メールフィルタの作り方

Rubyで作るmilter

須藤功平

株式会社クリアコード

2010/02/13

話題

- ✓ メールフィルタのこと
- ✓ milterの概要
- ✓ SMTPのこと
- ✓ milterの詳細
- ✓ Rubyでmilterを作る！

今日のみなさんのゴール

1. milterを説明できる
2. SMTPを話せる
3. Rubyでmilterを作れる

自分のこと

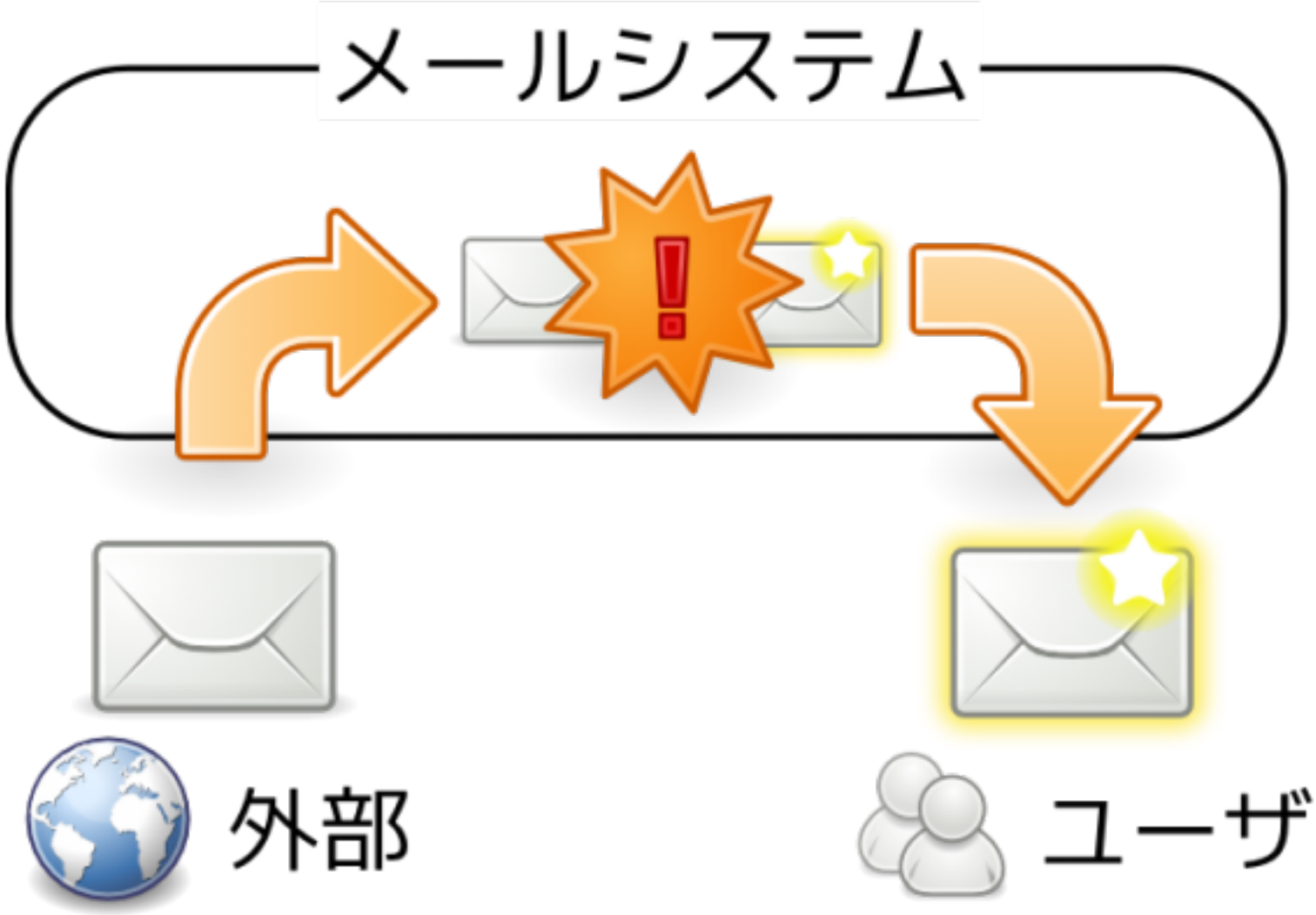
- 最近よく札幌に来る
  - 2009/06: OSC-do
  - 2009/12: 札幌Ruby会議02
  - 2010/02: LDD'10 Winter
- 実家の青森には年1回

札幌たいやき部

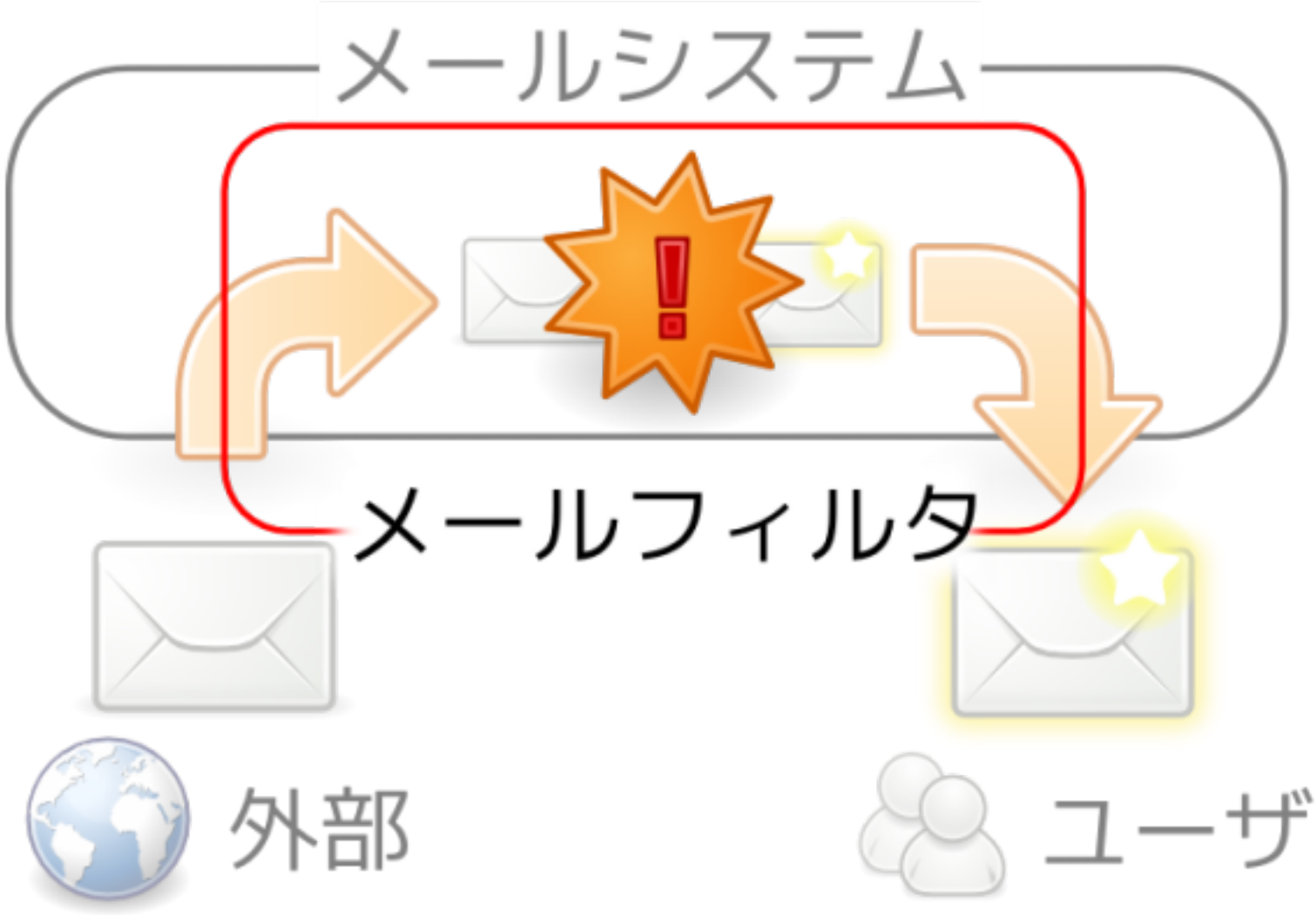
メールフィルタのこと

- ✓ メールフィルタのこと
- ✓ milterの概要
- ✓ SMTPのこと
- ✓ milterの詳細
- ✓ Rubyでmilterを作る！

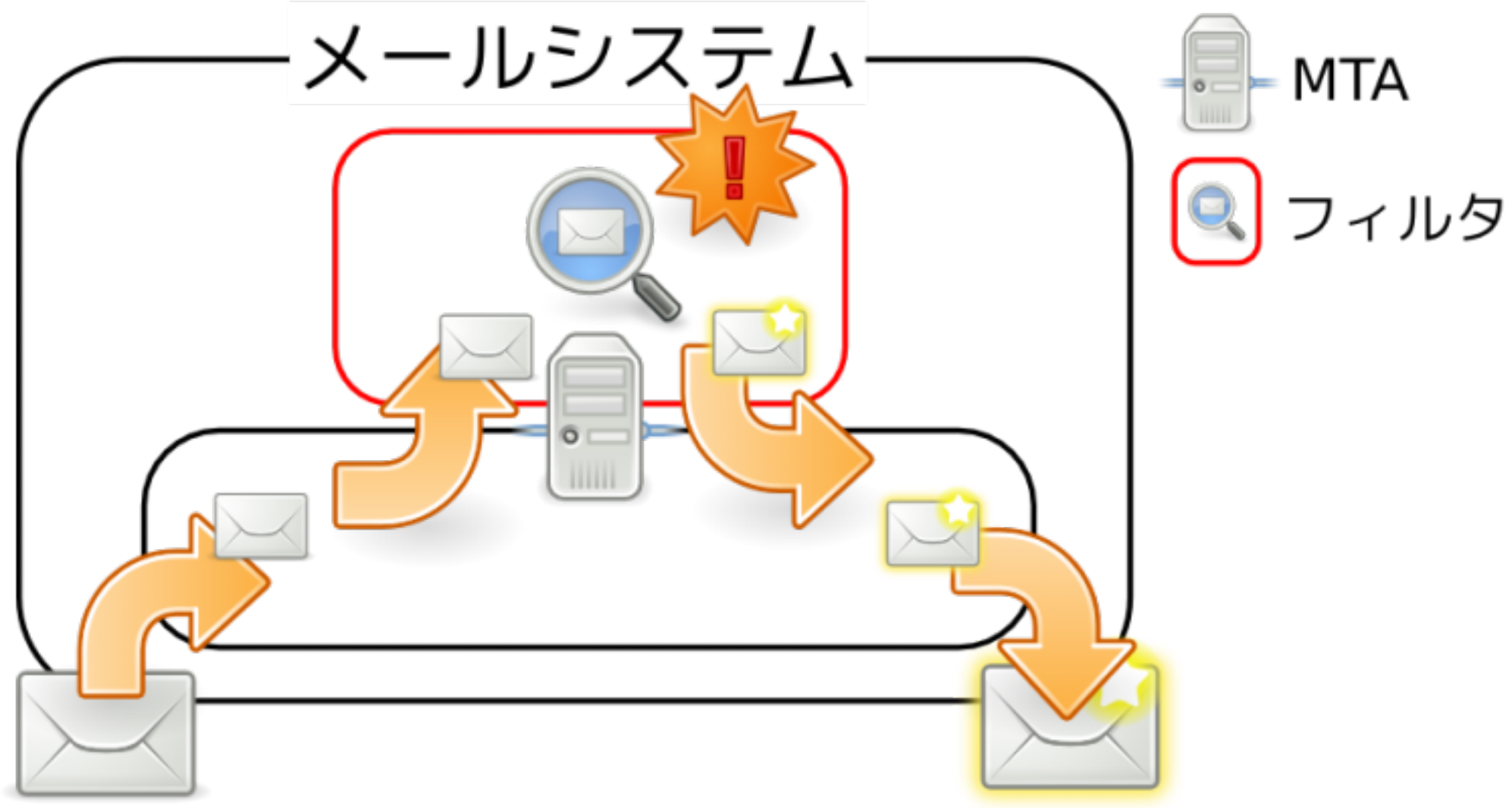
メールシステム



メールフィルタ



MTAにプラグイン



メールフィルタのまとめ

- メールシステムは持っている
- プラグインとして追加できる
- 用途はいろいろ
  - 迷惑メール対策・メールアーカイブ
  - アプリケーション連携

milterの概要

- ✓ メールフィルタのこと
- ✓ milterの概要
- ✓ SMTPのこと
- ✓ milterの詳細
- ✓ Rubyでmilterを作る！

ゴール(1)

1. milterを説明できる
2. SMTPを話せる
3. Rubyでmilterを作れる

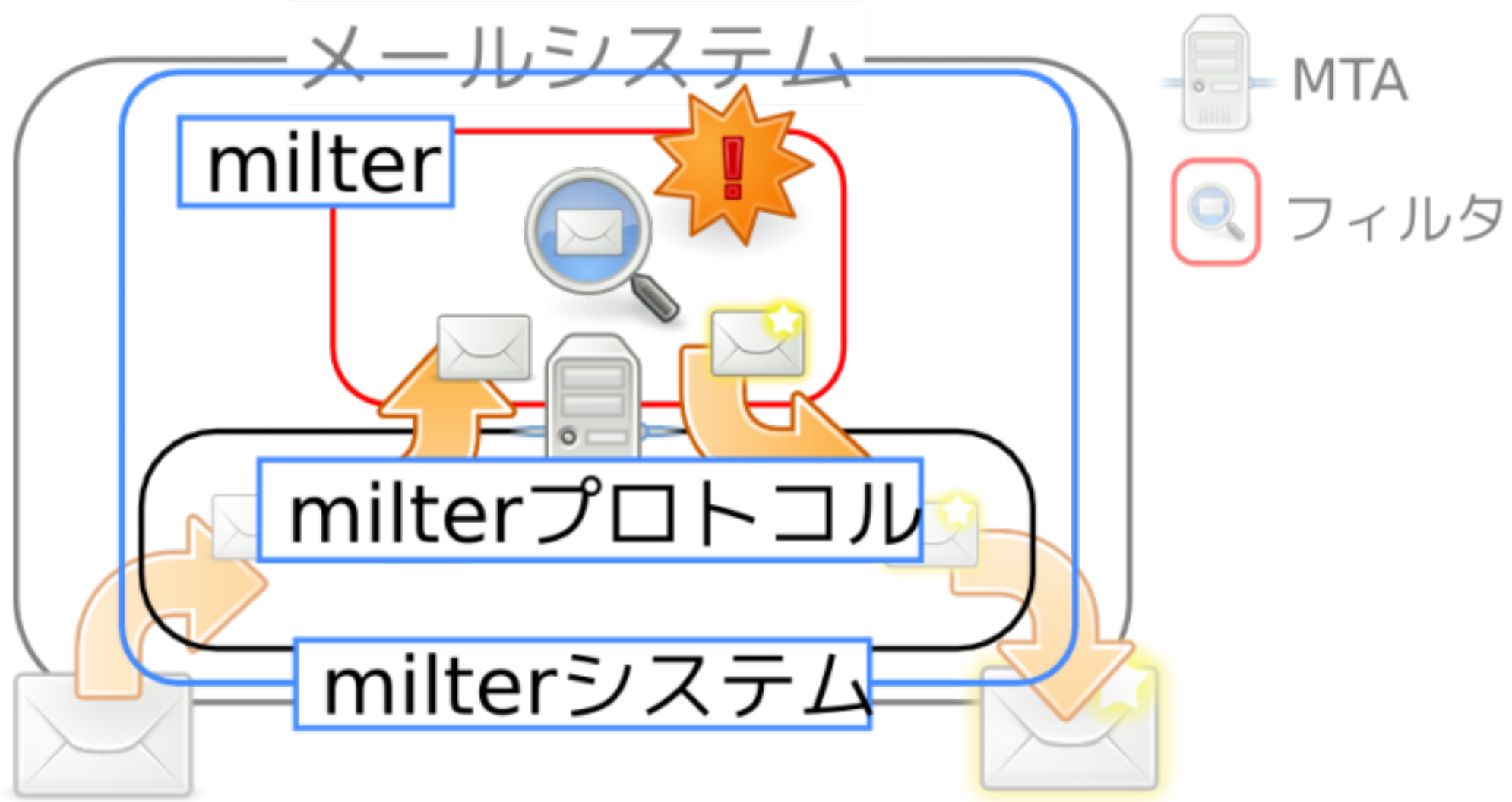
milter?

- メールフィルタ: mail filter
- 汎用の仕組み
  - Sendmail/Postfixなどで使える
  - MTAにプラグインタイプ
- そこそこ利用されている

milter関連用語

1. milterシステム
2. milterプロトコル
3. milter

milterシステム



milter概要まとめ

- 仕組み: milterシステム
  - MTAにプラグインタイプ
- MTAとmilterのやりとり: milterプロトコル
  - SMTPと密接
- フィルタ: milter

SMTPのこと

- ✓ メールフィルタのこと
- ✓ milterの概要
- ✓ SMTPのこと
- ✓ milterの詳細
- ✓ Rubyでmilterを作る！

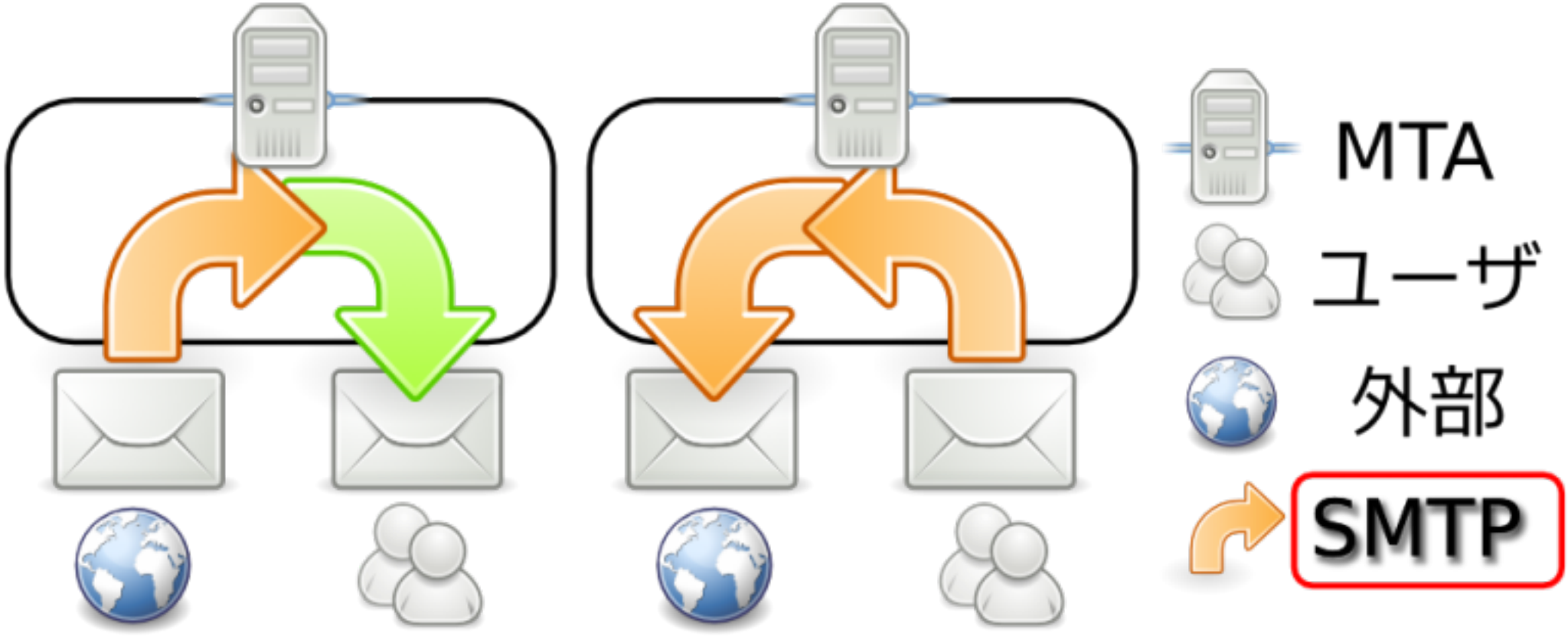
ゴール(2)

1. milterを説明できる
2. SMTPを話せる
3. Rubyでmilterを作れる

SMTP

- ✓ Simple: 簡単な
- ✓ Mail: メールを
- ✓ Transfer: 転送する
- ✓ Protocol: 方法

どこでSMTP?



簡単？

基本は4コマンド:

1. HELO/EHLO: あいさつ
2. MAIL FROM: 差出人
3. RCPT TO: 宛先
4. DATA: 本文

プロトコルの特徴

- ステートフル
  - 1接続 == 1セッション
- nメール/1セッション
  - MAIL FROM〜DATAをn回できる
- n宛先/1メール
  - RCPT TOをn回できる

SMTPのまとめ

- メール配送処理はほぼSMTP
- 基本は4コマンド
  - HELO → MAIL FROM →
  - RCPT TO → DATA
- ステートフル
  - セッションがある

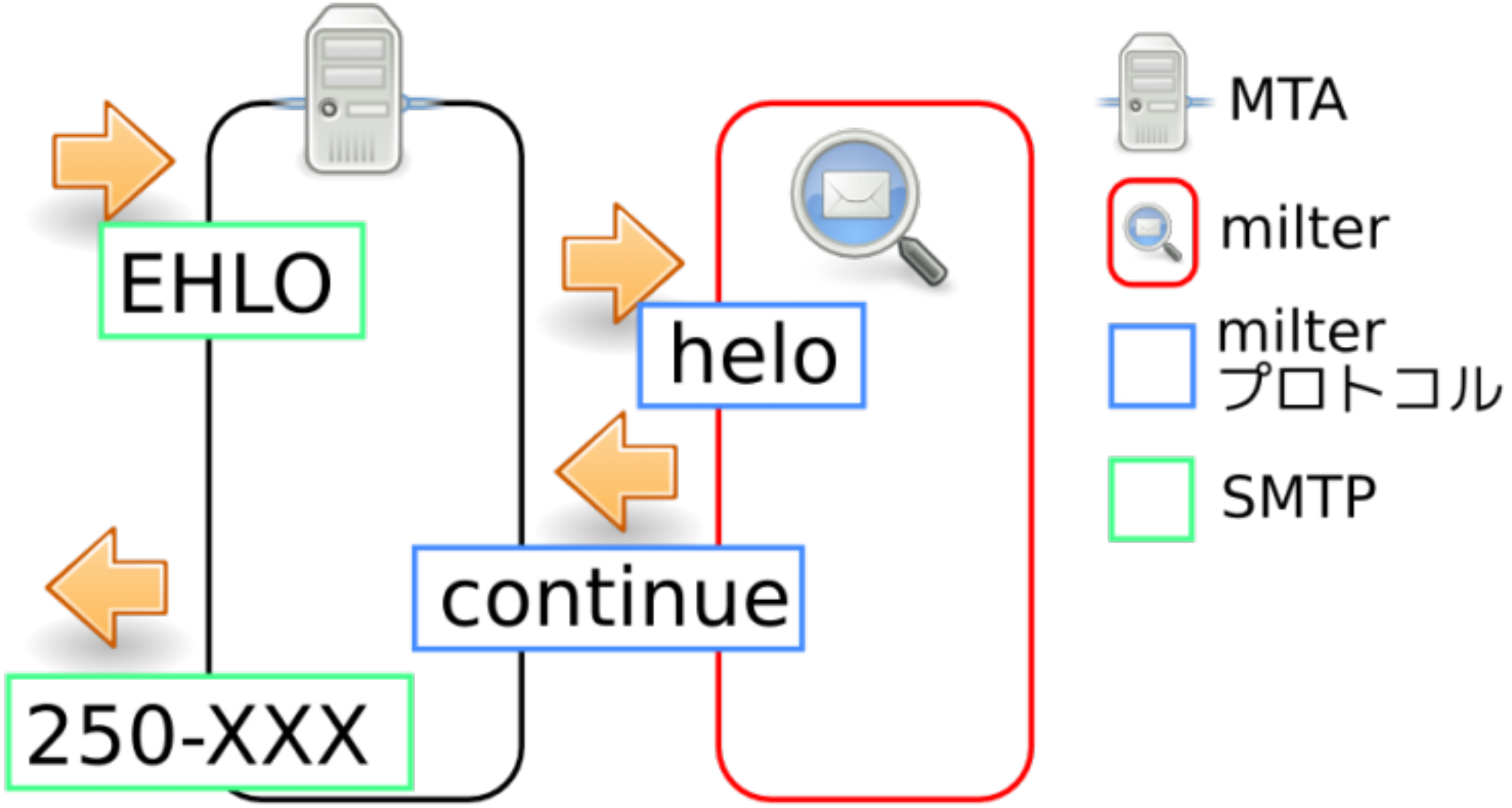
ゴール(3)

1. milterを説明できる
2. SMTPを話せる
3. Rubyでmilterを作れる

milterの詳細

- ✓ メールフィルタのこと
- ✓ milterの概要
- ✓ SMTPのこと
- ✓ milterの詳細
- ✓ Rubyでmilterを作る！

SMTPとmilterプロトコル

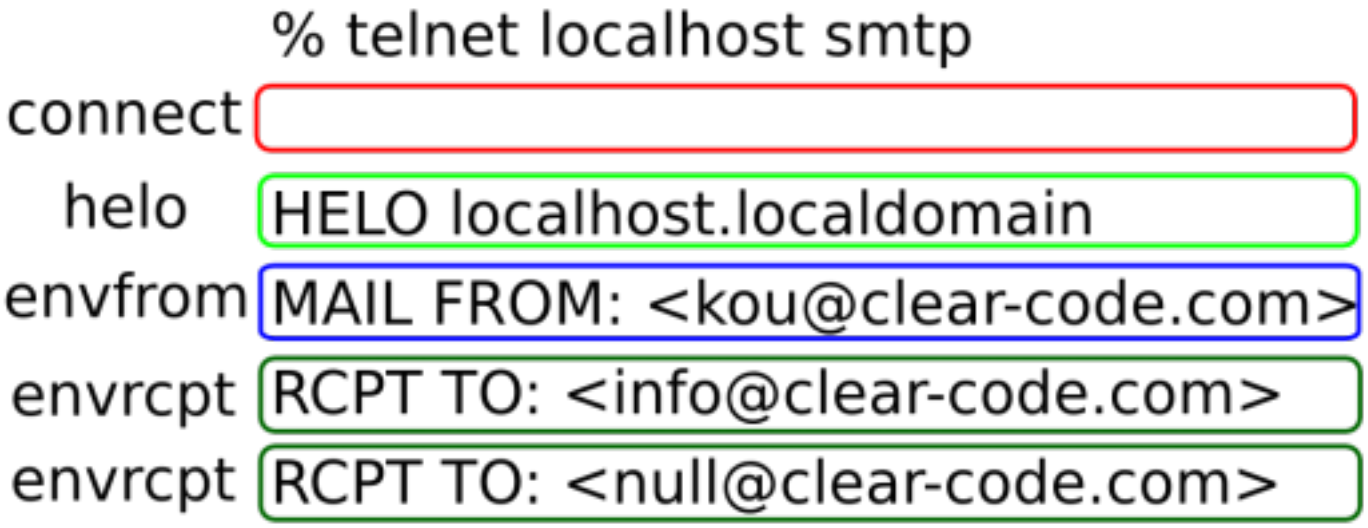


milterプロトコル

- SMTPと平行動作
- SMTPセッション == milterセッション
- セッション毎に情報を持つ

コマンド: メタ情報

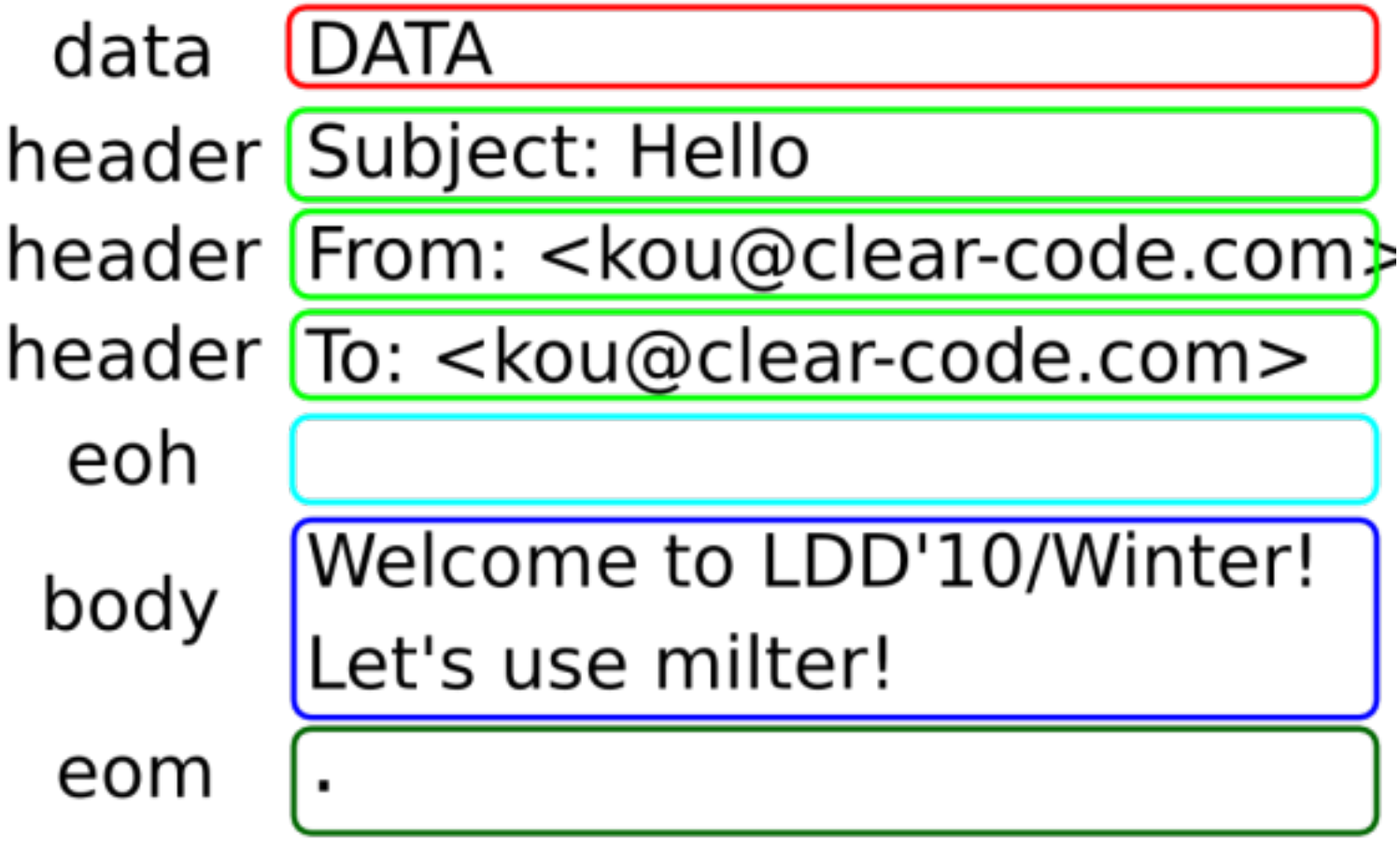
- ✓ connect
- ✓ helo
- ✓ envfrom
- ✓ envrcpt



コマンド: DATA

分解

- ✓ data
- ✓ header
- ✓ eoh
- ✓ body
- ✓ eom



セッション

- 同時に複数セッション
  - 1プロセス+nスレッド
  - 1プロセス+非同期処理
- セッション管理: ライブラリ

milter詳細のまとめ

- SMTPと密接
  - コマンドはSMTPのコマンドに対応
  - メッセージはパース済み
- 同時に複数セッションを処理
- セッション管理はライブラリ

Rubyでmilterを作る！

- ✓ メールフィルタのこと
- ✓ milterの概要
- ✓ SMTPのこと
- ✓ milterの詳細
- ✓ Rubyでmilterを作る！

ゴール(3)（リプライズ）

1. milterを説明できる
2. SMTPを話せる
3. Rubyでmilterを作れる

お題

メール検索（again）

材料

- ✓ Ruby
- ✓ groonga
- ✓ milter manager

扱うもの

- ✓ Subject
- ✓ From
- ✓ To
- ✓ 本文

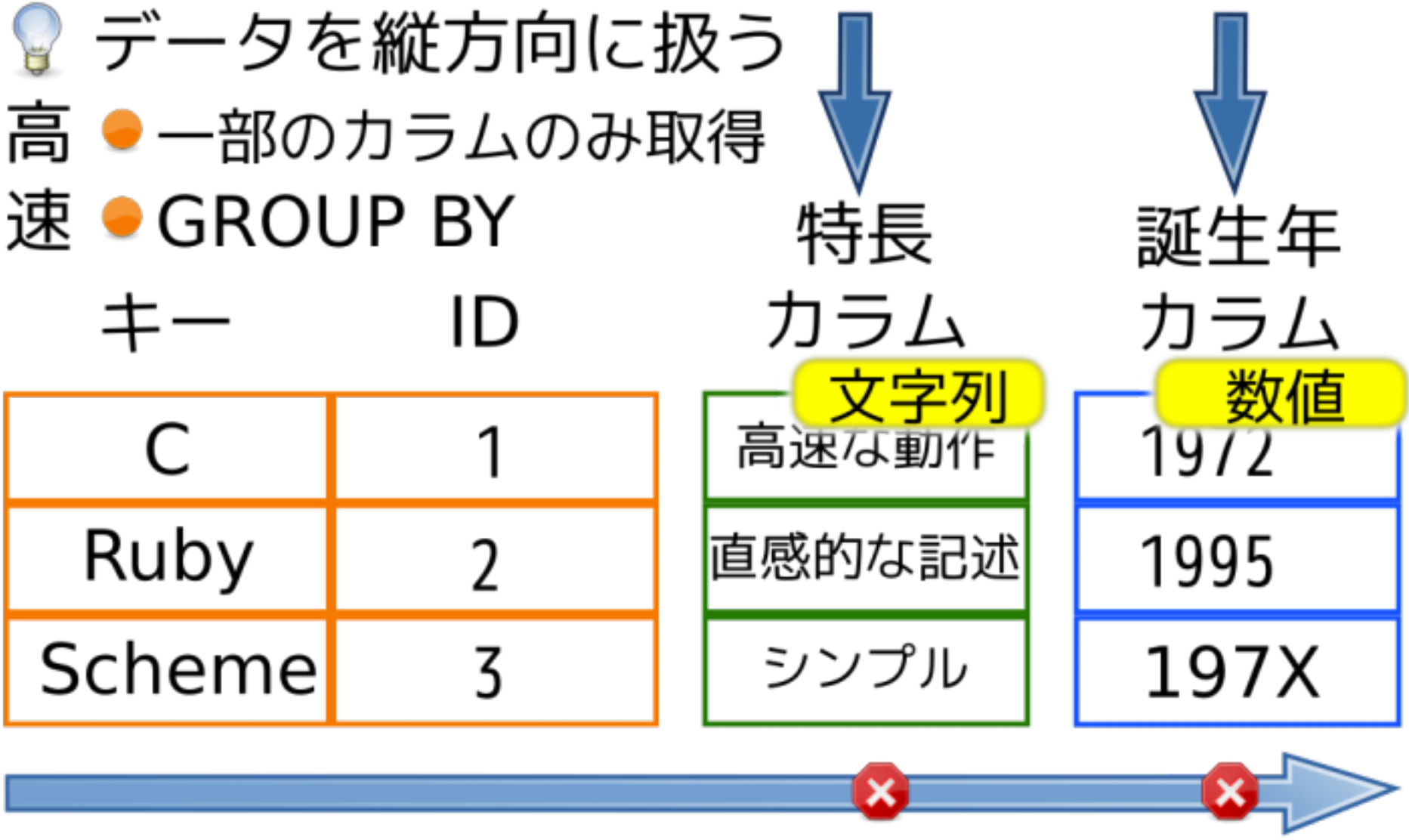
```
Subject: Rubyで作るmilter
From: <kou@clear-code.com>
To: <announce@clear-code.com>

2010/2/13（土）の
LOCAL DEVELOPER DAY'10/Winter
でRubyでmilterを作る方法を紹介します。
```

groonga?

- 全文検索エンジン
  - N-gram and/or 形態素解析
- カラム指向データストア
- キー管理
  - ハッシュテーブル
  - バイナリパトリシアトライ

カラム指向

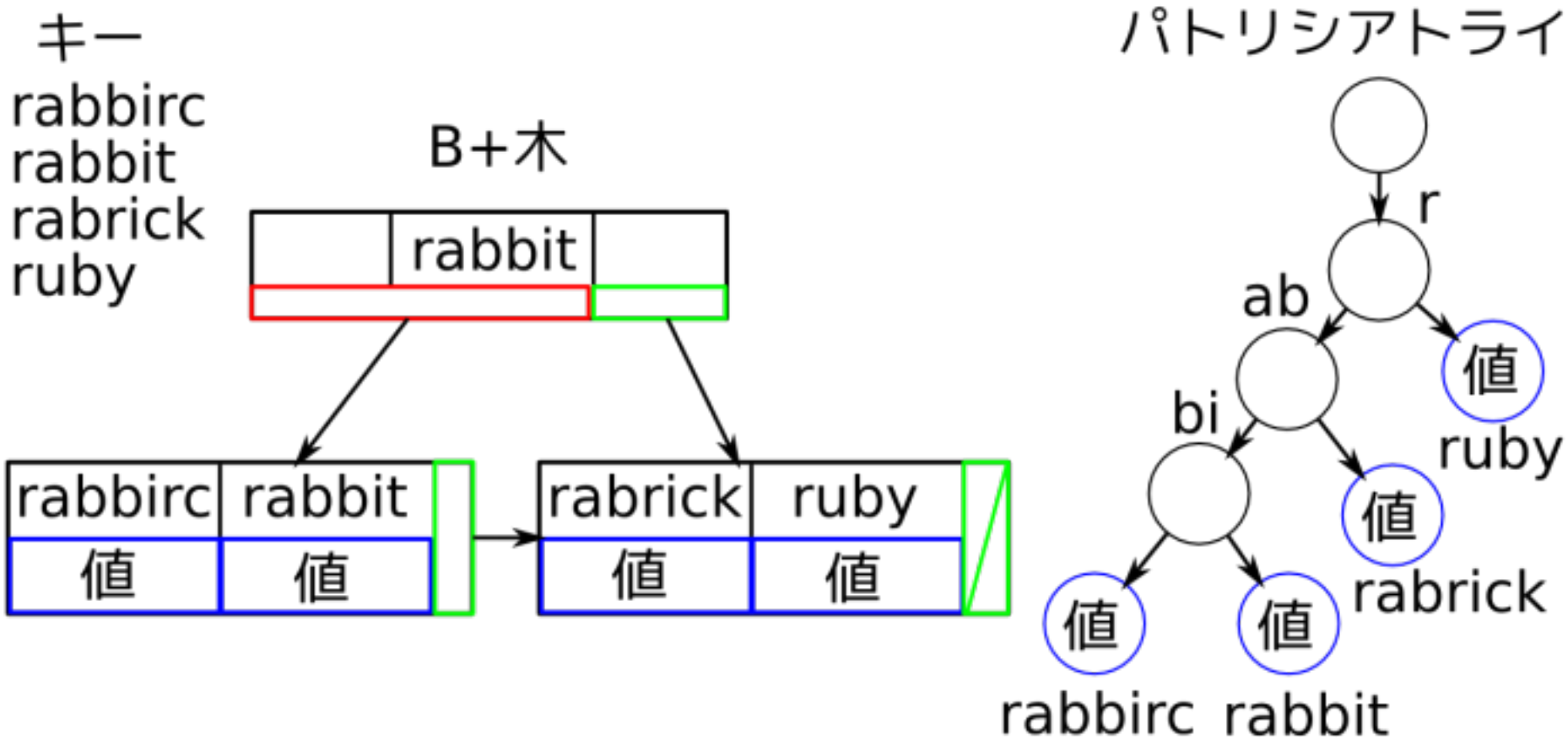
- データを縦方向に扱う
  - 高速
    - 一部のカラムのみ取得
    - GROUP BY



パトリシアトライ

- 木構造の一種
  - 文字列の検索に向いている
  - O(k): k = 文字列長
  - prefix検索・最長一致検索
  - キーワード検出（ニコニコ大百科）
- DBMではB+木採用が多い

B+木とパトリシア


キー
rabbirc
rabbit
rabrick
ruby
B+木
rabbit
rabbirc
rabbit
値
値
rabrick
ruby
値
値
パトリシアトライ
r
ab
bi
値
ruby
値
rabrick
値
値
rabbirc
rabbit

milter manager?

- milter支援ツール
  - 便利な管理機能を提供
  - デバッグ支援
- milterプロトコルを実装
  - ライブラリ
  - Rubyバインディング

ゴール(3)（リプライズ）

1. milterを説明できる
2. SMTPを話せる
3. Rubyでmilterを作れる

扱うもの

- ✓ Subject
- ✓ From
- ✓ To
- ✓ 本文

```
Subject: Rubyで作るmilter
From: <kou@clear-code.com>
To: <announce@clear-code.com>

2010/2/13（土）の
LOCAL DEVELOPER DAY'10/Winter
でRubyでmilterを作る方法を紹介します。
```

スキーマ: Messages

```
Groonga::Schema.define do |schema|
  schema.create_table("Messages") do |table|
    table.short_text("subject")
    table.short_text("from")
    table.short_text("to")
    table.text("body")
  end
  ...
```

スキーマ: Terms

```
  ...
  schema.create_table("Terms",
                      :type => :patricia_trie,
                      :key_type => "ShortText",
                      :default_tokenizer => "TokenBigram",
                      :key_normalize => true) do |table|
    table.index("Messages.subject")
    table.index("Messages.body")
  end
end
```

Rubyでmilter

```ruby
require 'milter'

class Session < Milter::ClientSession
  def connect(context, host, address)
    ...
  end
  def helo(context, fqdn)
    ...
  end
  ...
end
```

ヘッダ

```
def header(context, name, value)
  @values ||= {}
  case name
  when /\A(Subject|From|To)\z/i
    key = $1.to_s.downcase
    utf8_value = NKF.nkf("-w", value)
    @values[key] = utf8_value
  when /\AContent-Transfer-Encoding\z/i
    @encoding = value
  end
end
```

本文

```
def body(context, chunk)
  @values["body"] ||= ''
  @values["body"] << chunk
end
```

登録

```
def end_of_message(context)
  @encoding ||= nil
  nkf_option = "-w"
  nkf_option << " -MB" if @encoding == "base64"
  body = @values["body"]
  @values["body"] = NKF.nkf(nkf_option, body)
  messages = Groonga["Messages"]
  messages.add(@values)
end
```

表示

```
messages = Groonga["Messages"]
messages.each do |message|
  puts "-" * 80
  puts "Subject: #{message.subject}"
  puts "From: #{message.from}"
  puts "To: #{message.to}"
  puts
  puts message.body
  puts "-" * 80
end
```

検索

```
messages = Groonga["Messages"]
result = messages.select do |record|
  record["subject"].match(query) |
    record["body"].match(query)
end
result.sort([["_score", :desc]]).each do |message|
  puts "#{message.subject} (#{message.score})"
end
```

ゴール確認

1. milterを説明できる
2. SMTPを話せる
3. Rubyでmilterを作れる

まとめ

- メールフィルタ:
  - メールシステムには必須機能
- milter: メールフィルタの仕組み
- Rubyでmilterを書く環境がある
  - groongaも便利！

ありがとう！

また
札幌に来るとは
！！！！！