ダウンロード・データあります

パケットづくりではじめる ネットワーク入門

第53回 MACアドレス検索にハッシュを使う

坂井 弘亮

リスト1 前回作成した簡易スイッチング・ハブのプログラムではMACアドレスの検索処理は線形検索で行っている
MACアドレスの検索処理部分のみ抜粋

```
029:struct macaddr_entry *macaddr_entry_search(char
                                          macaddr[])
030:{
031:  struct macaddr_entry *entry;
032:  for (entry = macaddr_table; entry; entry =
                                       entry->next) {
033:    if (!memcmp(macaddr, &entry->macaddr,
                                    ETHER_ADDR_LEN))
034:      return entry;
035:  }
036:  return NULL;
037:}
```

● 今回行うこと…MACアドレス検索の高速化

前回は，簡易的なソフトウェアL2スイッチを作っていく手始めに，ソフトウェア的に動作する「簡易スイッチング・ハブ」を作成しました．

一般的にスイッチング・ハブは，接続されたノードのMACアドレスを学習テーブルに保持し，フレームをそのノードが存在しているポートに転送します．しかしMACアドレスには規則性がないため，転送先の決定のためにはMACアドレスをキーとした，学習テーブルの検索処理が必要となります．

前回作成した簡易スイッチング・ハブは，この検索を単純な線形検索で行っていたため，接続されたノードの数が増加した際のスケーラビリティがないものでした．今回はMACアドレスの学習テーブルの検索処理とデータ構造を修正し，ノードの数が増えてもスケールできるものにします．

MACアドレス・テーブルの検索処理

今回，MACアドレスの学習テーブルの検索処理のために実装した処理は，以下の2つです．

1. 検索処理のスケーラビリティを保つための工夫
2. 検索自体を高速化するための工夫

1と2は，高速化のための対策として同様に見えがちですが，異なる目的があります．

1はノード数が増加しても耐えられる，もしくは対応できるようにするための工夫です．

対して2は，そもそもの検索処理を単純に高速化するための工夫です．

● 前回実装した線形検索処理

前回作成した簡易スイッチング・ハブでは，MACアドレスの検索処理は**リスト1**のように実装しました．リンク・リスト構造を利用した線形検索になっています．

これはノード数が増加した際のスケーラビリティがありません．例えば10000ノードが接続された際に，最悪の場合にはフレームを受信するたびに10000回のループが行われてしまいます．転送の遅延時間を保証しなければならないようなときには，そのために高クロックで動作させるなどの工夫が必要です．これは装置のコストにそのまま直結してしまうでしょう．

● ハッシュ・テーブルを利用する

このようなときによく利用されるデータ構造が，ハッシュ・テーブルです．

ハッシュ・テーブルを利用する際にはまず，検索するキー（この場合はMACアドレス）から何らかの乱数性の高い値を計算しそれをインデックスとします．さらにテーブルはインデックスの最大値の数で分離し，エントリの登録時には，インデックスに対応するテーブルに登録するようにします．

これにより検索時には，インデックスに対応するテーブルのみ検索すればよいことになります．インデックスの最大値を増やすことでテーブルが広がりエントリが分散されるため，検索しなければならないループ回数を減らすことができます．

ハッシュ・テーブルには，

- 既存の処理に対して大きなデータ構造の変更なく適用しやすい（多くの場合，既存のデータ構造の先頭にハッシュ・テーブルを追加するだけで適用できる）
- さらにインデックスの上限を増やすことでエントリの総数に応じた調整がしやすい

本連載のプログラムのソースコードは以下の筆者のホームページからダウンロードできます．
ライセンスはKL-01というもので，組み込み機器などでも自由に利用できます．
http://kozos.jp/books/interface/ethernet2/