# In-Network Computing から見る
# データセンタネットワークの研究動向と課題

真壁　徹

2020/07/28

IS-RR-2020-001

# In-Network Computing から見る データセンタネットワークの研究動向と課題

真壁徹 [1, a)]　宇多 仁 [1]

2020 年 7 月 28 日

**概要**：データセンタの集約と大規模化に伴い，データセンタネットワークは広い帯域と高いスループット，低く安定した遅延，マルチテナント化など様々な要求に直面している．解決手段の 1 つに，ネットワークデバイスがデータの転送にとどまらず，データを処理する In-Network Computing がある．この調査は In-Network Computing の関連研究を通じ，データセンタネットワークの技術動向と課題を概観する．

**キーワード**：In-Network Computing，データセンタネットワーク

# The Research Trend & Discussion of Data Center Network from the view of In-Network Computing

TORU MAKABE [1,a)]　SATOSHI UDA[1]

July 28, 2020

***Abstract***: With the consolidation and increasing scale of data centers, data center networks are facing various requirements such as wide bandwidth, high throughput, low and stable latency and multi-tenancy. As a solution, there is In-Network Computing, which processes data in addition to data forwarding by network devices. This survey will examine changes in data center networks and issues to be solved through research trends in In-Network Computing.

***Keywords***: In-Network Computing, Data Center Network

## 1. データセンタネットワークの抱える課題

インターネットを通じて提供されるアプリケーションやクラウドコンピューティングの普及により，ネットワークトラフィックの増加が著しい. Cisco の公開する Global Cloud Index[1]からは，特徴的な傾向が得られる．

- 全世界のデータセンタ関連トラフィックは 2016 年から 2021 年まで年平均 24.7%と高い増加率で推移する．
- トラフィックの内訳はデータセンタからユーザの間が 14.9%, データセンタ間が 13.6%, データセンタ内が 71.5%であり，データセンタ内トラフィックの占める割合が高い．なおこの調査のデータセンタ内トラフィックには，ラック内で完結する通信は含まれておらず，それを含めると 90%を超える．
- 全世界のデータセンタのうち，Global Cloud Index がハイパースケールと定義する世界 24 の事業者の運用する大規模データセンタのトラフィック占有率は，2016 年の 39%から 2021 年には 55%まで上昇する．トラフィックのみならず，ハイパースケールデータセンタの占有率はサーバ数で 53%, 計算能力で 69%, 保管データ量で 65%に達する．

1 北陸先端科学技術大学院大学
Japan Institute of Science and Technology,
1-1 Asahidai, Nomi, Ishikawa 923-1292 Japan
a) tomakabe@jaist.ac.jp

この調査から分かる通り，増加するトラフィックの大部分はデータセンタ内部，特に大規模データセンタで生成，消費，保管される．主な要因は機械学習などデータ集約型アプリケーションの需要増である．規模の増大に合わせ，データセンタネットワークは次に挙げるような要求と課題に直面している．

**広帯域と高スループット**　増加するトラフィックを転送するため、ネットワークのリンクスピードは広帯域化している．サーバとスイッチ間で 100GbE，スイッチ間で 400GbE インタフェースを搭載可能な市販製品も珍しくない．次のマイルストーンとなる 800GbE も Ethernet Technology Consortium が仕様を公開済み[2]であり，ハイパースケールデータセンタを中心に普及が進むと考えられる．

一方で，その帯域を活かせないケースが見受けられる．ビデオ配信サーバへ 100GbE インタフェースを複数割り当てた Netflix 社の事例では，サーバの NUMA ノードを交差するデータフローがある場合にスループットが大幅に低下した[3]．インタフェースの帯域をサーバが活かせず，期待した性能が得られない例である．

**低遅延**　遅延はレスポンスとスループットに強い影響を与え，利用者の体験に直結する重要な性能指標である．大規模データセンタでは遅延の平均値だけでなく，99 パーセンタイル値など最大値近辺の値，いわゆるテール・レイテンシが課題となりやすい．なぜなら多数のサーバに処理を分散するアプリケーションでは，一部のサーバの応答の遅れがユーザへの応答時間に影響するからだ [4]．決定論的な遅延制御技術への期待はあるが，帯域と同様，ネットワークのデバイスやプロトコルだけでは解決できない．データを処理する終端ノードと統合的に解決すべき課題である．

**高拡張性**　従来のデータセンタネットワークは，サーバをアクセススイッチに収容し，その上位層でアグリゲーションスイッチへ集約のうえ，コアルータでデータセンタ外部に接続する構成が典型的であった．アグリゲーションスイッチをレイヤ 2 とレイヤ 3 の境界とし，大きなレイヤ 2 ドメインを構成するアプローチである．しかしデータセンタ内の通信，いわゆる East/West トラフィックの増加に伴い，フラッディングと学習モデルの非効率性，アグリゲーションスイッチの大型化，障害やメンテナンスにおける影響範囲の拡大など，大規模なレイヤ 2 ドメインの拡張性の課題が浮き彫りとなった．

現在はハイパースケールデータセンタを中心に， Leaf 層と Spine 層のそれぞれにスイッチを複数配置し，全ての Leaf と Spine を接続した CLOS トポロジの採用事例が増えている[5]．CLOS トポロジではスイッチの追加によるスケールアウトアプローチが可能であり，加えてスイッチ故障時の影響範囲を局所化できる．トラフィックの転送，ルーティングには主に BGP が用いられ，RFC 7938 で標準化されている[6].

**マルチテナンシ**　従来のデータセンタではユーザや用途別にネットワークを物理的に分割するアプローチが一般的であった．しかし昨今の大規模データセンタにおいては，迅速なサービスの提供や変更のニーズに応えるため，広帯域なリンクをサービスやユーザが共有し，論理分割のうえソフトウェアによって動的に構成を変更するマルチテナント型ネットワークが主流である．サーバやラック単位での障害を，設備の分散で解決できるハイパースケールデータセンタでは，サーバからスイッチへ広帯域なリンクを 1 つだけ持つシングルアタッチ型[7]も珍しくない．

データセンタネットワークの論理分割には，レイヤ 2 での VLAN が長く利用されてきた．しかし前述の通り，大規模なレイヤ 2 ドメインには拡張性の課題がある．解決策として，アンダーレイネットワークをレイヤ 3 ネットワークとし，その上にオーバーレイネットワークを構成し，複数のテナントを収容する手法がある．CLOS トポロジで構成したアンダーレイネットワークの上に，VXLAN でトンネリングしたオーバーレイネットワークを作る実装が代表的である．

VXLAN はイーサネットのフラッディングと学習モデルの問題を引き続き有するが，マルチプロコル BGP（MP-BGP）を利用して IP 情報と MAC 情報を交換するイーサネット VPN（EVPN）など解決法はある．

なお，レイヤを重ねずにマルチテナント化を実現する取り組みも注目されており，IPv6 セグメントルーティング（SRv6）の活用[8]がその代表例である．SRv6 は実現したいネットワークのポリシを中継ノードでなくパケットに保持できるため，トラフィックエンジニアリングなど，テナント分離の他でも大規模データセンタネットワークの有する課題を解決する技術として期待されている．

## 2. In-Network Computing の定義と分類，備えるべき要件

データセンタネットワークに対する要求のうち，性能，主にスループットと遅延に関する課題をユーザ視点で見ると，先に述べたようにネットワークだけでは解決が難しい．サーバなど終端ノードとの役割分担を見直すなど，抜本的な解決策が求められている．そのアプローチの 1 つが，In-Network Computing である．

In-Network Computing は新しい研究領域であり，定義や要件は議論の最中にある．この章では ACM SIGARCH[9]や IRTF Computing in the Network Research Group(COINRG)[10]での論考や議論を参考に，その定義と分類，備えるべき要件を整理する．

### 2.1 定義

In-Network Computing とは，従来トラフィック転送を担っていたネットワークデバイスが，ネットワーク内（In-Network）で演算や計算（Computing）を行うことを指す．

In-Network Computing はネットワークの性能に関する課題の解決にとどまらず，データセンタのトラフィック量の削減にも寄与できる可能性がある．なぜならネットワークデバイスがトラフィックを終端，処理することでサーバへのトラフィックを減らせるからである．

これまで Network Computing という In-Network Computing に似た言葉はあったが，多くはネットワークで繋がれたシステム，もしくはネットワーク内にあるコンピュータを意味していた. IRTF COINRG ではそれを Networked Computing と分類している[11].

また，In-Network Computing は新たな種類のデバイスをネットワークに加えるのではなく、スイッチや NIC といった既存のカテゴリに属するデバイスで実現する．フィルタリングやアドレス変換，負荷分散などパケットの転送にとどまらない処理を行うデバイス，いわゆるミドルボックスをネットワークへ挿入するアプローチは既に一般的であるが，このアプローチは In-Network Computing ではなく，Packet Processing であると IRTF COINRG では位置付けている[11]. しかし，サービス品質を考慮した動的な経路の最適化など Packet Processing も In-Network Computing を実現する要素である．広義の In-Network Computing には Packet Processing 技術を含めても良いだろう．

なお，仮想化されたサーバでネットワーク機能を実現する NFV(Network Functions Virtualization)は，IRTF COINRG の分類では Networked Computing のアプローチで Packet Processing を行なっていると解釈できる[11].

現時点で In-Network Computing を実現する代表的なデバイスはプログラム可能な NIC とスイッチである．これらのデバイスが市場で入手しやすくなったこと，また，デバイスの差異を吸収するデータプレーンプログラミング言語である P4(Programming Protocol-Independent Packet. Processors)[12]のエコシステム形成の本格化が，In-Network Computing の実現可能性を高めている．

以上から、本論文では In-Network Computing を「従来はネットワークデバイスと位置付けられたスイッチや NIC が，パケット転送にとどまらずに他の演算も行うこと。ただし，デバイスのプログラム可能性を活かした高度なパケット転送も含む」と定義する．

### 2.2 分類

IRTF COIN RG による In-Network Computing の 5 つのバリエーションを[11]を元に，In-Network Computing を分類する．

**Active Network および Programming the data plane of SDN switches** ネットワーク内でプログラムを動かすというアイデアは In-Network Computing が初めてではない．過去にも Active Network[13]という同様のコンセプトが提唱されていた．ネットワークデバイスが受け取ったパケットを Passive に転送するだけでなく，任意のプログラムで Active にパケットを処理することで，ネットワークの効率化や新たなアプリケーションを作り出すコンセプトを Active Network と呼ぶ．

Active Network には実現のアプローチが 2 つある．プログラム可能なネットワークデバイスを使う（Programmable Data Plane）方式と，パケットにプログラムを埋め込む（Packet-based programmability）モデルである．前者はプログラミングの難しさとデバイスへの依存性，後者はセキュリティやプライバシに関する懸念などから実装例は少なく，普及には至っていない．

しかし現在，前者の課題は P4 エコシステムの成熟などで解決できる可能性がある．In-Network Computing は Active Network コンセプトの再挑戦とも言える．

**Edge Computing** 利用者に近いネットワークデバイスへプログラムを配備するアプローチである．現在のエッジコンピューティングでは，汎用 OS を搭載したサーバや PC，アプライアンスが利用者側設備として一般的であり，アプリケーションは直接，もしくは仮想マシンやコンテナ形式で配布される．ネットワークデバイスでアプリケーションを動かすことにより，設備数の削減，ひいては故障点を減らすことにも繋がり，コスト削減や運用容易性、可用性向上に寄与する．

**Application-layer data processing frameworks** データ集約型アプリケーションの台頭はデータ量増加の主要因の 1 つであり、注目される適用領域である．ネットワークデバイスでデータを終端，処理することで性能向上やトラフィック削減を期待できる．

データ集約型アプリケーションにおいてデータ転送のスループットは処理時間に大きく影響を与えるため，アプリケーションをネットワークのどこに配置するかは重要な問題である．現在，分散型のデータ処理フレームワークはアプリケーションの配置にあたり，ネットワークに関する情報を間接的に取得，推測するにとどまっている[11]．ネットワークデバイスで実際の流量や性能を直接的に取得しアプリケーションの配置、スケジューリングに活かすことで，より高度な最適化を期待できる．

**Service Function Chaining (SFC)** Service Function Chaining(RFC7665) はサービスを構成する機能群へパケットを適切な順序で転送するフロー制御の仕組みであるが，In-Network Computing のバリエーションの 1 つと言える．従来の SFC ではサーバやミドルボックスにサービス機能を配置していたが，In-Network Computing ではプログラムがネッ

トワークデバイスで動くため，そのチェイニングを考慮する必要がある．

なお，RFC7665 ではサービス機能の宛先，ネクストホップを MAC アドレス，IP アドレスなどレイヤ 2，3 で明示してチェーンを構成するが，継続的な機能の追加や変更があり，パスを頻繁に再構成する環境では，再チェーンのオーバヘッドが課題となる．その解決のため，RFC8677 で名前ベースの転送を行う Name-Based Service Function Forwarder (nSFF)が提案されている．

### 2.3 備えるべき要件

IRTF COINRG による要件[14]を元に，In-Network Computing の実現に向け，環境が備えるべき技術的な要件を整理する．

#### 2.3.1 ネットワーク要件

**正確さ，精度** 先にテール・レイテンシ問題について述べたが，ベストエフォートではなく，決定論的に期待する性能を実現する能力がネットワークに求められる．遅延やパケット損失率が代表的な指標である．また，ネットワークデバイス上，もしくはネットワークに接続されたサーバのアプリケーション向けの計算資源，つまり Computing 用資源の正確な把握も重要である．なぜならネットワークだけでなく計算資源の状態も加味し，資源配備を最適化する必要があるからだ．つまりテレメトリの収集能力も求められる．

**並行性** ネットワークデバイス上にアプリケーションが分散され，それぞれが通信することによりコネクション数は劇的に増加する恐れがある．コネクション数がボトルネックとなり，帯域を活かせなくなる状況も考えられる．例えばストレージ，データベースに対する過多なコネクションは課題になるだろう．高い並行性を実現する技術が必要である．

**アドレッシング** 従来のインターネット，アプリケーションを中心としたアドレッシングは，性能の観点で In-Network Computing のユースケースにおいて最適とは言い難い．例えばクライアントにはサービスや機能としてのアドレスを伝え，機能を構成するネットワークやサーバなど計算資源には他の効率に優れるアドレッシングを採用するなど，新たな体系や方式を検討すべきである．

**情報共有** In-Network Computing で動的な最適化を実現するために，ネットワークデバイスとサーバは，アプリケーションの要件や資源の利用状況を交換，共有すべきである．アプリケーションは必要な CPU やメモリ量，遅延やジッタと行った要件をネットワークに伝え，ネットワークはそれに応じて構成を行う．

#### 2.3.2 計算資源要件

**特性を意識した配備** In-Network Computing における計算資源は CPU やメモリだけでなく，ネットワークプロセッサや GPU など多様であり，それぞれが特徴を持つ．よって用途に応じた資源の配備が求められる．例えば機械学習では学習と推論で適するプロセッサは異なる．特性に応じた計算資源を，ネットワークデバイス，サーバへどのように配備するか，専用か共有か，仮想化の要否など論点は数多い．

**ディスカバリ** ネットワークは要件に適し利用可能な計算資源がどこにあるかを発見や解決，つまりディスカバリできなければならない．ディスカバリで指定できる属性は重要な論点であり，プロセッサやメモリなど搭載資源の情報だけでなく，消費電力など状態を問い合わせできればユースケースは広がる．

**スケジューリング** ネットワークに配備されている資源をディスカバリした上で，要件を満たし，かつ最適な配置となるようにアプリケーションをスケジューリングできなければならない．

#### 2.3.3 管理要件

**クロスドメイン管理** データセンタネットワークの文脈において，複数のネットワークドメインを管理する重要性は，ユーザまで複数のネットワークを経由するエンドツーエンド通信と比較して低い．しかし複数のユーザが共有するマルチテナントネットワークでは論理的なクロスドメイン管理が課題となる．オーバレイで階層化するか，セグメントルーティングなどでフラットにするかは論点になるだろう．

**共同最適化** これまでの要件で述べたとおり，従来のネットワークの管理範囲を超え，サーバなどネットワークの範囲を超えた計算資源と合わせた最適化が必要となる．

## 3. 適用領域とユースケース

これまで述べた通り，In-Network Computing には高い性能や効率，品質の担保など様々な価値が期待されている．適用領域は産業 IoT からエンターテイメントまで幅広いが，このサーベイの趣旨に沿い，データセンタでの活用に焦点を当てる．アプリケーションや用途で 4 つのカテゴリに分け，それぞれ代表的なユースケースと研究事例を挙げる．

### 3.1 ネットワークアプリケーション

従来ミドルボックスやアプライアンスの機能，またはサーバのソフトウェアとして動作してきたネットワークサービスは，In-Network Computing の有望な適用領域である．

#### 3.1.1 DNS

In-Network Computing の要件で述べた通り，資源やサービスが分散する環境では，その配置先を発見，解決するディスカバリの仕組みが必要である．DNS は従来からあるディスカバリの代表例であり，ホストやサービスの名前と IP アドレスの対応を管理する．

ビジネス要件や環境の変化に柔軟に対応する手段として，アプリケーションの責務を小さくし，複数のアプリケーションを組み合わせてシステムを作るマイクロサービスアーキテクチャがある．マイクロサービスアーキテクチャの懸念の 1 つは，分散したサービスがそれぞれディスカバリ，名前解決を頻繁に行うことである．複数のサービスを組み合わせるマイクロサービスアーキテクチャにおいて，一部のディスカバリの遅延，つまりテール・レイテンシがサービス全体の応答性能に影響を与える．過多な問い合わせでコネクション管理テーブルなどサーバの資源を過度に利用する恐れもある．

マイクロサービスアーキテクチャの実装基盤として著名な Kubernetes[15]は，サーバ内に DNS キャッシュを配置することでその課題を解決している[16]が，その効果は Kubernetes で動くアプリケーションの名前解決に限られる．アプリケーションの実行環境に依存しない，DNS 自身の性能の底上げが求められる．

Woodruff ら[17]は，アプリケーションから透過的に利用できる In-Network DNS である P4DNS を提案している．P4DNS は FPGA(Field Programmable Gate Array)を搭載したネットワーク処理拡張ボードである NetFPGA SUME[18]に P4 で DNS を実装し，A レコードに対する問い合わせへの応答，キャッシング，再帰的問い合わせなど DNS の基本的な機能を持つ．そして DNS 問い合わせに関わらないパケットに対しては転送のみ行う．

P4DNS はその性能評価において，DNS のソフトウェア実装である NSD[19]と比較し，秒間問い合わせ数のスループットで 52 倍の性能を実現した．また遅延においても，NSD の中央値 122.25μs（99 パーセンタイル 181.73μs）に対し，中央値 3.33μs（99 パーセンタイル 3.35μs)を達成した．

なお，P4DNS のオーバヘッドを検証する目的で，DNS 処理を除きパケット転送に絞った同条件での性能も評価されている．その遅延は中央値 1.675μs（99 パーセンタイル 1.696μs）であった．この差が P4DNS の実行時パースやマッチ，アクション処理にかかる時間である．なお Woodruff らは利用した SDNet コンパイラ[20]の改善によって，P4DNS は同じ設計のまま，性能向上が期待できるとしている．

P4DNS のアプローチはアプリケーションに透過的でありスイッチや NIC など様々な配置が考えられる．その中でもサーバを集約する ToR（Top of Rack）スイッチへの配置が，遅延の低減と安定，問い合わせトラフィックのネットワーク末端での封じ込め，管理点の削減などの観点でバランスに優れているだろう．

なお，P4 と NetFPGA の組み合わせ固有の課題もあるが，著しい性能向上の反面，制約もある．例えばパース後パケットの状態保持や C 言語スタイルの文字列処理などプログラミングに関する難しさ，また，キャッシュ管理などコントロールプレーンの性能拡張性の確保し難さを Woodruff らは指摘している．

### 3.2 サーバのネットワーク機能のオフロード

トラフィックの増加，広帯域化によりサーバのネットワークスタックにかかる負担は大きくなっている．ネットワーク処理にかかる CPU の利用量が過大となれば，アプリケーションに影響する．よってサーバのネットワークスタックを NIC にオフロードする SmartNIC[21]は期待の大きな活用法である．

現在の SmartNIC は，プロセッサとして FPGA，ASIC NPU(Network Processing Unit)，SoC(System on Chip: CPU と NIC の同一チップでの組み合わせ)を使うものに分類できる．プロセッサの主な評価軸は性能，柔軟性，プログラミングのしやすさ，価格であり，ユースケースや目的に合わせて選定される．

従来から TCP のチェックサムやセグメンテーションのオフロードなどは NIC で行われてきたが，SmartNIC はそれに加えて幅広い用途に適用できる．TCP や UDP，RDMA などネットワークプロトコルのオフロードをはじめ，iSCSI や FCoE などのストレージ関連プロトコル，株式の高頻度取引などアプリケーション処理，暗号化やフィルタリングを例とするセキュリティ関連処理など，数多く挙げられる[21]．次に挙げる DDoS 緩和はセキュリティ関連処理の特徴的なユースケースである．

#### 3.2.1 DDoS 緩和

DDoS の緩和はハイパースケールデータセンタによって喫緊の課題である．キャッシュやリバースプロキシなど，インターネットとの界面となるサービスには，アプライアンスだけでなく汎用サーバも利用されている[22]．DDoS に耐える能力を有することはもちろん，汎用サーバのネットワークスタックが利用する資源を最小化し，本来提供すべき機能やサービスへより多くの資源を割り当てたい．

Miano[23]らは，DDoS が疑われるパケットをドロップする仕組みとして SmartNIC の活用を提案している．負荷の高い DDoS 処理を NIC へオフロードすることで，サービスやアプリケーションへより多くの資源，特に CPU を割り当てられる．

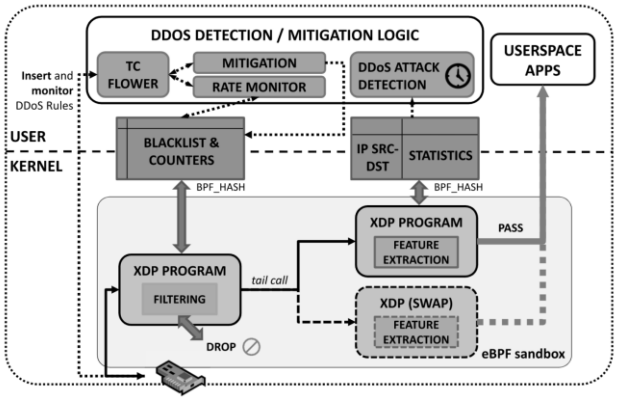


図 1 Miano らによる提案の概要 [23]

DDoS を緩和する方法の 1 つは，攻撃者と疑われる IP アドレスのブラックリストを用いたフィルタリングである．そこで SmartNIC による高速なハードウェアフィルタリングを期待するが，SmartNIC のフィルタリングに利用できるテーブルには限られた IP アドレスのエントリしか保持できない．Miano らが検証に用いた機器、構成で保持可能なエントリ数は 1〜2K ほどである[23]ため，上位の活発な攻撃者を優先してハードウェアフィルタリングし，その他はサーバホスト上で処理せざるを得ない．

そこで Miano らは SmartNIC によるフィルタリングだけでなく，extended Berkeley Packet Filter(eBPF)と eXpress Data Path(XDP)を利用したフィルタリングプログラムの組み合わせを評価している（図 1）. XDP によりカーネル空間内で，かつ NIC ドライバから TCP/IP スタックに渡る前のパケットを処理することで，高い効率を期待できる[24]. XDP はサーバホスト上だけでなく，SmartNIC など外部へのオフロードも可能である[25].

Miano らは 5 つのパターンを評価した．図 2 がパケットの破棄レートの評価結果であり，iptables(netfilter)よるソフトウェア処理（図中では Iptables），XDP をサーバホストまたは SmartNIC で動かす 2 パターン（図中では XDP Host と XDP SmartNIC），その 2 つに SmartNIC のハードウェアフィルタリングを組み合わせた 2 パターン（図中では HW＋XDP Host と HW＋XDP SmartNIC）である．

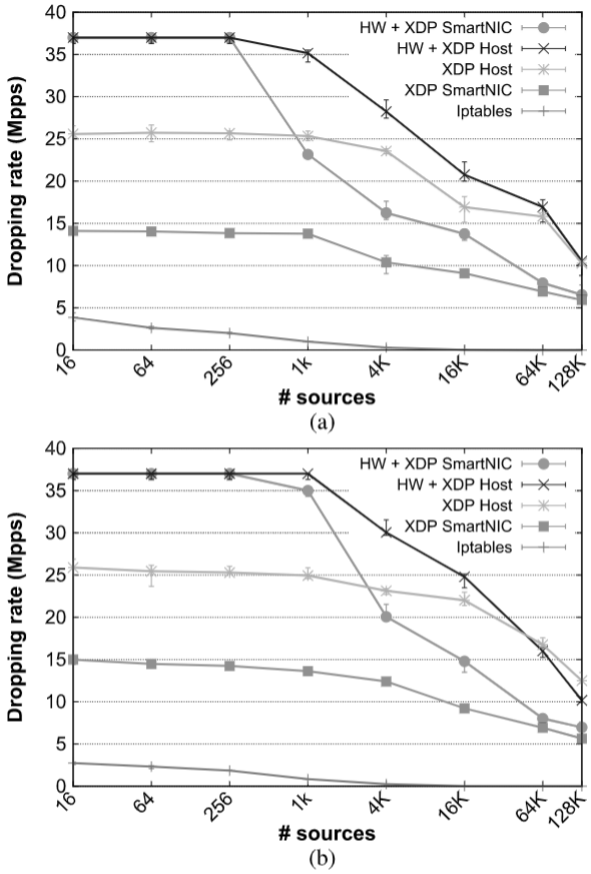


図 2 Miano らによる評価(パケット破棄レート)[23]

この評価で SmartNIC は 25Gbps のインタフェースを有しており，約 37Mpps がラインレートである．(a)は攻撃ソースが均等にトラフィックを生成した場合，(b)は正規分布の場合である．従来の TCP/IP スタックを利用した iptables の結果を他は大きく上回り，SmartNIC のハードウェアフィルタリングを利用したパターンでは，攻撃ソース数が 1K 近辺まではラインレートでの性能を達成している．これは SmartNIC に保持できる IP アドレスのエントリ数が影響していると考えられる．

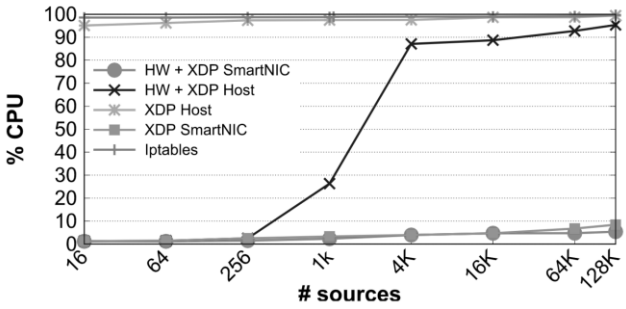


図 3 Miano らによる評価(ホスト CPU 利用率)[23]

図 3 はサーバホストの CPU 利用率である．SmartNIC を使わない 2 つのパターンでは攻撃ソースが少数の段階で飽

和したが，SmartNIC を使ったパターンでは期待通りサーバホストの CPU 利用率を低く維持，つまりオフロードできている．SmartNIC ハードウェアフィルタリングとサーバホストの XDP を組み合わせたハイブリッドのパターンで攻撃ソース数が 1K 近辺で利用率の急上昇が見られるが，図 2 から読み取れるようにパケットの破棄レート率も高く，ホストの CPU を利用してレートを上げていると分かる．

Miano らはこの研究で，DDoS 緩和処理の SmartNIC へのオフロードが有効であることを示したが，テーブルのサイズなどハードウェアの制約を意識したプログラミングが必要であると課題も提起している．

### 3.3 システムソフトウェア

アプリケーションに必要だが，責務を分離して外部化したい機能がある．アプリケーションの構成情報を保持するデータストアが代表的だ．このようなシステムソフトウェアも In-Network Computing が貢献できる領域である．

#### 3.3.1 分散合意

分散アプリケーションの設計においては，ネットワークの分断，故障などの障害に備え，データの整合性の確保が論点となる．例えばアプリケーションの構成や状態を管理する分散データストアは，そのうちの 1 つが利用できない状態であってもデータの不整合なくサービスを継続できることが望ましい．

分散したデータの更新で強い整合性を実現するためには，リーダを選出し，他はリーダの決めた値や順序に従う方法が実践的である．その際，何らかの合意を形成する必要があるが，Paxos[26]，Zookeeper Atomic Broadcast(ZAB)[27]，Raft[28]など実績あるアルゴリズムが存在する．

しかしこれらのアルゴリズムの実装では，強い整合性のトレードオフとして性能拡張性が犠牲になりやすい．SDN コントローラである Open Network Operating System(ONOS)[29]は Raft を採用しているが，高負荷時にコントローラが停止するリスクが指摘されている[30]．

また，合意形成に参加するコンポーネント間の遅延も性能拡張性や可用性に影響する．例えば災害対策のため Kubernetes のコントロールプレーンを距離の離れた複数のデータセンタへ分散したいというニーズがあるが，採用する分散データストアである etcd[31]が遅延によって性能と可用性の影響を受けやすく[32]，その制約を重視するとデータセンタ間の距離を確保し難い．

このような分散合意アルゴリズムの実装に関する課題を，In-Network で解決する研究が存在する[33][34][35]．スイッチや NIC へアルゴリズムを部分的に実装し，ハードウェアにより高速に，また，クライアントに近い場所で処理することで高スループットと低遅延を実現する．

表 1 In-Network Computing における
分散合意アルゴリズム実装の研究事例

| 研究 | アルゴリズム | 配置 | 言語 |
|---|---|---|---|
| Tu Dang ら[33] | Paxos | スイッチ | P4 |
| István ら[34] | Zookeeper Atomic Broadcast | NIC | 高位合成，Verilog，VHDL |
| Zhang ら[35] | Raft | スイッチ | P4 |

表 1 にアルゴリズム別の代表的な研究を挙げる．注目すべきはその配置先と実装に用いた言語である．

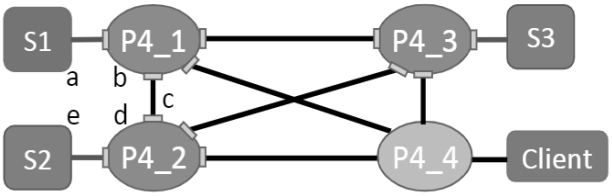


図 4 Zang らによる Raft-aware P4 スイッチの概念[35]

図 4 は Zang ら[35]による Raft-aware P4 スイッチ実験環境の概念図である．S1，S2 と S3 は Raft を使った分散ストレージである LogCabin[36]のノードを表し，S1 はリーダ，S2 と S3 はフォロワである．P4 スイッチは 4 台で構成され，P4_1～3 が Raft を意識して動く Raft-aware スイッチである．P4_4 は転送のみを行う．

Raft-aware スイッチは分散ストレージに対するハートビートへの応答や値の書き込みを代理する．また，クラスタへの新規メンバ追加などスイッチで完結しない処理は，分散ストレージへ要求を転送する．

このコンセプトで特に効果が期待できるのは，ストレージへの書き込み処理である．クライアントからの書き込み要求は Raft-aware スイッチで代理応答され，ストレージへの書き込みを待たない．ストレージに対してはバックグラウンドで値が書き込まれる．つまりクライアントへの応答には図 4 の a，e で表されるストレージへの書き込み処理が不要であり，低遅延での応答が期待できる．障害時の影響など，さらなる検証が必要ではあるが，代理応答するスイッチをクライアントと近接，つまり同じデータセンタに配置し，スイッチとストレージ間は非同期処理にできれば，ストレージは伝搬遅延が大きい遠隔のデータセンタでも配置できるため，災害対策へ寄与するだろう．

スイッチだけではなく NIC へ分散合意アルゴリズムを実装するアプローチもある．István ら[34]は ZAB のオフロードを行うフロントエンドを，FPGA を搭載する SmartNIC に実装した．図 5 がそのトポロジであり，ZAB の 3 つのフロントエンド（リーダ 1，フォロワ 2）はクライアントとスイッチを介して TCP/IP で通信するが，フロントエンドの間は

直接接続することで ZAB の合意形成を高速に行う．

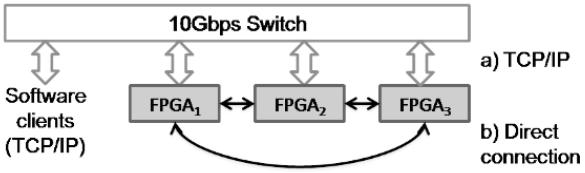


図 5 István らによる FPGA SmartNIC への ZAB 実装[34]

SmartNIC 同士を接続し特定の通信を高速化するアプローチは Microsoft の Catapult プロジェクト[37]での二次元トーラス構造でも知られている．

ここで挙げた研究ではアルゴリズムの実装に P4，もしくは FPGA 固有の言語を使用している．システムソフトウェアは専門性の高い技術者がコーディングする傾向があるが，分散合意など取り組む問題が複雑な領域ではより開発し易さが求められるだろう．Tu Dang ら[33]は P4 での実装で得た経験として，テーブル操作を中心とするプログラミングを習得するまでの時間，マクロが使える程度でモジュール化と再利用の仕組みがないこと，メモリレイアウト管理の難しさなどを指摘している．

### 3.4 ビジネスアプリケーション

データ集約型アプリケーションはデータセンタのトラフィック増加の主要因となっている．機械学習などデータ集約型アプリケーションを In-Network に配備することで，実行時間の短縮のみならず，トラフィックの削減も期待できる．

#### 3.4.1 ディープラーニング

ディープラーニングは近年，GPU の活用により大きくスループットが向上した領域であるが，学習対象のデータ量も増加しており，その性能拡張性は課題である．1 つのサーバに搭載可能な GPU には限りがあるため，一般的には複数のサーバ（ワーカ）へデータやモデルを分割し，並列に処理して解決する．各ワーカがそれぞれ処理した結果（勾配）を集約し，繰り返し学習して精度を高める．

集約方式にはワーカと独立したパラメータサーバで結果を集約し再配布を行う非同期型と，全てのワーカがタイミングを決めて結果を共有する同期型がある．非同期型はスループットに，同期型は精度に優れる．

Sapio ら[38]が提案する SwitchML は，スイッチに集約処理をオフロードし，結果をワーカへ同期的にマルチキャストすることでスループットと精度の両立を目指している[図 6].

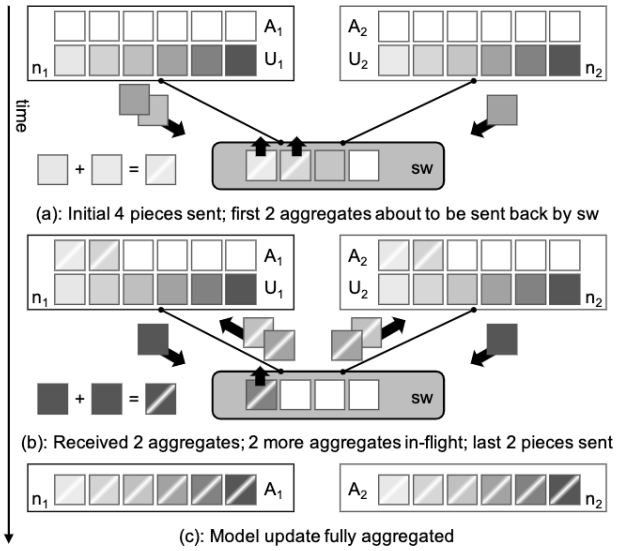


図 6 SwitchML による集約[38]

SwitchML は実装にあたって解決すべき 3 つの課題があった．

**CPU の制約**　ディープラーニングにおいて計算結果の集約は主に浮動小数点数のベクトルで行われる．しかし検証に用いたプログラマブルスイッチの CPU は浮動小数点数演算をサポートしていなかった．よってワーカ側で整数との型変換を行った．

**記憶域の制約**　それぞれのワーカが数百メガバイトの勾配を持つことは珍しくない．一方で検証に用いたプログラマブルスイッチでデータプレーン処理に利用できる記憶域は数十メガバイトであり，スイッチとしての転送テーブルなど他機能と共有する必要もあった．性能を考慮するとデータプレーン処理用の記憶域はチップ上の SRAM などで実現するのが妥当であり，容量の大幅な増加は短期的には期待できない．よって記憶域が少なく済むストリーム型の集約で解決した．

**パケットロス**　集約結果をスイッチからワーカへ同期して配布するため，パケットロスによる再送信の影響は大きい．SwitchML ではスイッチのデータプレーンでの複雑な実装を避ける目的で，ワーカ側でパケットロスを検知，再送信している．なお，ワーカからスイッチへ向けての再送信は勾配の再集約を招き精度低下につながるため，送受信を区別し集約結果の再送信は行わないよう工夫している．

表 2 SwitchML の性能評価[38]

| Model (abbrv) | Training throughput (image/s) | | | |
|---|---|---|---|---|
| | ideal | Multi-GPU | Horovod +NCCL | SwitchML |
| inception3 | 1132 | 1079(95.3%) | 99(70.6%) | 1079(95.3%) |
| resnet50 | 1838 | 1630(88.7%) | 911(49.6%) | 1412(76.8%) |
| vgg16 | 1180 | 898(76.1%) | 207(17.5%) | 454(38.5%) |

表 2 は SwitchML の性能評価結果である. 3 つの代表的な画像認識向けニューラルネットワークを 8 ワーカで処理した. 理論値（Ideal），サーバ 1 台に 8 基の GPU（Multi-GPU）を搭載した構成，ring all-reduce 型[39]の Horovod[40]，そして SwitchML を 8 サーバで構成した結果を比較している. 評価したニューラルネットワークによっては，ワーカ間の通信がネットワークを介さない Multi-GPU 構成と同等のスループットを達成している.

SwitchML は同期型で高精度，かつ高いスループットを実現するという目的を達した. また, ハードウェアの制約とパケットロスという課題も解決している．しかし検証すべき課題は残っており，その中でも特記すべきはサーバラックを超えた拡張性である．SwitchML は 1 つのスイッチを想定しており，ラック内のサーバを集約する ToR スイッチに向く．しかし計算規模や冗長性などから複数のラックが必要なケースを考慮し，複数スイッチでの階層化や機能分割が望まれる．SwichML の研究においてその課題は認識されており，ルートスイッチへの多段階集約で解決する案を示している．その案ではワーカ数に対して集約スイッチ数が少ない，つまり階層が少ない場合にルートスイッチの集約処理にかかる負荷が懸念されるため，ラックだけなくワーカの数に応じ，階層化や複数のルートスイッチを構成する必要がある.

## 4. 今後の課題と展望

In-Network Computing はデータセンタの抱える，主に性能に関する課題を解決する有望な手段であり，研究にとどまらず商用での実績も増えつつある. しかし同時に, 解決すべき課題は少なくない. この章では将来に向け, 課題と展望をまとめる.

### 4.1 課題

**プログラミングとテストの容易性** P4 エコシステムの本格化によって標準化が進めば，ハードウェアの多様性に対する懸念は解消されるだろう. しかし, ネットワークプログラミングが本質的に持つ難しさやデバイスの制約から，In-Network Computing のプログラミングに高い専門性が要求される状況は当面変わらないと考える．このサーベイで調査した研究においても，テーブル操作中心のプログラミングを習得する必要やモジュール化の難しさなど，多くの課題が指摘されている.

一方, 柔軟なプログラミングを志向するならば, ハードウェアによる高速な処理というネットワークデバイスの本来の長所を失いかねない. 可能性を過度に評価せず, トレードオフを考慮して議論する必要があるだろう.

また，従来のネットワークデバイスとはテストの考え方も異なる．これまでネットワークデバイスはベンダが出荷時にハードウェアへ固定的に埋め込んだ機能をテストしていた．しかしプログラマブルなハードウェアにはユーザがプログラムした機能を検証，テストするツールチェインが必要であり，十分とは言い難い．P4 プログラム向けにテストケースを自動生成する研究[41]は存在するが，さらなる議論と研究が必要だろう.

**障害からの回復性** ネットワークデバイスでアプリケーションを動かすのであれば，従来サーバやアプリケーションが回復の責務を有していたレイヤの障害に対して，回復手段を持たなければならない.

各デバイスが独立して動作し状態を持たないのであれば, 再起動や無効化で対処できるが，複数のデバイスが協調する場合や状態を持つサービスの回復は複雑である．これは In-Network Computing に限らない分散アプリケーションの課題であるが，プログラミングに制約のあるネットワークデバイスにおいて複雑な回復ロジックの実装は困難であり, シンプルな仕組みが求められる.

**アプリケーションの更新** アプリケーションが停止に至る原因は故障だけではない．アプリケーションの機能追加や修正に伴う更新作業も停止につながる．これは広い帯域を集約するスイッチで特に課題であり，Krude ら[42]はその影響を指摘している．例えばプログラムの更新に最大 50ms の停止を要するスイッチが 100GbE インタフェースを 64 ポート収容するケースでは，更新時に最大 37GiB のデータが影響を受ける.

Krude らは, 現状のプログラマブルスイッチが持つステージ（パース，マッチ、アクション）が 1 つにとどまり，実質 1 つのモノリシックなプログラムしか動かせないことを問題としている. そこで, ステージの入れ子構造による複数プログラムの並行動作とホットプラグ更新を提案した．この提案はマルチテナント，マルチアプリケーションの観点でも価値がある.

なお，アプリケーションが複数のネットワークデバイスに分散配備されるケースも考慮が必要である．バージョンの異なるプログラムが混在して問題が生じるならば，更新時の閉塞処理や経路制御など，アプリケーションとネットワークを統合的に制御する仕組みが求められる.

**消費電力** ハイパースケールデータセンタにおいて消費

電力はコストへの影響が大きな要素だが，その内ネットワークデバイスの電力消費量は大きくない. Google が 2017 年に行った調査[43]によると 5%である.しかし In-Network Computing の本格化により利用デバイスの種類や利用法が変化するならば，その増加が懸念される.

In-Network Computing の性能における特有の価値は低遅延である. 一方, スループットはアプリケーションによってはサーバの汎用 CPU でも目標を達成できる可能性があり，その選択において消費電力あたりの性能が主な論点となる.

Tokusashi ら[44]は処理量によって汎用 CPU とネットワークデバイスで電力消費の傾向が変わることに着目し，処理量に応じて汎用 CPU とネットワークデバイスをオンデマンドに切り替える提案をしている.

**セキュリティとプライバシ** In-Network Computing は主に性能でデータセンタネットワークに価値をもたらすが，それがセキュリティとプライバシを妥協する理由とはならない. 認証と認可, 信頼の範囲と粒度など幅広い観点での議論が必要である.

その中でも既に明確な課題として認識されているのが暗号化である．ユーザとサービスの終端ノード間でパケットが暗号化されるのならば, In-Network で可能なパケット操作は限定される. ヘッダなどメタデータの操作に徹する，または準同型暗号などを用いて暗号化したまま計算を行う必要がある[45].

**アプリケーションエコシステムとの分担と協調** In-Network Computing がネットワークの活用領域をアプリケーションの動作基盤へと広げている一方で，アプリケーションエコシステムがネットワークとの統合に取り組む動きもある．例えば Kubernetes エコシステムには，アプリケーション視点で必要なレイヤ 2，3 接続やネットワークサービスを割り当てる Network Service Mesh プロジェクト[46]がある. それぞれのエコシステムにおける取り組みは相反するものではないが，議論や実装の重複は否めない.

Kubernetes が代表的だが, 現在のアプリケーションエコシステムはオープンソースコミュニティが主導するケースが多い. 実装までのスピード感を重視し，リリース後にユーザのフィードバックを受け，改善のサイクルを短期間で繰り返すスタイルが好まれる．議論からの標準化を重視するネットワークのエコシステムとの間にある技術的，文化的なギャップをいかに埋め, 分担, 協調していくかは課題である.

### 4.2 展望

ユースケースによっては，In-Network Computing は既に実用段階にある．代表例は Microsoft のパブリッククラウドサービスで利用されている FPGA ベースの Azure SmartNIC[47]である．マルチテナントネットワークのためのカプセリング，NAT，ACL 適用，メータリングなどの処理をサーバの汎用 CPU からオフロードしている．アプリケーションから透過的に導入できる，アクセラレータとしての SmartNIC は珍しくなくなるだろう.

ただし, サーバ単体で完結せず, 複数のサーバの SmartNIC が同期して設定する必要がある用途では，相応のコントロールプレーンが必要である．エコシステムや製品市場が整うまでは，導入によって得られる効果が大きいデータセンタでの導入に限られるだろう.

一方でスイッチでの In-Network Computing はまだ研究の域を脱していない印象である．DNS などユーザアプリケーションから透過にできるネットワークアプリケーションやシステムアプリケーションから導入が期待されるが，インラインに配置するのであれば障害やメンテナンス時の影響が大きくリスクが高い．回復性や保守性についてさらなる研究が望まれる.

また、スイッチでの In-Network Computing はマルチテナント，マルチアプリケーションの実現も普及の鍵となるだろう. 特定の使い方のみに依存するスイッチは, 陳腐化の恐れがあるからだ. ディープラーニングはその好例で, 破壊的なアイデアがいつ提案されるか分からない．そしてそのアイデアはプログラマブルスイッチへの実装が適さないかもしれない. よって他の用途へ転用, 共用できる汎用性が必要である.

スイッチでの In-Network Computing は，アプリケーションを動かす基盤として価値を訴求する前に，パケット転送におけるプログラマブルスイッチの有用性と実用性を証明する必要があるだろう．SRv6 のカプセル化やサービスチェイニング，In-band Network Telemetry によるネットワークの可視化など，プログラマブルスイッチを活用できるユースケースとニーズは存在する．既存ネットワークへの部分的な適用から実績を重ねることで成熟すると考える.

なお, Ports ら[48]はユースケース, 用途を定量的に評価し, In-Network Computing の適性や適切な配備先を判断するための分類を提案している．パケットやワーキングセットのサイズによっては適さないとしながらも，漸近記法による定量化を試みている．分類には 3 つの評価軸がある.

**パケットあたりの操作数（Operations per packet）** 受信パケットあたりの操作数を示す. n をパケットサイズとすると，多くのユースケースでオーダーは$O(1)$もしくは$O(n)$である．$O(n)$を超えるものは処理量の懸念だけでなくプログラミングも複雑になりがちであり，制約のあるデバイスでは実装に困難が伴うだろう．分類にはそれぞれの頭文字を用い，C(Constant)，L(Liner)，G(Greater than liner)と表す.

表 3 Ports らによる分類[48]

| In-network primitive | Ops/packet | State/packet | Packet Gain | Class | Dominant |
|---|---|---|---|---|---|
| Network sequencing[49][50] | $O(1)$ | $O(1)$ | $O(\|replicas\|)$ | CC+ | Gain |
| Replicated storage[51] | $O(1)$ | $O(\|dataset\ size\|)$ | $O(1)$ | CLC | State |
| Caching[52][53] | $O(1)$ | $O(ln(\|dataset\ size\|))$ | $O(1)$ | CLC | State |
| DNN training(allreduce)[38][54][55] | $O(\|packet\|)$ | $O(\|packet\|)$ | $O(1/\|replicas\|)$ | LL- | Gain |
| DNN inference[56] | $O(\|input\ size\|^2)$ | $O(\|model\ size\|)$ | $O(1)$ | GLC | Ops |
| Database reductions[57] | $O(\|packet\|)$ | $O(\|elements\|)$ | $O(1/\|replicas\|)$ | LL- | Gain |
| Database hash joins[57] | $O(1)$ | $O(\|elements\|)$ | $< O(1)$ | CL- | State |
| Virtual networking[47] | $O(1)$ | $O(\|flow\ table\|)$ | $O(1)$ | CLC | State |
| In-band network telemetry[58] | $O(1)$ | $O(1)$ | $O(1)$ | CCC | Ops |

**パケットあたりの状態サイズ（State per packet）** パケット処理に際してどれだけの大きさのデータ，状態を保持するかを示す．n をパケットサイズとすると，パケットの大きさによらず定数であれば$O(1)$，パケットサイズに比例すれば$O(n)$とする．また，パケット個別のデータにとどまらないワーキングセットを保持する必要があればそれを s とし，オーダーは$O(s)$とする．操作数と同様に C(Constant)，L(Liner)，G(Greater than liner)と表す．L か G と評価できるユースケースでは，デバイスでプログラムが利用できるメモリ容量が配備先の判断ポイントとなる．

**パケットゲイン（Packet gain）** 受け取ったパケットに対し，どれだけの数のパケットを送信するかを示す．$O(1)$であれば，受信パケットと同数を送信する．$O(r)$は複製を r 送り，$O(1/r)$は集約のうえ送信することを表す．分類では-(less than constant)，C(Constant)，+(greater than constant)と表現する．-か+と評価できる場合，ネットワークトラフィックの削減やサーバの複製負荷の軽減を期待できる．

Ports らの定量化と分類は，客観的な判断の助けとなる．例えば表 3 に挙げた In-band network telemetry は全ての評価軸においてオーダーが定数でクラスは CCC と分類され，実装の難易度が高くないと判断できる．

分散アプリケーションの複製処理をネットワークデバイスにオフロードする Network sequencing のクラスは CC+で，パケットゲインが決定における支配的な要素である．このクラスはスイッチに配置することで複製の同報効果を期待できる．パケットあたりの操作数と状態サイズも定数であり，プログラミングや利用できるメモリに制約があるスイッチにも適用しやすい．

一方でパケットあたりの操作数と状態サイズが定数でなく，スイッチの制約に収まらないユースケースでは，より柔軟なプログラミングが可能で外部メモリも利用できる SmartNIC が配備先として適するだろう．

期待できる効果と難易度のバランスによっては，ネットワークデバイスでの処理に不向きと判断すべきケースもある．表 3 に挙げられていないユースケースの適性判断においても，Ports らの 3 軸評価による分類は有用である．

## 5. おわりに

本稿では，データセンタネットワークが抱える課題とその解決の方向性を，In-Network Computing の視点で概観した．一方で In-Network Computing の研究動向の網羅を目的とせず，データセンタネットワークに限定していることには注意が必要である．特に In-Network Computing はユーザに近いエッジでの活用も期待されており，それらについては IRTF COINRG[10]の参照を薦める．

なお、本稿の概要については[59]でも発表する。

インターネットを通じたアプリケーションの提供やクラウドコンピューティング，データ集約型アプリケーションの活用は今後ますます進み，それに合わせてデータセンタネットワークはさらなる変化を強いられるだろう．ゲームチェンジャーとなる技術が求められるが，その内の 1 つが In-Network Computing となることを期待している．

## 参考文献


[1] “Cisco Global Cloud Index: Forecast and Methodology, 2016-2021 White Paper (Updated November 19, 2018)”.

[2] “25 Gigabit Ethernet Consortium Rebrands to Ethernet Technology Consortium; Announces 800 Gigabit Ethernet (GbE) Specification”. https://ethernettechnologyconsortium.org/press-room/press-releases/25-gigabit-ethernet-consortium-rebrands-to-ethernet-technology-consortium-announces-800-gigabit-ethernet-gbe-specification-152/, (参照 2020-04-19).

[3] “NUMA Siloing in the FreeBSD Network Stack”. https://2019.eurobsdcon.org/talk-speakers/#numa , (参照 2020-04-08).

[4] Dean, J.. The tail at scale. Communications of the ACM 2013 vol: 56 (2) pp: 74-80

[5] “Introducing data center fabric, the next-generation Facebook data center network”. https://engineering.fb.com/production-engineering/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/, (参照 2020-04-12).

[6] “Use of BGP for Routing in Large-Scale Data Centers”. https://tools.ietf.org/html/rfc7938, (参照 2020-04-12).

[7] “Data Center Host to ToR Architecture”. https://docs.cumulusnetworks.com/cumulus-linux/Network-Solutions/Data-Center-Host-to-ToR-Architecture/#single-attached-hosts, (参照 2020-04-12).

[8] “LINE Data Center Networking with SRv6”. https://www.segment-routing.net/conferences/2019-09-20-SRv6-LINE-DC/, (参照 2020-04-12).
[9] “In-Network Computing”. https://www.sigarch.org/in-network-computing-draft/, (参照 2020-04-14).
[10] “Computing in the Network Research Group (coinrg)”. https://datatracker.ietf.org/rg/coinrg/documents/, (参照 2020-04-14).
[11] “Directions for Computing in the Network, draft-kutscher-coinrg-dir-01”. https://datatracker.ietf.org/doc/draft-kutscher-coinrg-dir/, (参照 2020-04-25).
[12] “P4 Language and Related Specifications”. https://p4.org/specs/, (参照 2020-04-25).
[13] Tennenhouse, D and Wetherall, D.. Towards an Active Network Architecture. ACM SIGCOMM Computer Communication Review Vol. 26, pp. 5-17
[14] “Requirement of Computing in network, draft-liu-coinrg-requirement-02”. https://datatracker.ietf.org/doc/draft-liu-coinrg-requirement/, (参照 2020-04-30).
[15] “Kubernetes”. https://kubernetes.io/, (参照 2020-04-30).
[16] “Using NodeLocal DNSCache in Kubernetes clusters”. https://kubernetes.io/docs/tasks/administer-cluster/nodelocaldns/, (参照 2020-04-25).
[17] Woodruff, J. Ramanujam, M. and Zilberman, N.. P4DNS: In-Network DNS. 2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)
[18] Zilberman, N. Audzevich, Y. Covington, A. and W. Moore, A.. NetFPGA SUME: Toward 100 Gbps as Research Commodity. IEEE Micro , vol. 34, pp. 32–41, 2014.
[19] “Name Server Daemon(NSD) ”. https:// nlnetlabs.nl/projects/nsd/about/, (参照 2020-04-25).
[20] Yazdinejad, A. Bohlooli, A. and Jamshidi, K.. P4 to SDNet: Automatic Generation of an Efficient Protocol-Independent Packet Parser on Reconfigurable Hardware. 8th International Conference on Computer and Knowledge Engineering (ICCKE 2018)
[21] Sabin, G and Rashti, M.. Security Offload Using the SmartNIC, A Programmable 10 Gbps Ethernet NIC. Proceedings of the IEEE National Aerospace Electronics Conference, NAECON. 2016 vol: 2016-March pp: 273-276
[22] “Cloudflare architecture and how BPF eats the world”. https://blog.cloudflare.com/cloudflare-architecture-and-how-bpf-eats-the-world/, (参照 2020-04-28).
[23] Miano, S. Doriguzzi-Corin, R. Risso, F. Siracusa, D. and Sommese, R.. Introducing smartnics in server-based data plane processing: The ddos mitigation use case. IEEE Access 2019 vol: 7 pp: 107161-107170
[24] Vieira, M. Castanho, M. Pacífico, R. Santos, E. Câmara, E. and Vieira, L.. Fast packet processing with EBPF and XDP: Concepts, code, challenges, and applications. ACM Computing Surveys 2020 vol: 53 (1)
[25] “eBPF Hardware Offload to SmartNICs: cls bpf and XDP”. https://www.netronome.com/products/agilio-software/agilio-ebpf-software/, (参照 2020-04-28).
[26] Lamport, L.. The Part-Time Parliament. ACM TOCS , 16(2):133–169, May 1998.
[27] Flavio, P. Junqueira, Benjamin C. and Reed, Marco Serafini.. Zab: High-performance broadcast for primary-backup systems. DSN’11 .
[28] Ongaro, D and Ousterhout, J.K.. In search of an understandable consensus algorithm. USENIX Annual Technical Conference , 2014.
[29] “Open Network Operating System”. https://www.opennetworking.org/onos/, (参照 2020-04-30).
[30] Hanmer, R. Jagadeesan, L. Mendiratta, V. and Zhang, H.. Friend or Foe: Strong Consistency vs. Overload in High-Availability Distributed Systems and SDN. 29th IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2018
[31] “etcd, a distributed, reliable key-value store for the most critical data of a distributed system”. https://etcd.io/, (参照 2020-04-30).
[32] “Hardware recommendations”, https://etcd.io/docs/v3.4.0/op-guide/hardware/#network, (参照 2020-04-29).
[33] Tu Dang, H. Canini, M. Pedone, F. and Souí, R.. Paxos Made Switch-y. ACM SIGCOMM Computer Communication Review Volume 46, Number 2, April 2016
[34] István, Z. Sidler, D. Alonso, G/ Zürich, E. Vukolić, M. and Vukoli, M.. Consensus in a Box: Inexpensive Coordination in Hardware. Open access to the Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16)
[35] Zhang, Y. Han, B. Zhang, Z. and Gopalakrishnan, V.. Network-assisted raft consensus algorithm. SIGCOMM Posters and Demos 2017
[36] “Logcabin: A distributed storage using raft” . https://github.com/logcabin, , (参照 2020-04-30).
[37] Putnam,A. Caulfiield, A. Chung, E. Chiou, D. Constantinides, K. Demme, J. Esmaeilzadeh, H. Fowers, J. Gopal, G. Gray, J. Haselman, M. Hauck, S. Heil, S. Hormati, A. Kim, J. Lanka, S. Larus, J. Peterson, E. Pope ,S. Smith, A. Thong, J. Xiao, P. and Burger D.. A reconfigurable fabric for accelerating large-scale datacenter services. Communications of the ACM 2016 vol: 59 (11) pp: 114-122
[38] Sapio, A. Canini, M. Chen-Yu, Ho. Nelson, J. Kalnis, P. Kim, C. Krishnamurthy, A. Moshref, M. Ports, K. and Richtárik , P.. Scaling Distributed Machine Learning with In-Network Aggregation. ArXiv ID 1903.06701v1
[39] Patarasuk, P. and Yuan, X. Bandwidth Optimal All-reduce Algorithms for Clusters of Workstations. Journal of Parallel and Distributed Computing , 69(2), 2009
[40] Sergeev, A. and Del Balso, M.. Horovod: fast and easy distributed deep learning in TensorFlow. ArXiv ID 1802.05799
[41] Nötzli, A. Khan, J. Fingerhut, A. Barrett, C. and Athanas, P.. P4pktgen: Automated test case generation for P4 programs. SOSR ’18: ACM SIGCOMM Symposium on SDN Research
[42] Krude, J. Hofmann, J. Eichholz, M. Wehrle, K. Koch, A. and Mezini, M.. Online Reprogrammable Multi Tenant Switches. In 1st ACM CoNEXT Workshop on Emerging in-Network Computing Paradigms (ENCP ’19)
[43] Barroso, L. Hölzle, U. Ranganathan, P. Martonosi, M.. The Datacenter as a Computer Designing Warehouse-Scale Machines Third Edition. Morgan & Claypool
[44] Tokusashi, Y. Dang, H. Pedone, F. Soul, R. and Zilberman, N.. The case for in-network computing on demand. In Proceedings of Fourteenth EuroSys Conference 2019 (EuroSys ’19)
[45] Burkhalter, L. Zurich, E. Hithnawi, A. Berkeley, U. Alexander, Z. Shafagh, H. Ratnasamy, S and Viand, A.. imeCrypt: Encrypted Data Stream Processing at Scale with Cryptographic Access Control. Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)
[46] “Network Service Mesh - The Hybrid/Multi-cloud IP

Service Mesh”. https://networkservicemesh.io/, (参照 2020-05-04).

[47] Firestonem D. Putnam, A. Mundkur, S. Chiou, D. Dabagh, A. Andrewartha, M. Angepat, H. Bhanu, V. Caulfield, A. Chung, E. Kumar, H. Somesh, C. Matt, C. Lvier, H. Lam, N. Liu, F. Ovtcharov, K. Padhye, J. Popuri, G. Raindel, S. Sapre, T. Shaw, M. Silva, G. Sivakumar, M. Srivastava, N. Verma, A. Zuhair, Q. Bansal, D. Burger, D. Vaid, K. Maltz, D. and Greenberg, A. Azure Accelerated Networking: SmartNICs in the Public Cloud. USENIX NSDI 2018 Networked Systems Design and Implementation pp 51-64

[48] Ports, D. and Nelson, J.. When Should the Network Be the Computer? Proceedings of the Workshop on Hot Topics in Operating Systems, HotOS 2019. 2019 pp: 209-215

[49] Li, J. Michael, E. and Ports, D. R. K.. Eris: Coordination-free consis-tent transactions using network multi-sequencing. In Proceedings of the 26th ACM Symposium on Operating Systems Principles (SOSP ’17) , Beijing, China, Oct. 2017. ACM.

[50] Li, J. Michael, E. Szekerres, A. Sharma, N. K. and Ports, D. R.K.. Just say NO to Paxos overhead: Replacing consensus with network ordering. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’16) , Savannah, GA, USA, Nov. 2016. USENIX.

[51] Jin, X. Li, X. Zhang, H. Foster, N. Lee, J. Soulé, R. Kim, C. and Stoica, I.. NetChain: Scale-free sub-rtt coordination. In 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18) , pages 35–49, Renton, WA, 2018. USENIX Association.

[52] Jin, X. Li, X. Zhang, H. Soulé, R. Lee, J. Foster, N. Kim, C. and Stoica, I.. NetCache: Balancing key-value stores with fast in-network caching. In Proceedings of the 26th ACM Symposium on Operating Systems Principles (SOSP ’17) , Beijing, China, Oct. 2017. ACM.

[53] Li, X. Sehti, R. Kaminsky, M. Andersen, D. G. and Freedman, M. J.. Be fast, cheap and in control with SwitchKV. In 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16) , pages 31–44, Santa Clara, CA, 2016. USENIX Association.

[54] Luo, L. Nelson, J. Ceze, L. Phanishayee, A. and Krishnamurthy, A.. Parameter Hub: A rack-scale parameter server for distributed deep neural network training. In Proceedings of the ACM Symposium on Cloud Computing , SoCC ’18, pages 41–54, Carlsbad, CA, USA, 2018. ACM.

[55] Sapio, A. Abdelaziz, I. Aldilaijan, A. Canini, M. amd Kalnis, P.. In-network computation is a dumb idea whose time has come. In Pro-ceedings of the 16th Workshop on Hot Topics in Networks (HotNets ’17) , Palo Alto, CA, USA, Nov. 2017. ACM.

[56] Fowers, J. Ovtcharov, K. Papamichael, M. Massengill, T. Liu, M. Lo, D. Alkalay, S. Haselman, M. Adams, L. Ghandi, M. Heil, S. Patel, P. Sapek, A. Weisz, G. Woods, L. Lanka, S. Reinhardt, S. K. Caulfield, A. M. Chung, E. S. and Burger, D.. A configurable cloud-scale DNN processor for real-time AI. In Proceedings of the 45th Annual International Sym-posium on Computer Architecture , ISCA ’18, pages 1–14, Piscataway, NJ, USA, 2018. IEEE Press.

[57] Lerner, A. Hussein, R. and Cudré-Mauroux, P.. The case for network accelerated query processing. In Proceedings of the 9th Conference on Innovative Data Systems Research (CIDR ’19) , Asilomar, CA, USA, Jan. 2019. VLDB / ACM.

[58] Kim, C. Bhide, P. Doe, E. Holbrook, H. Chanwani, A. Daly, D and Hira, M.. In-band network telemetry. https://p4.org/assets/INT-current-spec.pdf, 2016.

[59] 真壁 徹, 宇多 仁. In-Network Computing から見る データセンタネットワークの研究動向と課題. 信学技報, vol. 120, no. 125, IN2020-11, pp. 13-18, 2020 年 8 月.