一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
IA2016-60(2016-11)

ソフトウェアルータにおける細粒度の帯域制御

桑原　貴明†　　小谷　大祐††　　岡部　寿男††

† 京都大学大学院情報学研究科　〒 606–8501 京都府京都市左京区吉田本町
†† 京都大学学術情報メディアセンター　〒 606–8501 京都府京都市左京区吉田本町
E-mail: †akira@net.ist.i.kyoto-u.ac.jp, ††{kotani,okabe}@media.kyoto-u.ac.jp

あらまし　ネットワーク上の経路であるルータで細かい条件で帯域制御を行うことは，公平な帯域の利用を実現する手段の一つである．従来のハードウェア実装のネットワーク機器では，ネットワーク内のサーバに個別にルールを設定することが，リソース上の制約からホストの数が増えるにつれて難しくなる．一方でソフトウェア実装のものでは，リソースを大量に搭載できるため，ホストごとに個別にルールを設定することが可能であるが，ルールの行数が膨大になると，高速に処理することが難しくなる．本研究では，ソフトウェアルータを用いて，ネットワークの入口のルータにネットワーク内の全てのサーバに対する帯域制限を行うことで，トラフィック量の多い通信の元でもサービスの公平な利用を実現することを目指し、既存のソフトウェア処理で帯域制御を行える iptables hashlimit のルール数に対するパケット転送性能の評価を行った．
キーワード　帯域制御, ソフトウェアルータ, 細粒度

Fine-grained traffic control on software routers

Takaaki KUWABARA†, Daisuke KOTANI††, and Yasuo OKABE††

† Graduate School of Informatics, Kyoto University　Yoshida-Honmachi, Sakyo-ku, Kyoto, 606–8501 JAPAN
†† Academic Center for Computing and Media Studies, Kyoto University　Yoshida-Honmachi, Sakyo-ku, Kyoto, 606–8501 JAPAN
E-mail: †akira@net.ist.i.kyoto-u.ac.jp, ††{kotani,okabe}@media.kyoto-u.ac.jp

Abstract　We consider fine-grained traffic control on the ingress router in a tree-structured stub network site based on the source and/or destination IP address and/or port of each packet. We have evaluated packet forwarding rate according to the number of rules in iptables hashlimit. It is one solution for fairly use of network bandwidth to limit traffic forward rate with fine-grained condition in the ingress router. On a conventional hardware router, it is difficult to set up individual rules for a number of hosts in the network because of the resource restriction of the router. On the other hand, it is possible to set up as many rules as the memory resource allows on a software router, but it is not easy to maintain traffic forwarding rate. We aim to realize fair use of network bandwidth by setting up rate limit rules for all host in the network site at the ingress router with using software router.
Key words　Traffic Control, Software Router, Fine-grained

1. はじめに

ネットワークインフラは日々増強されている．特にこの数年は，ハードウェアの性能の向上により，その動きは活発である．一方で，ネットワークインフラが整備されるにつれて，インターネットを始めとして，より多くの人がネットワークを利用するようになり，ネットワークの上を流れるトラフィック量の増加も著しい．

トラフィック量の多さでネットワークの帯域を圧迫するトラフィックの例としては，帯域圧迫型の分散型サービス停止攻撃 (Distributed Denial of Service) がある．これは，踏み台となった大量のコンピュータから少しずつパケットを送りつけ，特定のネットワークあるいはホストの回線の帯域を埋めて通信を不可能にする攻撃である．DDoS 攻撃で流れるトラフィックは近年ますます増加しており，今後も増え続けることが予想できる．

帯域圧迫型の DDoS 攻撃のようなトラフィック量が膨大な通信は，末端になるに連れて回線が細くなる木構造をとるネットワークでは，特に特に大きな問題になる．入り口のネットワーク機器から通信の宛先のサーバまでの経路は，ネットワーク内

の他のサーバも使用しているため，経路を共有するその他のサーバも攻撃に巻き込まれる形で被害が拡大するためである．

このような状況に陥るのを防ぐためにも，ネットワークの管理者が意図しないトラフィック量の多い通信は，そのネットワーク内の一番上流のポイントで制限するのが望ましい．これを実現するには，ネットワーク管理者が一番上流のルータで，外部と通信するホストそれぞれに対して，個別に帯域制限のルールを記述する方法が考えられる．

しかし，既存の専用のハードウェアのルータでは．パケットを高速に処理するのために特殊なハードウェアを使用しており，これらのリソース上の制約からネットワーク内の全てのホストに対してルールを設定し，帯域制御を行うことが困難である．一方で，ソフトウェアルータでは，メモリや CPU は汎用のものを使用し，高速な検索アルゴリズムと検索に特化したデータ構造を用いてアプリケーションとしてルータが持つ機能を実現しているため，ハードウェアルータにあるリソース上の制約から帯域制御ルールを書くことができないといった制約はない．一方で，ソフトウェアルータでは，パケットのルーティングテーブルとの照合においてトラフィックのパターンによっては著しく性能が低下するケースが存在する可能性がある．

本稿の目的は，木構造をとるネットワーク内に存在する各々のサーバに対して，ネットワークの入り口に当たるルータで帯域制限のルールを記述することで，DDoS 攻撃などの管理者の意図と反するトラフィック量の多い通信を制限する方法を検討することである．

具体的には，宛先 IP アドレスと，宛先ポート番号の組のそれぞれに対して帯域制限を設定することで，不特定多数のホストから来る分散型の帯域圧迫型のトラフィックの下でも高速にパケットを処理することが可能なデータ構造について検討する．以降では，帯域制御の対象となる IP アドレスとポート番号の組をエントリ，帯域制御を行う単位をルールと表記する．本稿では，既存のソフトウェア処理による帯域制限の性能を確認するために，iptables hashlimit のエントリ数に対するパケットの転送性能の評価をおこなった．

以下に本稿の構成を述べる．第 2 章では，本研究の背景について述べる．第 3 章では，関連研究として既存のソフトウェア処理による帯域制御の例として iptables hashlimit と Lagopus を取り上げる．第 4 章では，大量に帯域制御のエントリを記述し，高速に検索するためのデータ構造を検討する．第 5 章では，既存の帯域制御の性能評価として，iptables hashlimit のエントリが大量に存在するときのパケットの転送速度を測定する．最後の第 6 章では今後の方針について述べる．

2. 背　　景

2.1 ハードウェアルータとソフトウェアルータ

ハードウェアルータの多くは開発元が設計したマシンと，マシンのハードウェアを考慮して設計された専用の OS を使用している．これにより ASIC(Application specific integrated circuit) によるパケットの高速な転送と，TCAM(Ternary Content Addressable Memory) を利用したルーティングテーブルの高速な検索が可能である．

一方でソフトウェアルータは，x86 などの汎用的なサーバに，OS あるいはアプリケーションとして展開される．近年は，ルータやスイッチ，ファイアウォールといった，従来は専用のハードウェアとして提供されていた機能をソフトウェアに置き換える NFV (Network Function Virtualization) [1] に関する議論が盛んに行われている．

ソフトウェアルータはハードウェアルータと比べて，パケット処理に特化したハードウェアを使用していないことや，そのままではパケットの到着のたびに CPU 割り込みをかけなくてはいけない点が性能を出す上でのボトルネックとなっている．一方で，ルータ自体がソフトウェア実装なために，ルーティングテーブルなどのパケット処理に使用できるメモリリソースを大量に利用できることや，柔軟な処理の実装を得意とすること，設置や撤去，設定が容易なためネットワークを柔軟に構成できることなどの利点がある．

本研究の目的は，上流のルータで帯域制御ルールおよびエントリを記述することで，帯域圧迫型のトラフィックを高速に処理可能なルータの実現である．ソフトウェアルータの柔軟性は，この目的を実現するための重要な性質であるため，帯域制限を実現する機器としてソフトウェアルータを取り上げる．

2.2 DDoS 攻撃

DDoS 攻撃は，DoS 攻撃が分散する複数のホストからなされたものである．DoS 攻撃は，ネットワークスタックの性質あるいは脆弱性を利用して行われる攻撃である．一つの DoS 攻撃が及ぼす影響が小さくても，分散した大量のホストからの DDoS 攻撃になると，攻撃対象のネットワークあるいはホストは大きな影響をうけることになる．DDoS 攻撃の概要を図 1 に示す．

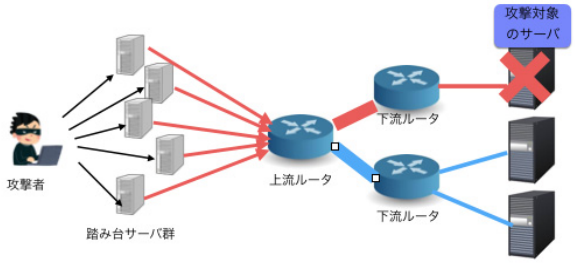

図 1　DDoS 攻撃の概要

本研究で対象とする DDoS 攻撃として，特定のサービスが標的の帯域圧迫型の UDP Flood を取り上げる．UDP Flood は，対象のホストに UDP パケットを大量に送りつけ，帯域を埋める攻撃である．DDoS 攻撃の規模は．年々規模が大きくなっている．ネットワークに接続する端末がより多くなり，かつネットワークインフラも増強されることが予想されるため，これから先に DDoS 攻撃の規模が大きくなることはあっても小さくなることは考えられないため，ネットワークの帯域に対して拡張性のある，DDoS 攻撃を耐えることのできる仕組みが必要である．

本研究では，図 1 のように末端になるにつれて帯域が細くな

るような，ネットワークが木構造をとる場合の DDoS 攻撃について考える．また，それぞれのサーバにはサービスが複数動作していると仮定する．

分散型の DoS 攻撃である DDoS 攻撃では，送信元のパケットが偽装されていたり，一つのホストから到着するトラフィック量は大きくないため，パケットの宛先の情報を使用して帯域制限を行うことを考える．このとき宛先 IP アドレスそれぞれについてのみについて帯域制限を設定すると，そのサーバで動作しているサービスの一つが，IP アドレスごとに与えられている帯域を占有する可能性がある．このような状況を防ぐため，図 2 のように，トラフィックの宛先の IP アドレスとポート番号の組み合わせを単位として，サーバで動作するサービス単位で細粒度に帯域制限を設定し DDoS 攻撃に対処することを考える．

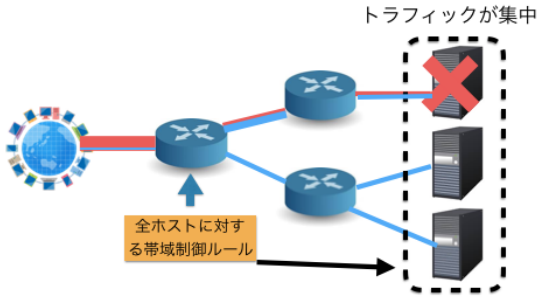

図 2　DDoS 攻撃への対処

3. 関 連 研 究

3.1 iptables hashlimit

iptables hashlimit [4] は，Linux の標準ファイアウォールである iptables の拡張モジュールとして提供されている，IP アドレスやポート番号単位での帯域制限を行う機能である．hashlimit では，送信元 IP，送信先 IP，送信元 port，送信先 port それぞれの組み合わせで帯域制限を行うことができる．iptables hashlimit では，帯域制限を行うためのトラフィックのカウンタに，トークンバケツアルゴリズムを使用している．トークンバケツアルゴリズムの概要を図 3 に示す．ルータに入ってきたパケットは，トークンバケツに定期的にたまるトークンを消費することで，転送することが可能になる．

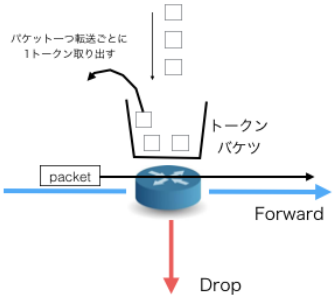

図 3　トークンバケツアルゴリズム

hashlimit モジュールでは，トラフィックのバースト量の指定や，エントリの生存期間などの様々な条件が設定できる．また，hashlimit モジュールでは指定した IP と port の組の管理にハッシュテーブルと線形リストを使用している．hashlimit テーブルの概要を図 4 に示す．

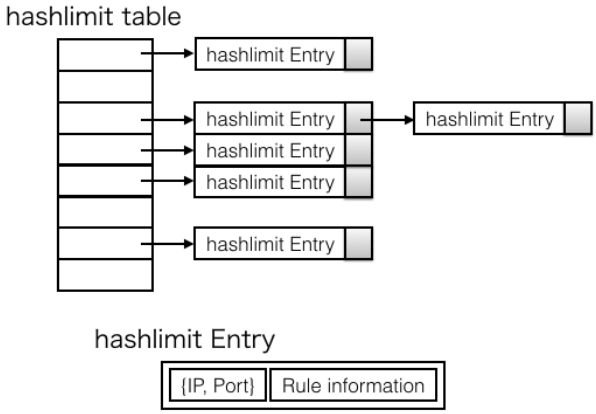

図 4　hashlimit テーブルの概要

hashlimit テーブルの一つ hashlimit エントリは，宛先と送信先の IP アドレスとポート番号の組と，パケット送信のトークン情報やエントリの更新といった帯域制限のエントリ，次の hashlimit エントリへのポインタなどで構成されている．hashlimit テーブルの検索は，IP アドレスとポート番号からハッシュ値を計算し，ハッシュテーブルの 1 バケットを特定する．ハッシュテーブルのバケットは線形リストであり，目的の hashlimit エントリがないか線形探索で検索する．

hashlimit では，宛先 IP アドレス，送信元 IP アドレス，送信元ポート番号，送信先ポート番号の 4 つの組で hashlimit のエントリを作成する．このうち二つを指定したとしても，大量のホストから分散したトラフィックが到着した場合には，hashlimit エントリの数が膨大になり到着するパケット一つ一つを検索するのに時間がかかり性能が落ちると考えられる．

3.2 Lagopus

Lagopus [2] は，OpenFlow 1.3 に準拠したソフトウェアスイッチである．Lagopus では，パケットの I/O 処理に DPDK (Data Plane Development Kit) [3] を使用して，OS の割り込み処理を回避している．また，Lagopus では OpenFlow 1.3 の仕様書 [5] で策定されている Meter Table 機能が実装されており，Meter Table を利用した帯域制限が設定できる．

Lagopus の帯域制限の処理部分は，入出力処理を担当する DPDK のライブラリを利用して行われる．DPDK のトークンバケツアルゴリズムでは，図 5 に示すようにバッファが二つ用意されている．この二つのバッファは committed token buffer と excess token bucket という．Lagopus では，committed token bucket のサイズに 1 秒間のパケットの転送速度が設定され，excess token bucket にサイズが許容するトラフィックのバースト値が設定される．committed token bucket に入っているトークンの数がパケットの転送に必要なトークンの数よりも少

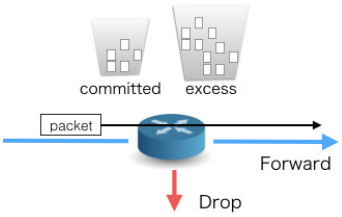

図 5　Lagopus(DPDK) のトークンバケツアルゴリズム

ないときに，excess token bucket のトークンが使用される．

Lagopus のパケット処理では，到着したパケットを OpenFlow FlowTable の flow entry と照合 (match) し，そのパケットに対して処理を実行 (action) する．Lagopus の帯域制限を行うときのパケット処理の流れを図 6 に示す．

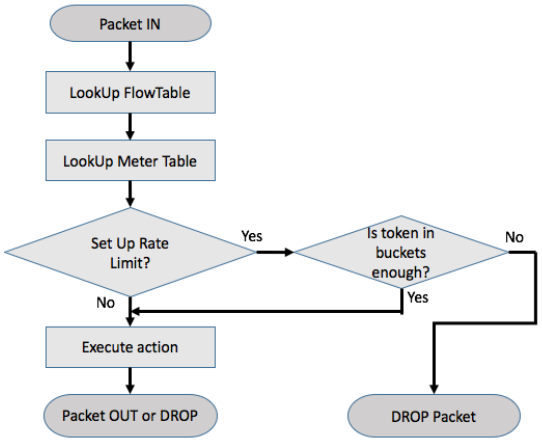

図 6　Lagopus の帯域制限のパケット処理の流れ

Lagopus は，FlowTable から，パケットに適合する flow entry を検索する．適合する flow entry が存在した場合には，flow entry に対応する meter entry を Meter table から検索する．このとき meter entry を参照して帯域制限が行われる．

4.　帯域制限の検討

4.1　帯域制限についての検討

まずネットワーク内の各サーバは，サービスを提供するために使用しているポート番号以外は閉じていると仮定する．この場合，上流のルータで帯域を圧迫する管理者の意図に反するトラフィックの帯域制限は，ネットワーク内に存在するサーバおよびサーバ上で動くサービスに対して，宛先 IP アドレスと宛先ポート番号の組のそれぞれに対して，帯域制限を設定すれば実現できる．しかし，宛先 IP アドレスと宛先ポート番号との組だけでも，サーバの数が大量にあればエントリの数が膨大になるため，エントリが大量にテーブルに存在するときでも高速に検索できるようなテーブル構造が必要になる．

4.2　ハッシュテーブルについての検討

iptables hashlimit のハッシュテーブルのエントリは宛先 IP アドレス，送信元 IP アドレス，宛先ポート番号，送信元ポート番号の 4 つの要素すべてを一つのハッシュテーブルで扱っている．これは指定した IP アドレスとポート番号の組が少ない時には高速に検索できる．しかし，図 4 に示すようなハッシュテーブルと線形リストによるデータ構造は，hashlimit エントリの生存期間超過に関する hashlimit エントリの削除のようなエントリ管理を行う際は都合が良いが，追加されるエントリが多くなるほど線形探索を行う部分が多くなる．

本稿では，ハッシュテーブルのエントリを分けることで，エントリの検索効率をあげて帯域制限を行うことを考える．検討するテーブルのデータ構造を，図 7 に示す．

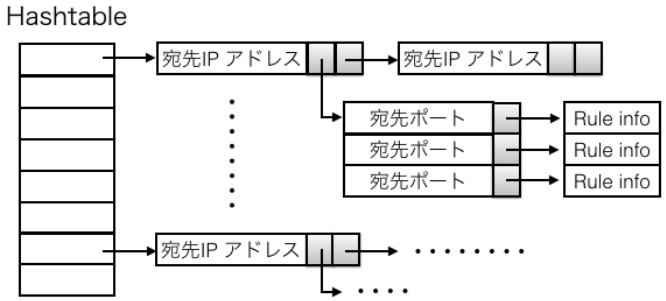

図 7　検討するハッシュテーブルの構造

このデータ構造では，パケットが到着して帯域制限エントリを確認するとき，最初に IP アドレスを検索し，次にその宛先 IP アドレスの宛先ポート番号を検索して，帯域制御に関する情報にたどり着く．検索する対象を段階ごとに分けることで，ハッシュテーブルの線形リストの長さが短くなり，大量にエントリが存在する場合でも，高速に検索できるようになると考える．

5.　実　　験

ここでは，iptables の拡張モジュールである，hashlimit の性能評価を行う．

5.1　実験環境

実験の想定として，図 2 のような特定のサーバ宛てに来る分散したトラフィックを考える．ネットワークにはサーバが複数存在していて，そのサーバの上には，外部のネットワークと通信するサービスが動作しているとする．実験環境のネットワークを図 8 に示す．

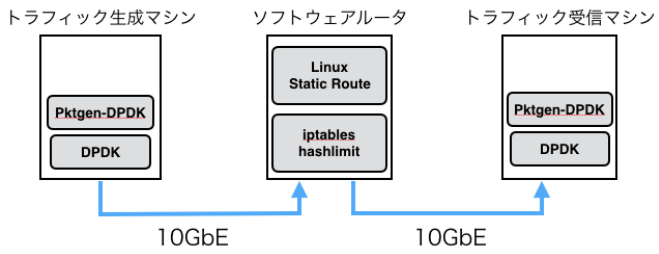

図 8　実験ネットワーク

トラフィック生成マシンから送信されたパケットは，ソフト

ウェアルータを経由して，トラフィック受信マシンに到達する．このとき，ソフトウェアルータ上の iptables hashlimit のエントリの数に対するソフトウェアルータのパケットの転送速度を測定する．また実験に使用したマシンを表 1 に示す．

5.2 転送速度の評価

図 8 のトラフィック生成マシンから，送信元の IP アドレスとポート番号を分散させたトラフィック受信マシンへとパケットを送信し，トラフィック受信マシンで受信できたパケットを測定した．実験に使用したパケットの情報を表 2 に示す．

パケットの生成には pktgen-dpdk を使用した．パケットは表 2 に示す範囲内で送信元の IP アドレスとポート番号をランダム設定して送信した．今回の実験では，測定の利便性の理由からパケットの宛先を分散させるのではなく，パケットの送信元の情報を分散させて行ったが，hashlimit テーブルに格納されて検索対象となる hashlimit テーブルのエントリの総数は同数である．また，この実験はソフトウェアルータの CPU のコア数は 1 でおこなった．hashlimit のそれぞれのエントリに対して一秒間に送信できるパケット数を 10000 個に設定し，それを超えてパケットを送付できるバースト値を 10000 万に設定した．hashlimit エントリ数に対するパケットの転送速度を結果を図 9 に示す．

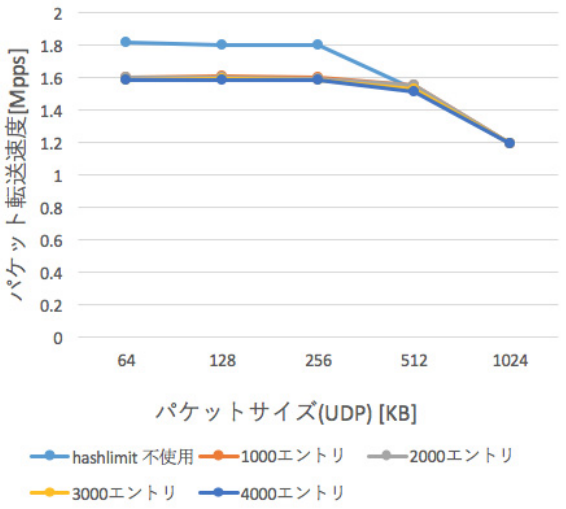

図 9　hashlimit エントリが存在する時のパケットの転送速度

図 9 中の判例は，パケットがソフトウェアルータを通る際の，hashlimit エントリの数を表す．図 9 から，エントリ数が 1000 から 4000 までのパケットの転送速度は，エントリ数にかかわらず，hashlimit エントリが存在する場合は，エントリが存在しない場合に比べて速度が約 13%低下した．

6. 今後の方針

本稿では，iptables hashlimit のエントリ数が 4000 までのときのパケットの転送速度を測定した．その結果エントリ数が 4000 までのとき，パケットの転送速度は約 13%程度であった．本稿ではトラフィック量の多い帯域専有型のトラフィックとして，主に DDoS 攻撃を取り上げた．今後は，DDoS 攻撃に対して，高速にパケット処理を行うことができるように，帯域制限のルールあるいはエントリを扱えるデータ構造について検討を深める．

文　献

[1] “ETSI : Network Functions Virualisation" http://www.etsi.org/technologies-clusters/technologies/nfv, 2016/10

[2] “Lagopus switch" http://www.lagopus.org/, 2016/10

[3] “Data Plane Development Kit" http://dpdk.org/, 2016/10

[4] “iptables-extensions" http://ipset.netfilter.org/iptables-extensions.man.html, 2016/10

[5] “OpenFlow 1.3.0 Specification" https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf, 2016/10

表 1　実験環境一覧

役割	マシン	CPU	RAM	NIC	OS
ソフトウェアルータ	Supermicro Super Server /X11SSL-F x86_64	Xeon E3-1230 3.40GHz 4Core8Thread	DDR4 2133MHz 16GB	Intel X540-T2 10GBaseT Dual	Ubuntu 16.04 LTS linux-4.4.0-42
トラフィック生成	Dell R210 II x86_64	Xeon E31220 v2 3.10GHz 4Core4Thread	DDR3 1333MHz 16GB	Intel X520-T1 10GBaseT Single	Ubuntu 16.04 LTS linux-4.4.0-43
トラフィック受信	HP DL320e x86_64	Xeon E3-1220 v5 3.10GHz 4Core4Thread	DDR3 1600 MHz 16GB	Intel X520-DA2 10GBaseT Dual	Ubuntu 16.04 LTS linux-4.4.0-38

表 2　送信したトラフィックの情報 (UDP Packet)

hashlimit テーブルのエントリ数	Src IP range	Src Port range	Dst IP range	Dst IP range
0(hashlimit 不使用)	10	400	1	1
1000	10	100	1	1
2000	10	200	1	1
3000	10	300	1	1
4000	10	400	1	1