情報通信ネットワーク入門

尾崎 博一 著

コロナ社

ま　え　が　き

情報通信ネットワークやインターネットに関する教科書，解説書は数多く出版されている。タイトルを見て本書を手にした人はそれらといったい何が違うのだろうと思うかもしれない。本書の特徴は，まず記述が平易でひとりで読み進められるということである。また，本書を読むために予備知識はほとんど必要でない。高校卒業程度の知識があればだれでも読み進めることができるはずである。しかし，基礎的な内容の解説だけに終始しているかというとそうでもない。他の入門的な教科書にはあまり書かれていない比較的高度なこと（しかし重要なこと），たとえば実際に使われているディジタル変調の方式（2.4.3 項，2.4.4 項），無線 LAN のフレームの詳細（4.10.2 項），TCP の輻輳制御の詳細（6.6.3 項），HTTP のメッセージの詳細（7.5.2 項）などが本書には含まれている。

本書の執筆は 2023 年 2 月に完了したが，可能な限りこの時点の最新の技術動向を反映するようにした。情報通信の技術は日進月歩であり，ある時点で最新の技術であったものがその後廃れて使われなくなることは普通に起こる。しかし，基本的なところは変わらないのである。本書では基本を丁寧に説明しつつ最新の技術動向を紹介している。

読者には情報関連の資格取得を目指す人も多いと思う。本書の各章末の演習問題には基本情報技術者試験，応用情報技術者試験，ネットワークスペシャリスト試験等の過去問題から多くを収録した。本書で学んだ内容がどのような設問として問われるかがわかるようになっている。各章の問および演習問題の解答例は巻末に示した。また，一部解答例の補足はコロナ社の本書紹介ページ†にアップした。

† https://www.coronasha.co.jp/np/isbn/9784339029369/

本書の構成は次のとおりである。第1章は序論であり，最低限知っておくべき知識を整理し情報通信ネットワークの概要を説明している。第2章はディジタル通信技術の解説である。高校数学で習う三角関数の基礎がわかっていれば理解できるはずである。第3章は通信プロトコルとはいったいどういうものであるかを説明する。第4章は私たちの最も身近にあるネットワーク（LAN）の解説である。第5章と第6章はプロトコルの中でも特に重要なTCP/IPの解説である。第5章のIPは基本的な内容が中心であるが，第6章のTCPではこの種の入門書としてはかなり詳しい内容まで紹介した。第7章はインターネットサービスとそのプロトコルの解説である。ユーザには直接見えないDHCPとDNSを最初に解説しているのは，この2つはインターネット通信を支える基本的なプロトコルだからである。第8章はブロードバンド通信と移動通信の解説である。普段何気なく使っているPCやスマートフォンの裏側でどんな技術が使われているかを理解していただきたい。第9章はセキュリティに関する解説である。心構えや人的対策などは省略して技術的な内容に特化して記述した。

本書を大学や専門学校の半期の授業で使用する場合，第1章，第3章，第8章をそれぞれ1回，その他の章をそれぞれ2回の講義とすれば，全15回で完了することができる。もっともこれに従う必要はなく担当される先生の裁量で適宜変えていただいて差し支えない。

本書では紙数の関係で割愛せざるを得なかった内容が多々あるが，読者は巻末の参考文献を参照し，さらに深い知識を身につけていただきたいと思う。

最後に本書の内容について有益なご教示をいただいた北海道情報大学の中島潤教授，廣奥暢准教授に深く感謝いたします。また，出版に際してお世話になったコロナ社の諸氏に感謝いたします。

2023年6月

著　　者

目　　次

第1章　序　　論

第2章 ディジタル通信技術

第3章　通信プロトコル

第4章　LAN

第 5 章　IP とルーティング

第 6 章　TCP と UDP

第7章 インターネットサービスとプロトコル

第8章 ブロードバンド通信と移動通信

第 9 章　ネットワークセキュリティ

トを受け入れられるようになる。これを「ウィンドウが開く」と表現する。ウィンドウが開くと受信側はそれを送信側に ACK で伝える決まりになっている。送信側はこれを受けてセグメントの送信を再開する。

6.6.3 輻輳制御

LAN，特に有線 LAN の内部ではネットワークの混雑が発生することは稀であるから，前節で述べたフロー制御さえあればセグメントを順調に転送することができる。しかし，インターネットを介して離れた端末にセグメントを転送する場合は，フロー制御だけでは不十分である。インターネットには多くの TCP コネクションが同時に存在し，同一の経路（伝送路）上にも多くのパケットが流れている。各端末がフロー制御だけに基づいて，パケットを次々に送り出すとネットワーク上で混雑や渋滞が発生することがある。間にあるルータの受信バッファがいっぱいになると，その後到着するパケットは破棄されてしまう。その場合，TCP は再送を行うから，混雑はますますひどくなる。ネットワーク上で発生するパケットの混雑のことを**輻輳**（ふくそう）（**congestion**）と呼ぶ。

TCP には輻輳を回避するための仕組みがあり，その動作を**輻輳制御**（congestion control）と呼ぶ。輻輳制御では輻輳がなるべく発生しないようにセグメントの送信数が調整される。フロー制御が相手端末の受信バッファの空き状況に応じて送信量を調整するのに対して，輻輳制御はネットワークの混雑状況に応じて送信量を調整するのである。

〔1〕 **輻輳の検出**　　さて，輻輳制御を行うためにはネットワークの混雑状況を知る必要があるが，TCP は直接的にそれを知ることができない。それは輻輳が TCP の下にあるネットワーク層で発生する現象だからである。したがって，TCP は何らかの方法で混雑状況を推定するより仕方がない。そこで TCP は 2 つの現象に着目する。ひとつは送信時に起動するタイマーのタイムアウト発生である。相手から一定時間内に ACK が返ってこない場合は，途中で混雑が発生しており，パケットが破棄されたと判断するのである。もうひとつは，

同じ ACK の重複受信（**重複 ACK**）である。TCP には順序違いのセグメントを受信した際には，正しいセグメントを要求する ACK をただちに返さなければならないという規則があることはすでに述べた（6.5.2 項参照）。相手が順序違いのパケットを受信するということは，ネットワーク内で発生している混雑のためにパケットが途中で破棄されたと判断するのである。TCP は重複 ACK を 3 回受信すると回復動作に入る。重複 ACK を 3 回受信するということは，最初に受け取った（正常な）ACK を含めると全部で 4 回同じ ACK を受信することになる。

〔2〕 **スロースタートと輻輳回避**　輻輳の検出は以上のように行うが，送信量の調整は前節で述べた送信ウィンドウ（スライディングウィンドウ）の中にさらに**輻輳ウィンドウ**（congestion window）という小さなウィンドウを作って行う。輻輳ウィンドウは連続して送信できるセグメント数にさらに枠をはめる働きをする。

コネクション確立直後の輻輳ウィンドウの初期値は基本的に 1 セグメントである。したがって，最初の送信は 1 セグメントだけ行い，これに ACK が返るとウィンドウをスライドさせて輻輳ウィンドウを 1 セグメント分増やす。つまり，2 セグメント連続して送信できるようになる。2 セグメント送信して 2 つ ACK が返ると今度はウィンドウサイズを 4 セグメントに増やす。つまり，ACK がひとつ返るごとに輻輳ウィンドウを 1 セグメント分増加させるのである。この動作を続けると輻輳ウィンドウは 1 → 2 → 4 → 8 → 16 →…のように指数関数的に増加し，一度に連続して送信するセグメントが増えていく。この動作を**スロースタート**（slow start）という。送信セグメント数が急速に増えるためスローとは奇妙な名前にも思われるが，「最初は 1 セグメントからゆっくり始める」という意味でスロースタートと呼んでいる。以上のように輻輳ウィンドウは最初，急速に成長するが，あらかじめ決められた大きさに到達するとスロースタートをやめて次の動作に移る。この閾値のことを**スロースタート閾値**（slow start threshold：**ssthresh**）という。

スロースタートの次のフェーズ（動作）は**輻輳回避**（congestion avoidance）と呼ばれる。輻輳回避では輻輳ウィンドウを少しずつ大きくしていく。輻輳ウィンドウ内のすべてのセグメントは連続的に一挙に送信される。この時にACK待ちのタイマーはひとつだけ起動する（セグメントごとではない）。そしてRTT後にこれらに対するACKがやはり連続的に戻ってくる。これらのACKによりウィンドウ全体が大きくスライドし，次のセグメント群を送信できるようになる。輻輳回避のフェーズでは，送信した輻輳ウィンドウ内のセグメントに対するすべてのACKに対して輻輳ウィンドウを約1セグメント増やす。たとえば，輻輳ウィンドウ内の10セグメントを送信し，10個のACKが戻ってくると輻輳ウィンドウを11セグメントに拡大するのである。ただし，すべてのACKが戻ってきた時点で1セグメント増やすのではなく，そういう割合になるようにひとつひとつのACKに対してバイト単位で少しずつ増やしていくという方法を取る。輻輳回避では，輻輳ウィンドウの増加はスロースタートに比べてはるかに緩やかなものとなる。

輻輳回避フェーズにおける輻輳ウィンドウサイズの増加には限界がある。それは相手端末から通知されるウィンドウサイズである。相手端末から通知されるウィンドウサイズを特に**広告ウィンドウサイズ**（advertised window size）と呼ぶ。広告ウィンドウサイズを超えてセグメントを送信しないことは前項のフロー制御で述べた。**図6.8**はスロースタートと輻輳回避における輻輳ウィンド

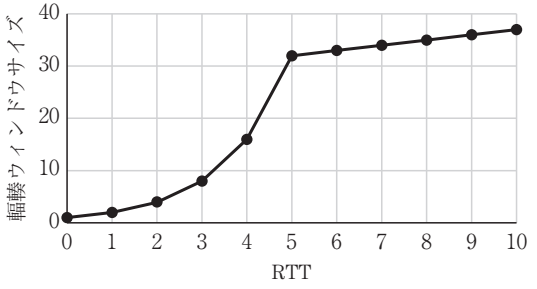
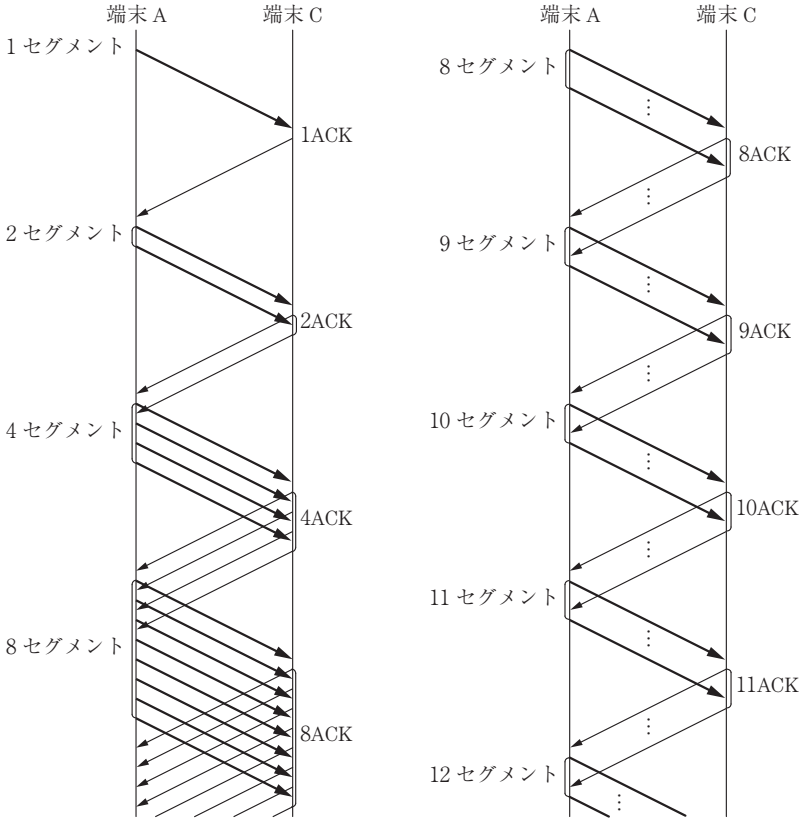


図6.8　スロースタートと輻輳回避における輻輳ウィンドウサイズ

ウサイズの変化を示している。この例では ssthresh を 32 MSS としている。スロースタートで始まり，5×RTT の時点で輻輳回避に切り替わっている。

図 6.9 はスロースタートと輻輳回避における送受信の様子を時系列的に表したものである。この図では簡単のために ssthresh を 8 MSS としている。



（a） スロースタートの動作　　（b） 輻輳回避の動作

図 6.9　スロースタートと輻輳回避

〔3〕 **輻輳検出時の処理**　　さて，輻輳ウィンドウが増加を続ける中で送信タイマーのタイムアウトが発生したとしよう。この場合，TCP はネットワークに輻輳が発生していると判断し，輻輳ウィンドウのサイズを 1 セグメントに減らし，スロースタートから再開する。つまり，送信量を一挙に減らして輻輳

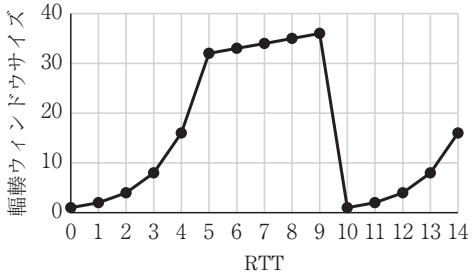


図 6.10　タイムアウト発生時の
輻輳ウィンドウの変化

ウィンドウの増加を最初からやり直す。**図 6.10** は，輻輳回避中にタイムアウトが発生した時の輻輳ウィンドウの変化を示している。9×RTT の時点でタイムアウトが発生し，輻輳ウィンドウサイズを 1 セグメントに減らし，スロースタートから再開している。

次に重複 ACK を 3 回受信した場合を考える。その場合は，まず，途中で破棄されたと思われる 1 セグメントだけをすぐに再送する。これを**ジェイコブソンの高速再送**（Jacobson's fast retransmit）と呼ぶ。そして，スロースタート閾値（ssthresh）をその時点のフライトサイズの半分まで減少させる。また，輻輳ウィンドウのサイズはこの新しい ssthresh に MSS を 3 個分付け加えた値にセットする。重複 ACK を受信している間，ウィンドウはスライドしないので，新しいセグメントは基本的に送信できないが，それでは転送効率が悪いので重複 ACK を受信するたびに暫定的に 1 セグメントずつ輻輳ウィンドウを増加させる。この一時的な輻輳ウィンドウサイズの増加を**インフレーション**（inflation）と呼ぶ[5]。インフレーションで拡大した輻輳ウィンドウ内に新たに送信できるセグメントがあれば送信する。

輻輳発生を検出して輻輳ウィンドウサイズを減少させる動作はわかりやすいが，なぜインフレーションのような動作を行うのだろうか。それは次の理由による。重複 ACK を受信するということは，相手端末は何がしかのセグメントを受信していることを意味している。したがって，たまたま 1 パケットだけ破棄された，あるいはそれほどひどい輻輳は発生していないと判断して，できる

だけセグメントの送信を続けるのである。上に述べたフライトサイズ×1/2+3 MSS の 3 MSS は重複 ACK 3 回分の受信に対応したものである。インフレーションがしばらく続くにしても，やがて正常な ACK が返るから，その時点で輻輳ウィンドウのサイズを先ほど更新した ssthresh に再度設定し直す。つまり，インフレーションで暫定的に増やした分を一挙に帳消しにするのである。

正常な ACK が返ると輻輳ウィンドウは大きくスライドし，新しいセグメント群を送信できるようになる。そして輻輳回避の動作を再開する。以上述べたように，重複 ACK を 3 回受信した場合はスロースタートからやり直しをするのではなく，ssthresh と輻輳ウィンドウを減少させて輻輳回避の動作を継続するのである。重複 ACK 3 回の受信から高速再送とインフレーションによるセグメントの送信を経て，正常 ACK の受信で輻輳回避に戻るまでの過程を**高速回復**（fast recovery）と呼ぶ。

図 6.11 は，輻輳回避中に重複 ACK を 3 回受信した時の輻輳ウィンドウの変化を示している。この図では簡単のために高速回復による過渡的な状態（輻輳ウィンドウのインフレーション）は省略している。9×RTT の時点で重複 ACK 受信による輻輳ウィンドウの減少が起きているが，スロースタートに戻るのではなく輻輳回避動作を再開している。また，**図 6.12** には高速再送と高速回復の動作例を示す。この例ではセグメントを 4 個送信し，その先頭のセグメントが破棄された場合の動作を示している。図 6.12（a）は輻輳ウィンドウ内のセグメントとスライドの様子を示し，同図（b）はセグメント送受信の様子を表

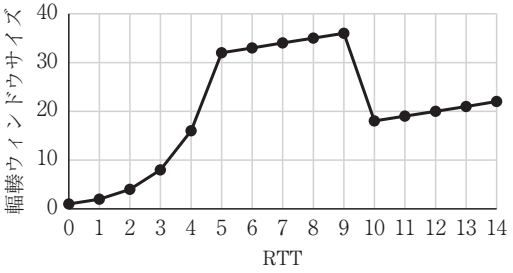
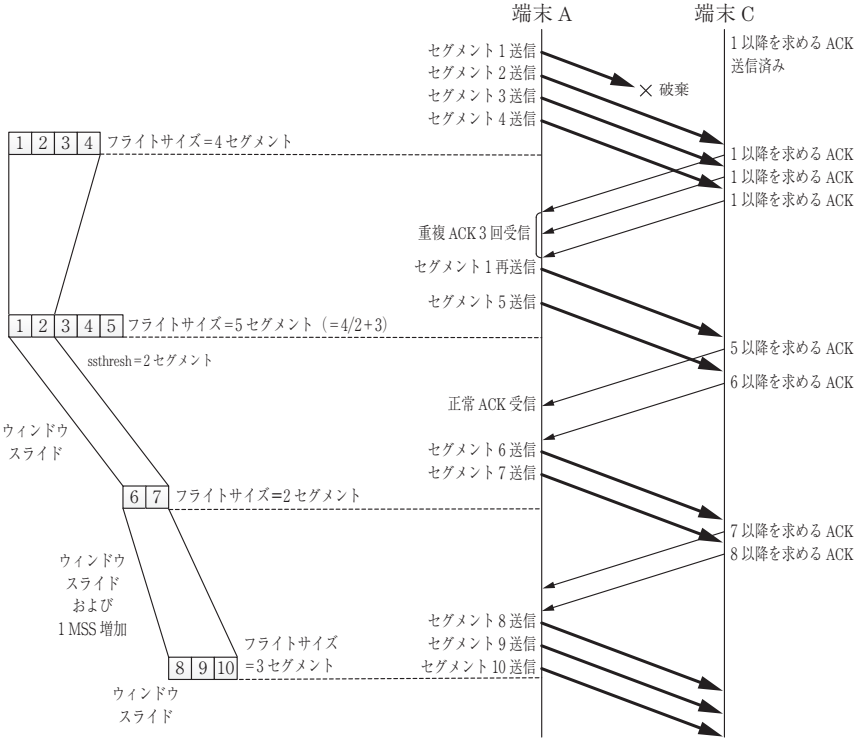


図 6.11　重複 ACK を 3 回受信した時の輻輳ウィンドウの変化



（a）　輻輳ウィンドウの中身とスライドの様子　　（b）　セグメント送受信の様子

図 6.12　高速再送および高速回復の動作例

している。

以上述べた輻輳制御は最も標準的な方法であり通称 **Reno** と呼ばれる。Reno を基本として送信ウィンドウ内の複数のセグメント欠落に対して回復動作を速めた改良版は **New Reno** と呼ばれる。ただし，New Reno が Reno と表記されている場合がある。標準的な方法以外にもいろいろな輻輳制御の方法が提案され，実際に使われている。特に伝送速度が高速で RTT が長い場合，スロースタートによる輻輳制御では伝送効率が悪いため，3 次関数の曲線を利用して急速に送信量を増加させる **cubic** という方法が現在では広く用いられている。

索　　引

——著 者 略 歴——

1983 年　京都大学工学部電気工学科卒業
1985 年　京都大学大学院工学研究科修士課程修了（電子工学専攻）
1985 年　日本電気株式会社（NEC）勤務（～2011 年）
2007 年　会津大学大学院非常勤講師（～2014 年）
2009 年　博士（コンピュータ理工学）（会津大学）
2011 年　北海道情報大学教授
　　現在に至る

情報通信ネットワーク入門

Introduction to Information and Communication Networks

2023 年 8 月 24 日　初版第 1 刷発行　★

検印省略

著　者　尾崎博一（おざきひろかず）
発 行 者　株式会社　コロナ社
　　代 表 者　牛来真也
印 刷 所　新日本印刷株式会社
製 本 所　有限会社　愛千製本所

112-0011　東京都文京区千石 4-46-10
発 行 所　株式会社　**コ ロ ナ 社**
CORONA PUBLISHING CO., LTD.
Tokyo Japan
振替 00140-8-14844・電話 (03) 3941-3131 (代)
ホームページ　https://www.coronasha.co.jp

ISBN 978-4-339-02936-9　C3055　Printed in Japan　（齋藤）