OpenFlow を利用した QoS 認証サービスプレーン導入に関する基礎的検討

宮田宏[†1]　並木美太郎[†1]　佐藤未来子[†1]

プラント等でのプロセス制御において Backhaul を利用した制御が期待されている．Backhaul を用いたプロセス制御には実時間性を確保するために QoS の利用が求められるが，QoS には優先度の詐称を防ぐ手段が無い．著者らはこのような詐称を防ぐために QoS 認証機能を提案してきた．本稿では，この QoS 認証機能をネットワークレイヤに依存せず，よりスケーラブルで冗長性を持ったものにするための課題を整理し，その解決策を検討した．検討の結果としてルータ以外での QoS 認証機能がないこと，スケーラビリティがないこと，認証機能の更新性がないことを示し，検討に基づき QoS 認証技術を，OpenFlow を用いてオフロードする方式を設計した．本稿では，その設計と，最も根幹となるスイッチから QoS 認証機能を呼び出すための機能の試作による評価を示す．

An Fundamental Study of QoS Authentication Service Plane with OpenFlow

HIROSHI MIYATA[†1]　MITAO NAMIKI[†1]　MIKIKO SATO[†1]

1. はじめに

近年，石油や化学プラントのような Industrial Automation System (ICS)では，Wireless Sensor Network (WSN)を導入し，より簡便に多くのプラント内の情報を収集し，制御を行いたいという要求がある．これを実現するために制御室と WSN を接続するための有線や無線を含むプラントワイドなネットワーク(Backhaul)の導入が進められている．また，プラントには監視カメラや保全端末等のような，ネットワーク要件や管理者の異なる種々のサブシステムが導入されており，これらのサブシステムも Backhaul を共有したいと言う要求もある．(図 1)

このように，プラントにおいてもネットワークを積極的に活用しようという流れがある一方で，STUXNET[1]に代表されるようなプラントを標的とした攻撃も顕在化しており，サイバーアタックへの懸念が広がっている．プラントを標的とした攻撃は，プラントの誤制御を引き起こし，操業の中断やプラントの爆発等の取り返しのつかない事態を招く可能性がある．

Backhaul を導入するにあたっては，このようなリスクへの対策が不可欠となっている．Backhaul のガイドラインを述べている[2]によれば，サブシステムを Backhaul に接続する際には，サブシステム内を保護するために Firewall 等を用いた境界防衛を行う事を推奨するなど，多種サブシステムの接続する Backhaul はそれぞれのサブシステムにとって安全ではないと位置づけられている．†

プロセス制御を行うためにはスケジュールされた時間に機器間のデータ授受が必要であるが，Backhaul には多様なトラフィックが混在することになるため通信の遅延やジッタが生じる．そこで，実時間通信を提供するために Quality of Service (QoS)の利用が求められている．しかしながら，DiffServ[3]等の QoS 技術は，ネットワーク機器がパケットの優先度を特定するために用いる属性値(QoS 属性)を認証する機能を持たない．すなわち，だれもが優先度の高いパケットを送信する事ができ，優先処理を行う機器はその値を検証することなく指定された優先度に応じて優先処理を行う．したがって，悪意を持ったユーザが無線等を用いて Backhaul に接続できれば，高優先度になりすましたパケットを送信し，プロセス制御パケットの遅延や紛失を起こす攻撃が可能である．本稿ではこのような攻撃を QoS Spoofing と呼ぶ．QoS Spoofing を防ぐためには，経路上のネットワーク機器が QoS 属性値の認証を行う必要がある．そこで，本稿では QoS 属性値を認証するための機能を QoS 認証サービスプレーンという群で提供するための技術に

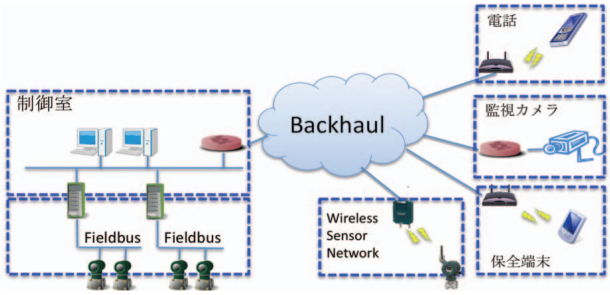


図 1 プラントにおける Backhaul 利用

†1 東京農工大学
Tokyo University of Agriculture and Technology

ついて検討する．

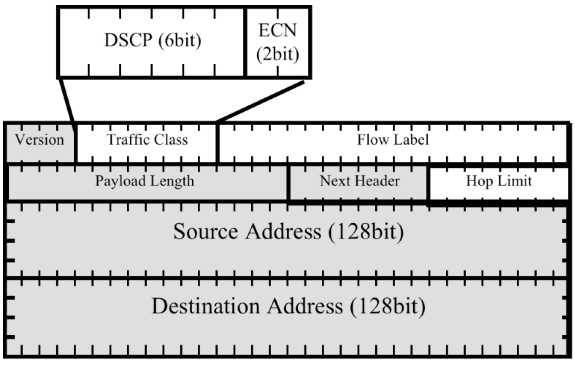
本稿の 2 章では既存技術について述べ，既存技術に対する本技術の違いを明確にする．3 章では課題と目標を明確にし，4 章では設計を，5 章では試作の状況を説明する．6 章で考察を示し，最後に 7 章でまとめと今後の課題について述べる．

2. 既存技術

QoS Spoofing 攻撃の本質的な問題点は，途中経路の機器が QoS 属性の偽造を検出できる仕組みが無いと言う点である．例えば，現在使われている Diffserv とよばれる QoS 技術においては, パケット内の IP ヘッダ内にパケットの優先度を示す属性値である DSCP 値を持つが，途中のネットワーク機器はこの値を変更可能である．さらに，実際に優先処理を実行する途中のルータにおいてこの値が正しい権限を持った機器によるものである事を認証する仕組みは提供されていない．

パケットの偽装を防ぐための技術として広く普及している技術に IPsec がある[4]. IPsec は送信者と受信者が共通の秘密鍵を持ってパケットの認証や暗号機能を提供する．IPsec の中でも， AH[5]というプロトコルは，IP ヘッダの認証機能を提供する．しかしながら，AH においても Flow Label や DSCP 値等一部のフィールドを認証対象外としている(図 2). また, 認証対象となるフィールドに対しても，受信者は改竄を検出できるが，途中経路の機器が改竄を検出する目的では利用できない．

パケットの偽装を End-to-End だけでなく途中の各ネットワーク機器において検出する技術が Packet Level Authentication (PLA) [6]において提案されている．PLA では電子証明書をパケットに付加し公開鍵暗号方式を用いたデジタル署名により改竄を検出している．一般に公開鍵暗号方式は共有鍵暗号方式に比べ多くの計算処理を必要とするため，パケット単位での認証には負荷が高く利用されてこなかったが，PLA では，楕円鍵暗号方式を用いれば公開鍵暗号方式であっても高速に処理ができるという点に着目し，公開鍵暗号方式を利用している．また，PLA は電子証明書をパケットに付加する事で暗号鍵の配布の必要がないという利点がある．しかしながら，PLA においても，IPsec と同様に途中継路で書き換えられる可能性のあるフィールドについては認証対象から外している．Diffserv で利用する DSCP は途中のルータが書き換える可能性があるため認証されない．



DSCP: differentiated services codepoint
ECN: Explicit Congestion Notification
網掛部は Authentication Header の認証対象フィールド

図 2 IPsec の保護対象フィールド

また，QoS の技術には DiffServ 以外にも IntServ [7]や，FlowLabel を使った QoS 技術 [8]のようにいくつかの提案が有り，それぞれの QoS の技術はパケットの優先度分類に用いるフィールドが異なるため，認証対象となるフィールドも異なる必要がある．しかしながら，PLA ではこのような検討がなされておらず，利用できる QoS 技術としては IntServ に限定されてしまう．さらに，[9]によれば楕円鍵暗号方式を用いることにより演算が高速になったとはいえ，ソフトウェアで認証した場合，デジタル署名に約 2.37 ミリ秒,ルータ一台あたりの検査時間には約 6.0 ミリ秒必要であるとの報告がある．例えば，End-to-End で二台のルータを利用し，受信機器でも同様に認証を行うとした場合，単純計算で 20m 秒の遅延が起こるため，実時間通信においては影響が大きいと言える．また，PLA を用いる事で増えるパケットサイズは 89～125 Octet であり, プロセス制御で用いられる FF-HSE[10]において実時間通信が求められる Publisher/Subscriber 通信を IPv6 パケットに格納した場合のサイズである 104～168 Octet と比較して大きいと言える．このため，制御用の優先帯域は PLA を利用しない場合の 1.5 倍～2.2 倍必要となる. これらの事から PLA はプロセス制御における QoS の保護には負荷が大きいと言える．

Chris らは，PLA と同様に IPsec 等では提供できない Hop-by-Hop でのパケットの認証を提供するために，TinySec[11]を提案している．TinySec の特徴は，1)センサネットワーク向けの技術でありエネルギー，遅延，帯域に対する影響が 10%以下である，2)共有秘密鍵を用いてパケット全体の Massage Authentication Code(MAC)を生成しパケットの完全性を保証する，3)メッセージの送信元の認証，4)メッセージ暗号化機能の提供，5)リプレイ攻撃の防止機能の提供，と言ったものが挙げられる．

TinySec では，1)の設計方針により PLA の持つ帯域や遅延に対する影響の大きさは低く抑える事ができているが，同時にパケットフォーマットも省電力無線向けに最適化を図っているため，IP ネットワークや Ethernet との互換性が無く，TinySec 対応の WSN での利用に限定される．

ここまで述べてきたように, 先行技術では IP ネットワーク上で QoS Spoofing を防ぐと言う観点を持ったものは無い．そこで, 著者らは sQoS [12]において QoS Spoofing を防ぐための技術を提案してきた．この技術はパケットの QoS 属性に対して送信機器で認証値を作成し，経路上のルータが

QoS 処理する前に認証値を認証する(図 3). sQoS は以下のような特徴を持つ. 1)保護すべき QoS 属性を選択する事が可能である(QoS 技術独立), 2)共有秘密鍵を用いた HMAC[13]を用いて QoS 属性の改竄を検出可能である(QoS 属性保護), 3)シーケンス番号を用いる事で正しいパケットをキャプチャし再送する攻撃への対策が可能である(Replay 防止), 4)パケットの認証値やシーケンス番号, 保護フィールドの情報等の必要な情報は全て IPv6 の Hop-by-Hop Option 拡張ヘッダ[14](HbH)に格納している(IPv6 互換), 5)HMAC によりパケットの認証処理が高速に実現できるため実時間性通信への影響が極めて低い(高速処理). 2)や 3)で用いる技術は IPsec のものと共通する部分が多いが, 途中の各ネットワーク機器で認証を行うと言う点で IPsec にはない QoS 属性の検証を提供している. 4)は Tinysec に欠けている Ethernet や IP への互換性を提供している. 1)や 5)は PLA に欠けていた QoS Spoofing 対策と言う観点を盛り込んでいる.

sQoS は既存技術に無かった QoS Spoofing への対策を目指して提案された技術であるが, 以下のような問題点も持つ. 1)IPv6 互換であるためルータでの処理が前提となり, 途中経路の Layer2(L2)スイッチでは対応できない. すなわち Layer 依存性が有る, 2)sQoS ではルータへ QoS 属性の認証機能を組み込む必要があるが, 一般的にルータはパケットの転送処理機能については ASIC 等を用いて高速に行えるものの, 演算処理に利用する CPU の性能は高くなく, HMAC のように演算を要する処理を, 実時間性を提供しながら大量に処理することができない. すなわち, スケーラビリティに問題がある, 3)QoS 属性認証プログラムにセキュリティ強化等の修正が必要になったとき柔軟に置き換えられない. すなわち, プログラム更新性の問題がある.

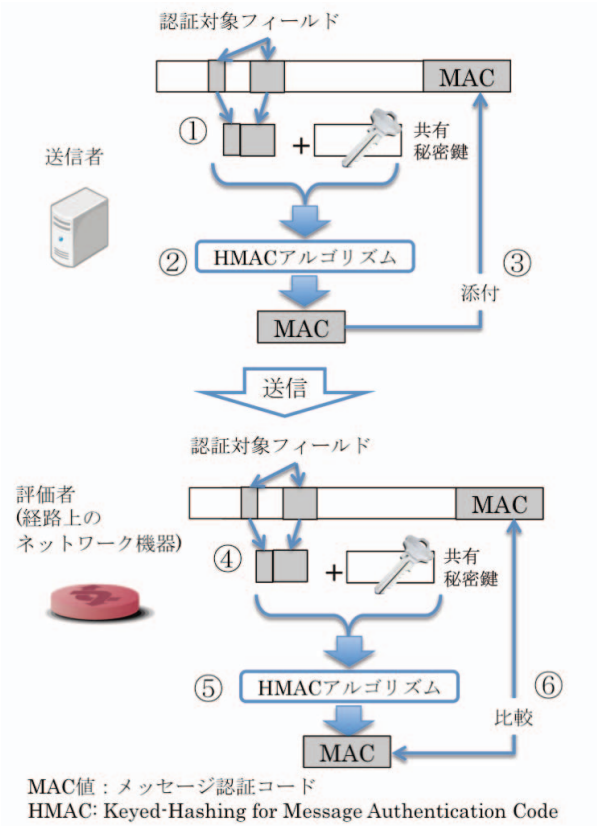


図 3 sQoS の処理概要

3. 課題と目標

セキュリティポリシやアプリケーションの異なる多種サブシステムのトラフィックが混在する Backhaul を用いて安全にプロセス制御を行うためには, QoS Spoofing というプロセス制御に特有の脅威を防ぐ必要がある. 先行技術においてもパケット単位でのセキュリティを提供するものがあったが, そのほとんどが実時間通信に対する攻撃である QoS Spoofing 対策という観点をもたなかった. 唯一この観点を持った sQoS にも, 1)Layer 依存性, 2) スケーラビリティ, 3)プログラム更新性といった三つの問題点がある.

そこで本稿では, Backhaul を用いた安全かつ柔軟なプロセス制御を行う事を目的とし, その実現のための技術を QoS Spoofing 対策の観点を持つ sQoS を基に検討する. 検討にあたっては, sQoS の持つ三つの問題点の解決が不可欠である. 本稿では, これらの課題に対する検討を進め, 解決策を提案し, 提案する手法の最も根幹となる機能について設計し, 実現性と有用性を検証する.

以下に, 先行技術である sQoS の三つの問題点に対する検討項目を示す.

3.1 Layer 依存性

sQoS は IPv6 の HbH ヘッダを用いているが, IP ルーティングを行わないスイッチはこのヘッダを検証しない. QoS の技術は Layer 毎に異なり, ルータはネットワーク層である IP ヘッダに含まれる QoS 属性値に基づき処理を行い, スイッチは Ethernet ヘッダに含まれる QoS 属性に基づき処理を行う. それぞれの QoS 属性値は 6bit, 3bit となっており, 分類できるクラスの数が異なるうえ, ルータとスイッチの優先制御の動作も異なる. したがって, End-to-End で一貫した優先制御を行うためには, 共通の QoS 属性値を用い, 共通の優先制御処理を行うことに加えて, 共通な認証技術が必要となる. これを実現するためには, ネットワーク機器が従来の処理対象ヘッダに留まらず, 複数の層のヘッダを評価し処理を行うことが求められる. そこで, ネットワーク機器の QoS 機能と QoS 属性認証機能の Multi-Layer 対応を検討する.

3.2 スケーラビリティ

[15]によればプラントにおけるセンサやアクチュエータの数は 16,000 個と非常に多く, 無線機器の普及等によりその数は増える傾向にある. したがって, 現時点で大量のトラフィックを処理できることに加え, 将来トラフィックが増えた際に処理能力を柔軟に拡張できる必要がある. そこ

で，このような処理能力拡張性を検討する．

さらに，このような拡張を複数の機器を用いて段階的かつ自律的に行えれば，同時に冗長性も提供できるためさらに有効である．ただし，sQoS はリプレイ防止のためのシーケンス番号を利用しているため，複数機器にフロート毎のステートを持つ．自動的に処理を分散してスケーラビリティを提供する場合には，ネットワーク機器内でフロー毎に管理されているステートに一貫性が必要である．そこで，リプレイ対策のステート管理方法を検討する．

3.3 プログラム更新性

認証に用いられる暗号化やハッシュ技術は日進月歩であり，今日安全な技術も将来にわたり安全である保障は無い．したがって，sQoS で利用する認証技術は適宜見直しが必要である．ネットワーク機器によっては特定の暗号アルゴリズムを ASIC 内に持つものもあるが，全ての機器が共通に持つ訳ではなく，また将来にわたり利用できるとは限らない，したがって，利用する認証技術を適宜更新できるようにネットワーク機器からの独立性を検討する．

4. 検討

3 章で示した四つの検討項目に対して検討した内容を以下に示す．

4.1 Multi−Layer 対応

前述したように Layer 依存性を解決するためには，Layer 間で共通して利用できる QoS を認証する技術が必要である．このような技術では認証のための情報を格納するヘッダが必要となるが，Ethernet ヘッダはルータ等の機器が置き換えてしまうことからこの要件を満たさない．一方，IP 層以上のヘッダであれば End-to-End で情報が維持される．したがって QoS 認証に利用する属性は IP 層以上のヘッダに格納すべきである．同様の理由で End-to-End で共通して利用できる QoS 属性 IP 層以上のヘッダ内情報である必要がある．このような属性情報を用いて，サブネット内通信，サブネット間通信において共通の QoS 認証機能と QoS 制御を提供する．このような Multi-layer の情報を利用した QoS 機能を本稿では Unified QoS と呼ぶ．

4.2 処理能力拡張性

QoS の属性検証機能(以下 sQoS ノードと呼ぶ)のスケーラビリティを提供するためには，計算能力の限られたネットワーク機器から QoS 属性検証機能を外部の機器にオフロードし，計算能力の高い外部の計算能力を活用する事が重要である．このような構成をとる事で，転送機能と sQoS ノードを別々に増強可能である．また， QoS ノードは一台の機器で提供するのではなく，複数の機器を利用できるようにする事で段階的な増強，負荷の分散，冗長化を図る事が可能となる．このような特徴を利用するためにはオフロード先の sQoS ノードを柔軟に切り替えられる Flexible な接続機能を備えればよい．

QoS ノードをオフロードする際，認証機器とスイッチ間の通信に通常のデータ転送経路を利用すると，認証のために本来守るべきネットワーク帯域を消費する事になってしまい，認証対象パケットを送付する事で結果的に実時間性通信を阻害する事が可能となってしまう．したがって，認証のための通信には，通常の守るべきネットワークとは異なるネットワークを用いて，このような攻撃の影響を避ける必要がある．このように通常の転送経路とから独立した通信経路を本稿では Out-of-Band 通信路と呼ぶ．

4.3 リプレイ対策のステート管理

sQoS ではリプレイ攻撃対策のためにフロー毎のシーケンス番号を用いており，sQoS ノードは内部的にそのステートを保持する必要がある．したがって，sQoS ノードを切り替えるためには，sQoS ノード間で，内部で管理しているステートを共有しなければならないという制約がある．そこで，本研究ではステートの管理を不要とするために，シーケンス番号ではなく時刻を用いたリプレイ防止対策を利用可能とする．具体的には，送信者がパケットを生成する際に，ヘッダに Time-stamp を格納する．途中のネットワーク機器は一定の許容時間を定義し，受信したパケットが受信した時刻に比べ一定以上古い物は破棄する．

時刻を用いる場合，許容された一定の時間内はリプレイを許すことになるため厳密にはリプレイを防いでいないが，許容する時間を短くすることでリプレイされるパケットの数を減らすことができる．したがって実時間性を守るという観点では十分であるといえる．また，ICS ネットワークでは，操作の記録やリアルタイム通信等の要求が高いため，一般に高い精度の時刻同期を採用している．したがって，時刻の利用は ICS に適したアプローチであるといえる．

4.4 機器独立性

機器独立性を提供する際にもオフロードは有効に働くと考えられる．また，オフロードすることで，認証方式の変更で計算負荷が高くなった場合でも，容易に計算能力を拡充する事が可能である．したがって，ネットワーク機器からの独立性を提供するためにも，オフロードを用いる．sQoS ノードのオフロードの際の検討内容は既に 4.2 で示した物と同一である．

4.5 検討のまとめ

sQoS の持つ三つの課題に対策するための四つの検討項目に対する検討結果をまとめる．まず，以下の三点を基本方針とする．(1)sQoS ノードはネットワーク機器から外部機器にオフロードする，(2)ネットワーク機器としては Multi-Layer 対応の機器を用いる，(3)先行研究の sQoS を改良して利用する．

この上で，以下の四つの機能を実現し，sQoS の持つ三つの問題点を解決する．

(F1)Unified QoS

IP 層以上の複数の情報に基づきレイヤによらず利用で

きる QoS の認証技術と QoS 技術により，Multi-Layer 対応を解決する．

(F2)Flexible 接続

複数のオフロード先の QoS ノード群を用意し，スイッチを適切な sQoS ノードのインスタンスに柔軟に接続する技術により処理能力拡張性と機器独立性を解決する．

(F3)Out-of-Band 通信路

ネットワーク機器が QoS 処理する前に，sQoS ノードを専用のネットワークを用いて呼び出す技術により安全なオフロードを可能とし処理能力拡張性と機器独立性を解決する．

(F4)Time-stamp ベースリプレイ対策

時刻情報をパケットに持たせる事で，リプレイ対策のステート管理を解決する．

5. OpenFlow を利用した QoS 認証サービスプレーン

図 4 に 4 章の検討に基づきプラント内の機能を配置し，各機能の役割を以下に示す．

(1) **スイッチ**

Backhaul を構成するネットワーク機器としては，OpenFlow 対応のスイッチを用意する．OpenFlow のスイッチは OpenFlow コントローラからの指示に基づき，Multi-Layer の情報を元にパケットを分類し，処理することができる．すなわち，(F1)Unified QoS の QoS 機能を提供するために必要である．

(2) **OpenFlow コントローラ**

OpenFlow コントローラは OpenFlow スイッチの動作を制御する機能であり，スイッチ内の帯域制御は OpenFlow コントローラから操ることが可能である．OpneFlow スイッチを使う上では必須な機能である．すなわち，(F1)Unified QoS の QoS 機能を提供するために必要である．

また，OpenFlow スイッチの経路制御や帯域制御は

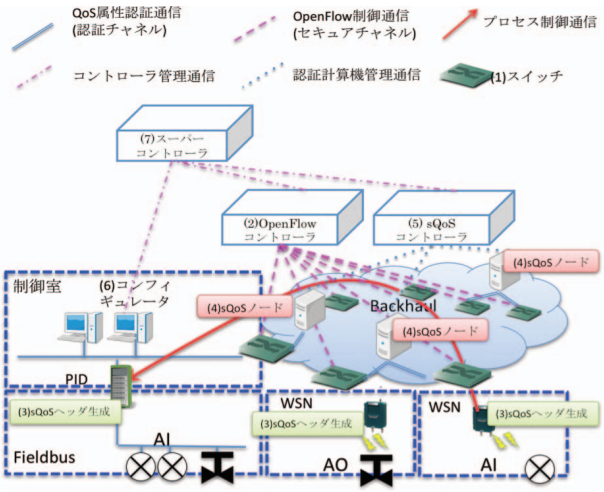


図 4 プラント内機能配置図

OpenFlow コントローラから自在に操ることが可能である．したがって，(F3)Out-of-Band 通信路の提供にも必要である．

(3) **sQoS ヘッダ生成**

sQoS ヘッダを生成する機能であり，本機能はパケットの送信元で動作する．パケットの送信時に QoS 属性や時刻等から sQoS ヘッダを生成する．IPv6 の拡張ヘッダに挿入する sQoS ヘッダを生成するため，経路上で失われる事が無く，(F1)Unified QoS の QoS 認証機能を実現するために必要な機能である．また，sQoS ヘッダに時刻情報を入れるため (F4)Time-Stamp ベースリプレイ対策にも必要である．

(4) **sQoS ノード**

受け取った sQoS ヘッダの認証を行う機能であり，スイッチの外部に配置される．本機能はフレーム全体をアプリケーションで検査する事が可能であり，(F1)Unified QoS の QoS 認証機能を実現するために必要な機能である．また，sQoS ヘッダの時刻情報を検証するため(F4)Time-Stamp ベースリプレイ対策にも必要である．

(5) **sQoS コントローラ**

sQoS ノードの動作状況を把握し，スイッチが利用する最適な sQoS ノードを決定し，設定を行う．sQoS ノードの負荷に応じて計算能力の高いものをスイッチに紐付ける事で，台数を増やし負荷を分散したりすることができる．本機能は(F2)Flexible 接続に必要な機能である．

(6) **コンフィギュレータ**

従来からプロセス制御に存在する機能であり，プロセス制御における制御通信要件の管理やスケジューリングを行う機能である．センサやアクチュエータの通信の設定はこの機能が行う．sQoS コントローラは本機能の持つ管理情報を用いる事で最適な sQoS ノードを選択できるため，(F2)Flexible 接続に必要な機能である．

(7) **スーパーコントローラ**

コンフィギュレータや，認証コントローラ，OpenFlow コントローラらの要件を受け，これらの間の要件の調整を行う機能であり，最終的な判断を行う機能である．スイッチに最適な sQoS ノードを接続させるため，(F2)Flexible 接続に必要な機能である．

このような構成をとる事で，Unified QoS 機能をスイッチおよび sQoS ノードにおいて提供し，これらの間を条件に応じて Flexible に Out-of-Band 通信路を用いて接続することが可能となる．この結果，sQoS のもつ三つの問題点を解決できる．本稿では複数 sQoS ノードを用意し，これらの同一サービス群から，各スイッチに最適な sQoS ノードを自動的に選択し接続する機能をサービスプレーンと呼ぶ．

6. 設計

6.1 OpenFlow スイッチの利用

Unified QoS を実現するためには，Multi-layer の情報を利用した QoS 機能を提供できるネットワーク機器が必要で

ある．そこで，Multi-Layer のヘッダを評価し，Layer によらない動作をする事ができる OpenFlow スイッチを利用する．パケットの分類のルールや QoS 制御内容はコントローラから事前に設定される．

6.2 Time-stamp の利用

sQoS では QoS Spoofing を防ぐための情報を sQoS ヘッダに格納している．リプレイ攻撃を防ぐシーケンス番号も同様に sQoS ヘッダに含まれる．本研究では，4.3 章の検討内容を受け，リプレイ防止のためにシーケンス番号に加え時刻を利用可能にするため，sQoS ヘッダを拡張する．

図 5 に本提案で利用するために改良した sQoS パケットフォーマットを示す．sQoS ヘッダは IPv6 の HbH ヘッダを利用する．4byte あるヘッダ部に含まれる Option Type にはこの HbH ヘッダに含まれるオプションが sQoS ヘッダである事を示すために 0x1E という値を設定する．

認証対象フラグは 1byte の長さを持ち，それぞれのビットが認証対象となるフィールドを示す．これにより DSCP, IP アドレス等のさまざまな QoS 属性を認証することが可能となる．

アンチリプレイフラグ(ARF)にはアンチリプレイフィールドに入れる値の種別を指定する．0 はシーケンス番号を示し， 1 は時刻(マイクロ秒)を示す．

アンチリプレイフィールドには同一パケットの再送による QoS 阻害攻撃を防ぐための値を格納する．シーケンス番号が指定された時は，送信者が送信するパケット毎に 1 つずつ増やしていくシーケンス番号を格納する．時刻が指定された時には，送信者が送信時刻を格納する．時刻を指定された時，受信したパケットの持つ時刻が，受信時刻と比べ 1 ミリ秒おそければ破棄する．これにより，Time-stamp ベースリプレイ対策が可能となる。

MAC には HMAC アルゴリズムにより生成した MAC 値を格納する．本提案では HMAC-SHA1-96[17]を用いるため，12byte の長さを持つが，この長さは利用するアルゴリズムにより変わるものである．

6.3 Out-of-Band 通信路の設計

本研究では，sQoS には無かった機能として，スイッチから sQoS ノードをオフロードし，サービスプレーンとして利用する．一般的に，OpenFlow ネットワークにおいてパケットの検査を行うような場合，二つの手法がとられる．一つはパケットを別の機器に転送しそこで処理を行い，処理を行った機器から最適な経路制御に基づきパケットを宛先に対して転送する．例えば，Firewall の例では，あるスイッチにおいて検査対象パケットを検出すると，そのスイッチから強制的に Firewall へ送られ，その後他のスイッチへ転送される．もう一つは OpenFlow コントローラへ通知し，判断を待つ手法である．前者の場合，パケットは検査を受けるためにネットワーク帯域という守るべき資源を消費することになる．例えば，スイッチから QoS 属性の認証機器にパケットを送る際，スイッチの優先 Queue を消費することになる．したがって，このような構成では QoS Spoofing を防げない．後者の場合，コントローラとの通信は TCP で行うためオーバーヘッドが大きいうえ，コントローラに負荷が集中するという問題がある．そこで，本研究では，本来の通信経路を保護するために，検査用のパケットを通常の転送経路とは独立した別のネットワークに送る構成を提案する．このネットワークを認証チャネルと呼ぶ．

提案する構成におけるパケットの流れを図 6 に示す．OpenFlow 対応スイッチに到着したパケットの DSCP 等の QoS 属性は QoS 処理する前に認証する必要がある．したがって，認証対象となるパケットをうけとったスイッチは，そのパケットをまず sQoS ノードに転送する．認証チャネルに流すパケットは正しいパケットであるか未確認であるため，本来の転送経路の帯域を利用してはならない．そこで，独立したネットワーク上に認証チャネルを用意する．

OpenFlow において，コントローラはパケットの転送先としてスイッチのインタフェース番号を指定する．このため，認証チャネルへの転送を指定する場合，コントローラは，認証チャネルに割り当てられた特定のインタフェース番号を指定する．

パケットを受け取った sQoS ノードはパケット内の指定された QoS 属性の認証を行い，正しいパケットであればスイッチに戻し，不正なパケットであれば破棄する．スイッチはパケットが戻ってきたときにはそのパケットの持つ QoS 属性が正しいものであると判断し QoS 制御を行う．

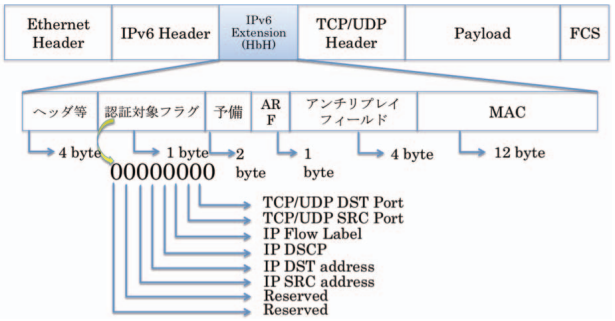


図 5 改良した sQoS ヘッダフォーマット

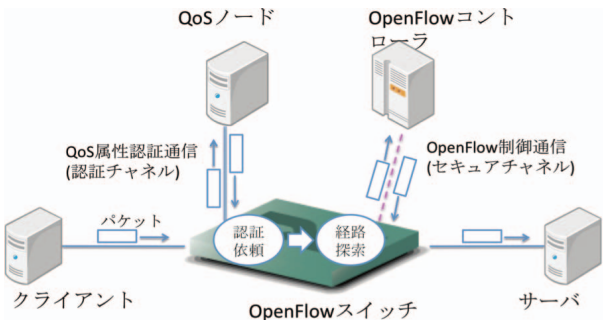


図 6 パケットフロー

このような構成をとる事で，Out-of-band 通信を実現できる．戻されたパケットは信頼されるものであるためにこのパケットを戻す際にも専用の Out-of-Band 通信路を利用する．

6.4 認証チャネルタイプの設計

認証チャネルは実時間性を阻害しない高速性をもち，不正なパケットが流入する事を防ぐ必要がある．さらにサービスプレーンを提供し，自律的に高信頼でスループットを保証できるような柔軟性が必要である．そこで，速度を重視した認証チャネルと，柔軟性を重視したチャネルの二つのチャネルを用意する．

(1) 物理認証チャネル

一つ目のタイプは，物理インタフェースを用いるものである．最も単純な構成では，スイッチとコントローラを Ethernet ケーブルで直結し，その接続したインタフェース番号を転送先として指定する．このような構成をとった場合，単純に結線のみで構築する事ができることに加え，割り当てられたインタフェースを認証チャネル専用に利用する事が可能となり，暗号化も必要ではないため高速なチャネル内の通信が可能となる．ただし，このような構成は，物理的なネットワークケーブルの配線が必要となり，また物理的なネットワークインタフェースが必要となるため，少数のスイッチが少数の sQoS ノードを利用する際に有効である．また，物理的な接続を要するため，静的な運用に向いている．

(2) 論理認証チャネル

もう一方のタイプは論理的なインタフェースである．具体的には，Ethernet Over IP(Etherip)[18]のトンネルインタフェースを利用する．物理的な認証チャネルは物理的な配線を伴うため，自動的な負荷分散等には利用できないが，論理的なインタフェースを用いれば，自由に Etherip トンネルの対向機器を切り替える事が可能となる．トンネルの設定は，OpenFlow からは独立した既存のトンネルの設定機能で行う．これにより，Flexible 接続を実現する．

6.5 認証チャネルタイプの選択

単純な構成であれば物理インタフェースタイプの認証チャネルが適しているが，サービスプレーンとして sQoS の認証機能群を提供し，QoS 属性認証処理の負荷や，通信要件等を加味して自律的に最適な認証機能を選択するためには論理インタフェースタイプの利用が必須となる．

前述したように，認証チャネルのトラフィックは通常の転送ネットワークから独立させる必要がある．そこで，sQoS ノード群とスイッチの持つ認証チャネル用の物理インタフェースを独立したネットワークに接続する．各々のスイッチと sQoS ノードの間の Etherip トンネルはこのネットワーク上で構築し，このネットワークには認証チャネル以外のトラフィックは流さない．

論理的なインタフェースを用いた認証チャネルを用いる場合，スイッチが利用する sQoS ノードの IP アドレスに対してトンネルを構築する．トンネルの終点に指定した IP アドレスが Unicast であれば特定の sQoS ノードに転送されるが，Anycast アドレス[19]であれば自動的に最も近いノードに転送される．Unicast は高度な判断を用いてより厳密に sQoS ノードを選択する際に有効である．Anycast は簡便に最も近い sQoS ノードを自動選択する際に利用できる．また，接続した sQoS ノードからのレスポンスがなくなった場合，自動的に他のアクティブな sQoS ノードに切り替える事ができる．しかしながら，sQoS ノードの負荷状況等は加味しないため，最も近いノードが最適な sQoS ノードであるとは限らない．また，自動的に切り替える場合も一定の時間待たなければならないなどからあくまで簡易的な自動化，冗長化手法である．

7. 試作

6 章に示したシステム設計において最も根幹となる機能は，オフロードした sQoS ノードに対する Out-of-Band 通信路を用いた認証機能の呼び出しである．そこで，スイッチが受け取ったパケットを Out-of-Band 通信路に転送し，認証を受けたパケットがスイッチに戻ってくることと，戻ってきたパケットが本来の転送先へ転送されることを検証するために sQoS ノードと OpenFlow コントローラを試作した．本章では試作したシステムについて述べる．

7.1 試作システムの構成

本試作システムの構成を図 7 に示す．また，表 1 に各機器の仕様を示す．以下に各機器について説明する．

(1) クライアント

sQoS ヘッダを持つパケットを生成するためのクライアントアプリケーションが動作する機器である．クライアントアプリケーションは試作したものであり，優先度を示す DSCP 値に EF の値をもつ UDP パケットを用い，送信するパケットに sQoS ヘッダを含む．

(2) サーバ

クライアントから送信された UDP パケットを受け取ることが目的のアプリケーションである．サーバも試作したものである．

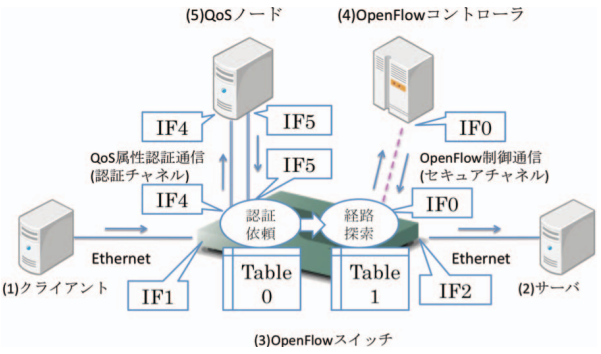


図 7 試作システムの構成

表 1 機器の仕様

機器	H/W	OS	利用 S/W
OpenFlow コントローラ	CPU:インテル Core i7-2600 (クアッドコア /HT 対応 / 定格 3.40GHz/TB 時最大 3.80GHz/L3 キャッシュ 8MB) Mem:8GB DDR3 SDRAM(PC3-10600)	Ubuntu 12.10	Trema-edge (Head repository 2013/5/9)
QoS 属性認証用計算機		Ubuntu 12.04.1 LTS	ebtables ip6tables sQoS 認証機能
OpenFlow スイッチ			Openvswitch (Head repository 2013/5/9)
送信機器			sQoS 対応クライアント/サーバ
受信機器			

表 2 テーブル 0 の設定

Priority	Match	Instruction
200	In-port = 5	Goto Table 1
101	DSCP==EF HbH Exist == TRUE	OUT Port 4
100	DSCP==EF HbH Exist == FALSE	DROP
0	Any	Goto Table 1

表 3 テーブル 1 の設定

Priority	Match	Instruction
101	DST Addr == 10.0.1.1	OUT Port 1
100	DST Addr == 10.0.1.2	OUT Port 2
0	Any	OUT Port 0xfffffffd (Controller)

(3) **OpenFlow スイッチ**

OpenFlow に対応したスイッチであり，Open Source Software (OSS) の Openvswitch[20] を利用している．OpenFlow コントローラから制御される．インタフェース (IF)の 1 から 5 までを制御対象 IF としている．

(4) **OpenFlow コントローラ**

OpneFlow スイッチを制御する機能を持った機器である．制御機能は OSS の Trema-edge[21]を元に試作したものである．

(5) **sQoS ノード**

試作した sQoS 認証機能が稼働する機器である．受信したパケットを認証し，QoS 属性値が正しければパケットをスイッチに返し，正しくなければ破棄する．

7.2 認証チャネル設定

本構成では基本的な動作の検証が目的であるため，認証チャネルには最も単純な物理インタフェースを用いている．sQoS ノードの選択は接続時に固定的に決定されるため，論理的なインタフェースを用いる際に必要となる sQoS ノードの選択手法は利用していない．

スイッチは0〜5番までの6つの物理インタフェースを持つため認証チャネルには 4 番と 5 番を割り当て，4 番はスイッチから sQoS ノードへの通信に，5 番は sQoS ノードからスイッチに戻ってくる通信に利用する．

sQoS ノードにも 0〜5 番の 6 つの物理インタフェースを用意してありスイッチの 4 番と 5 番のインタフェースをそれぞれ sQoS ノードの 4 番と 5 番のインタフェースに接続する．

7.3 OpenFlow の設定

OpenFlow スイッチは複数のフローテーブルを持つ．それぞれのフローテーブルは複数のフローエントリからなり，各フローエントリはフローの特定条件(Match)と，それに適合したフローの処理内容(Instruction),処理したパケットの数(Counter)，優先度(Priority)からなる．スイッチはパケットを受信すると若い番号のテーブルの優先度の高いエントリから該当のエントリを検索し，該当するエントリが見つかったらそのエントリの Instruction にある処理を行い，パケットの転送，破棄もしくは必要に応じて順に古い番号のテーブルの処理を行う．パケットを転送してしまうとそこでそのパケットに対する一連の処理は終了する．すなわち，一つのパケットに対して転送処理は一度のみ許されている．

本研究では，一つのパケットに対し認証のための転送処理と，本来の宛先に向けた転送処理を連続的に行うことを求めている．そこで，これらの２つの処理を連続して行う機構を確認するために，QoS 属性の認証のための認証チャネルへの転送ルール処理用と，フロー毎の本来の宛先にむけた経路情報を記述する転送用のテーブルの二つのテーブルを用い，二つの転送処理を連続的に行えることを確認する事で, Out-of-Band な認証チャネルが利用可能である事を検証した．これらのテーブルをそれぞれ，テーブル 0，テーブル 1 とする．具体的なテーブルの内容を表 2，表 3 に示す．このテーブルに則った処理を以下に示す．

本構成では，高い優先度 DSCP 値(EF)をもったパケットは sQoS ヘッダを持つ必要があるという前提を持つ．スイッチに届いたパケットが DSCP に EF の値を持っており，かつ sQoS ヘッダを持っていれば，認証チャネルであるインタフェース 4 を用いて sQoS ノードに転送する(テーブル 0 [101]). DSCP が EF を持つが sQoS ヘッダを持たないものは破棄する(テーブル 0 [100]). DSCP 値が EF でないものはテーブル 1 へすすむ(テーブル 0 [0])，指定されたインタフェースに転送される．

sQoS ノードは認証チャネルから送られたパケットを sQoS の動作に則り処理する．すなわち，QoS 属性の認証値が正しければパケットはスイッチのインタフェース 5 に返信される．正しくなければ破棄される．

返信されたパケットを受信したスイッチは再度テーブル 0 を検索する．このパケットは，より Priority の高いインタ

フェース 5 から入力されたパケット向けのエントリにマッチするため，テーブル 1 へ進む(テーブル 0 [200]) ，

テーブル 1 では，宛先アドレスが受信機器(10.0.1.2)のパケットはインタフェース 2 へ転送し(テーブル 1 [100])，宛先アドレスが送信機器(10.0.1.1)のパケットはインタフェース 1 へ転送する(テーブル 1 [100])よう指定されている．何れのエントリにもマッチしないパケットはコントローラに転送し，処理方法の指示を待つ(テーブル 1 [0])．この設定による動作を確認することで，パケットを一度スイッチの外に転送しながらも，あたかもスイッチの内部で処理を行ったかのように通常のパケット転送処理を継続して行える事を確認できる．

7.4 sQoS ノードの設定

sQoS ノードではユーザスペースのアプリケーションにより QoS 属性の認証を行うため, 転送されたパケットをアプリケーションに届ける必要がある．しかしながら，パケットの宛先 MAC アドレスは元の宛先アドレスであるため sQoS ノードはこのパケットを受信できない．そこで, sQoS ノードのインタフェース 4 およびインタフェース 5 をブリッジ設定とし，宛先によらずパケットを受信できるようにする．また，このように入力と出力のインタフェースを同一のブリッジ設定にする事でパケットの Ethernet ヘッダの書き換えが起こらず，受信したパケットそのものをスイッチに返信する事が可能となる．

このパケットの認証を行うためにアプリケーションへこのパケットを届ける必要がある．そこで，パケットをアプリケーションに直接届けられるフレームワークを提供する ebtables[22]と ip6tables[23]，netlink[24]を用いる．ebtables によりパケットを sQoS ノードに吸い上げ, ip6tables にわたし, 転送処理対象とする．ここで, 経路制御の前に ip6tables の netlink を用いて sQoS 認証機能アプリケーションにパケットを届ける．パケットが正しければ，認証機能はパケットを転送処理に戻す．正しくなければ，パケットを破棄する．

8. 評価と考察

8.1 評価方法

本試作の目的は，最も根幹となる機能である，オフロードした sQoS に対して Out-of-Band 通信路を用いた認証機能の呼び出しである．この検証を行うための図 6，表 1 に示した構成において，OpenFlow スイッチに対し表 2，表 3 の二つのテーブルの設定を行い，以下の順にパケットの処理が行われることを検証した．1) クライアントからは DSCP 値に EF を持つ UDP パケットを送信し, 2)OpenFlow スイッチに到着したパケットが Out-of-Band 認証チャネルを経由して sQoS ノードへパケットが到達すること．3)sQoS ノードがパケットの QoS 属性値の認証を行いスイッチに返すこと．4)戻されたパケットが OpenFlow コントローラに送られること，5)パケットの転送先が OpenFlow コントローラから OpenFlow スイッチに設定されること，6)OpenFlow スイッチに到着したパケットがサーバに到着すること．

パケットの流れは各機器において Wireshark[25]を用いて観察することで確認した．

8.2 評価結果

この結果, 1)クライアントから送信された sQoS ヘッダを持ったパケットが OpenFlow スイッチへ到着し，2)OpenFlow スイッチが Out-of-Band の認証チャネルを経由して sQoS ノードへ転送し，3)sQoS ノードが sQoS ヘッダを認証し，正しければスイッチにパケットを戻し，4)スイッチに戻されたパケットは OpenFlow コントローラに送られ, 5)OpneFlow コントローラがパケットの転送先を指示し, 6)パケットがサーバに到達することが確認できた．しかしながら，3)の処理を行う際に，sQoS ノードの IF5 から送信されたマルチキャストパケットが，IF4 から受信される事象が観測されており, 処理時間等の測定には至っていない．しかしながら, OpenFlow スイッチにおける二つのテーブルがそれぞれ sQoS 認証目的，通常の経路制御目的で利用できており，この結果 Out-of-Band の認証チャネルを用いた sQoS 処理のオフロードが利用できることまでは確認ができている．

8.3 考察

本試作では本提案に不可欠な Out-of-Band を用いた認証チャネルを用いた sQoS ノードの呼び出しの基本動作を確認した．認証対象となるパケットの転送処理は確認できたが, sQoS ノードにおいて認証対象パケットではないパケットの再受信現象が見られており sQoS ノード側の設計を見直す必要がある．この問題の解決できれば，本認証チャネルの動作原理はパケット転送前の検証処理を非常に高速に提供できる可能性がある．例えば，先行研究の sQoS では一台のルータでの認証処理および転送処理で 137 マイクロ秒程度であったが，本提案での認証処理はネットワーク層の処理を行っていない分軽い処理となっている．一方でスイッチでの転送処理が増えているが，スイッチの転送時間は 4 マイクロ秒程度と小さいため，結果的に sQoS に近い速度がでるものと予想される．

9. まとめ

本稿では、著者らがこれまでに提案を行ってきた sQoS の課題を解決するために， sQoS を拡張し，OpenFlow と組み合わせる技術検討を行った．本提案では，QoS 属性の認証機能をネットワーク機器から第三の計算機にオフロードし，認証チャネルという通常の転送経路から独立した通信路を用いて認証通信をおこなう．また、第三の計算機資源を群として提供し，認証チャネルを自律的に構築する事で常に要件に見合ったサービスを利用できるサービスプレーンを提案した．これにより，sQoS の持つ 1)Layer 非依存,

2）スケーラビリティ，3)プログラム更新といった課題を解くことができる．

本提案は評価・試作段階であり最も根幹となる機能の検証を行っている状況であり，認証チャネルの返信パケットの処理に課題が残されている．今後はこの課題を解決し，QoS に対する効果や実時間通信に対する評価，論理的なインタフェースを用いたパフォーマンスの評価についても進めていく予定である．

なお，本稿で示した OpenFlow の活用方法は，ネットワーク機器においてパケットを転送する際に任意の事前処理を行うための汎用的なフレームワークとして利用可能である． OpenFlow はある一定のパケットへの処理をスイッチに行わせる機能を持つが，経路制御やパケットの一部フィールドの書き換えに限定される.したがって，このようなフレームワークは OpenFlow 機器の制約を取り外し，高度なパケット処理の可能性を低供することができるため OpenFlow ネットワークの可能性を大きく広げることができる．

参考文献

1) R. Langner, "Stuxnet : Dissecting a Cyberwarfare Weapon," IEEE Security & Privacy vol. 9 no. 3 pp.49-51, May/June 2011.
2) ISA, ISA-TR100.15.01-2012 Backhaul Architecture Model: Secured Connectivity over Untrusted or Trusted Networks, 2013
3) Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.
4) Kent, S. and K. Seo: Security Architecture for the Internet Protocol, RFC 4301, December 2005.
5) Kent, S.: IP Authentication Header, RFC 4302, December 2005.
6) Dmitrij Lagutin: Redesigning Internet - The Packet Level Authentication architecture, Licentiate’s Thesis - HELSINKI UNIVERSITY OF TECHNOLOGY, May 2008.
7) Braden, R., Clark, D., and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, ISI, MIT, and PARC, June 1994.
8) Q Hu, B Carpenter. Survey of proposed use cases for the IPv6 flow label. RFC6294, June 2011
9) Billy Bob Brumley: Implementing Cryptography for Packet Level Authentication, Proceedings of The 2008 International Conference on Security & Management—SAM’08, 2008
10) Fieldbus Foundation: System Architecture, FF0581-1.3, October, 2003.
11) C. Karlof, N. Sastry, and D. Wagner, “TinySec: Link Layer Security for Tiny Devices”, In Proceeding of SenSys '04 Proceedings of the 2nd international conference on Embedded networked sensor systems
12) 宮田宏，並木美太郎，佐藤未来子: “OpenFlow ネットワークの産業用制御システム応用の基礎的検討”，情報処理学会研究報告[システムソフトウェアとオペレーティング・システム] 2012-OS-122(1), 1-9, 2012-07-25
13) Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, February 1997.
14) Deering, S. and R. Hinden: Internet Protocol, Version 6 (IPv6) Specification, RFC2460, December 1998.
15) Shimomura, Takanori, and Kazunori Miyazawa. "A multicast tunneling system for IP-based Control Systems." ICCAS-SICE, 2009. IEEE, 2009.
16) MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review 38, 2 (April 2008).
17) Madson, Cheryl, and Rob Glenn. "The use of HMAC-SHA-1-96 within ESP and AH." (1998).
18) Housley, Russell, and Scott Hollenbeck. "Etherip: Tunneling ethernet frames in ip datagrams." Network Working Group, Request for Comments 3378 (2002).
19) Narten, T., et al. "RFC 4861-Neighbor discovery for IP version 6 (IPv6), 2007." (2011).
20) http://openvswitch.org/
21) http://github.com/trema/trema-edge/
22) De Schvymer, Bart, and Nick Fedchik. "ebtables/ipt. ables interaction on a Linux? basedbridge." 2003 [2006-05-15]. http://ebtables. sourceforge. net/br_fw_ia/br_fw_ia. html (2003).
23) http://www.netfilter.org/projects/iptables/
24) He, Kevin Kaichuan. "Kernel korner: why and how to use netlink socket." Linux Journal 2005.130 (2005): 11.
25) http://www.wireshark.org/