■3 群（コンピュータネットワーク）- 4 編（トランスポートサービス）

# 6 章 その他のトランスポートサービス

## ■概要■

### 【本章の構成】

■3 群 - 4 編 - 6 章

# 6-1 SCTP

執筆中

■3 群 - 4 編 - 6 章

## 6-2 XCP（eXplicit Control Protocol）

（執筆者：大崎博之）[2013 年 6 月 受領]

XCP（eXplicit Control Protocol）は MIT のディナ・カタビ（Dina Katabi）と ICIR のマーク・ハンドレー（Mark Handley）が開発した輻輳（ふくそう）制御プロトコルである [1].

XCP では，エンドホスト（送信側ホストおよび受信側ホスト）と，それらのエンドホスト間の経路上のルータが協調して輻輳制御を行うことにより，広帯域・高遅延ネットワークにおいて高いスループットを実現する．通常の TCP は，エンド – エンド原則により，エンドホスト間のみで輻輳制御を行う．TCP のエンドホストは，ネットワーク中のルータの輻輳状況をパケット棄却の有無から推測し，自身の転送レートを調整する．一方 XCP は，経路上のルータが輻輳の状況をエンドホストに明示的に通知する．XCP のエンドホストは，経路上のルータから通知された輻輳状況をもとに転送レートを調整する．

XCP の動作の概要は以下の通りである．

XCP はウィンドウ型のフロー制御の一種であるが，送信側ホストのウィンドウサイズ（ラウンドトリップ時間内に転送できるパケット数）の増減量をルータが計算し，明示的なフィードバックとして送信側ホストへ通知する．パケットのヘッダ（輻輳ヘッダ）には，送信側ホストのウィンドウサイズ，送信側ホストが計測したラウンドトリップ時間，ウィンドウ制御に使用するウィンドウサイズの増減量（フィードバック値）が格納されている．

まず，送信側ホストは，パケット送信時に，送信するパケットの輻輳ヘッダに，計測したラウンドトリップ時間，現在のウィンドウサイズ，フィードバック値の初期値（送信側ホストが期待するウィンドウサイズの増加量）を書き込む．これによって，送信側ホストの状態を XCP ルータに通知する．

次に，パケットが XCP ルータに到着すると，XCP ルータは，輻輳ヘッダに記録されている情報をもとに適切なフィードバック値を計算する．XCP ルータは，パケットの輻輳ヘッダに格納されているフィードバック値が，XCP ルータが計算したフィードバック値よりも大きければ，輻輳ヘッダ中のフィードバック値を XCP ルータが計算したフィードバック値に更新する．その後，パケットを下流のノードに転送する．

パケットが受信側ホストに到着すると，受信側ホストは ACK （ACKnowledgement）パケットを送信側ホストに返送する．このとき，データパケットの輻輳ヘッダを ACK パケットの輻輳ヘッダに複製する．これにより，XCP ルータの輻輳情報を（受信側ホスト経由で）送信側ホストに通知することが可能となる．

最後に，送信側ホストが ACK パケットを受信すると，ACK パケットの輻輳ヘッダに格納されているフィードバック値を現在のウィンドウサイズに加算することによって，送信側ホストの転送レートを調整する．

経路上のルータが，明示的にエンドシステムに輻輳状況を通知するメカニズムは古くから存在する．例えば，ICMP（Internet Control Management Protocol）におけるソースクエンチ（Source Quench）メッセージを用いることにより，経路上のルータから送信側ホストに対して，明示的に転送レートの抑制を要求することができる．同様に，TCP ヘッダ中の ECN（Explicit Congestion Notification）フィールドを用いることにより，経路上のルータから送信側ホストに対して，明示的に転送レートの抑制を要求することができる．

XCP の特徴は，(1) ルータにおける公平性の実現と利用率の最大化を，それぞれ独立した二種類のコントローラで実現すること，(2) ルータにフロー単位の情報を管理させることなく，フロー単位の公平性を実現すること，の 2 点にある．

第 1 に，XCP は，ルータにおける公平性の実現と利用率の最大化を，それぞれ独立した二種類のコントローラ（公平性コントローラ及び効率性コントローラ）で実現している．一般に，輻輳制御プロトコルでは，ルータに収容されている複数のフロー間の公平性を維持すると同時に，ルータの出力リンクの利用率を最大化することが求められる．従来の輻輳制御プロトコルは，フロー間の公平性の実現とリンク利用率の最大化を単一のコントローラで制御していた．一方，XCP では，フロー間の公平性を実現する「公平性コントローラ」と，リンク利用率の最大化を実現する「効率性コントローラ」が独立している．このような独立した制御により，迅速な効率性の制御が可能となり，広帯域・高遅延ネットワークにおいても高いスループットを実現する．

第 2 に，XCP は，ルータにフロー単位の情報を管理させることなく，フロー単位の公平性を実現している．ルータにおいてフロー単位の公平性を実現するためには，通常，ルータがフロー単位の情報（例えば，フローを識別するための情報や，該当フローの利用帯域など）を管理する必要がある．しかし，ルータがフロー単位の情報を管理する場合，ルータが収容するフロー数が増えるにつれて，ルータが管理すべき情報量も増大する．このため，ルータがフロー単位の情報を管理する場合には，フロー数に対するスケーラビリティを実現できない．XCP では，フロー単位の情報は，ルータが管理するのではなく，パケットの輻輳ヘッダに格納するというアイデアによって，フロー数に関するスケーラビリティを実現している．

XCP は，もともと TCP のための輻輳制御プロトコルとして考案されたが，TCP 以外のトランスポートプロトコルへの適用も可能である．2003～2006 年の間に，BSD UNIX への実装や，XCP プロトコルのインターネットドラフト作成が行われた [2]．

**■参考文献**

1) D. Katabi, M. Handley, and C. Rohrs, “Congestion control for high bandwidth-delay product networks,” in Proceedings of ACM SIGCOMM 2002, pp. 89-102, Aug. 2002.
2) A. Falk, Y. Pryadkin, and D. Katabi, “Specificaton for the explicit control protocol(XCP),” IETF Internet Draft: draft-falk-xcp-spec-02.txt, Nov. 2006.

■3 群 - 4 編 - 6 章

## 6-3 DCCP（Datagram Congestion Control Protocol）

（執筆者：大崎博之）[2013 年 6 月 受領]

DCCP（Datagram Congestion Control Protocol）は，リアルタイム系アプリケーション向けに設計された，データグラム転送のためのトランスポートプロトコルである [1]．DCCP では，転送されるデータの信頼性は保証されない．つまり，DCCP の送信側ホストは，ネットワーク中でパケットが廃棄されても，パケットの再送を行わない．

DCCP では，DCCP を利用するアプリケーションが，輻輳 (ふくそう) 制御プロファイルを選択することによって，使用する輻輳制御のアルゴリズムを選択できる．DCCP によってサポートされている輻輳制御プロファイルには，CCID （Congestion Control IDentifier）と呼ばれる識別子が割り当てられている．DCCP の送信側ホスト及び受信側ホストは，コネクション確立時に，それぞれが利用可能な輻輳制御プロファイルの CCID を交換し，転送中に使用する輻輳制御プロファイルを決定する．現在，DCCP の輻輳制御プロファイルとして，CCID2（TCP 型輻輳制御プロファイル）[2] 及び CCID3（TFRC 型輻輳制御プロファイル）[3] が提案されている．

DCCP の TCP 型輻輳制御プロファイルでは，TCP と同様に AIMD（Additive Increase and Multiplicative Decrease）型のウィンドウ制御を行う．AIMD 型のウィンドウ制御では，送信側ホストがネットワークの輻輳を検出するまで，ウィンドウサイズ（ラウンドトリップ時間内に転送できるパケット数）を加算的に増加させる．ネットワークの輻輳を検出すると，送信側ホストは，ウィンドウサイズを乗算的に減少させる．そのため，TCP 型輻輳制御プロファイルを用いた DCCP のパケット送出レートは，ラウンドトリップ時間程度のタイムスケールで変動することになる．このため，TCP 型輻輳制御プロファイルは，例えば，受信側ホストである程度のデータをバッファリングするような，ストリーミングアプリケーションなどに適している．

DCCP の TCP 型輻輳制御プロファイルは，以下の 4 点において TCP の輻輳制御と異なっている．

1 番目に，DCCP では，ACK 比率（ACK Ratio）という機構を用いて，受信側ホストから送信側ホストへの ACK パケットに対しても，輻輳制御が行われる．受信側ホストが，送信側ホストに対してどの程度の ACK パケットを返送するかは，ACK 比率によって決定される．具体的には，ACK 比率が $R$ のとき，DCCP の受信側ホストは，$R$ 個のデータパケットを送信側ホストからが受信するごとに，ACK パケットを一つ送信側ホストに返送する．

2 番目に，DCCP は信頼性のないトランスポート層通信プロトコルであるため，DCCP の送信側ホストはパケットの再送を行わない．TCP の輻輳制御機構では，パケット廃棄が発生した場合，そのパケットが再送パケットかどうかを識別するが，DCCP ではそのような処理は行われない．

3 番目に，DCCP では，パケット廃棄が発生した原因を，受信側ホストから送信側ホストに通知することができる．これは，受信側ホストから送信側ホストへの ACK パケット中に含まれる，データ廃棄オプションによって実現されている．例えば，パケット廃棄が，伝送路のビット誤りによって発生したものか，もしくは受信側ホストのバッファあふれにより発生したものなのかを，受信側ホストから送信側ホストへ通知することができる．

4 番目に，DCCP は，フロー制御を行わない．つまり，AIMD 型のウィンドウ制御だけを行い，TCP の輻輳制御で行われているような，広告ウィンドウを用いた受信側ホストのバッファ管理は行わない．

一方，DCCP の TFRC 型輻輳制御プロファイルでは，TCP 型輻輳制御プロファイルで生じるパケット送出レートの変動を抑えながら，TCP の輻輳制御と公平に帯域を共有する（TCP-friendly となる）ように輻輳制御を行う．TFRC 型輻輳制御プロファイルでは，受信側ホストが中心となって輻輳制御を行う．つまり，TFRC 型輻輳制御プロファイルを用いた DCCP では，受信側ホストがネットワークの輻輳を検出し，これを送信側ホストに通知する．送信側ホストは，受信側ホストから通知された輻輳情報（パケット棄却イベント率）に基づき，送信側ホストからのパケット送出レートを調整する．TFRC 型輻輳制御プロファイルを用いた DCCP では，送信側ホストからのパケット送出レートが大きく変動しないため，例えば，受信側ホストであまりバッファリングを行わないような，ストリーミングアプリケーションなどに適している．

DCCP の TFRC 型輻輳制御プロファイルは，パケット棄却が発生した原因を，受信側ホストから送信側ホストに通知することができるという点で，TFRC の輻輳制御と異なっている．これは，TCP 型輻輳制御プロファイルと同様に，受信側ホストから送信側ホストへの ACK パケット中に含まれる，データ廃棄オプションによって実現されている．

**■参考文献**

1) E. Kohler, M. Handley, and S. Floyd, “Datagram congestion control protocol (DCCP),” Request for Comments (RFC) 4340, Mar. 2006.
2) S. Floyd and E. Kohler, “Profile for datagram congestion control protocol (DCCP) congestion control id 2: TCP-like congestion control,” Request for Comments (RFC) 4341, Mar. 2006.
3) S. Floyd, E. Kohler, and J. Padhye, “Profile for datagram congestion control protocol (DCCP) congestion control id 3: TCP-friendly rate control (TFRC),” Request for Comments (RFC) 4342, Mar. 2006.

■3 群 - 4 編 - 6 章

# 6-4 UDP-Lite

（執筆者：西田佳史）[2011 年 1 月 受領]

## 6-4-1 UDP-Lite の概要

UDP-Lite（Lightweight User Datagram Protocol）は，軽量版の UDP として 2004 年に標準化された新しいトランスポートプロトコルである．UDP-Lite の仕様は RFC3828[1)]で規定されている．UDP-Lite は基本的に UDP と同等の機能を提供するが，異なるプロトコルとして 136 のプロトコル番号が割り当てられている（UDP は 17）．

UDP-Lite と UDP の違いは，チェックサム機能にある．UDP では IPv4 を利用する場合，UDP セグメント全体に対するチェックサムを検証するか，チェックサムを全く検証しないかのどちらかを選択する必要がある．また IPv6 を利用する場合は，UDP セグメント全体に対するチェックサム検証が必須となっている．これに対して，UDP-Lite ではセグメント中の指定した部分に対してのみチェックサムの検証を行うことができる．

UDP-Lite が作られた背景には以下の点があげられる．

アプリケーションの中には，パケットの内容の一部が破損していても，そのパケットを廃棄せずに受信した方がよいものもある．典型的な例としては，動画の転送において H.264 など，ある程度のデータの誤りを許容できるコーデックを利用する場合が考えられる．このような場合，トランスポート層でチェックサムを検証して一部が欠損したデータを破棄してしまうよりも，欠損したデータをそのままアプリケーション層に渡した方が効率が良い．

リンク層のプロトコルのなかには，パケットの一部により強力なデータ検証，補償（例えば FEC など）を行う機能を提供するものもある．トランスポート層においてペイロード中のどの部分が重要であるかを示すことにより，このようなリンク層プロトコルの機能を応用できる可能性がある．

## 6-4-2 UDP-Lite のパケットフォーマット

**図 6･1** に UDP-Lite のパケットフォーマットを示す．

UDP-Lite の UDP とほぼ同じフォーマットをもつが，UDP の Length フィールドに対応す

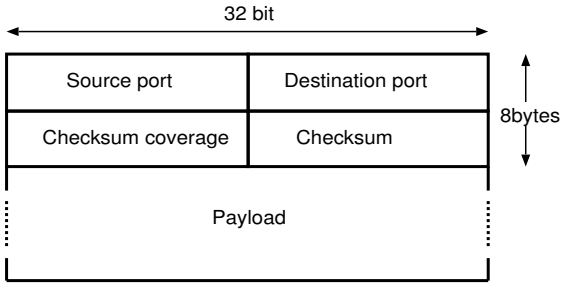


**図 6･1** UDP-Lite **のパケットフォーマット**

る部分が Checksum Coverage フィールドである点が異なる．Length フィールドを持たない UDP-Lite の場合，ペイロードの大きさは IP ヘッダ中の Total Length（IPv6 の場合 Payload Length）フィールドから計算することになる．

Checksum Coverage フィールドには，チェックサムのカバーする範囲の長さをバイト数で格納する．UDP-Lite のチェックサムのカバー範囲は，UDP-Lite ヘッダの先頭バイトから，Chesksum Coverage で指定された長さまでとなる．Cheksum Coverage フィールドに 0 をセットする場合は，セグメント全体がチェックサムのカバー範囲となる．また UDP-Lite では，UDP-Lite ヘッダは必ずチェックサムでカバーされなければならない．以上のことから，Checksum Coverage フィールドのとり得る値は，0 もしくは 8 以上となる．

UDP-Lite のチェックサム計算に利用される擬似ヘッダ（Pseudo Header）は，UDP と同じフォーマットであるが，Length フィールドの値を IP ヘッダ中のフィールドから計算する点が異なっている．

### 6-4-3 Linux 実装における UDP-Lite

UDP-Lite は Linux kernel 2.6.20 以降でサポートされている．UDP-Lite を利用するアプリケーションは，以下のように IPPROTO_UDPLITE を指定して socket を作成する．

```
int s = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDPLITE);
```

また，Checksum Coverage を指定する場合は，送信側のアプリケーションは setsockopt システムコールを用いて，以下のような設定を行う．

```
int cscov = 108;
setsockopt(s, SOL_UDPLITE, UDPLITE_SEND_CSCOV, &cscov, sizeof(int));
```

この操作により，100 バイトのアプケーションデータ（および 8 バイトの UDP-Lite ヘッダ）がチェックサムで保護されることとなる．この場合，100 バイト以降のアプリケーションデータに破損があったとしても，アプリケーションはパケットを受信できる．

UDP-Lite 利用する受信側アプリケーションは，setsockopt で特に設定を行う必要はないが，付加的な機能として，Checksum Coverage の最小値を設定することができる．この場合，受信側は以下のような設定を行う必要がある．

```
int cscov = 20;
setsockopt(s, SOL_UDPLITE, UDPLITE_RECV_CSCOV, &cscov, sizeof(int));
```

この操作により，アプリケーションが Checksum Coverage が 20 バイト未満のパケットを受信することはなくなる．

**■参考文献**

1) Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., Ed. and G. Fairhurst, Ed., “The User Datagram Protocol (UDP)-Lite Protocol”, RFC 3828, Jun. 2004.