MySQL Network Protocol in depth

Yoshinori Matsunobu

Senior MySQL Consultant
Professional Services APAC
Sun Microsystems

Yoshinori.Matsunobu@sun.com

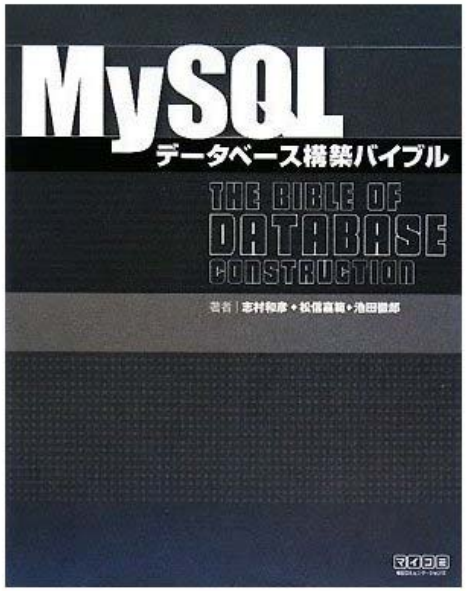
内容

- プロトコル解説
  - 認証時のパケットフロー
  - INSERT文実行時のパケットフロー
  - SELECT文実行時のパケットフロー
  - プリペアードステートメント実行時のパケットフロー
  - サーバーサイドカーソルのパケットフロー
- プロトコルの活用
  - tcpdumpとtsharkによる動的クエリ解析
- 簡単なドライバプログラムのデモ
  - Erlangからの接続・操作
- とみたまさひろ さんによる解説
  - Rubyドライバ Ruby/MySQLおよびMySQL/Ruby
- 将来的なアイデア
- ディスカッション

プロトコル解説の資料

- 英語
  - http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol

- 日本語
  - 「MySQL構築バイブル」毎日コミュニケーションズ



前提知識

数値の表現方法

●エンディアン
すべてリトルエンディアン
(例:257、4バイト表現であれば
0xFF 02 00 00)

●固定長の場合
16進数表現

●可変長の場合
Length Coded Binary
　先頭1～9バイトで値を表現。
1バイト目が:
　0-250 : その値が実際の値
　251: NULL
　252: 後続の2バイトが実際の値
　253: 後続の4バイトが実際の値
　254: 後続の8バイトが実際の値

文字列の表現方法

●固定長文字列
そのまま

●可変長:NULL終端
　文字列の終端を0x00で埋める

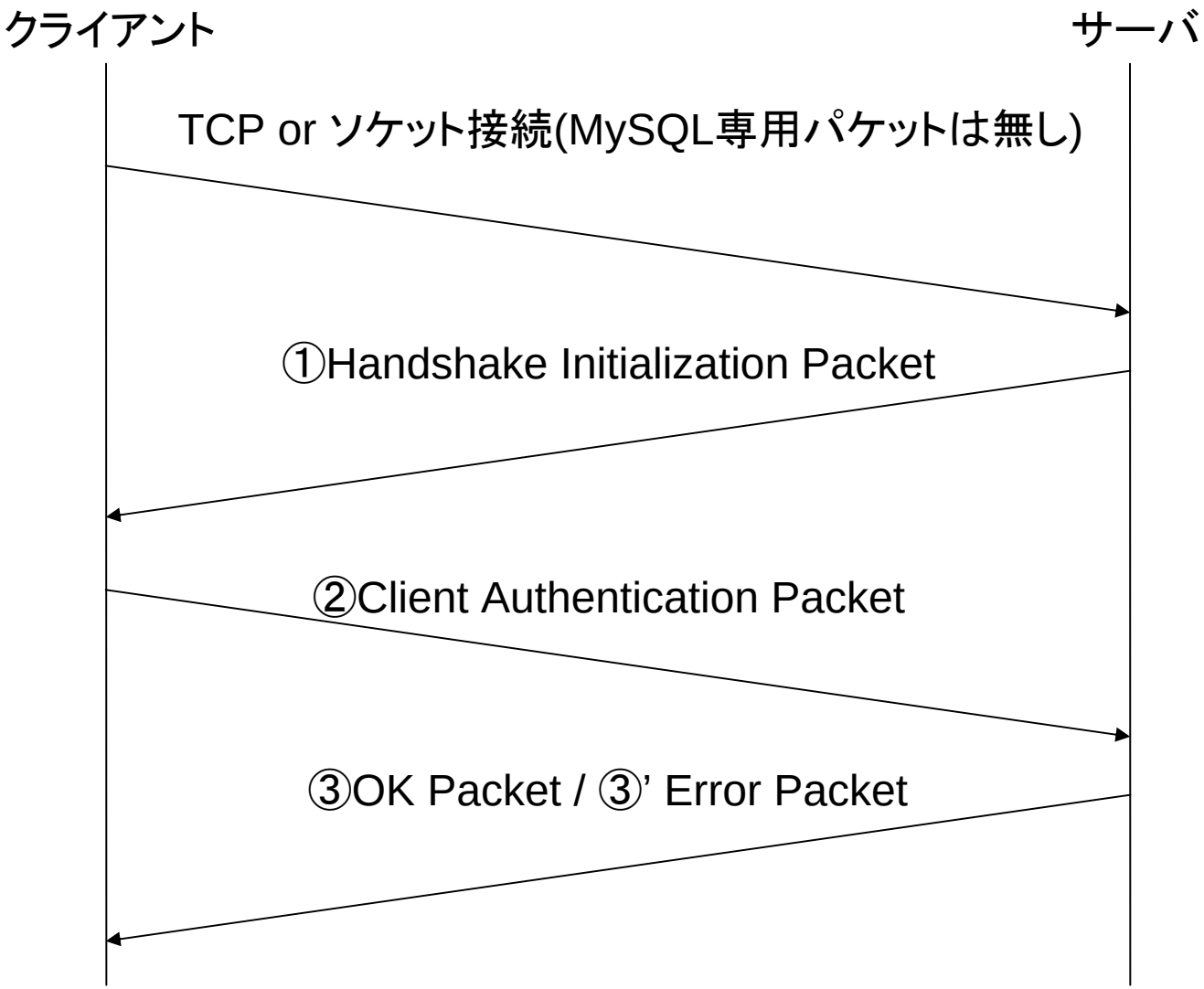
●可変長:Length Coded String
　LCBと同様だが、文字列長を1～9バイトで表現し、
実際の文字列はその直後から続く
　0x00終端とは違い、文字列の中に
0x00を含めることができる。
　また、長さが最初から分かっているので
メモリ確保を最適化できる

例:abc　→ 0x03 61 62 63

あると便利なツール

- パケットアナライザ
  - Ethereal
  - tcpdump
- 統合開発環境/デバッガ
  - Microsoft Visual Studio
  - NetBeans
  - Eclipse
  - ddd/gdb

認証時のパケットフロー



Client Authentication Packet

・ユーザー名、データベース名は0x00終端
・パスワードは平文では流れない
　・パスワードおよび、Handshake時の暗号化の種を組み合わせて20バイトのランダム文字列を生成
　参照： password.c#scramble()
　ネイティブドライバは、このscramble処理を自力で実装する必要がある
・データベース名を直接指定して接続すれば、後で「USE database_name」1回分のパケットを省略できる

更新系SQL文の実行

クライアント→サーバ　：

①Command Packet

サーバ→クライアント　：

②OK/Error Packet

検索系SQL文の実行

クライアント→サーバ　：

①Command Packet

サーバ→クライアント　：

②ResultSet Header Packet　1パケット

③Field Packets　列数分のパケット

④EOR Packet　1パケット

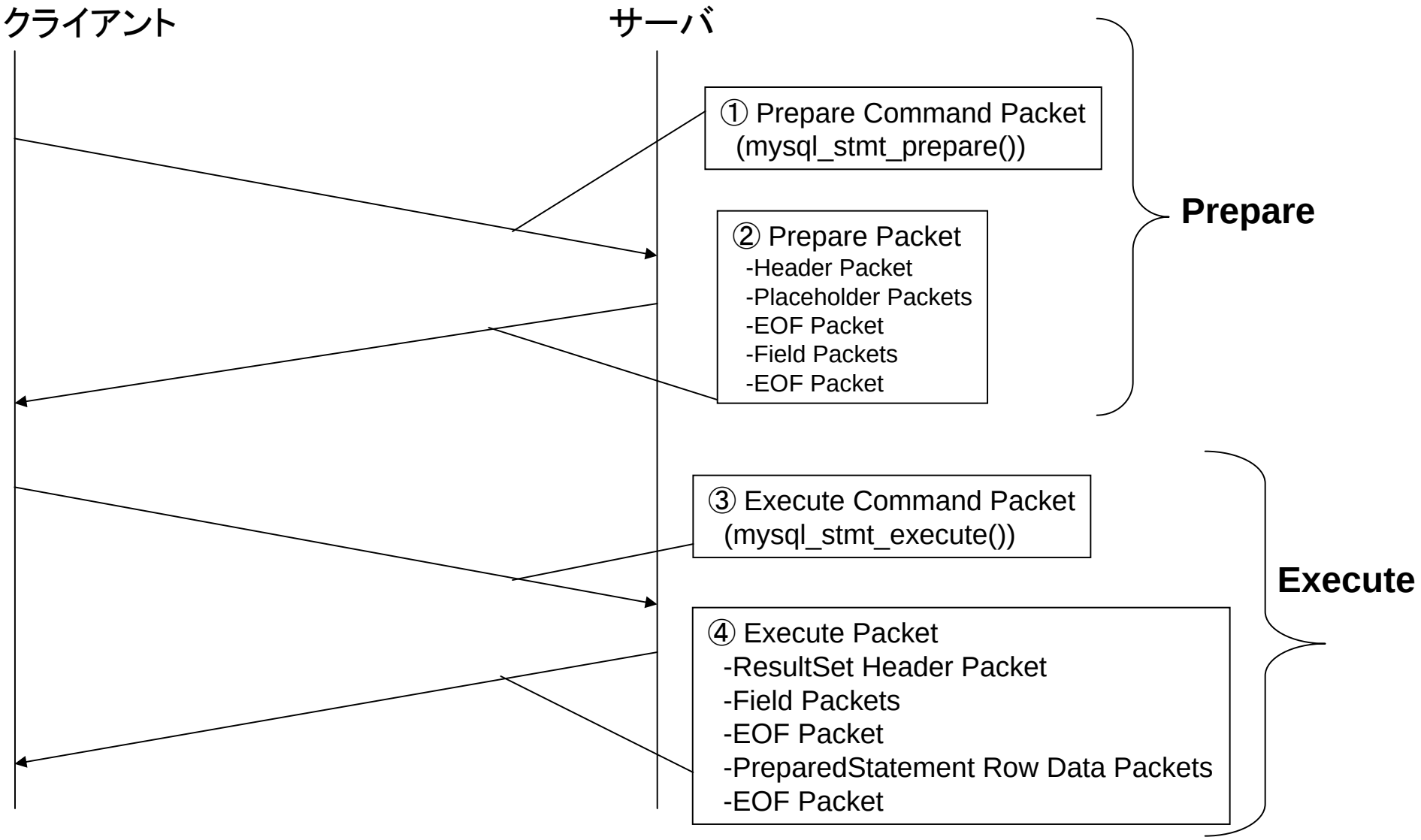
⑤Row Data Packets　レコード数分のパケット

⑥EOF Packet

デモ

- ログイン – SELECT – INSERT – SELECT時のパケットフロー
- ソースコード上のポイントをいくつか

プリペアードステートメント

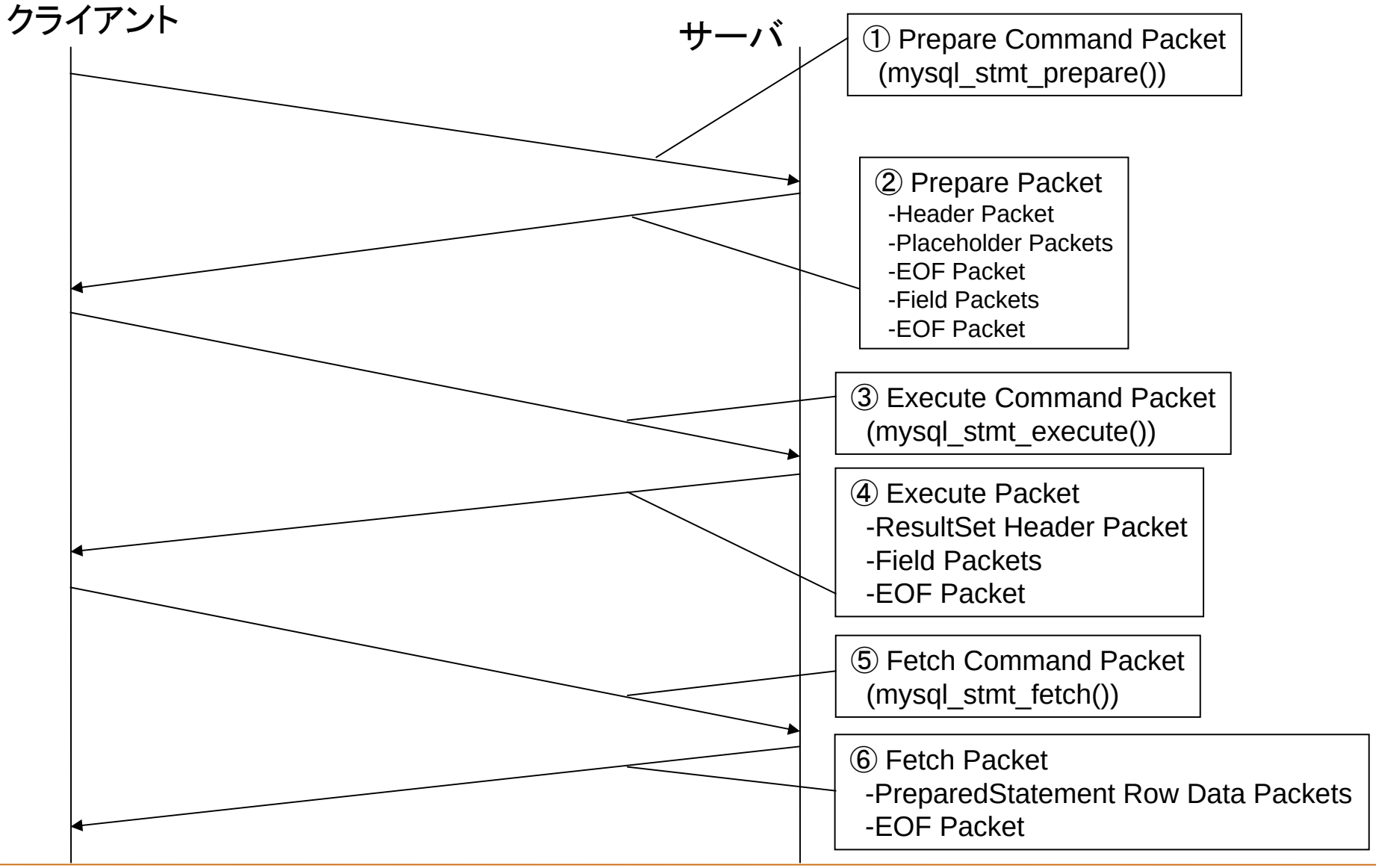


プリペアードステートメントのパケット

- Prepare、Executeのそれぞれで1往復ずつ発生する
- アプリケーションでは、
  プリペアードステートメントキャッシュを使うのが定石
  - Prepare済みのオブジェクトをキャッシュしておき、以降では Executeだけ行なう

- プリペアードステートメントキャッシュが効く場合、通常のSQL文よりも実行効率が良い
  - SQL文の構文解析が不要
  - 実行計画の作成は毎回行なわれてしまう (MySQL実装上の制約)

サーバーサイドカーソル



現場での応用：動的なSQL文の取得

- 本番環境において、一定時間の間、実行されたクエリの収集をしたいことがある
- General Query Logは本番ではOFFにすべき
- スロークエリログは実行時間1秒未満のものは出ず、解析には今一歩不便

- MySQL 5.1
  - Dynamic General Query Log (一般ログを動的に有効/無効化)
  - Microslow Log (実行時間1秒未満のクエリをスロークエリログに出す)

- パケットの中で、SQL文は平文で流れることが分かったので、それ「だけ」をキャプチャリングすれば目的を達成できる
  - COM_QUERYはパケット番号「3」
  - もっと単純に、クライアント→サーバ向きのパケットだけ取っても良い
- デモ：tcpdump + tsharkによるキャプチャリング

- MySQL Enterpriseユーザは、Enterprise MonitorのQuery Profilerによって、同じことをGUIベースで実現できる
  - EXPLAINの結果や取得件数を表示したりなど、非常に実用的

デモ

- 関数型プログラミング言語Erlangによる自作ドライバ
- 接続、INSERT、SELECT

とみたさんによるRubyドライバの解説

- とみたまさひろ さん
  - Rubyドライバの作者
    - ネイティブドライバ Ruby/MySQL
    - C APIを呼ぶラッパードライバ MySQL/Ruby
  - http://tmtm.org/ja/tdiary/
  - tommy@mysql.gr.jp

将来の計画/アイデア

- Drizzleでの計画
  - Full Duplex (全二重通信)
    - クライアントから巨大なパケットを送っている最中に、サーバからエラーを返すことができる
  - 連続送信
    - 複数のコマンドを、クライアント→サーバに対して一度に(片道1回で)送信
    - 後のコマンドの結果を、前のコマンドの結果よりも先に受け取ることも可能
  - 読み取り専用のフラグ
    - MySQL Proxyなどに、書き込みをマスターに、読み込みをスレーブに振り分けさせるのに便利

Q&A