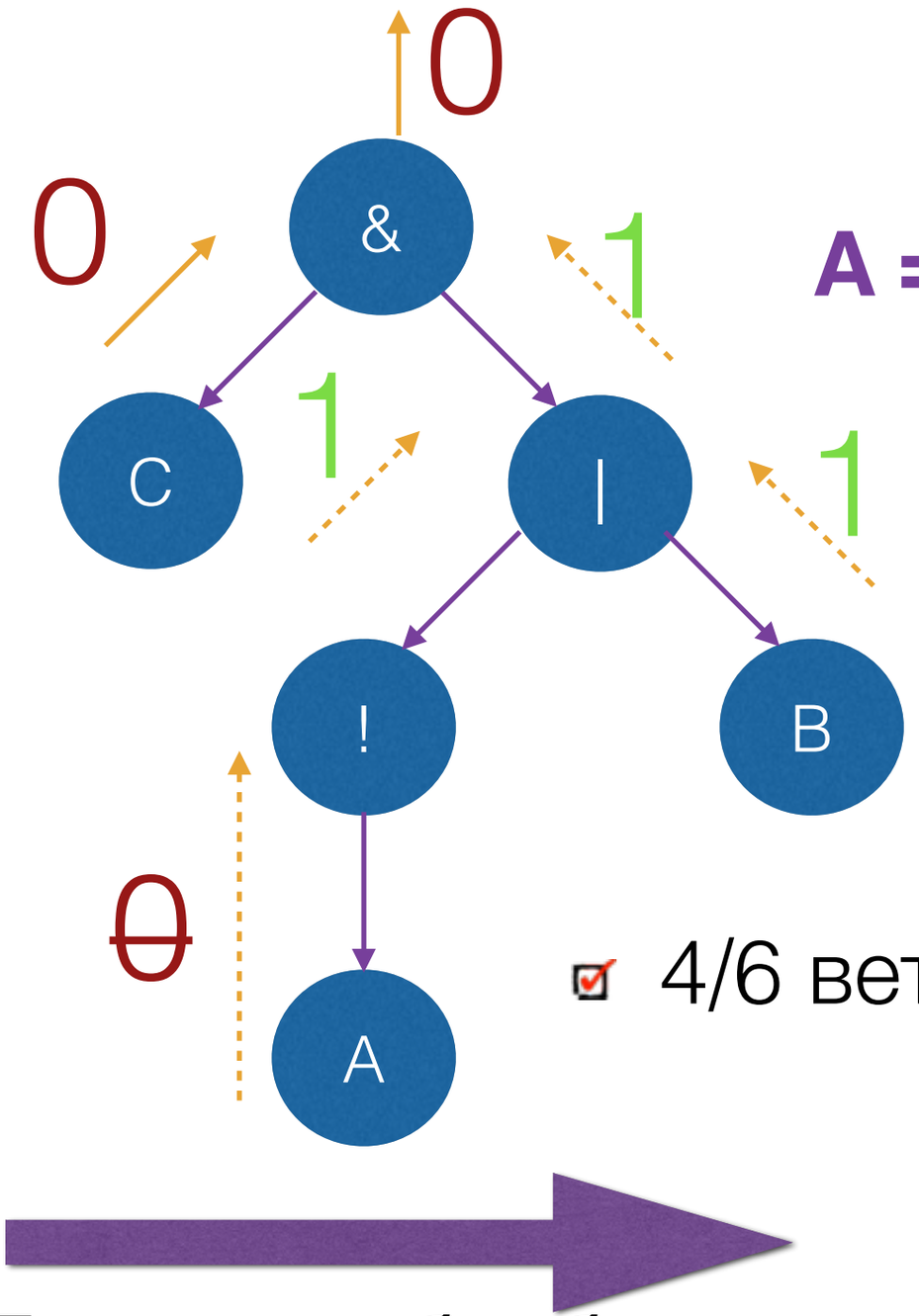
- Элементы в дереве упорядочиваются по аналогичному “жадному” алгоритму

- Регулярные выражения компилируются при помощи **PCRE JIT** (+100-150% скорости)

- Код на Lua компилируется **LuaJIT**

Оптимизации AST

Исключение лишних ветвей



A = 0, B = 1, C = 0

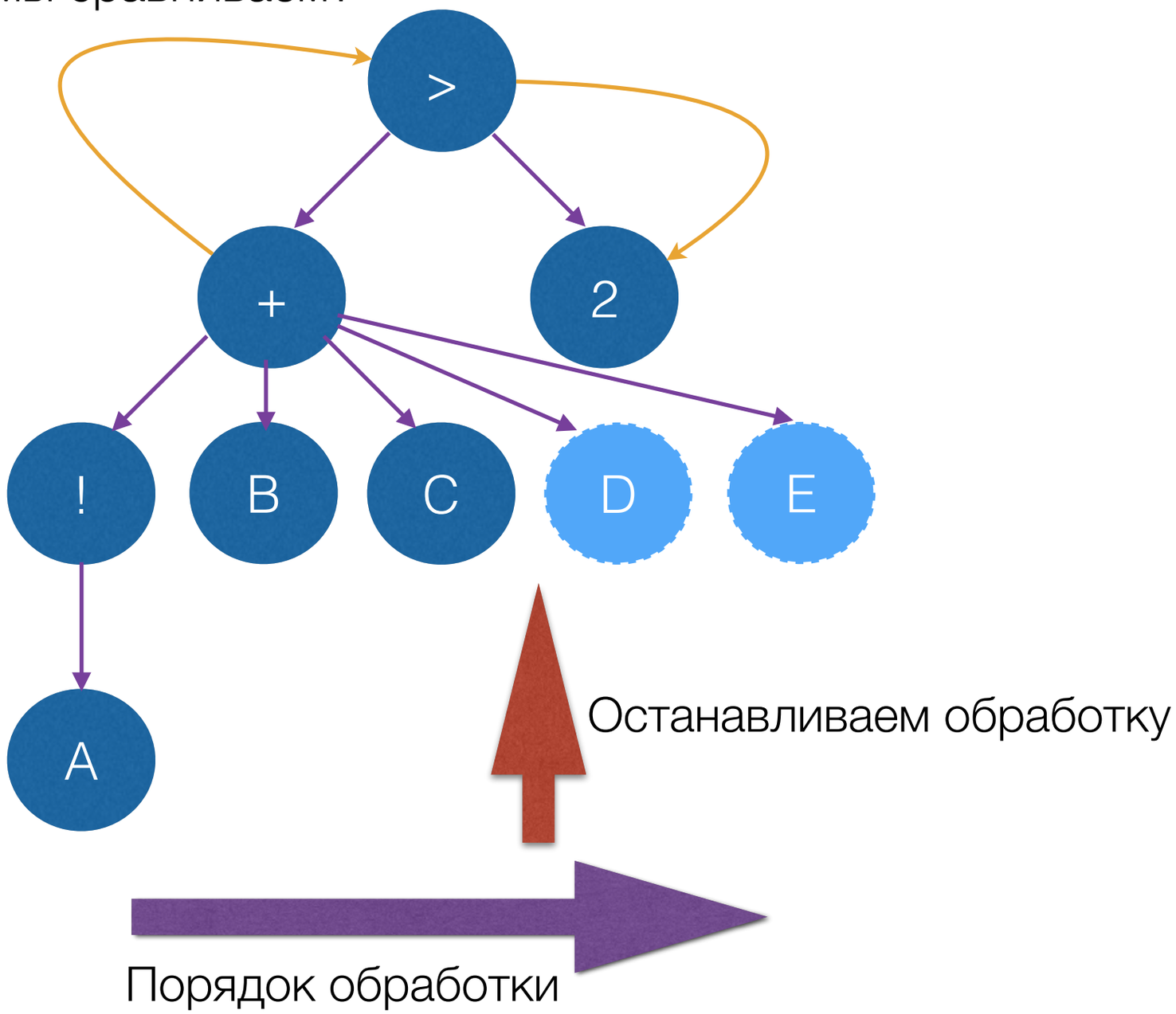
- ☑ 4/6 ветвей пропущено

Порядок обработки

Оптимизации AST

N-арные операции

Что мы сравниваем?



Оптимизации библиотеки C

Оптимизации библиотеки C

- Собственное API для ведения логов:
  - умеет работать с syslog/file/console
  - фильтрует повторяющиеся сообщения и сообщения с не-ASCII символами

Оптимизации библиотеки C

- Собственное API для ведения логов:
  - умеет работать с syslog/file/console
  - фильтрует повторяющиеся сообщения и сообщения с не-ASCII символами
- Собственная реализация **printf**:
  - работает со структурами rspamd (строки фиксированной длины, int*_t)
  - не зависит от локалей
  - не пытается вычислять требуемую длину буфера

Оптимизации библиотеки C

Строковые функции

Оптимизации библиотеки C

Строковые функции

- Оптимизированные функции работы с **base32/base64**:
  - работа с выровненными данными;
  - оптимизация циклов для 64-х битных платформ

Оптимизации библиотеки C

Строковые функции

- Оптимизированные функции работы с **base32/base64**:
  - работа с выровненными данными;
  - оптимизация циклов для 64-х битных платформ
- Реализация приведения текста к нижнему регистру (ASCII и UTF8 версия)

Оптимизации библиотеки C

Строковые функции

- Оптимизированные функции работы с **base32/base64**:
  - работа с выровненными данными;
  - оптимизация циклов для 64-х битных платформ
- Реализация приведения текста к нижнему регистру (ASCII и UTF8 версия)
- Использование строк фиксированной длины

Оптимизации библиотеки C

Оптимизации библиотеки C

- Быстрые **хеш** функции (*xxhash* и *blake2*)

Оптимизации библиотеки C

- Быстрые **хеш** функции (*xxhash* и *blake2*)
- Быстрое **шифрование** (ассемблерные реализации)

Оптимизации библиотеки C

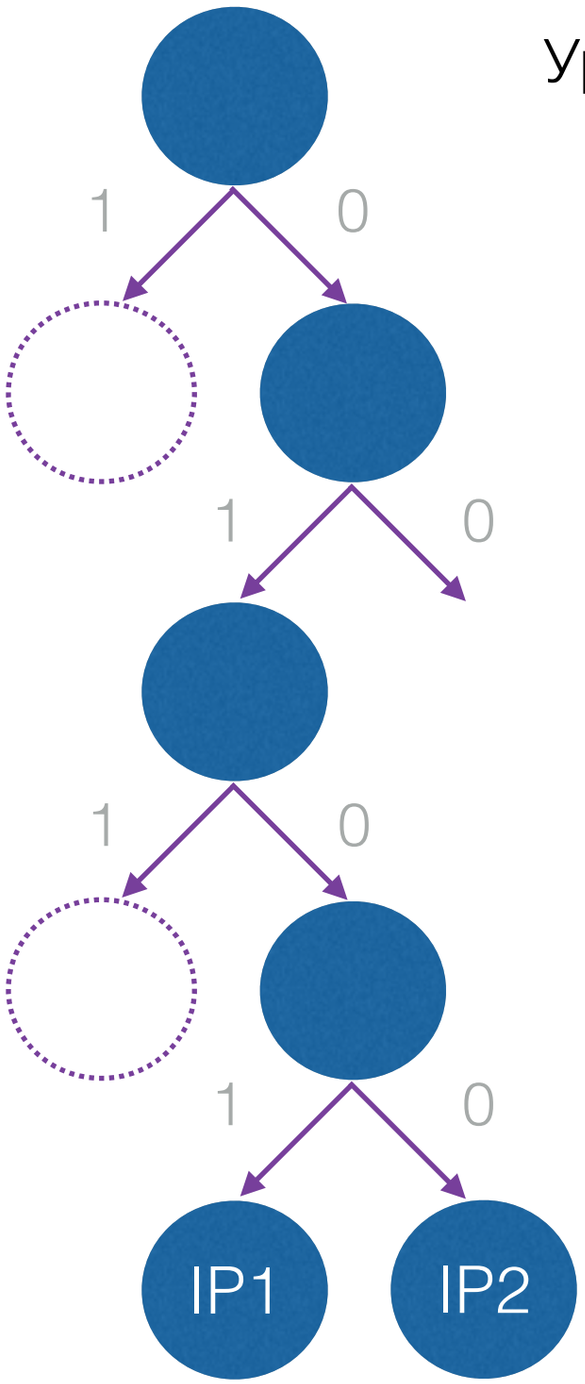
- Быстрые **хеш** функции (*xxhash* и *blake2*)
- Быстрое **шифрование** (ассемблерные реализации)
- **Выравнивание** данных

Оптимизации библиотеки C

- Быстрые **хеш** функции (*xxhash* и *blake2*)
- Быстрое **шифрование** (ассемблерные реализации)
- **Выравнивание** данных
- Поддержка инструментов анализа производительности и возможность оценки скорости работы каждого правила

Хранение IP адресов

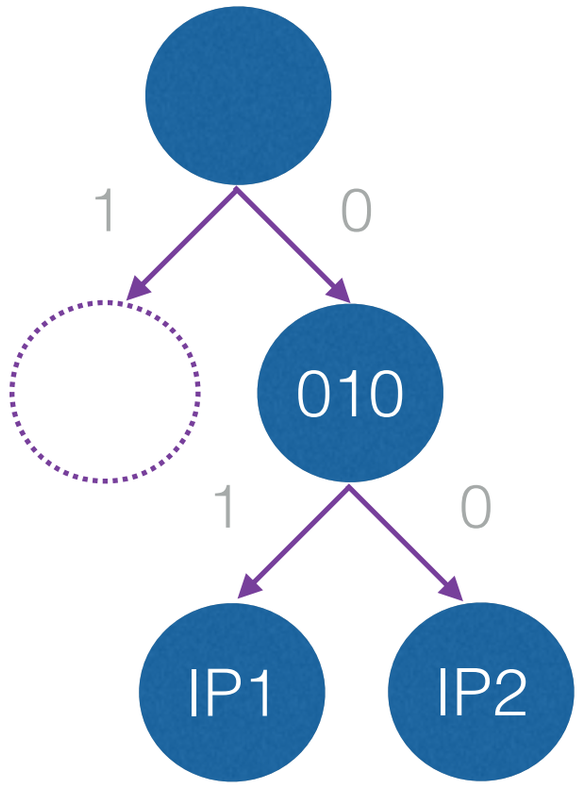
Несжатое radix дерево



Уровень на каждый бит: **32** уровня для IPv4
128 уровней для IPv6

Хранение IP адресов

Компрессированное radix дерево



Часть IV: Безопасность

Основные принципы

Основные принципы

- Писать безопасный код на plain C **трудно**

Основные принципы

- Писать безопасный код на plain C **трудно**
- Основные угрозы:
  - Взаимодействие с **DNS**
  - Прослушивание трафика (шифровать надо все данные, передаваемые по сети)
  - Вредоносные сообщения

Примеры проблем

Примеры проблем

- В письме может содержаться очень много данных для проверки в DNS (например, сотни тысяч URL)

Примеры проблем

- В письме может содержаться очень много данных для проверки в DNS (например, сотни тысяч URL)
- SPF записи могут иметь бесконечную рекурсию

Примеры проблем

- В письме может содержаться очень много данных для проверки в DNS (например, сотни тысяч URL)
- SPF записи могут иметь бесконечную рекурсию
- Нужны лимиты на число запросов для одного письма

Защита DNS

Защита DNS

- Криптографический генератор DNS id

Защита DNS

- Криптографический генератор DNS id
- Пул сокетов с постоянной ротацией

Защита DNS

- Криптографический генератор DNS id
- Пул сокетов с постоянной ротацией
- Рандомизация портов для запросов

Защита DNS

- Криптографический генератор DNS id
- Пул сокетов с постоянной ротацией
- Рандомизация портов для запросов
- Фильтрация входных данных (+ IDN кодирование)

Шифрование трафика

Шифрование трафика

- Безопасность -> скорость -> простота

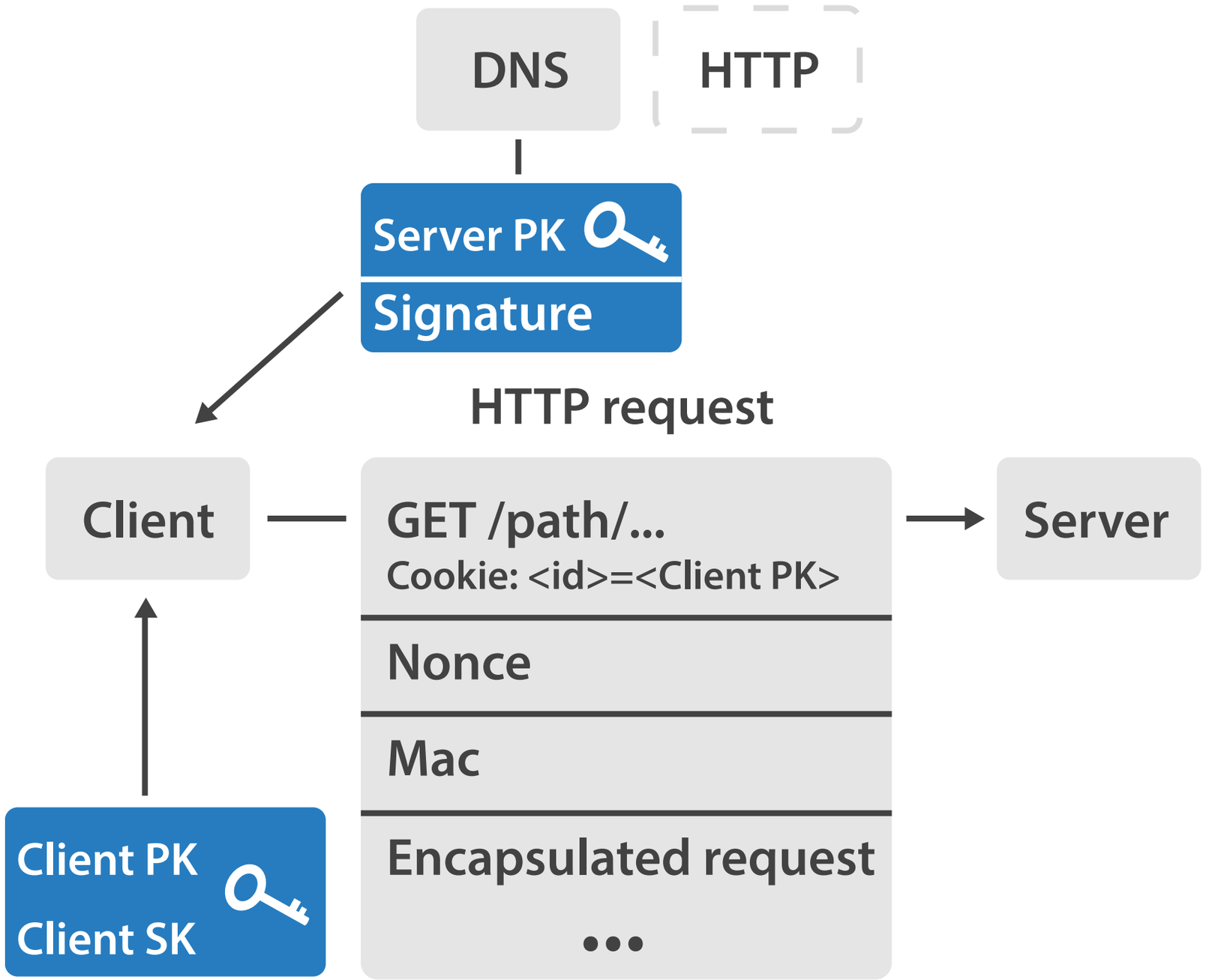
Шифрование трафика

- Безопасность -> скорость -> простота
- TLS сложно интегрировать в non-blocking IO

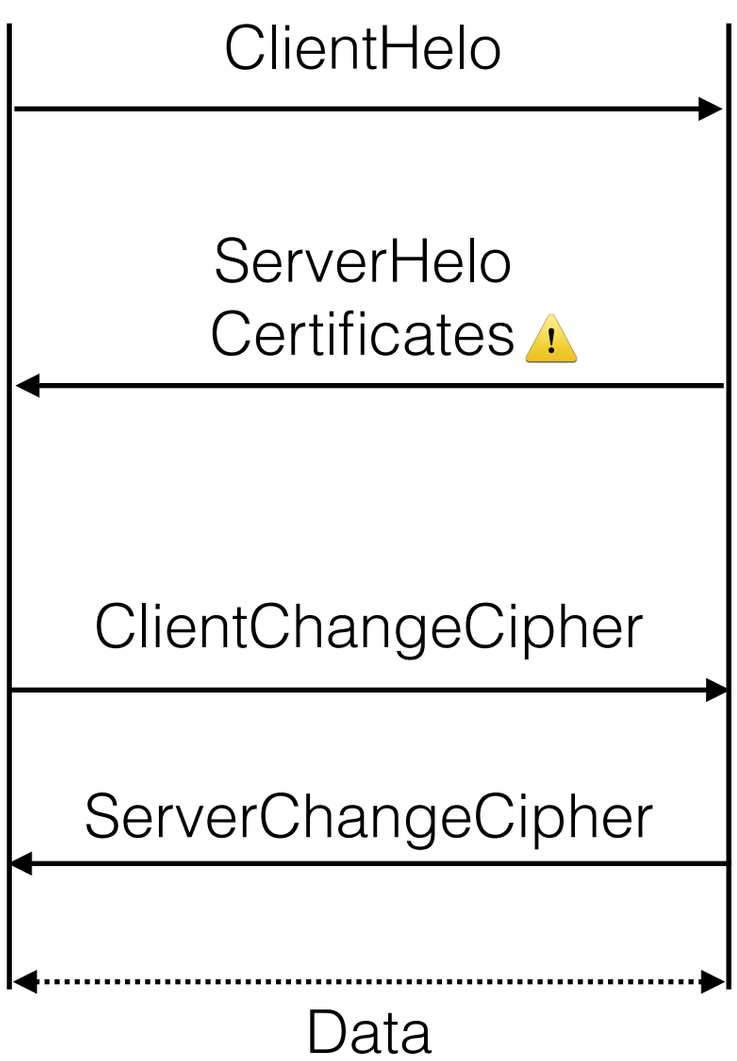
Шифрование трафика

- Безопасность -> скорость -> простота
- TLS сложно интегрировать в non-blocking IO
- TLS делает много ненужного, увеличивая CPU load и latency

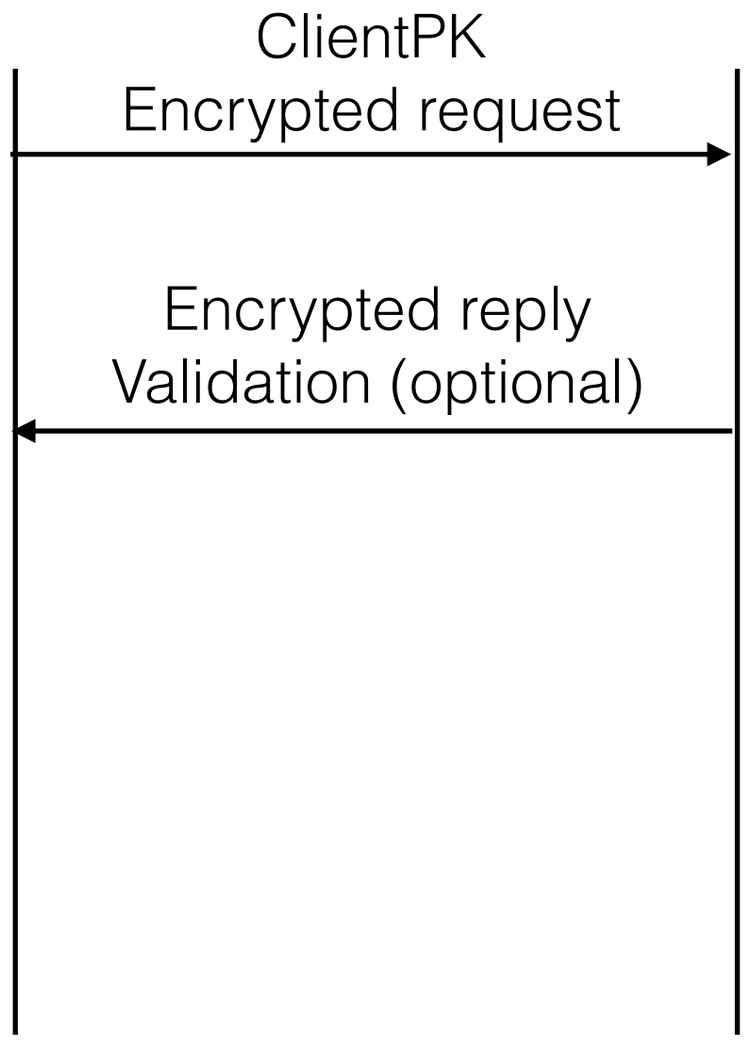
Протокол HTTPCrypt



Установление соединения



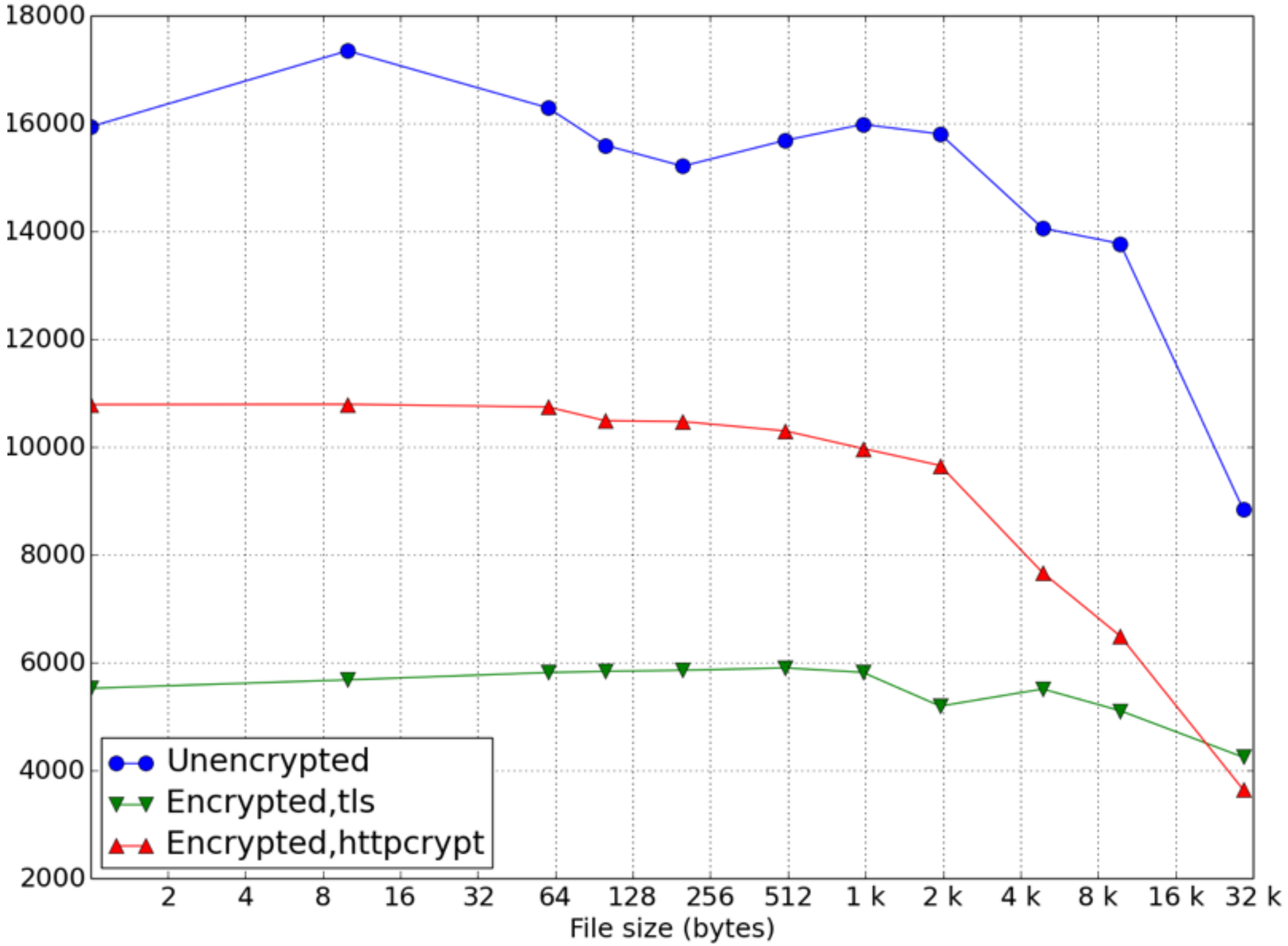
TLS handshake



HTTPCrypt handshake

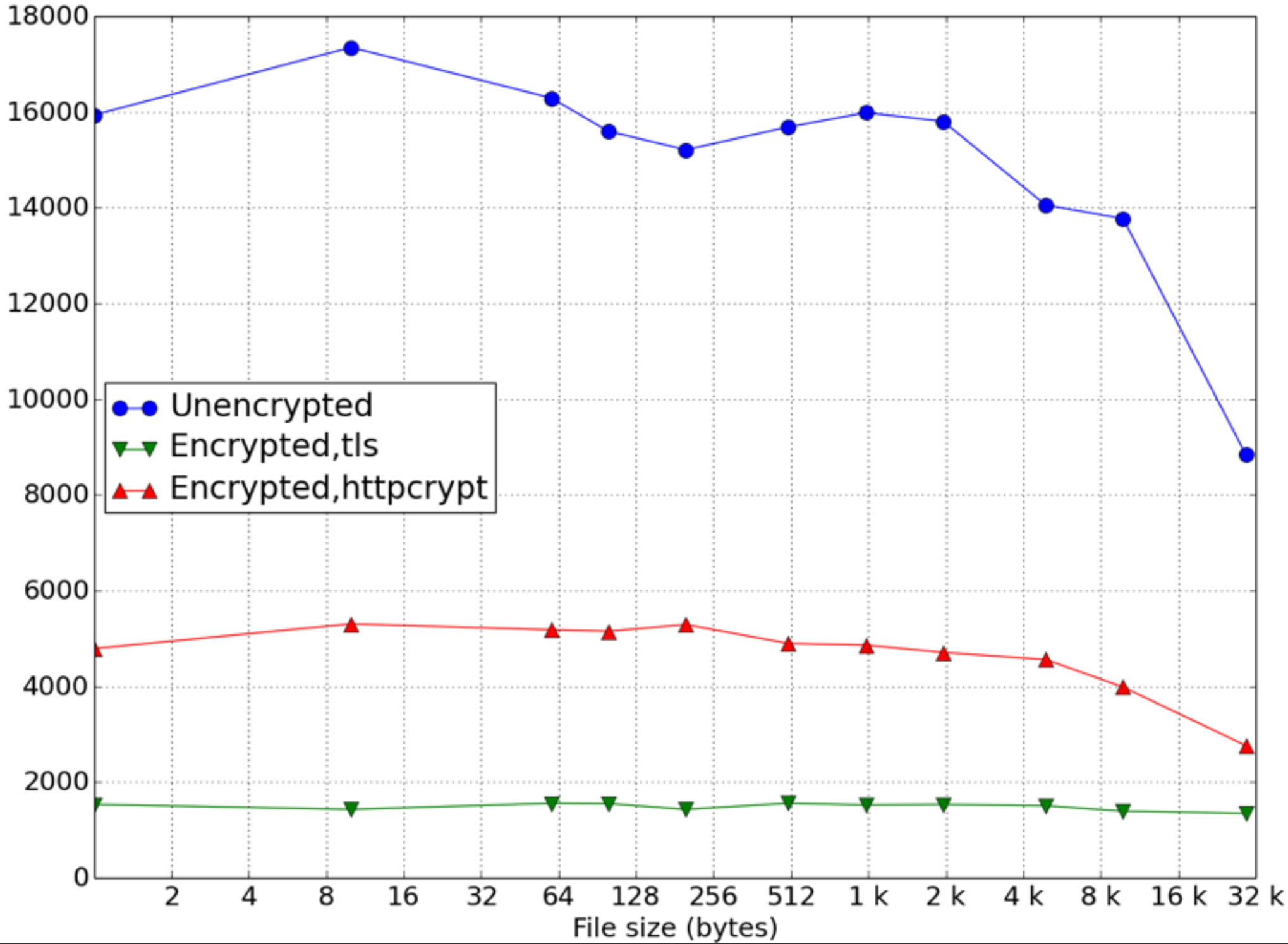
Пропускная способность

Шифрование кэшированных соединений



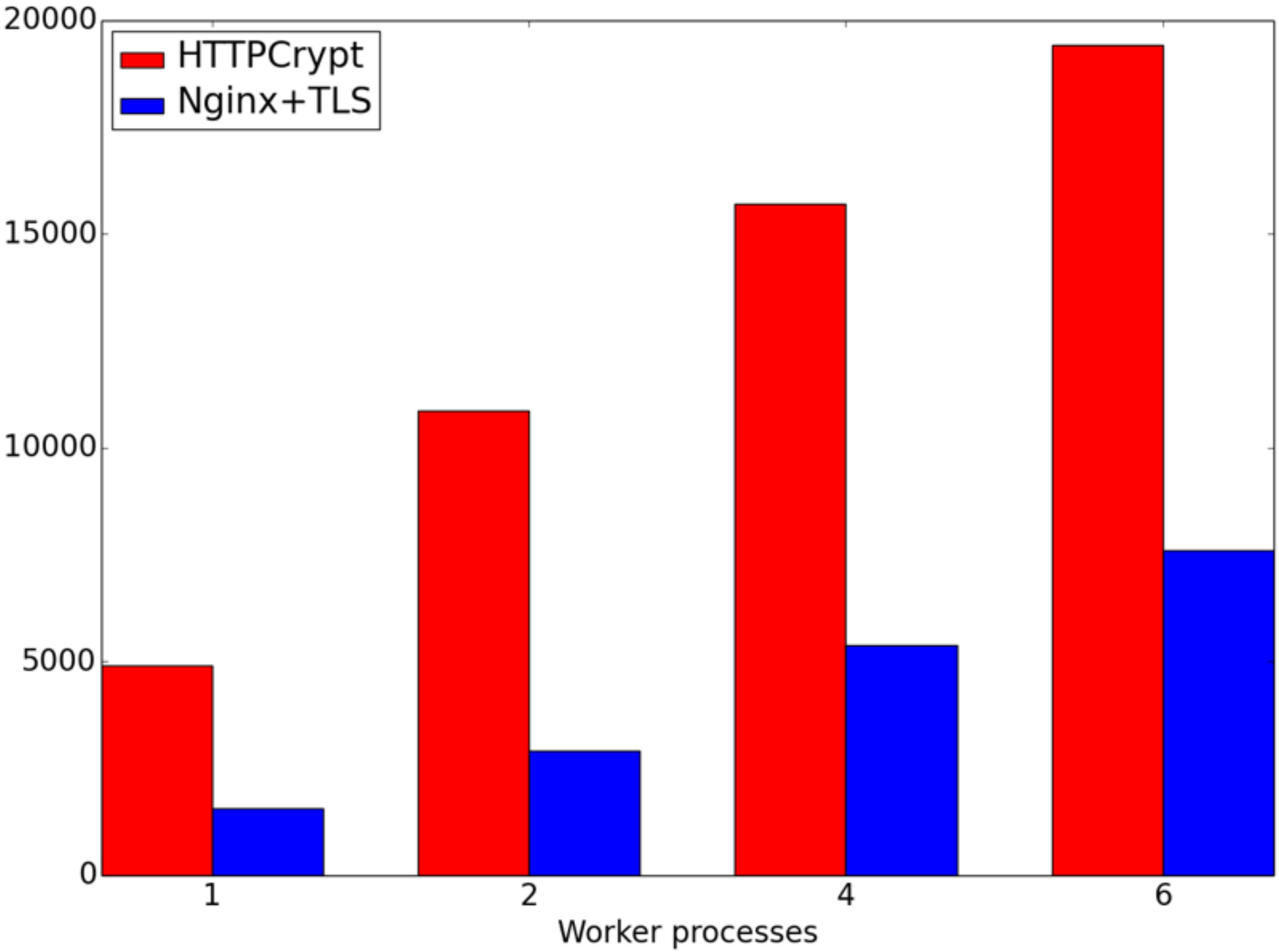
Пропускная способность

Шифрование новых соединений



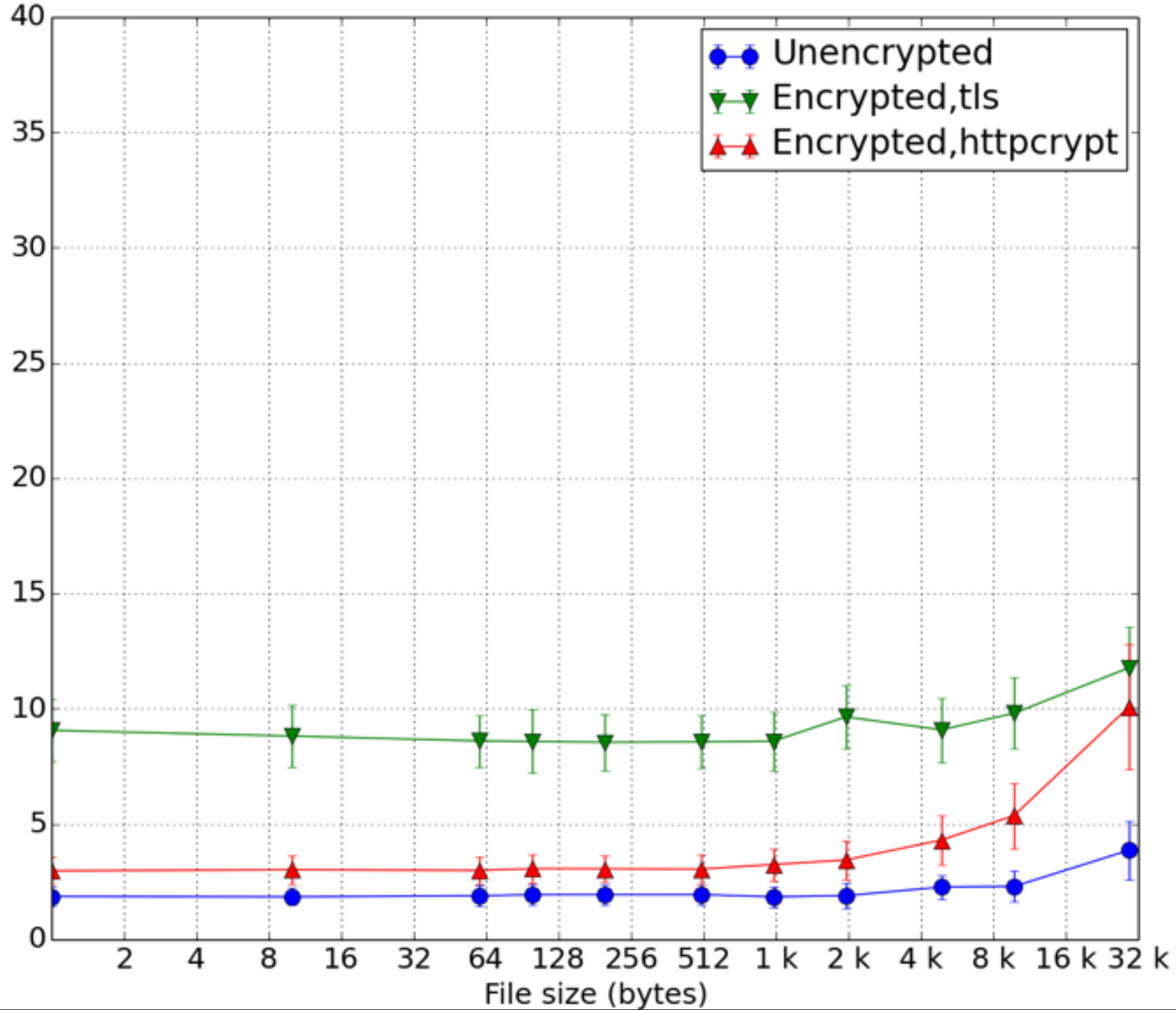
Пропускная способность

Шифрование новых соединений



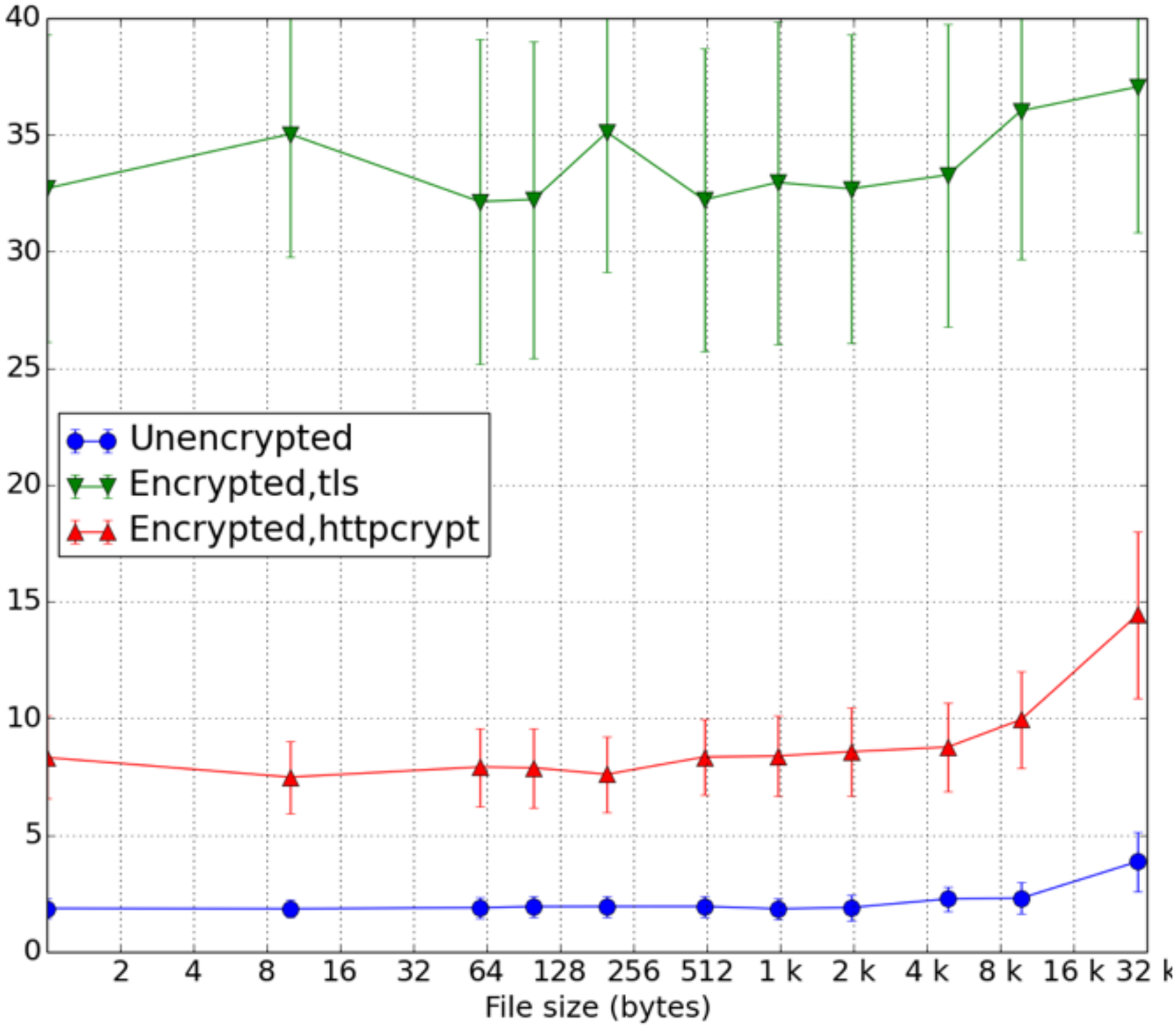
Задержка соединения

Шифрование кэшированных соединений



Задержка соединения

Шифрование новых соединений



Анализ производительности

Почему HTTPCrypt работает быстро

Анализ производительности

Почему HTTPCrypt работает быстро

- Более быстрая асимметричная криптография (ECDH)

Анализ производительности

Почему HTTPCrypt работает быстро

- Более быстрая асимметричная криптография (ECDH)
- Отсутствие подписи и ее проверки (как требуется в TLS)

Анализ производительности

Почему HTTPCrypt работает быстро

- Более быстрая асимметричная криптография (ECDH)
- Отсутствие подписи и ее проверки (как требуется в TLS)
- Шифрование без промежуточного копирования (на месте)

Анализ применимости

Анализ применимости

- HTTPCrypt применяется для работы с клиентами сканера

Анализ применимости

- HTTPCrypt применяется для работы с клиентами сканера
- Есть проект JavaScript реализации для защиты Web интерфейса

Анализ применимости

- HTTPCrypt применяется для работы с клиентами сканера
- Есть проект JavaScript реализации для защиты Web интерфейса
- Аналогичный метод применяется для работы с хранилищем хешей

Анализ применимости

- HTTPCrypt применяется для работы с клиентами сканера
- Есть проект JavaScript реализации для защиты Web интерфейса
- Аналогичный метод применяется для работы с хранилищем хешей
- Библиотека rdns, используемая для работы с DNS, поддерживает DNSCurve, который позволяет шифровать DNS запросы теми же конструкциями

Выводы

Выводы

Выводы

- Rspamd спроектирован, чтобы обеспечивать высокую производительность

Выводы

- Rspamd спроектирован, чтобы обеспечивать высокую производительность
- Rspamd обладает большим набором правил и плагинов, поставляемых “из коробки”

Выводы

- Rspamd спроектирован, чтобы обеспечивать высокую производительность
- Rspamd обладает большим набором правил и плагинов, поставляемых “из коробки”
- Весь трафик, отправляемый rspamd по сети, можно шифровать

Вопросы?

Всеволод Стахов
https://rspamd.com

Дополнительные материалы

Администрирование rspamd

Основы

- Rspamd устанавливается из пакетов/портов для большинства систем и работает “из коробки”
- Предоставляется сервис нечетких хешей, обучаемый из honeypot’ов
- Web интерфейс для большинства повседневных задач

Администрирование rspamd

Ручная настройка

- Секции

```
section {
  key = “value”;
  number = 10K;
}
```

- Массивы

```
upstreams = [
      “localhost:80”,
      “example.com:8080”,
]
```

- Переменные

```
static_dir = “${WWWDIR}/“;
filepath = “${CURDIR}/data”;
```

- Макросы

```
.include “${CONFDIR}/workers.conf”
.include (glob=true,priority=2) “${CONFDIR}/conf.d/*.conf”
.lua { print(“hey!”); }
```

- Комментарии

```
key = value; // Single line comment
/* Multiline comment
/* can also be nested */
 */
```

Администрирование rspamd

Компоненты конфигурации

- Основные правила, веса и другие настройки используются по умолчанию после установки
- *rspamd.local.conf* используется для **добавления** опций
- *rspamd.override.conf* используется для **переопределения** опций
- *rspamd.local.lua* содержит локальные правила на языке lua

Глобальные опции

Настройки процессов

Настройки весов

Опции статистики

Конфигурация плагинов

Администрирование rspamd

Обучение

- Для обучения статистики нужно примерно равное количество примеров спама и хама

- Обучение статистики и хешей делается сразу для всех серверов (защита от повторного обучения)

- Утилитой **rspamadm** можно выполнять слияние баз статистики и баз хешей

Администрирование rspamd

Правила регулярных выражений

```
-- Outlook versions that should be excluded from summary rule
local fmo_excl_o3416 = 'X-Mailer=/^Microsoft Outlook, Build 10.0.3416$/H'
local fmo_excl_oe3790 = 'X-Mailer=/^Microsoft Outlook Express 6.00.3790.3959$/H'
-- Summary rule for forged outlook
reconf['FORGED_MUA_OUTLOOK'] = string.format('(%s | %s) & !%s & !%s & !%s',
                    forged_oe, forged_outlook_dollars, fmo_excl_o3416, fmo_excl_oe3790, vista_msgid)
```

Администрирование rspamd

Правила на Lua

```
rspamd_config.R_EMPTY_IMAGE = function(task)
  local tp = task:get_text_parts() -- get text parts in a message

  for _,p in ipairs(tp) do -- iterate over text parts array using `ipairs`
    if p:is_html() then -- if the current part is html part
      local hc = p:get_html() -- we get HTML context
      local len = p:get_length() -- and part's length

      if len < 50 then -- if we have a part that has less than 50 bytes of text
        local images = hc:get_images() -- then we check for HTML images

        if images then -- if there are images
          for _,i in ipairs(images) do -- then iterate over images in the part
            if i['height'] + i['width'] >= 400 then -- if we have a large image
              return true -- add symbol
            end
          end
        end
      end
    end
  end
end
```