■3群（コンピュータ -ソフトウェア）－ 3編 ネットワーク層

3章 IGP（Interior Gateway Protocol）

■3群 － 3編 － 3章

3-1 RIP

（執筆者：甲藤二郎）[2009年8月 受領]

RIP（Routing Information Protocol）はIGP（Interior Gateway Protocol）して広く使用されている経路制御プロトコルの一つである．RFC 1058 [1] でRIP Version 1が，RFC 2453 [2] でRIP Version 2（RIP-1）が定められ，現在は主にRIP Version 2（RIP-2）が使用されている．

IP網では，経路の選択はホップバイホップで行われる．パケットの中継ノードであるルータには経路表（Routing Table）が保持され，受け取ったパケットの宛先アドレスに応じて次の中継ノードを決定し，転送する．この動作を各々の中継ノードが繰り返すことで，パケットは最終的な目的地まで配送される．

ネットワークが小規模な場合は，各ノードの経路表は手動で設定してもよい．しかし，ネットワークの規模が大きくなると経路表の手動管理は困難になり，各ノードが定期的に経路情報を交換し，経路表を動的に更新する仕組みが必要になる．前者を静的経路制御（Static Routing）と呼び，後者を動的経路制御（Dynamic Routing）と呼んでいる．現在IP網で使用されている動的経路制御方式は以下の三種類に分類される．

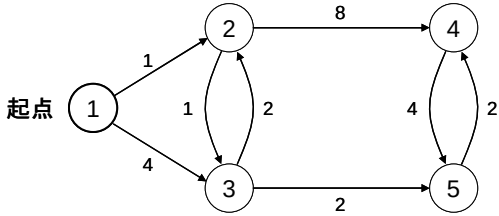
- 距離ベクトル経路制御（Distance Vector Routing）
- リンク状態経路制御（Link State Routing）
- パスベクトル経路制御（Path Vector Routing）

RIPはこの中の距離ベクトル経路制御に分類される．

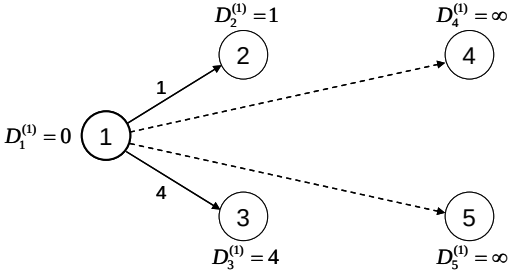
3-1-1 Bellman-Fordアルゴリズム

距離ベクトル経路制御では，各ノードが宛先ノードとそこに至る距離を互いに情報交換し，その情報を元に各ノードでは宛先ノードまでの最短経路を計算・更新する．また，各ノードは，宛先，宛先までの距離（コスト），転送リンク（転送先ノード）の情報を，ベクトルの形式で経路表に保持する．距離のメトリックとしては，通常，ホップ数を用いる（昔のARPANETでは遅延時間を用いていた）．この距離ベクトル経路制御において最短経路を計算・更新するアルゴリズムは，Bellman-Fordアルゴリズムと呼ばれている [3]．

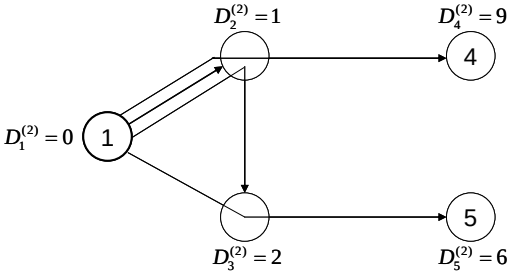
図3・1(a) に示すようなノード1を起点とするネットワーク（有向グラフ）において，ノード1からその他のノード2,3,4,5への最短経路を求める問題を考える．このとき，ノード i からノード j に向かうリンク (i, j) の距離（コスト）を d_{ij} とし，図中には各リンクの距離の値を示している．またリンク (i, j) が存在しない場合は $d_{ij}=\infty$ とする．Bellman-Fordアルゴリズムでは，初めに高々一つのリンクを含む場合の最短経路を求め，次に高々二つのリンクを含む場合の最短経路を求め，以下同様に最短経路を求めていく．よって，リンクの数が k の条件下で求めた最短経路は，リンクの数が k 以下の条件下における最短経路を表すことになる．この「リンクの数が k の条件下で求めた最短経路」はまた，「k 回の情報交換の後に求められた最短経路」と言い換えることができる．



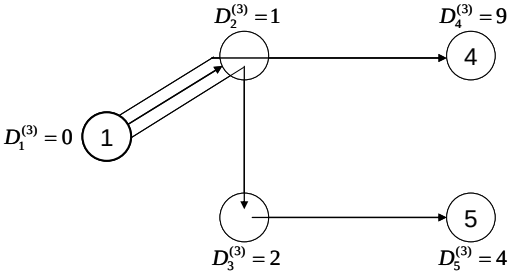
(a)　ネットワーク構成とリンクコスト



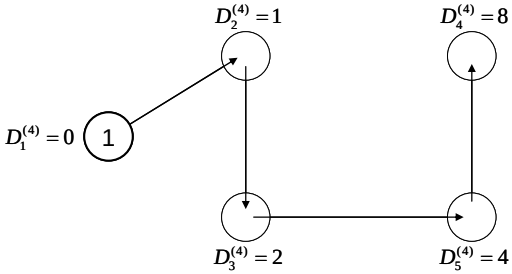
(b) 高々一つのリンクを用いた最短経路（k=1）



(c) 高々二つのリンクを用いた最短経路（k=2）



(d) 高々三つのリンクを用いた最短経路（k=3）



(e) 最終的な最短経路の木（k=4）

図 3・1　Bellman-Ford アルゴリズムの反復動作

$D_i^{(k)}$を，ノード 1 からノード i までのリンクの数が k 以下の場合の最短距離とする．ノード 1 に関しては，すべての k に対して $D_1^{(k)} = 0$ とする．また，ノード i に隣接するすべてのノードの集合を V_i と表す．このとき，Bellman-Ford アルゴリズムは以下のように表すことができる．

初期条件： $k=0$ とし，すべての $i \neq 1$ に対して $D_i^{(0)} = \infty$

ステップ 1： すべての $i \neq 1$ に対して $D_i^{(k+1)} = \min_{j \in V_i} \left(D_j^{(k)} + d_{ji} \right)$

ステップ 2： $D_i^{(k+1)} = D_i^{(k)}$が成立すれば終了．終了しなければステップ 1 に戻る．

図 3・1(b)～(d)は，それぞれ k=1, 2, 3 の場合に求められる最短距離 $D_i^{(k)}$とそれを与える最短経路を示している．1 回の情報交換後（k=1 の場合）は隣接ノードの情報しかわからないが，2 回，3 回と情報交換を繰り返すにつれて離れたノードの情報が反映され，最短距離が改善されて行く．そして，**図 3・1(e)**は，4 回の情報交換後（k=4 の場合）に収束した，ノード 1 からノード i に至る最終的な最短距離と最短経路を示している．

Bellman-Ford アルゴリズムは，トポロジの変更が無ければ，正しい最短経路に収束することが証明されている．また，ノード間の情報交換を分散非同期に実行した場合（RIP の場合）でも，タイムアウト等によって経路の有効性が保証されていれば，初期条件にかかわらず正しい最短経路に収束することも示されている．ただし，ノードやリンクに故障が発生し，経路情報が変化すると，経路の更新が緩やかにしか実行されないという，実用上は致命的な問題点がある．

また，ネットワークのノード数を N とすると，Bellman-Ford アルゴリズムは最大 N-1 回の繰返しで停止し，それぞれの繰返しにおいて，最大 N-1 個のノードに関する処理が行われる．さらに，各ノードごとの最小化を，少なくとも N-1 個の場合について行う必要がある．このことから，Bellman-Ford アルゴリズムの計算時間の総計は $O(N^3)$で見積もられる．

3-1-2 無限カウント問題

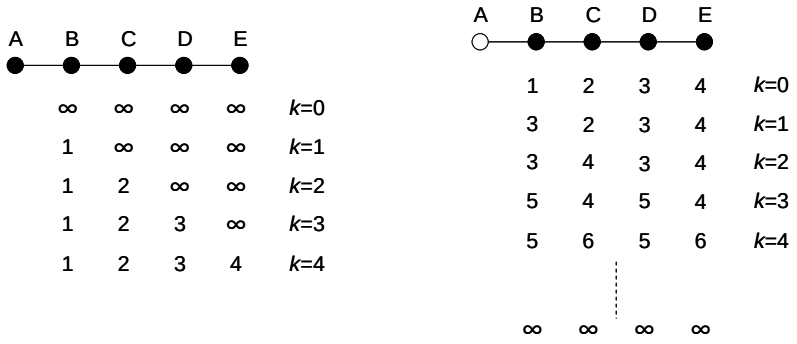
Bellman-Ford アルゴリズムには，正しい最短経路に収束するまでの時間がかかる，という問題がある [3, 4]．「良い」経路の出現時の経路更新は迅速に行われるが，ノードやリンクに障

害が発生した場合など，「悪い」経路が出現した際の経路更新の反応が遅い．

図 3・2 は，複数のノードが直線的に接続されたネットワークにおいて，ノード A が新たに利用可能になった場合（**図 3・2(a)**），および，ノード A がダウンもしくはノード A と B を結ぶリンクに障害が発生した場合（**図 3・2(b)**）に，ノード A からノード B,C,D,E に至る最短距離がどのように更新されるかを示している．

図 3・2(a) において，ノード A とその他のノードとの距離の初期値は∞である（k=0）．最初の情報交換後（k=1），ノード B はノード A の存在と距離 1 で到達可能なことを確認し，自身の経路表にノード A に関する情報を追加する．その他のノードは，ノード A はまだダウンしている（あるいは存在していない）と考えている．次の情報交換で（k=2），ノード C はノード B 経由の距離 2 でノード A に到達可能なことを知り，自身の経路表にノード A に関する情報を追加する．以降の情報交換のたびに，ノード D，ノード E もノード A の存在を知り，それぞれの経路表にノード A に関する情報を追加し，高々4 回の情報交換によって最短経路の更新が完了する．

図 3・2(b) は，当初はすべてのノードが問題なく通信していたものとする（k=0）．ここで突然ノード A が通信不能になり，最初の情報交換で（k=1），ノード B はノード A から情報が送られてこないことを知る．しかし，ノード C は，ノード A に距離 2 で到達可能である旨，ノード B に伝えてくる．ノード B は，ノード C の伝える経路が，ノード B を経由するものなのか，独立した経路によるものなのか，知ることができない．このため，ノード B はノード C からの情報を採用し，自身からノード A への最短距離を 3 として，経路表を更新する．次の情報交換では（k=2），今度はノード C が，隣接ノードからノード A までの最短距離が 3 になったことを知る．そこで，ノード C は，自身からノード A への最短距離を 4 として，経路表を更新する．この操作を繰り返しながら，各ノードからノード A への最短距離は，徐々に徐々に無限大に近付いていく．



(a) ノード A が新たに利用可能になった場合　　(b) ノード A が突然通信不能になった場合

図 3・2　Bellman-Ford アルゴリズムの無限カウント問題

このように，Bellman-Ford アルゴリズムでは，「良い」経路は迅速に伝播されるが，「悪い」経路は遅々としか伝播されない．原理的に，各ノードは，隣接ノードが保持する距離の最小

値よりも 1 だけ大きい値しか持つことがない．現実的には，無限大を最長経路長＋1 などの有限値で打ち切ればよいが，問題を解決しているわけではない．この問題は，無限カウント（count-to-infinity）問題として知られている．

3-1-3 水平分割とトリガ更新

無限カウント問題に対する対策は数多く提案されているが，問題を完全に解決するものは無い．これらの中から，RIP では，「毒戻しを伴う水平分割」（split horizon with poisoned reverse）と「トリガ更新」（triggered updates）と呼ばれる対応策を使用している．

図 3・2(b) において，ノード C は，隣接するノード B と D に対して，自身のノード A への距離情報を報告している．これに対して毒戻しを伴う水平分割では，ノード C は，ノード D に対してはノード A への距離情報を報告するが，ノード B に対してはノード A への距離を無限大であると報告する．これはすなわち，経路を学習した隣接ノード（この場合は B）を記憶しておき，そのノードに対しては距離情報を無限大にして返す（毒戻し），という戦略である．ノード D と E も同様の動作を実行する．これによって，ノード A に関する障害発生後，情報交換のたびに 1 ホップずつ「悪い」経路が伝播され，各ノードの経路表は迅速に更新されることになる．なお，当該ノードに無限大の距離情報を返すのではなく，距離情報を返さない方式も知られており，「単純水平分割」（simple split horizon）と呼ばれている．一般的に，毒戻しを行う方が収束時間を早められるが，オーバーヘッドは若干増加する．ルータ要件に関する RFC 1812[5] では，RIP における水平分割の実装を義務付け，毒戻しの実装を推奨している．

水平分割は多くの場合に有効に機能するが，うまく機能しない場合もある．**図 3・3** はその具体例を示したものであり，4 個のノードを接続するリンクの距離はそれぞれ 1 であるとする．まず，C と D をつなぐリンクがダウンしたものとする．水平分割を用いる場合，A と B は，D への距離は無限大との情報を C に報告する（毒返し）．C は D に到達不能であることを検知し，A と B に報告する．一方で，B は A に対して D まで距離 2 で到達可能との情報を報告し，A もまた B に対して同様の情報を報告する．よって，ノード A，B は，それぞれ互いを通じてノード D まで距離 3 で到達可能と考えてしまう．この更新は情報交換の度に繰り返され，両者から D までの距離は 1 ずつ無限に増加することになり，これは無限カウント問題にほかならない．

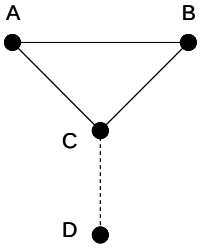


図 3・3　水平分割が有効に機能しないネットワークの例

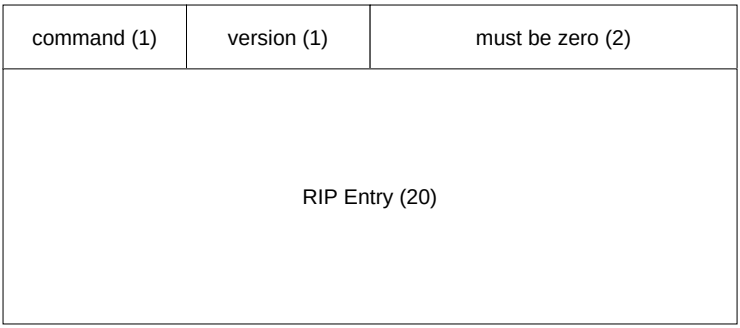
RIP では，通常それぞれのノードは 30 秒ごとに経路情報を隣接ノードに送信し，また，隣

接ノードから 180 秒間何も受信しない場合は，当該する経路は無効であると判断する [1,2,5]．また，これまでの手法は，経路情報をいつ隣接ノードに報告し，経路更新をいつ行うかを明確に記したものではなかった．これに対してトリガ更新は，あるノードがネットワークの変化を検出すると，即座に変更された経路情報を隣接ノードに報告し，隣接ノードも順次変化を伝播し，経路の更新を実行するものである．これによって経路の迅速な変更が期待され，無限カウント問題の収束時間の短縮にも貢献する．ただし，依然として無限カウント問題を解決するものではない．

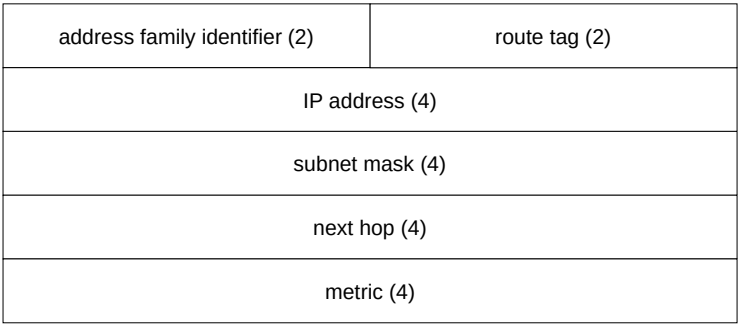
3-1-4 プロトコル仕様

RIP は UDP ベースのプロトコルである．RIP を使用する各ノードは，UDP ポート 520 番上でデータグラムの送受信を行うプロセスをもつ．前述のように，30 秒ごとに経路情報を隣接ノードに送信し（経路表の更新），180 秒間何も応答がない場合は経路が無効になったと判断する（タイムアウト）．

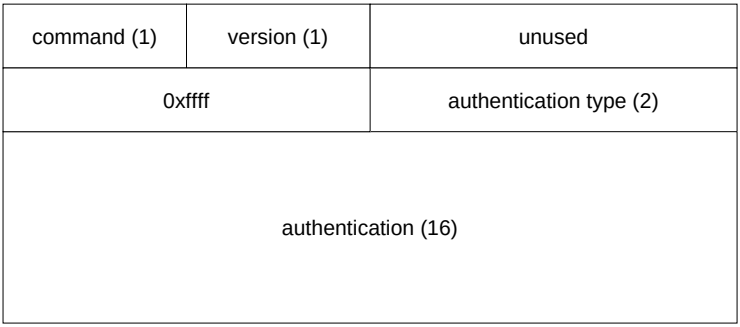
図 3・4 は RIP メッセージのパケットフォーマットを示している．**図 3・4 (a)** は RIP-1/RIP-2 共通のパケットフォーマットである．括弧内の値は，各フィールドのバイトサイズを表す．コマンド（command）とバージョン（version）は RIP-1/RIP-2 共通のフィールドであり，二つを合わせて RIP ヘッダと呼ぶ．コマンドは要求（request）と応答（response）が用意されており，フィールド値としてはそれぞれ 1，2 を取る．要求コマンドは，経路表のすべてまたは一部の送信を要求するのに使用する．応答コマンドは，経路表のすべてまたは一部を含むメッセージであり，要求に対する応答である場合と，要求とは無関係に生成した経路更新情報の場合がある．バージョンは RIP-1 か RIP-2 かを区別するフィールドであり，それぞれ 1，2 の値を取る．RIP エントリはバージョンごとに異なる 20 バイトのフィールドであり，RIP-2 の場合の具体的な構成を **図 3・4 (b)** に示している．この RIP エントリは，一つの RIP メッセージ内に最大 25 個存在してもよい．



(a) RIP-1/RIP-2 共通パケットフォーマット



(b) RIP-2 エントリ



(c) RIP-2 認証メッセージ

図 3・4　RIP のパケットフォーマット

図 3・4 (b) において，アドレスファミリ識別子（address family identifier）はアドレスのタイプを通知する．通常は AF_INET（値は 2）のみがサポートされる．経路タグ（route tag）は，BGP と RIP の連携などを想定し，経路とともに伝播すべき経路の属性情報を通知するために用意されている．IP アドレス（IP address）には宛先ノードの IPv4 アドレスを記載する．サブネットマスク（subnet mask）には，IP アドレスに適用されるサブネットマスクを記載する．このフィールドが 0 の場合は，このエントリにはサブネットマスクが含まれていないことを示す．次ホップ（next hop）には，記載した経路エントリに関して，指定した宛先ノードにパケットを転送するための次ノードの IP アドレスを記載する．次ホップに指定された IP アドレスは，サブネット上で直接到達可能なものでなければならない．このフィールドが 0.0.0.0 の場合は，パケットは，このメッセージの送信元を経由すべきであることを示す．最後にメトリック（metric）は，1～15 までの値を含み，宛先ノードまでの距離の値を示す．また，メトリックの値が 16 の場合は無限大に対応し，宛先ノードが到達不可であることを示す．

図 3・4 (c) は，特に認証のために定義された RIP メッセージのパケットフォーマットを示す．アドレスファミリ識別子の位置に記載された 0xffff が RIP-2 認証メッセージとしての識別子になっており，エントリの残りの部分は認証データの通知に使用される．RFC 2453 に記載さ

れている唯一の認証タイプはパスワードであり, 認証タイプの値は 2 が割り当てられている．残りの 16 バイトの認証フィールドは平文のパスワードを含み, 左詰めで余った部分には 0x00 がパディングされる．0xffff から認証フィールドまでのバイト数は 20 バイトで RIP エントリのサイズに等しく, RIP メッセージの残りの部分には最大 24 個の通常の RIP エントリを含めることができる．

■参考文献

1) C. Hedrick: “Routing Information Protocol”, IETF RFC 1058, Jun. (1988).
2) G. Malkin: “RIP Version 2”, IETF RFC 2453, Nov. (1998).
3) D. Bertsekas and R. Gallager (八星監訳): “Data Networks”, オーム社, (1990).
4) A.S. Tanenbaum (水野他訳): “Computer Networks”, ピアソンエデュケーション, (1999).
5) R. Baker: “Requirements for IP Version 4 Routers,” IETF RFC 1812, Jun. (1995).