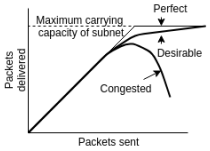
コンピュータネットワーク特論 レポート課題 #3
コンピュータサイエンス専攻 江畑 拓哉 (201920631)

1 5.3 Congestion Control Algorithms と 5.4 Quality Of Service の説で学んだことを、A4 用紙 1 ページ以内でまとめなさい。

Congestion Control Algorithms (輻輳制御アルゴリズム) はネットワークの輻輳を防ぐことを目的としている。具体的には次のような状況が発生しないようにすることを目的としている。端末 A, B から 1Gbps しか通信できないルータを経由して、それぞれ 1Gbps の通信を行いたいと要求しているものと想定する。この場合ルータは送れないパケットを一旦キューに溜めておく。しかしそのキューが一杯になってしまうと、溢れたパケットはなくなってしまう。このような溢れてしまう状況を防ぐことが目的になっている。

パケットの送信料とパケット到達量の関係についてのグラフを以下に示す。完璧に輻輳を制御できた場合には、線形にパケット到達量が増えていき、サブネットの最大キャリー容量 (リンク容量) に到達すると、それ以上パケット到達量が増えないように制御される。対して理想的な場合には、リンク容量に達する直前で線形に増えていたパケット到達量が緩やかな増加となり、やがてキャリー容量に達する。しかし輻輳制御に失敗してしまった場合には、キューから溢れてしまったパケットの再送パケットのために到達パケットが減少してしまう。また輻輳が起きてしまうと同様の理由で送信量も減らす必要も生まれてしまう。



輻輳が起きないようにするためのアプローチとしては、provisioning、traffic-aware routing, addmission control、traffic throtting、load shedding がある。これらは前者から順に対策に時間がかかり、輻輳が起こる前に対策を行うことが出来る。(後者のほうが対策に時間がかからず、輻輳が起きてから対処する。) 例えば provisioning は、輻輳が起きる前に輻輳しないように設備を増強するアプローチである。

traffic-aware routing では、輻輳を避けて通信経路を選択するアプローチであるが、問題点としてその経路が安定しないという点がある。具体的な例として、あるグループ West と East では互いの通信に C-D 経由と E-F 経由の 2 種類の経路が選択できると考える。もし C-D が輻輳を起こしてしまった場合、もう片方の E-F 経由の通信路に端末間の通信が集中してしまうことになり、結果として次に E-F 経由の通信が増加してしまう。そして次に E-F で輻輳が起きてしまったために、今度は C-D 経由の通信が増加してしまう。これを繰り返すことで経路が安定しなくなってしまう上に、C-D と E-F を均等に通信に用いることができない。

Admission Control とは、Connection-Oriented の場合にのみ利用可能なアプローチである。まずネットワーク上に仮想的なリンク容量を設定した通信路 (Virtual Circuit、VC) を作る。次にこの VC についてパケット量を監視し、もしそれを超えそうな接続要求があれば、その通信を拒否するか、あるいはリンク容量を超えない範囲で通信を行えないか要求元に確認を取る。例えばリンク容量 1Gbps の VC に対して、700Mbps が既に確保されてしまっているときに、新たに 400 Mbps の接続要求があったとする。するとその接続は拒否するか、300Mbps での接続に変更しても構わないかを確認する。また Admission Control は traffic-aware control と組み合わせて利用することが可能で、これによって輻輳を起こしているルータを取り除いてから最短経路の VC を求めることが出来るようになる。

Traffic Throtting は輻輳しそうになったら通信量を減らしてもらうという発想に基づいたアプローチである。これは輻輳を起こしそうであると判断する基準として、キューに入ってから出るまでの時間を用いている。さらにキューに入ってから出るまでの時間はキュー長に依存しているという性質を用いて、平均的なキュー長を求めることで輻輳判断を行っている。式としては $d_{new} = \alpha d_{old} + (1-\alpha)s$ で、α は $0 < \alpha < 1$ の定数、 d_{old} は最後に求めた平均のキュー長、 s は現在の平均キュー長、d_{new} は現在のキュー長さである。 つまり d_{new} がある程度以上大きくなれば、輻輳を起こしそうであると判断する。

Traffic Throtting で輻輳を起こしそうであると判断されたならば、Choke Packets や ECN、Hop-by-Hop Backpressure を用いてルータの周囲にこれを通知する。

Choke Packets とは、送信量を減らす要求を送信元に送る手法で、 d_{new} が大きくなると choke パケットを送信元に送り、送信元はそれを受け取ると送信量を減らすというプロセスになっている。

ECN (Explicit Congestion Notification、明示的輻輳通知) とは、パケットに輻輳に関するフラグを持たせることで送信元に輻輳を通知する手法である。例を用いて説明するために、端末 A から端末 B へルータ R を経由して通信する状況を想定する。まず A → B の送信パケットに、輻輳なしであることを示すよう IP ヘッダ内にフラグを設定して B へ送信する。次に途中の R で輻輳が起こったとすると、先程輻輳なしであるとしていたフラグを輻輳ありに変更して B へ中継する。B は輻輳ありであることを当該フラグから検知して、A への返信を行うパケットに TCP ヘッダ内へフラグをコピーして A に送信量を減らすように要求する。これは Choke Packats に比べて、ルータが普通のパケットの中継に専念することが出来るという利点がある。

Hop-by-Hop Backpressure とは、送信元だけでなく途中のルータに対して自分 (輻輳しそうなルータ) に送信量を減らしてもらうように Choke パケットを送る手法である。これによって、送信元にのみ送信量を減らしてもらうよりも早く輻輳対策を行うことが出来る。しかしこの仕組みを実装するには、各ルータに沢山のキューが必要になってしまうという問題点がある。

Load Shedding では、輻輳が起きたらパケットを廃棄するというアプローチである。キューに入り切らないパケットがあれば、wine 方式や milk 方式を用いてパケットを廃棄する。 wine 方式はデータ通信向きで、新たに入ってきたパケットを廃棄する。milk 方式は音声や動画像の通信向きで、古いパケットを廃棄する。また RED (Random Early Detection) という手法では、TCP が パケット廃棄 (つまり ACK が返らない状態) を元に暗黙的な輻輳判断を行っているという性質を利用している。詳しくは、ルータのキューに対してある一定の基準値を定めておき、これを超えると、超えた長さに応じた確率で入ってくるパケットを廃棄する仕組みになっている。この仕組みによって早期にパケットを送る量を減らしてもらうことが出来るようになる。

Quality Of Service (QoS) とは通信によって得られたサービスの品質を示し、例えば次のようなケースについて評価することが出来る。状況として端末 A と B がそれぞれメール、動画のストリームをルータ R を経由して送信したいものとする。これら 2 つのパケットは R のキューに入ることになるものの、動画のストリームが一定間隔でパケットが送られるのに対して、メールは局所的に密にデータが送られるため、キューの中を見ると、動画のストリームのパケットは等間隔に並ばない。そのためルータからデータが送信される際には動画ストリームのパケットはまばらに送信されてしまうことになる。このような状態を QoS が低いと評価することが出来る。

QoS を高くするための要求として、次の例の表で通信する内容 (Application) ごとにどのような通信特性が必要であるのかを眺めることが出来る。表中の Reliability はデータが確実に相手に届く信頼性を示し、Delay は届くまでの時間を示す遅延、 Jitter はパケットとパケットの届く間隔のゆらぎを示し、Bandwidth は帯域 (送信速度) を示している。

Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Videoconferencing	Low	High	High	High

QoS を低下させないためには次に紹介していくような手法が存在する。
Traffic shaping は、ルータを用いて送信量を型にはめ、ネットワーク上に流すパケットを一定以下に保つアイデアに基づいている。手法としては A leaky Bucket と A token Bucket が存在している。 A leaky Bucket では端末からネットワークにデータを流す前に、Bucket にパケットを溜め、設定されている一定時間あたりのパケット量ずつネットワークにデータを流す。これに対して A token Bucket では Bucket に一定時間ごとに溜まる token を消費した分だけパケットを送信することが出来る手法である。こちらは A leaky Bucket に比べて送信可能なパケット量を柔軟に調整することが出来るという点で優れている。尚 Bucket を溢れてしまったパケットについては捨てられてしまう点には注意が必要である。

Packet Scheduling は、フローごとにキューを別々に持ち、それぞれのキューの先頭から順番にパケットを取り出して送信するアイデアに基づいている。Round-robin fair queueing では、それぞれのキューから一つずつ先頭から順番に取り出す手法である。対して WFQ (Weighted Fair Queueing) では、それぞれのキューに対して重み付けを行って、その重みの分ずつパケットを取り出す手法である。

Admission Control は、通信を行う前に、その通信の QoS を満たすだけの通信資源があるかを確認し、そうでなければ VC を設立しないようにする仕組みを示している。フローが要求する品質の仕様の例としては次のようなものが挙げられる。この表にあるパラメータを指定して VC が設立できるかどうかを判断する。

Parameter	Unit
Token bucket rate	Bytes/sec
Token bucket size	Bytes
Peak data rate	Bytes/sec
Minimum packet size	Bytes
Maximum packet size	Bytes

Integrate Service としては、RSVP(Resource reSerVation Protocol) という手法がある。これはルータに対して資源を予約するプロトコルである。具体的には、動画を送信したいサーバとそれを受け取る端末、その間にいくつかのルータがある状況を想定する。またサーバ側に近い側のルータを上流のルータ、端末に近いルータを下流のルータと呼ぶ。まずサーバは例えば 100Mbps 必要である旨のメッセージを端末へ送る。端末はそのメッセージを受け取り、100Mbps の資源を予約する胸のメッセージをサーバ方向に送る。すると当然下流から順番にルータを経由することになるが、このときそれぞれのルータは次のルータへメッセージを送る前に自分が 100 Mbps の資源を確保できることを確認し、確保できれば次のルータへ、そうでなければ中継しない。全てのルータで予約できたとしたら、件のメッセージはサーバへ到達し、通ってきたルートで通信が開始される。但しこの手法はネットワークの規模が大きくなると実現が困難になってしまうという問題がある。

Differentiated Services としては、DiffServ(Differentiated Services) という手法がある。これはパケットをクラスに分類し、そのクラスに応じてルータの処理を変える。分類に用いる標準的なクラスとしては EF (Expedited Forwarding)、AF (Assured Forwarding) などがある。

EF とは、Expedited packets という特別なクラスに属するパケットを優先的に、ルータに届き次第すぐに送信するのに対して、Regular packets という一般的なクラスに属するパケットは Expedited packets がなければそれを送信する。

AF とは、まずパケットを優先度ごとに 4 つのクラスに分類しそのクラス番号をマークする。そして短時間でパケットを送りすぎないように送信量を型にはめる shaper、送り過ぎている際にパケットを廃棄する dropper らを通される。その後クラスに応じたキューからスケジューリングに従ってパケットを取り出し送信する。