ホワイトペーパー

NFViアクセラレーション
SRv6

インテル® FPGA プログラマブル・アクセラレーション・カード N3000 による IPv6 セグメント・ルーティングのアクセラレーション

著 者

SRv6 ソリューション・チーム
HCL Technologies

Chuck Tato
Intel Corporation
コミュニケーションズ・ビジネス事業部
ディレクター

目 次

はじめに

セグメント・ルーティング (SR) は、ネットワーク機能仮想化(NFV) やソフトウェア・デファインド・ネットワーク (SDN) といった最新のアーキテクチャーで求められる要件に対応できる新しいテクノロジーです。SR は、ネットワーキング・プログラマビリティー、サービス機能チェイニング (SFC)、プロトコルのシンプル化、トラフィック・エンジニアリング、モバイル・ネットワークと固定ネットワークのコンバージェンスを実現する統合ソリューションを提供します。

SR では、ソース・ルーティング方式を利用します。ノードは、「セグメント」と呼ばれる命令の順序付きリストとしてインスタンス化された SR ポリシーを使用してパケットをハンドリングします (通常はセグメント識別子 (SID) を参照)。セグメントは、トポロジー (指定されたパスを経由してパケットを転送する) またはサービスベース (NFV/SDN アーキテクチャー内では、コンテナや仮想マシン (VM) により実装された仮想ネットワーク機能 (VNF) でサービスを指定) のいずれかで任意の命令を表すことができます。各セグメントは、SR ノードに対してはローカル・セマンティックを、SR ドメイン内ではグローバル・セマンティックを持つことができます。このテクノロジーを活用することで、ネットワークはフローごとまたはアプリケーションごとの状態を維持する必要がなくなります。

SR アーキテクチャーは、マルチプロトコル・ラベル・スイッチング (MPLS) や IPv6 などのさまざまなデータプレーン上にインスタンスさせることが可能ですが、ここでは、IPv6 上に実装するセグメント・ルーティング (SRv6) に着目して説明します。SRv6 では、ソフトウェアのみの実装から、SRv6 の特定機能をハードウェアへオフロードしたり、純ハードウェア・ベース製品に実装するなど、効率性に優れたさまざまな実装オプションが利用可能です。

このホワイトペーパーでは、利用可能な SRv6 ソフトウェアの実装例についての概要を説明していますが、ベクトルパケット処理 (VPP) データプレーンに対してインテル® FPGA プログラマブル・アクセラレーション・カード N3000 (インテル® FPGA PAC N3000) 用に構築されたソリューションの紹介が、このペーパーの主な目的です。このソリューションでは、そのパフォーマンスが、貴重な CPU サイクルとリソースの節約をもたらします。

概 略

- 2 ページ「背景」では、アクセラレーター・ソリューションと SRv6 プロトコルの背景について簡単にまとめています。
- 3 ページ「SRv6 ソフトウェア実装」では、現時点で可能な主な SRv6 ソフトウェア実装について説明します。
- 4 ページ「SRv6 アクセラレーションの手法」では、VPP データプレーンと関連する処理の詳細をベースに、SRv6 アクセラレーションの手法について説明します。
- 5 ページ「SRv6 のスループット・パフォーマンス」では、SR ダイナミック・プロキシーによるサービス機能チェイニングのパフォーマンス・データを示します。

背景

NFVi アクセラレーション・ソリューション

ネットワーク機能仮想化インフラストラクチャー(NFVi) で提案されるアプローチでは、データセンター内で IPv6 ルーティング・ベースのアーキテクチャーを利用します。さらに、SRv6 ベースのファブリックをデータセンターで使用すると、必要とされる基本プロトコルスタックを削減することが可能です(例えば、データセンターの Border Gateway Protocol (BGP) と Internal Gateway Protocol (IGP) により、コア・ネットワークとの相互接続がシンプルになります)。また、トラフィック・エンジニアリングのための Equal Cost Multi Path (ECMP) を暗黙的に処理できるようになり、イーサネット VPN (EVPN) オーバーレイの実装もシンプルになります。これにより、データプレーン・サーバー内に配置されるトンネル・エンドポイント (TEP) まで SRv6 ドメインを拡張できます。

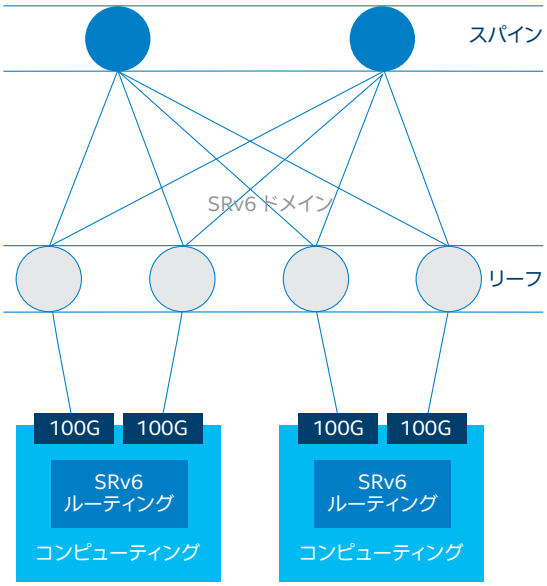

図 1. データセンター・アーキテクチャー

NFV 業界では、NFV/SDN の新しいワークロードに対応するには、標準のネットワーク・インターフェイス・カード (NIC) 製品ではパフォーマンス制限が避けられないと認識されはじめています。ネットワークでこういったパフォーマンス面での制限があると、サーバーでネットワーキング専用のものより多くのコンピューティング・リソースが必要となり、実際の VNF で使用できるリソースが限られてしまうことになりかねません。ネットワーキングに対してこの追加のオーバーヘッドを受け入れても、同じネットワーキング機能をハードウェアで実現できるソリューションと比べ、パフォーマンスが予測しにくくなります。

インテル® FPGA アクセラレーション・ソリューションなら、NFVi をオープン・プラットフォームとして維持しながら、ローレベルの特定のネットワーク機能をハードウェアにオフロードし、データパスのパフォーマンスと予測可能性を高め、CPU のサイクルとコア数をセーブすることができるため、サーバーに追加した CPU リソースを VNF やワークロードに割り当てることも可能になります。

インテル® FPGA PAC N3000 では、インテル® Quartus® Prime 開発ソフトウェアを含めた標準のインテル® FPGA 開発環境を利用して FPGA ベースのソリューションを再プログラミングできるため、オフロードしたほぼすべてのタイプの機能で高いレベルのパフォーマンスを実現できます。

ここで紹介する HCL ソリューションは、SRv6 プロトコル処理をインテル® FPGA PAC N3000 にオフロードした実装に基づいています。

このソリューションは、通信事業者の仮想化導入環境で、大きく 2 つのメリットをもたらします。

- ハードウェアのオープン・プラットフォームを使用したまま、データプレーンのスループットを大幅に向上
- 導入済みのハードウェア・ベースを変更することなく、新しい SRv6 機能へのスムーズなネットワーク・アップグレードが可能

SRv6 の概要

SRv6 ベースのアーキテクチャーを構築するのは、ソース・ルーティングの手法です。「セグメント」と呼ばれる命令の順序付きリストを使用して、一連の中間ノードを経由して宛先までパケットを転送します。SR ドメイン内で、IPv6 セグメントは特定の IPv6 ルーティング・ヘッダーにより転送されます。これが SRv6 ヘッダーと呼ばれる、IPv6 ヘッダー領域内の拡張ヘッダーです。ネットワーク・セグメントには次の 2 種類があります。

- 隣接セグメント : 特定リンクで転送を許可するローカルな (ノードへの) セグメント。
- プレフィックス・セグメント : ネットワーク・プレフィックスに接続されたグローバルな (SR ドメインへの) セグメント。ノードセグメントはプレフィックス・セグメントの特殊なケースです。

さらに、サービス機能チェイニングのコンテキストでは、サービスセグメント (ノードからローカル) がパケットに適用される特定サービスを表します。

特定のパスを指定するには、セグメントの順序リストを含むセグメント・ルーティング・ヘッダー (SRH) をエッジノードで挿入する必要があります (図 2 を参照)。この情報は、ルーティング・プロトコルまたは SDN コントローラーのコンフィグレーションから取得できます。

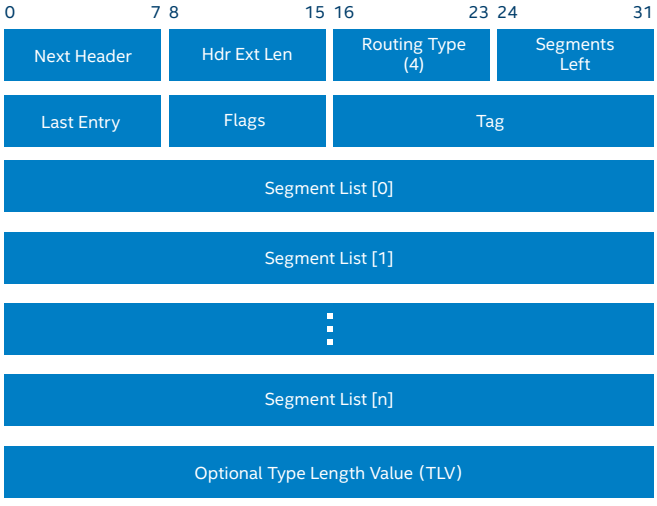

図 2. SRv6 ヘッダーのフォーマット

SRv6 では、セグメントは通常の 128 ビット IPv6 アドレスとしてエンコードされます。セグメントリストは SRH 内に格納され、最大 128 個のセグメントを追加できます。セグメントはセグメントリストに降順で格納され、例えば、インデックス 0 のセグメントは、最後に実行される命令に相当します。

以下に簡単な SRH フィールドの説明を記載します。

- Next Header : SRH の後に続くヘッダーの種類。
- Hdr Ext Len : SRH ヘッダーの長さ、単位は 8 バイト。最初の 8 バイトは含まれません。
- Routing Type : 常に 4 に設定。
- Segments Left (SL) : 残りのセグメントの数。
- Last Entry : セグメントリスト内で最後のセグメントのインデックスを指定。
- Flags : クリーンアップ・フラグは、最後から 2 番目のセグメント・エンドポイント SRH ストリッピングの対象となるフラグ。
- Tag : 同じプロパティーを共有するパケットクラスに属するパケットのマーキングに使用。
- Type-Length-Value (TLV) : SRH への追加データの格納に使用する 3 タプル。この 3 タプル構造により、可変長の TLV を実装できます。Length フィールドで Value に含まれるバイト数を指定し、最後に各 TLV タイプに対するユニークな識別子が Type に格納されます。

SRH を導入する場合、SRv6 パケットに対する新しいパケット処理ルールを定義する必要があります。

SRv6 パケット処理に関係するノードは次の 3 種類です。

- ソース SR ノード : SRH を IPv6 パケットに挿入するノード。IPv6 パケットの送信元であるエンドホスト、または受信パケットを SRH とともに IPv6 の外部ヘッダーにカプセル化する SRv6 イングレスルーターのいずれかになります。
- トランジット・ノード : 宛先アドレス (DA) がノードに属する SID ではない IPv6 パケットを転送するノード。単に IPv6 の DA に基づいてパケットを転送する通常の IPv6 ルーターとして動作します。
- SR セグメント・エンドポイント・ノード : DA がノードに属する SID である IPv6 パケットを受信するノード。

Internet Engineering Task Force (IETF) が公開している SRv6 ネットワーク・プログラミングに関するドラフト版では、SRv6 アーキテクチャーの定義に加え、SRv6 SID の概念と SRv6 SID に関連すると考えられる機能の詳細が記載されています。SRv6 SID は LOC:FUNCT としてエンコードでき、この中のロケーター部分が SID で最上位の L ビットで、ネットワークの場所、つまり SRv6 ノードを示します。機能の部分が下位 128-L ビットであり、そのノードに実装される機能を示します。各機能に引数 (ARGS) を追加する必要がある場合は、LOC:FUNCT:ARGS::. の形式で FUNCT の直後に続けます。IETF によるこのドラフト版には、SID に関連する「既知の」機能についての定義も含まれています。

サービス機能チェイニングの SR プロキシー

SRv6 では、経由する VNF を SID の順序付きリストでマッピングすることによって NFV SFC に対応します。SR 非対応の VNF の場合、コンピューティング・ノードは SR プロキシー機能を実行することができ、サービス機能で処理できるようにパケットを変更できます。SR プロキシーの一般的な動作では、ローカルでインスタンス化したサービスセグメントを介して VNF 宛てのトラフィックをインターセプトし、SR 関連情報を削除してそのトラフィックを修正した後、IFACE-OUT VNF インターフェイスに送信します。その後、IFACE-IN を介して VNF から SR プロキシーにトラフィックが返送されると、SR 情報がリストアされ、転送が行われます。

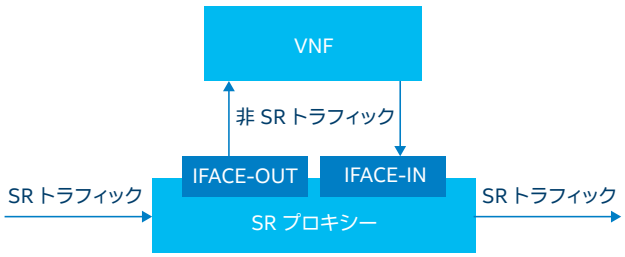

図 3. 汎用 SR プロキシー

IETF では、次の 4 種類の SR プロキシー方式が定義されています。

- 静的プロキシー (VPP で使用可能)
- 動的プロキシー (VPP で使用可能)
- 共有プロキシー
- マスカレーディング / プロキシー (VPP で使用可能)

後半で記載している SRv6 パフォーマンス・データは、SR 動的プロキシーによるシナリオに基づきます。

SRv6 ソフトウェア実装

SRv6 データプレーンの実装は、Linux* や fd.io VPP をはじめとするオープンソース・ソフトウェア機構とハードウェア / ASIC 通信ベンダー・プロバイダーの両方から利用できるようになりました。IETF の下で相互運用性テストも報告されています。このホワイトペーパーでは、アクセラレーションされた VPP データプレーンのみのパフォーマンス・データを紹介していますが、次のセクションでは Linux* カーネルベースの実装についても簡単に触れておきます。Linux* カーネルのパフォーマンスに関する一部のデータは資料で入手可能です。

Linux* の SRv6 実装の概要

SRv6 の実装は、Lightweight Tunnel (LWTunnel) として Linux* Kernel 4.10 に統合されています。SRv6 SID は、IPv6 ルーティングの FIB エントリーとして構成されます。iproute2 ユーザー空間ユーティリティーも、追加となった SID と関連動作に対応するように拡張されました。

Linux* 4.18 カスタム SRv6 機能は、End.BPF フックを介した拡張 Berkeley Packet Filter (eBPF) を使用して Linux* カーネルへ実装可能です。現在サポートされるビヘイビアは、トランジットとして T.insert、T.encaps、エンドポイントとして End、End.T、End.X、End.DX2、End.DX6、End.DT6 を含みます。

SREXT は、セグメント・ルーティングの基本機能と拡張機能の両方を実現するカーネルモジュールです。これはスタンドアロンの SRv6 実装としても、既存の SRv6 カーネル実装の補完としても使用することができます。SREXT は完全独立な「My Local SID table」に対応し、これまでの Linux* カーネル・エンドポイント・セットに End.AD と End.AM を追加します。

VPP の実装

VPP での SRv6 の最初のサポートは、2017 年 4 月にリリースされたバージョン 17.04 から始まりました。現時点で、End、End.X、End.DX6、End.DT6、End.DX4、End.DT4、End.DX2、End.B6、End.B6.Encaps の各 SRv6 エンドポイントがサポートされています。さらに、T.insert、T.encaps の各ポリシー動作にも対応済みです。SR の手法で極めて重要なプログラマビリティー・オプションを完全にサポートするために、VPP では、開発者がアドホック・アプリケーション・プログラミング・インターフェイス (API) を介して、プラグインとして VPP に接続できるカスタム SRv6 関数を簡単に作成できるようになっています。プラグインには新しいグラフノードを介してカスタム動作するコードが必要となりますが、VPP には基本的な SRv6 機能がすでに実装済みです。SR 非対応の SRv6 エンドポイント実装では、サービス機能チェインのプロキシー関数を VPP プラグインとして利用できます。

SRv6 アクセラレーションの手法

SRv6 対応 NFVi 環境内のコンピューティング・ノードは、インテル® FPGA PAC N3000 によって提供される追加のアクセラレーションにより恩恵を受けることができ、インフラストラクチャーとシームレスに統合され、導入とネットワーキングが簡素化されます (図 4 を参照)。

これは通信事業者にとって、このようなソリューションを既存環境に導入し、管理できるようにするためには、オーケストレーション・サポートの開発が不可欠であるということを意味します。

ソリューション管理アプリケーションを開発する基本フレームワークとして活用できるように、インテルはオープン・プログラマブル・アクセラレーション・エンジン (OPAE) を開発しました。ここで提案する SRv6 アクセラレーション手法では、SRv6 処理の全体を VPP/DPDK ソフトウェアで実行する VPP ベースの SRv6 実装のパフォーマンス改善に取り組みます。ハードウェアと VPP ソフトウェア間の SRv6 動作で機能の分割を新たに定義して、CPU 負荷の高い処理をインテル® FPGA PAC N3000 にオフロードするというのが基本的な考え方です。インテル® FPGA PAC N3000 と VPP 間のインターワーキングは、VPP 内に新しいグラフノードを導入することでサポートされます。VPP はエンコード / デコード / ハードウェア関連の情報の処理を監視します。

結果として、スループット・パフォーマンスの向上は基本的に、ハードウェア内にオフロードされた SRv6 処理によって削減された CPU コアサイクルと、新たに加わったハードウェア・インターワーキングで必要となる追加の処理との差分によって与えられます。

SR 動的プロキシーによるオフロード

このユースケースをサポートするための基本的な考え方は、SRH のカプセル化 / カプセル化解除と、その結果としての SR キャッシュ処理をインテル® FPGA PAC N3000 にオフロードするというものです。入力方向では、インテル® FPGA PAC N3000 が SID 検索を実行し、その検索結果に基づいてアウター IPv6/SRH ヘッダーを削除し、代わりに適切なメタデータヘッダーを付加して、VPP で内部フレーム / パケットを処理できるようにしています。出力方向では、SmartNIC 上で内部フレーム / パケットがメタデータとともに受信され、その特定のフローに使用される SRH キャッシュエントリーをインテル® FPGA PAC N3000 に指示します。この手法では、経路検索は VPP 内で実行されます。

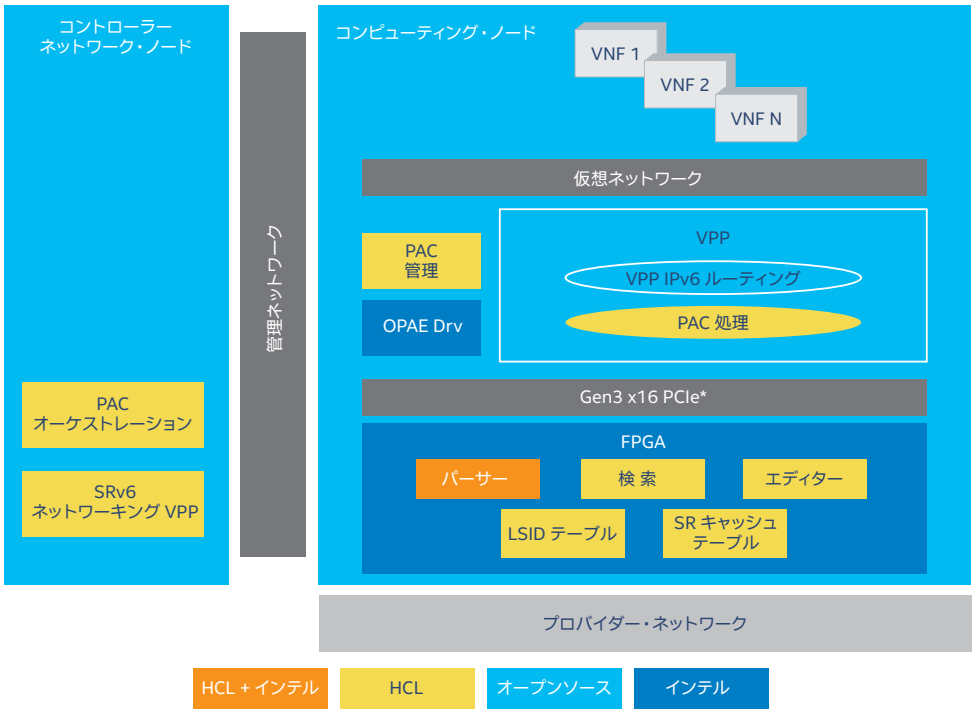

図 4. OpenStack* リファレンス・アーキテクチャー内のアクセラレーター

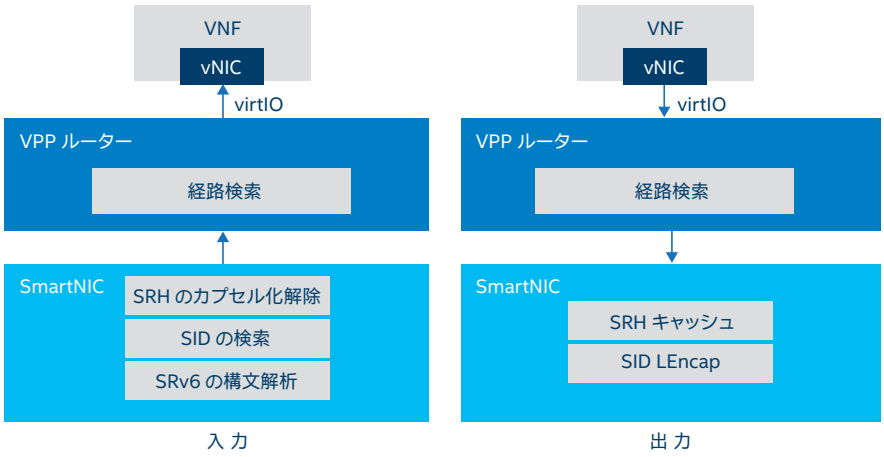

図 5. 基本的な処理

SRv6 のスループット・パフォーマンス

実験では、アクセラレーション実装とアクセラレーション未実装の両方のシナリオで、End.AD2 の動作のスループット・パフォーマンスをテストしました。

スループット測定の定義は IETF RFC 1242 に準拠し、1 秒当たりのパケット数（pps）で記録しています。許容耐力に応じて、Non Drop Rate（NDR）または Partial Drop Rate（PDR）を定義します。ここでは、0.5% のしきい値に設定した PDR ベースのパフォーマンス測定値を示しています。

テスト環境のセットアップ

テスト環境に使用したセットアップを図 6 に、構成設定を表 1 に示します。

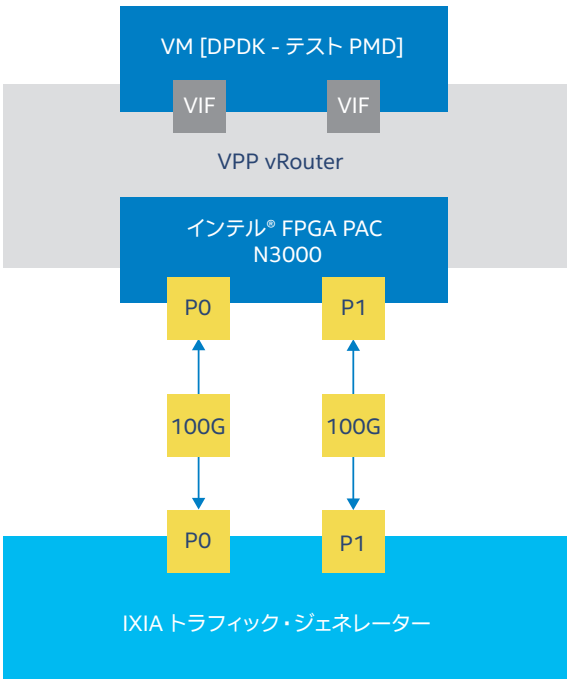

図 6. テスト環境のセットアップ

テスト対象デバイス（DUT）の構成	
CPU	ファミリー 6、モデル 85、モデル名：インテル® Xeon® Platinum 8180M CPU（2.50GHz）
ソケット	CPU ソケット 1 を使用
コア / ソケット	56（物理 28 + 論理 28）
インテル® HT	無効
インテル® VT	有効
インテル® ターボ・ブースト	有効、3GHz
RAM	192GB
SmartNIC	PAC N3000 1x4x25G
トランシーバー	QSFP28 – SR4、光ファイバー MTO ケーブル（12 芯）を使用
ホスト・オペレーティング・システム（OS）	CentOS* 7.4 カーネル 3.10
Huge Pages	2MB
データプレーン開発キット（DPDK）のバージョン	19.05
VPP のバージョン	19.08
QEMU のバージョン	2.6

表 1. DUT の構成

End.AD2 ケースに一致する SRv6 トラフィックを双方向に送信します。フルラインレートから始め、ロスが指定したしきい値を下回るまで Cisco* TRex トラフィック・ジェネレーターの両ポートのレートを低下させています。「パススルー」モードのインテル® FPGA PAC N3000 を使用して、両方のレガシー VPP に対して同じテストを繰り返しました。

結 果

192B、512B、Internet Mix（IMIX）の 3 つの異なるパケットサイズでパフォーマンスを測定しました。IMIX のトラフィック・プロファイルは表 2 に示すとおりです。

パケットサイズ（単位：バイト）	パケットの分布	バイトの分布
192	58.33%	22%
576	33.33%	47%
1,500	8.33%	31%

表 2. IMIX プロファイル

パケットサイズ 192B の結果を図 7 に示しています。表 3 は、パケット当たりのコアサイクル数（CPP）の削減率です。スループット・パフォーマンスは平均 CPP に反比例しています。

パケットサイズ（単位：バイト）	CPP の削減
192	84%
512	72%
IMIX	67.5%

表 3. パケット当たりのサイクル数の削減

短パケットでは、CPU キャッシュ処理と比較して CPP の節約の影響が大きいため、結果はパケットサイズが短いほどパフォーマンスが向上することを示しています。図 7 に示すとおり、VPP ベースのソリューションでは 12 コア使用する最大スループットに、アクセラレーション・ソリューションでは 8 コアのみで到達しています。このパフォーマンスの向上は、512 バイトのパケットサイズでも確認されました。IMIX プロファイルから分かるとおり、1,500 バイトのパケットが一定の割合を超えると、パフォーマンス向上の達成可能なゲインはさらに低くなります。

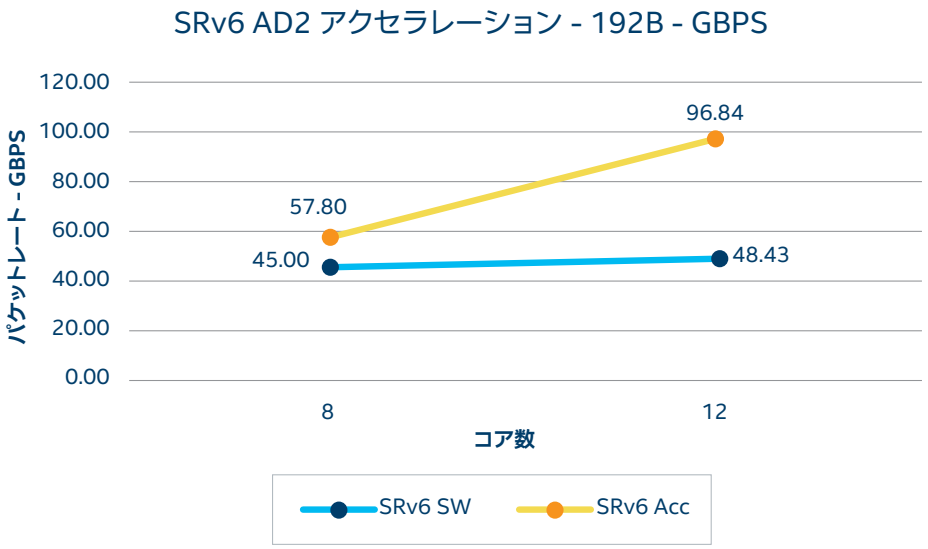

図 7. 192 バイトのパケットサイズでのパフォーマンス・データ

まとめ

SRv6 への VPP や DPDK の実装により、高いスループット・パフォーマンスを示す最適化されたソリューションがすでに提供されています。ここで提案したインテル® FPGA PAC N3000 を使用するアクセラレーション・ソリューションを利用することで、全体的なスループットが向上し、パフォーマンスを予測しやすくなり、CPU コア / サイクルを解放してほかの VNF やワークロードに割り当てることができるようになります。また、パフォーマンスをさらに向上させることもできますが、これを実現するには、ハードウェア内にネットワークと VM 間のルーティング機能とインターフェイスを配置し、インテル® FPGA の機能を可能な限り最大限に活用する必要があります。

HCL のエンジニアリング / R&D サービスについて

HCL Technologies の一部門である HCL ERS は、革新的な製品とソリューションで市場参入を目指すテクノロジー主導の組織を支援します。HCL が実現するのは、柔軟なエンゲージメント・モデルを通じた顧客企業との連携による、迅速な製品開発、新しいテクノロジーの採用、産業向けの俊敏なサービス提供を組み合わせた、ワールドクラスの顧客体験です。また、関連ソリューションを展開するエコシステムを構築して、市場でのリーダーシップを確立します。HCL は、半導体、通信とネットワーキング、家電機器、ソフトウェア、オンライン、サーバーとストレージ、医療機器、航空宇宙と防衛、自動車、工業生産の各分野で、顧客向けのエンジニアリング製品、ソリューション、プラットフォームを開発しています。

詳細情報

詳細については、erx@hcl.com までお問い合わせください。

Twitter :
http://twitter.com/hclers/（英語）

HCL のブログ :
https://www.hcltech.com/blogs/（英語）

HCL のウェブサイト
http://www.hcltech.com/engineering-services/（英語）

参考資料

1. C. Filsfils ほか、「Segment Routing Architecture」、Internet Requests for Comments RFC Editor RFC 8402。参照先 : https://tools.ietf.org/html/rfc8402/（英語）
2. C. Filsfils ほか、「SRv6 Network Programming」、IETF Internet-Draft。参照先 : https://tools.ietf.org/html/draft-ietf-spring-srv6-network-programming-00/（英語）
3. C. Filsfils ほか、「IPv6 Segment Routing Header (SRH)」、IETF InternetDraft。参照先 : https://tools.ietf.org/html/draft-ietf-6man-segment-routing-header-18/（英語）
4. F. Clad ほか、「Service Programming with Segment Routing」。参照先 : https://tools.ietf.org/html/draft-xuclad-spring-sr-service-programming-02/（英語）
5. C. Filsfils ほか、「SRv6 interoperability report」、IETF Internet-Draft。参照先 : https://tools.ietf.org/html/draft-filsfils-spring-srv6-interop-02/（英語）
6. 「VPP」。参照先 : https://wiki.fd.io/view/VPP（英語）
7. 「SRv6: Segment Routing for IPv6」。参照先 : https://docs.fd.io/vpp/17.07/srv6_doc.html（英語）
8. S. Bradner、「Benchmarking Terminology for Network Interconnection Devices」、Internet Requests for Comments RFC Editor RFC 1242、1991 年 7 月。参照先 : https://tools.ietf.org/html/rfc1242/（英語）
9. TRex - 現実的なトラフィック・ジェネレーター。参照先 : https://trex-tgn.cisco.com/（英語）
10. 「srext – a Linux kernel module implementing SRv6 Network Programming model」。参照先 : https://github.com/netgroup/SRv6-net-prog/（英語）
11. A. Abdelsalam ほか、「Performance of IPv6 Segment Routing in Linux Kernel」、2018 14th International Conference on Network and Service Management（CNSM）、2018 年、ローマ、414 ~ 419 ページ
12. P. Lapukhov ほか、「Use of BGP for Routing in Large-Scale Data Centers」、Internet Requests for Comments RFC Editor RFC 7938、2016 年 8 月。参照先 : https://tools.ietf.org/html/rfc7938/（英語）

性能に関するテストに使用されるソフトウェアとワークロードは、性能がインテル® マイクロプロセッサー用に最適化されていることがあります。

SYSmark* や MobileMark* などの性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。製品の購入を検討される場合は、他の製品と組み合わせた場合の本製品の性能など、ほかの情報や性能テストも参考にして、パフォーマンスを総合的に評価することをお勧めします。詳細については、http://www.intel.com/benchmarks/(英語) を参照してください。

性能の測定結果は、2019 年 10 月時点のテストに基づいています。また、現在公開中のすべてのセキュリティー・アップデートが適用されているとは限りません。詳細については、公開されている構成情報を参照してください。絶対的なセキュリティーを提供できる製品またはコンポーネントはありません。

このソリューションでは、コストのかかる CPU サイクルとリソースを節約したパフォーマンス結果が示されています。

インテルは、サードパーティーのデータについて管理や監査を行っていません。原典を確認し、ほかの情報も参考にして、参照しているデータが正確かどうかを確認してください。

インテルは、製品やサービスの内容を、いつでも予告なく変更できるものとします。本資料に記載した情報、製品、サービスの適用や使用により生じた損害について、インテルは、書面で明示的に合意した場合を除き、一切の責任や義務を負いません。公開された情報を利用する場合や、製品やサービスをご注文の場合は、事前に最新バージョンのデバイス仕様を入手されることをお勧めします。

Intel、インテル、Intel ロゴ、Quartus、Xeon は、アメリカ合衆国および / またはその他の国における Intel Corporation またはその子会社の商標です。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

WP-01295-1.0/JP