第 II 部

制御ネットワークの IP 化

第 2 部
制御ネットワークの IP 化

第 1 章　はじめに

TACA ワーキンググループでは、PC のように豊富なリソースを持っていないデバイスがネットワークに接続される環境を想定し、それを可能とする技術の研究や開発を行っている。扱っている問題の例を挙げると、大量のデバイスをどのように運用・管理するかや、計算能力の低いデバイスでいかにしてセキュリティを確保するかなどである。

本年度は、12 月までに 6 回の定期ミーティングを開催し、主に制御系ネットワーク（Process Automation、Factory Automation、Building Automation）に IP 技術を取り入れ、ネットワークを構築、運用するにあたって直面する課題について研究を行った。

具体的には、センサなどに代表される非常に多数で処理能力の低いノードを、安全かつ低コストに設定するための設定プロトコルについて、既存環境との共存などの課題を含め、研究・開発を行った。また、実際の運用において、ポリシ策定の権限委譲、ゲストノードの取り扱いに関する方式に関しても議論を行った。

以下では、TACA ワーキンググループで本年度検討した課題をまとめた論文を紹介する。第 2 章は、2005 年 7 月に岩手県で開催されたマルチメディア、分散、協調とモバイル（DICOMO2005）シンポジウムで発表された論文であり、第 3 章は 2006 年 1 月に米国アリゾナ州で開催された SAINT2006 で発表された論文である。

第 2 章　制御ネットワークの IP 化におけるセキュリティを考慮したノード自律設定システムの一検討

2.1 はじめに

近年、建築物に付随するさまざまな機器（ファシリティ）に対してネットワークを利用した高機能化へのニーズが高まっている。例えばビルにおいては、照明装置や空調装置などの制御をネットワークを利用して緻密にコントロールすることにより、建物全体のエネルギー消費量を下げることができる。また、工場などにおいては各プロセスの保守、管理、点検などをネットワークを利用して管理することによって、さらなる運用効率の向上が期待されている。以後、これらファシリティを接続するネットワークを制御ネットワークと呼ぶ。これらの要求を受け、制御ネットワークにはすでにさまざまな規格が存在し、LonWorks[116][1]、BACnet[11][2]などといった標準化が進められているプロトコルがある。

一方で、現在はさまざまな機器と制御対象を接続し、連携させたいという要求が強い。例えば自分の携帯電話から直接部屋のエアコンを制御する、あるいは警備会社のシステムとフロアのカメラを接続してモニタする。このような状況においては、制御ネットワークは前述のような専用のプロトコルではなく、現在最も利用されているネットワークプロトコル、すなわち Internet Protocol（IP）の利用が求められている。IP の利用により、既存のネットワークインフラストラクチャを利用した高度な応用が期待できる。また、現在は管理システム側の IP 化が進んでおり、IP 化された制御ネットワークがインターネットと接続しない場合においても、管理システムとの通信プラットホームの共通化によるコスト減が期待できる。同様に、IP によるインフラストラクチャの共通化は、運用ノウハウの共有やプログラマ育成効率に代表されるソフト的なコストも軽減する。加えて、

1　LonWorks is a registered trademark of Echelon Corporation.
2　BACnet is a registered trademark of ASHRAE.

ネットワークの物理層のハードの共有化による全体コストの削減も期待できるため、制御ネットワークの IP 化への期待は高い。さらに、多数のセンサーやアクチュエータがネットワークに接続されることが予想されるため、アドレス空間が広く自動設定機能が基本機能として含まれている IPv6 の採用が、制御ネットワークにおいて強く望まれている。

しかし、制御ネットワークの IP 化にはまださまざまな課題が残されている。まず、センサーやコントローラなどを IP 接続した場合、ネットワーク管理者はこれらの多数の IP ノードを低コストで設定、管理しなければならない。セキュリティも重要な課題である。従来は制御ネットワークは仕様も公開されていない場合があり、一般のユーザが制御ネットワークへ接続するのは難しかった。しかし IP 化されると、一般のユーザが誤って接続したり、悪意を持ったユーザによる制御ネットワークへの攻撃が従来に比べ容易になる。

さらに IP 化とは別の問題として、従来の制御系のネットワークデバイスの設置では、ネットワーク設計に従って、ある固有 ID をもったデバイスを指定された位置に設置する必要があった。これは、実際の施工を複雑化させる要因となっている。例えば、同じ形式の空調装置が 10 台ある場合「どの空調装置をどこに置く必要があるのか」を考えなければならない。ネットワークがなければ、任意の空調装置を任意の位置に設置できる。しかし、ネットワークの設定が含まれている場合には、10 階に置かなければならない空調装置を 9 階に設置することはできない。

本稿では、これらの問題に着目し、制御ネットワークのような非常に多数のノードが IPv6 を利用して接続される環境において、管理者が低い設定コストで、各ノードを安全に設定、管理できる手法を提案する。提案方式では、Kerberos[161][3]を用い、Property Server と呼ぶ各ノードの設定情報を保持するサーバを導入し、ノードがこれらを自律的に発見することでこれを達成する。

2.2 提案方式

2.2.1 要求条件

本稿では、制御ネットワークにおいて、多数の管理対象となるノードを接続した場合においても、管理者が初期設定情報を容易に、かつ安全に設定できるシステムを提案する。ここでいうノードの設定情報とは、例えばノードが照明のスイッチであった場合、どの照明装置に対してオン／オフを伝えれば良いかなどの情報や、そのスイッチが物理的にどこに設置されたかなど、ノードの情報を指す。また、ノードから通知される、ノードの現在の状態などに関する情報も含まれる。これには例えば現在の IP アドレスなどが挙げられる。多数のノードを設定する場合に、管理者はそれぞれのノードに対して一台一台設定するよりも、ネットワークのどこかにノードの設定情報を集約しておき、各ノードが自律的に自己の設定情報を取得して起動するほうが、総合的なコストが低くなる。一方で、ノードがネットワークを介して設定を行なう場合、なりすましや盗聴に対する防御も必要である。

これらのことを踏まえ、まず制御ネットワークの IP 化に対する要求条件を以下に挙げる。

1. 管理コスト
   制御ネットワークは通常、専任の管理者の存在を想定できるため、家庭用ネットワークのような完全な無設定となるシステムを目指す必要はない。しかし、ノード数が膨大であるため、個々のノードそのものへの設定は必要最小限にとどめ、それぞれのノードが可能な限りネットワークを通して自律的に設定できるアーキテクチャが望まれる。
2. セキュリティ
   既存の公開されたネットワーク技術を用いるため、制御ネットワークでの盗聴などさまざまな攻撃が想定される。このため、すべてのノードにおいて通信の認証／秘匿性／完全性といったセキュリティを確保できる枠組が必要である。
3. 管理対象のノード数に対するスケーラビリティ
   前述のように制御ネットワークに参加するノードは非常に多くなることが想定されるため、ノード数に対してのスケーラビリティがシステムには求められる。
4. 低コストノードでの運用可能性
   一般に部品コストの低いノードは計算能力も低く、特に公開鍵暗号系のような多倍長整数演算が必要な処理を行なうことは現実的でない場合が多い。このような低コストノードが参加可能で、かつ全体のセキュリティが低コストノード

3　Kerberos is a trademark of the Massachusetts Institute of Technology.

の参加によって下がることのないシステムが必要となる。

5. ノードの配置および置換の容易性
   同一種類のデバイスについては、施工時にはそれらを区別することなく配置でき、ネットワーク管理者は後から固有の設定を行なうことができることが望ましい。さらに、セキュリティや同一の設定を保ちながら、ネットワーク管理者の観点で簡便にノードの交換ができることが望ましい。

本稿では、これらの要求条件を満たすシステムを提案する。以下各空調装置や照明装置、スイッチやセンサーなどの管理対象となるノードを単にデバイスと呼ぶ。

2.2.2 提案方式のシステム構成

提案システムの概要を図 2.1 に示す。提案システムは Kerberos をベースとしたシステムである。Kerberos では基本的には公開鍵暗号系を必要としないため、計算能力の低いノードも使用可能である。Kerberos では各ノードは realm と呼ばれる管理ドメインに所属し、principal と呼ばれる識別子によって管理される。

提案システムでは、デバイスの個体認証のための realm と、抽象化された機能（例えば入口にあるスイッチ）の設定および運用のための realm という 2 種類の realm を利用する。個体認証が終了した段階でその個体に抽象化された管理運用のための名前を割り当て、管理運用のための realm へと移行させる。これによりデバイスの個体と実際の機能を分離することができ、同一種類のデバイスの物理的配置や、それらの交換などに自由度を持たせることが可能となる。

提案システムでは、デバイスの設定は大きく分けて 2 つのフェイズで行なわれる。まず、各デバイスが、対象となる制御ネットワークに参入可能かどうかを認証するフェイズである。このフェイズを起動フェイズと呼ぶ。起動フェイズでは、デバイスの個体認証を行ない、デバイスに管理運用名を割り当てる。提案システムでは、定常状態での運用においては各デバイスは管理運用名で識別される。管理運用名を与えられたデバイスは、その管理運用名に割り付けられた設定を自律的に取得する。このフェイズを管理運用フェイズと呼ぶ。

全てのデバイスは、デバイス識別子（Device ID、DID）を持っているものとする。DID には例えば EUI-64[118] などが利用できる。加えて、この DID に対応するデバイス固有鍵（Device Key、Dk）を持っているものとする。すべてのデバイスは IPsec[157] を利用できるものとし、IPsec の鍵交換プロトコルは Kerberos を利用した鍵交換方式である Kerberized Internet Negotiation of Keys（KINK）[275] を使用できるものとする。起動フェイズと管理運用フェイズでは Kerberos の realm が異なり、それぞれ異なる Key Distribution Center（KDC）を使用する。以後、起動フェイズで使用される KDC を KDC-B と呼ぶ。KDC-B はすべてのデバイスの DID と Dk を共有する。一方管理運用フェイズおよび定常運用状

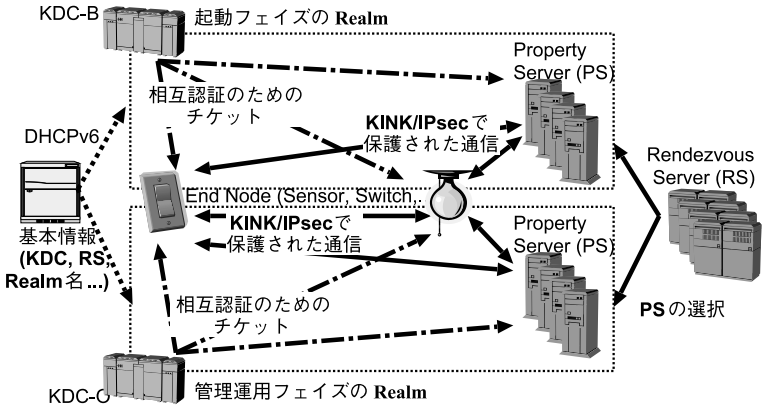

図 2.1. 提案システムの概要：すべてのノードは 2 系統の Kerberos の管理下におかれる。各ノードは起動フェイズの realm にある Property Server（PS）から管理運用名を与えられ、管理運用フェイズの realm にある PS から自律的に設定情報を安全に取得する。各 PS は Rendezvous Server を通して発見される。

態で利用される KDC を KDC-O と呼ぶ。

Property Server（PS）はデバイスの設定情報を保持する。PS はデバイスと同一のネットワークに存在してもよいし、またインターネット上に存在してもよい。また PS もデバイス同様 Kerberos クライアントであり、KDC-O と鍵を共有しており、IPsec および KINK を利用できるものとする。

提案方式では、あるデバイスの設定情報をどの PS が持っているかを保持するデータベースを持つ。これを Rendezvous Server（RS）と呼ぶ。本提案方式では RS としてローカルな DNS[188] サーバを利用する。

また、提案システム内には、KDC-B, KDC-O および RS のアドレスを提供するため、DHCPv6[54] サーバがあるものとする。DHCPv6 には, Kerberos に必要な情報を付与するための拡張が行なわれているものとする。この拡張には、例えば対象となる KDC が管理する realm 名の通知などが挙げられる。

2.2.3 通信モデル

以下図 2.2 を例に、デバイスの起動時における通信手順を示す。

D1 は DHCPv6 サーバ、KB1 は KDC-B、KO1 は KDC-O、RS1 は RS を表す。また、Kerberos の起動フェイズの realm 名を BOOTREALM.LOCAL、管理運用フェイズにおける realm 名を EXAMPLE.COM、とする。デバイス N1 の DID を DID1、デバイス固有鍵を Dk1 とする。N1 の PS を PS1 であるとし、N1 の管理運用名は n1.example.com となるものとし、n1.example.com の PS は PS2 であるとする。

なお、Kerberos では KDC とノードの時刻同期が必要であるが、この同期は文献 [42] に従うものとする。

1. デバイスの起動フェイズにおける principal 名の決定
   デバイスは起動後、まず自己の起動フェイズにおける principal 名を決定する。提案方式では、principal 名はデバイスに与えられた DID を 16 進数で表記し、4 bit ごとに区切ったものとする。例えば、DID1 が 0123:4567:89ab:cdef であるとすると、N1 の principal 名は 0.1.2.3.4.5.6.7.8.9.a.b.c.d.e.f となる。
2. KDC-B の発見
   N1 はまず DHCPv6 を使用して起動に必要な情報を得る。この情報には、Realm 名とそれに対応する KDC のアドレス、RS のアドレス、NTP サーバのアドレスなどが含まれる。この例では、D1 より、realm 名として EXAMPLE.COM とその KDC である KO1、BOOTREALM.LOCAL とその KDC である KB1 の組が得られる。また、RS のアドレスとして RS1 を得る。N1 は、得られたそれぞれの KDC に対して、Ticket Granting Ticket（TGT）の取得を試みる。起動フェイズにおいては、KDC-B しか Dk1 を知らないため、TGT の取得に成功した KDC が KDC-B であることがわかる。
3. PS の発見
   N1 は、得られた RS のアドレスのうちの一つに、`_propertyserver_._tcp.rev(principal).REALM` の SRV レコードを問い合わせる。ここで、rev(principal) は principal 名の中の label を逆順に並べたものであり、REALM の部分は DHCPv6 で得た realm 名である。よってこの例では、
   `_propertyserver_._tcp.e.d.c.b.a.9.8.`

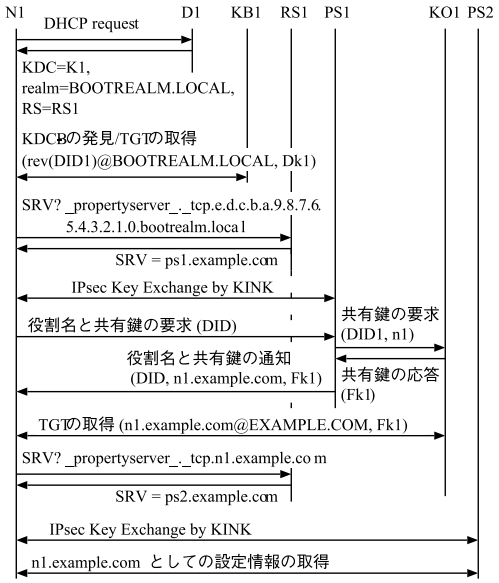

図 2.2. 通信モデル：ノード N1 はまず DHCPv6 を利用して KDC-B を発見する。DID を元に、RS を介して PS を発見する。発見した PS から管理運用名を取得する。Kerberos の realm を管理運用フェイズの realm に移行し、その設定情報を取得する。通信には KINK/IPsec を利用する。

7.6.5.4.3.2.1.0.bootrealm.local

に対する SRV レコードを RS1 に問い合わせる。この結果、N1 は PS として、PS1 のアドレスを得る。

4. PS との KINK を利用した鍵交換
   N1 は PS1 に対して KINK を利用して IPsec 通信のための鍵交換を行なう。
5. PS1 からの管理運用名および管理運用共有鍵の取得
   N1 は IPsec を利用して、PS1 へ DID1 を通知し、管理運用名と、それに対応する KDC-O との共有鍵を要求する。この共有鍵を管理運用共有鍵と呼ぶ。この結果、管理運用名は realm EXAMPLE.COM における principal 名となる。PS1 は、DID1 に対応する管理運用名は n1 であると設定されているので、KDC-O に対して DID1 と n1 を通知し、管理運用共有鍵の生成を要求する。KDC-O はこの要求を受けて管理運用共有鍵 Fk1 を生成し、与えられた管理運用名との共有鍵として設定し、PS1 へと通知する。PS1 は、この内容を N1 へ通知する。
6. 管理運用フェイズへの移行
   N1 は、得られた管理運用名 n1 と管理運用共有鍵 Fk1 を使って、管理運用フェイズへの realm である EXAMPLE.COM へと参加する。
7. 管理運用フェイズにおける PS の発見
   起動フェイズにおける PS の発見と同様、管理運用名に対応する PS を発見する。この結果、PS として PS2 を得る。N1 は PS2 に対して（管理運用フェイズにおける realm のノードとして）KINK を利用して IPsec 通信のための鍵交換を行う。
8. PS からの設定情報の取得
   N1 は IPsec を利用して、PS2 から n1.example.com としての設定情報を取得する。

2.3 考察

2.3.1 管理コスト

提案方式では、設定情報を PS に集約可能であり、管理者は個々のノードに接続する必要がなく管理コストを低減できる。また、PS は分散配置可能であるため、例えば空調系のノードの PS は建物内に配置し、一方防犯系ノードの PS は警備会社内に配置するといった管理主体の分散も可能である。

また、個々の管理対象ノードには、EUI-64 のような一意な識別子と、それに対応する鍵という 2 つの情報のみを擦り込んでおけばよい。これはそのノードがどのネットワークで利用されるかに依存しないため、任意の時点、例えば工場出荷時などに設定可能であり、ノードの生産者にとっても負荷は低い。ノードは接続された時点で自律的に PS を探し出すため、ノードの利用者は KDC に鍵を登録し、PS の設定に専念できる。よって提案システムの管理コストは低いといえる。

2.3.2 セキュリティ

すべてのノードは KINK を利用した IPsec を利用できるため、PS とノード、あるいは設定後に行なわれる、空調の ON/OFF といったノード間の通信はすべて IPsec が利用可能なため、盗聴やなりすましという攻撃は難しいといえる。一方、KDC の発見に利用する DHCPv6 においては、攻撃者は容易に DHCPv6 サーバになりすませる。ここで通知されるのは KDC のアドレスであるため、攻撃者は虚偽の KDC を通知することはできるが、KDC との共有鍵を知らない限りこの後の通信を成り立たせることは困難である。ゆえに、DHCPv6 のメッセージが正しいかどうかは、与えられた KDC と相互認証可能であるかで判断できる。

PS を発見するために RS に問い合わせを行なうが、この応答も攻撃者は容易に偽装可能である。しかし、ノードは PS との通信に KINK/IPsec を利用するため、KDC の場合と同様に誤った PS に問い合わせのパケットを投げさせることは可能であるが、その後の通信を成り立たせることはやはり困難である。また必要であれば、RS との通信にも KINK/IPsec が適応可能である。

RS の発見には DHCPv6 を使用するが、悪意のあるユーザが DHCPv6 サーバになりすまし、誤った KDC および RS の情報をノードに対して与えることができる。しかし、KDC や PS になりすますことは困難であるので、ノードは誤った設定情報であることを検出できる。また、正しい DHCPv6 のパケットを遮断しないかぎり、各ノードは KDC との相互認証が可能になるまで TGT の取得を繰り返すことによって、最終的に正しい KDC と接続できる。加えて、このフェイズはノードが定常状態になる前のブートストラップ時の過渡状態であり、定常状態で

の性能に影響はない。

さらに、ノード間の通信にも KINK を利用した IPsec が利用可能である。よってノード上のアプリケーションも、IPsec が提供するセキュリティ機能が利用可能となる。同様にノード上で実行されるアプリケーションプログラマも、特別なセキュリティメカニズムを意識することなくセキュリティメカニズムを利用できるため、プログラミングのコストを引き上げることがない。

また、管理運用フェイズが終了した後は、デバイス固有鍵は必要としない。よって管理運用共有鍵を不揮発な記憶領域に保存可能であれば、デバイス固有鍵そのものを管理運用フェイズへの移行時に KDC-B 側で無効にすることも可能である。

2.3.3 管理対象ノード数に対するスケーラビリティ

KDC への負荷は基本的に各ノード起動時および KINK による鍵交換時に発生する。よって、定常運用状態における、各ノードが交換するパケットあるいは張るコネクションに比較すると、KDC への通信量は多くないと考えられる。

一方で、ノードが多数同時に起動するような場合には KDC へのアクセスが集中することが考えられる。KDC が適切に処理要求パケットを破棄することによって、KDC 自体の処理を行なうことができる。ノード側では KDC との接続がタイムアウトした場合には充分かつランダムな待ち時間をとって再度 KDC と接続すればよい。この手続きによって発生する遅延は定常状態の性能に影響を与えるものではなく、起動時の過渡状態から定常状態への遷移時間へ影響を与えるだけである。さらに、一般的な制御ネットワークにおいては、一ヶ所の電源を投入した時にすべてのノードが起動するような運用は行なわれないのが通常である。

過渡状態から定常状態への移行時間も問題となる場合には、必要な L2 容量の推定と、適切な KDC の分散配置が必要となる。KDC に含まれる情報は静的なものであり、繁雑に更新されることがないため、KDC 自身の多重化は容易である [156]。しかし管理する KDC の台数が増加するため、この問題は KDC の管理コストと定常状態への移行時間とのトレードオフになると考えられる。

また、管理対象施設（ビルや工場など）が大きい場合や複雑な場合には管理ポリシーなどを分ける必要が発生する可能性がある。この場合には Kerberos の管理空間、すなわち realm を分割する必要がある。分割した複数の管理空間で相互運用性については、Kerberos の inter-realms が適用可能である。

一方で、PS の台数は管理対象の数や特性によって変化するため、柔軟に配置できる必要がある。提案方式では、ノードと PS の関係は、RS に記述するため、PS の台数および位置は自由に設定できる。また RS 自身についても、既存の DNS をそのまま利用可能であるため、ノード数や問い合わせ数に応じて RS の台数を設定できる。

2.3.4 低コストノードでの運用可能性

提案システムは Kerberos を利用したシステムである。IPsec の鍵交換プロトコルにも、公開鍵暗号系を利用する Internet Key Exchange（IKE）[104] を利用せず、Kerberos ベースの KINK を使用する。提案システムはセキュリティメカニズムの中に公開鍵暗号系の方式をまったく必要としないため、計算能力の低いノードにもセキュリティを提供できる。

2.3.5 ノードの配置および置換の容易性

提案システムでは、まず起動フェイズでどのデバイスがどの役割を果たすデバイスであるかを認証した上で決定し、管理運用フェイズでその詳細な設定情報が与えられる。よって、デバイスの物理的な施工後に、それぞれのデバイスの位置に応じた管理運用名を割り当てることが可能である。さらに、DID と管理運用名との関係は、PS 上に設定してあるため、故障などの理由でデバイスの交換が発生しても、起動フェイズにおける DID と管理運用名の関係のみを変更すればよく、物理的な機器の施工者とネットワーク管理者の作業を分離できる。

2.4 おわりに

本稿では、多数のノードが接続されると想定される制御ネットワークの IPv6 化において、多数のノードの設定を低いコストで、かつ安全に行なうことを支援するシステムの検討を行なった。提案方式は Kerberos と IPsec を用いてセキュリティを提供する。また、ノードの設定情報を管理するサーバである Property Server（PS）と、PS をノードが自律的に発見できる機構を提案した。提案方式では、ノードが起動時に必要な情報はノードの識別子と Kerberos のため

の鍵のみであり、他の設定情報は自律的に発見し設定するため管理コストが低い。また、鍵交換プロトコルには公開鍵暗号系を使用しない KINK を利用するため、多倍長整数演算などの負荷の高い処理を必要としない。よって、性能の低いノードでもセキュリティを確保できる。さらに、対象となるノードの個体認証とそれらの役割に応じた設定のフェイズを分けることにより、ノードの物理的な配置および置換が容易となる。

今後の課題としては、PS における情報の表現方式と、その情報を実際に交換するためのプロトコルの規定などがあげられる。また、提案システムのプロトタイプ実装及びその性能評価などを行なっていきたい。

第 3 章 A Prototype of a Secure Autonomous Bootstrap Mechanism for Control Networks[4]

Abstract

There are many kinds of control networks, which have been used in various non-IP network areas, such as BA (Building Automation), FA (Factory Automation) and PA (Process Automation). They are introducing IP and face the issues of security and configuration complexity. The authors have proposed a model which intends to solve the issues while satisfying restrictions, i.e. small embedded devices, isolated networks and private naming system/name space, which are required when introducing new functionality into existing control networks. Secure bootstrap sequence and device-to-device communication using the chain of trust are the points of the model. This paper shows the practicability of the model through implementing the model experimentally.

3.1 Introduction

Control networks are different from IP (Internet Protocol) with regard to their history, purposes and technology. There are numerous standards of the control networks, e.g. FOUNDATION fieldbus[5][82], PROFIBUS[6][221], MODBUS[7][190], BACnet[8][15] and LonWorks[9][59], which have been used in various non-IP network areas, such as BA (Building Automation), FA (Factory Automation) and PA (Process Automation). Multiple standards coexist within a single system usually because the system's requirements are diverse. Many standards are introducing IP as a transport technology, e.g. FOUNDATION fieldbus HSE, PROFInet, MODBUS/IP and BACnet/IP.

The following are the issues that the control networks are facing.

- Security

  Security of the control networks has not been considered sufficiently. For example, current specifications of FOUNDATION fieldbus and MODBUS do not mention security. BACnet has the capability of network security, however, but it is insufficient[107, 312]. Improvement is on-going[229]. Security of LonWorks supports only server authentication using challenge-response before starting a session. The security of LonWorks is weaker than BACnet because mutual authentication and packet based security, i.e. authentication, integrity and confidentiality, are not provided.

  They must be as concerned with security as IP networks are because security incidents have occurred on them, and there are concerns for safety of social infrastructure[25, 94, 219].

4 This research is supported/funded by the Ministry of Internal Affairs and Communications of Japan.
5 FOUNDATION fieldbus is a registered trademark of the Fieldbus Foundation.
6 PROFIBUS is a registered trademark of PROFIBUS International.
7 Modbus is a registered trademark of Modicon, Inc.
8 BACnet is a registered trademark of ASHRAE.
9 LonWorks is a registered trademark of Echelon Corporation.

- Configuration complexity

  Devices are manually configured in the fields whereas their user interfaces are not so powerful, like PCs. However, the number of devices are increasing because precision is required in measuring and controlling. For example, a BA system of a large building complex in Japan has 170,000 control points with 16,500 devices[10]. This will present not only the cost of engineering but also the possibility of human errors in the future if a labor-saving mechanism is not introduced.

The following are restrictions that should be accepted when introducing new functionality into the control networks.

- Small embedded devices

  The small embedded devices commonly used in the control networks have limited computational performance because of their restricted requirements of cost, physical size and power consumption. Some devices will have more powerful CPUs in the future. At the same time, low-power CPUs will survive because choice of CPU depends upon not only cost performance but also power consumption which has an impact against battery operation or bus width which has an impact on circuit size.

- Isolated network environments

  The control networks do not always require connectivities to the Internet even though introducing IP. It is the user's choice whether to connect to the Internet. Therefore, functionalities introduced into them have to work well under an isolated network environment.

- Private naming system and private name space

  Information of the control networks, not only the traffic but also device's name, has to be confidential, because the information can help to indicate corporate activities, e.g. the capability of plants. Therefore, the naming system should be closed to the public if operators desire. It is also important not to force device's identity to be global unique if most of the devices should not be accessed by outside. For the above two reasons, DNS is not an appropriate naming system for them.

We have proposed a model[123], which intends to solves the above issues while satisfying the above restrictions when introducing IP into the control networks. It is important for them to inherit existing property when introducing new functionality because they have large property, e.g. specification, operational knowledge and applications. This is the reason the model is a framework whose functions are addressed to ether below the application layer or the middleware instead of inventing new control network protocols. In this paper, we show the practicability of the model by implementing it experimentally. This paper shows an overview of the model in Section 3.2, details of the model in Section 3.3, the prototype system in Section 3.4, considerations through prototyping in Section 3.5 and related work in Section 3.6.

3.2 Proposed Model

3.2.1 Network Security

The authors have already studied a security mechanism[208] which can satisfy the restrictions described in Section 3.1. We will use this mechanism in our proposed model. The following are the features of the security mechanism.

- Communication is protected by IPsec[157] which provides IP packets with confidentiality, integrity and authentication with the other end. IPsec is useful because its enforcement is independent from applications and sharable among them. IPsec is applicable to small embedded devices due to not using public key cryptography.
- It is important for IPsec to share a secret, which is called IPsec SA (Security Association), between both ends. Key exchange protocols will be important in facilitating the

10 http://www.echelon.com/about/press/2003/echelon_mori.htm

sharing of a secret if running IPsec on small embedded devices because these devices do not have a powerful user interface like a PC, which makes manual keying difficult. The security mechanism uses not IKE (the Internet Key Exchange)[104] but KINK (Kerberized Internet Negotiation of Keys)[244] for the key exchange protocol. IKE is the most popular key exchange protocol for IPsec. However, it is not suited to small embedded devices because the Diffie-Hellman key exchange is mandatory. KINK can work well on small embedded devices because KINK is based upon Kerberos[11][161], where public key cryptography is not mandated.

- In the security mechanism, a node's identity is in the manner of Kerberos, i.e. a principal-id, which is a combination of a realm name and a principal name.

3.2.2 Auto-configuration using a Directory Service

To simplify the configuration process, the model provides the device's application layer with an auto-configuration mechanism. The basic ideas of the auto-configuration are 1) to minimize preinstalled information in a device, 2) to acquire most information from servers located in networks. IP address configurations are beyond the scope of the model because it can be done by DHCP (Dynamic Host Configuration Protocol) in IPv4 or RFC2462 in IPv6, with which the model can be combined. The auto-configuration requires not only name/address resolution like DNS but also general data handling, e.g. searching, getting and updating data. We introduce our own directory service named PS (Property Server)[162]. The following are the features of PS.

- It is not a prerequisite condition for PS to connect to the Internet because PS does not require global tree structures like DNS.
- PS maintains a device's attributes as metadata of the device's identity. A typical example is that an IP address IP_{FOO} is an attribute value of an attribute type $ATTR_{IPaddress}$, and the attribute, i.e. the type and the value, belongs to a device's identity FOO as metadata.
- PS supports two types of transactions, i.e. PUT and GET. PUT sets/updates attributes in PS. GET acquires attributes from PS. Any request of transaction has search conditions which designate attributes to be affected. For example, identity/IP address resolution is done by GET transaction, where search conditions is the value of $ATTR_{IPaddress}$ belonging to the name FOO, which returns IP address(es) IP_{FOO}.
- PS's protocol uses XML for the future extension.
- In the proposed model, every node belonging to a system has to use the security mechanism described in Section 3.2.1. Illegal access to PS by outside can be prohibited with IPsec security policy simply. An access control list can be introduced into PS if accurate restrictions are required.

3.2.3 Bootstrap Sequence using the Chain of Trust

When considering secure auto-configuration, devices have to discover a trusted PS, then exchange data with PS under secure communication channels. The following bootstrap sequence, which we call the Chain of Trust, satisfy the above requirements.

1. Devices can trust Kerberos server KDC (Key Distribution Center). It is a prerequisite condition of Kerberos.
2. Devices should trust PS which trusted KDC shows.
3. Devices register their own information, e.g. a principal-id and IP address(es), to trusted PS. The information will be used for discovering peers (see Section 3.2.4).
4. Devices should trust data which trusted PS

11 Kerberos is a trademark of the Massachusetts Institute of Technology (MIT).

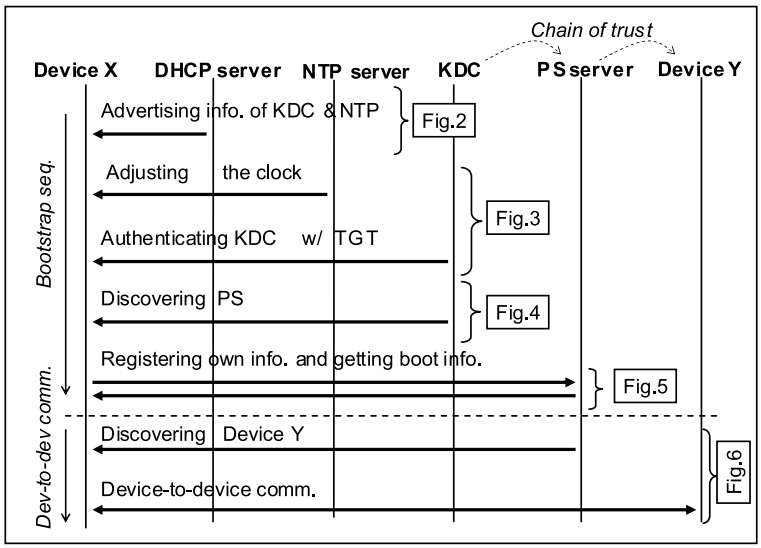

Fig. 3.1. Messages of the proposed model

provides. Then devices can complete the sequence. The communication is protected by IPsec.

Therefore, the minimum information with which a device has to be pre-installed is a principal-id and a key shared with KDC, i.e. a secret key of Kerberos. Other information can be acquired from PS.

3.2.4 Device-to-Device Communication

In the control networks, communication is occupied by control messages and notification messages between devices, e.g. controllers, sensors and actuators. Therefore, devices have to discover trusted peers, then to exchange messages with them under secure channels. The Chain of Trust can be applied in that case.

1. Devices search their peers using PS because they have already known trusted PS. (see Section 3.2.3).
2. Devices should trust peers which trusted PS provides. The communication is protected by IPsec.

3.3 Details of the Model

Figure 3.1 shows the bootstrap sequence and the device-to-device communication using the chain of trust. Details of each function is shown in Figure 3.2 through Figure 3.6.

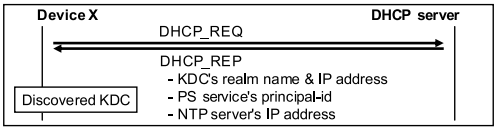

Fig. 3.2. KDC Discovery

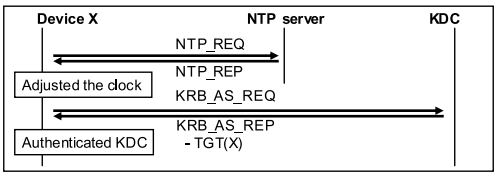

Fig. 3.3. Authenticating KDC

1. KDC Discovery (KDCD)
   DHCP server(s) advertise KDC related information, e.g. KDC's IP address(es) and realm name(s) where KDC offers authentication services and NTP related information, e.g. NTP server's IP address(es), (see Figure 3.2).
2. Authenticating KDC
   Device X needs to authenticate KDC advertised by DHCP server (see Figure 3.3). First, X adjusts its clock with NTP server because Kerberos requires that every device synchronizes its clock to prevent replay attacks. Second, X authenticates KDC, which X learned with DHCP, through verifying a given TGT (Ticket Granting Ticket).

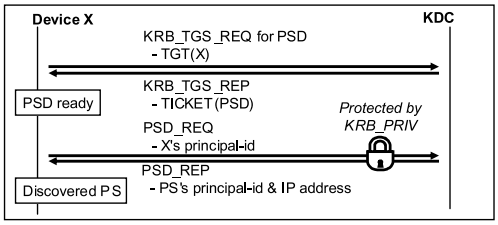

Fig. 3.4. PS Discovery

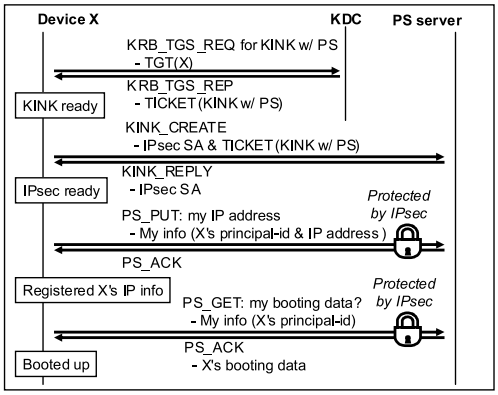

Fig. 3.5. Booting up

3. PS Discovery (PSD)

   Device X acquires PS's information from KDC (see Figure 3.4). X shows its principal-id to KDC. Then KDC returns PS's information, i.e. PS's principal-id and IP address(es). X has to acquire a service ticket for PSD, e.g. TICKET(PSD), prior to the above procedures. KRB_PRIV messages, which need a service ticket, are used for protecting PSD because IPsec/KINK can not be used at this moment.

4. Booting up

   Device X registers its own information to PS which is used for device discovery, and acquires its boot data from PS. Then, the bootstrap sequence are completed (see Figure 3.5).

   First, X acquires a service ticket for KINK with PS, e.g. TICKET(KINK w/ PS), from KDC. Second, IPsec is established between X and PS after exchanging KINK messages. Third, X registers its own information, e.g. a principal-id and IP address(es), which other devices or servers can use for discovering. Fourth, X acquires its boot data. Then X completes the bootstrap sequence.

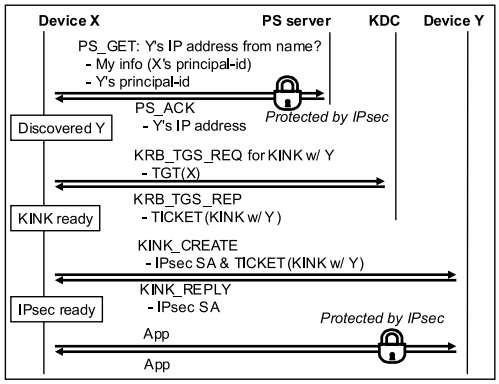

Fig. 3.6. Device-to-device communication

5. Device-to-device communication

   Device X can discover device Y using PS, i.e. device discovery, then starts the device-to-device communication with Y (see Figure 3.6).

   First, X discovers Y through PS. A typical example is that X resolves Y's IP address from Y's identity like DNS. IPsec between X and PS has already been established at the bootstrap sequence. Second, X acquires a service ticket for KINK with Y, e.g. TICKET(KINK w/ Y), from KDC. Third, IPsec is established between X and Y after exchanging KINK messages. Then the devices can exchange application messages which are protected by IPsec.

3.4 Prototype System

We implemented the model to examine its practicability, i.e. object code size and performance, experimentally. Table 3.1 shows the specifications of an experimentally prototyped device, whose CPU is H8/3029 (Renesas Technology Corp.), which has cryptographic hardware in a Xilinx's FPGA. Renesas's H8 family is a popular low-end CPU in Japan[12]. Table 3.2 shows the specifications of servers which were used for the system.

12 http://www.assoc.tron.org/jpn/research/data/survey2003J.pdf

Table 3.1. The spec. of the device

H/W	H8/3029@20 MHz, Crypto H/W@20 MHz (3DES, MD5)
OS, IP	uC/OS-II w/ Original IP stack
IPsec	ESP (3DES-CBC, HMAC-MD5)
Kerberos	MIT-1.2.4 based (etype: des-cbc-md5)
KINK	`draft-ietf-kink-kink-06` based

Table 3.2. The spec. of servers

DHCP	CPU: pentium-III@1.2 GHz, MEM: 128 MB, OS: freebsd4.10R
NTP, KDC	CPU: pentium-III@750 Mhz, MEM: 896 MB, OS: linux2.6.8, Kerberos: Heimdal-0.6.2, KINK: racoon2
PS	CPU: celeron@1.7 GHz, MEM: 1 GB, OS: linux2.6.8.1

3.4.1 Object Code Size

Table 3.3 shows the code size of the device. The total size will be 20 K bytes greater if the cryptography, i.e. 3DES and MD5, is implemented by software instead of hardware.

Table 3.3. Object code size of the device (K bytes)

Module	Size	Module	Size
OS	50	Kerberos	166
IP (v4/v6)	126	KINK	50
IPsec	8	Crypto	2
		App	16
		total	419

3.4.2 Performance of the Bootstrap Sequence

Table 3.4 shows processing time of each function on the device, whose conditions are with and without cryptographic hardware. The values without parentheses mean net processing times, and the values in parentheses mean waiting times from sending a request til receiving a reply. KINK_I and KINK_R mean the processing of KINK initiator and responder. $\text{PS}_{PUT,\,IPsec}$ means PS's PUT transaction of a client side which is for registering device's IP address and includes the overhead of IPsec ESP. $\text{PS}_{GET,\,IPsec}$ means PS's GET transaction of a client side which is for getting device's boot data (512 bytes) and includes the overhead of IPsec ESP. The waiting times of KINK_I, i.e. the values in parentheses, are varied because they depend upon the performance of a peer. The waiting time 112 msec occurs where a device initiates KINK with PS. The waiting time 213 msec occurs where a device initiates KINK with another device which has cryptographic hardware. The waiting time 421 msec occurs where a device initiates KINK with another device which does not have cryptographic hardware.

Those values exclude the processing time of IP address configurations, i.e. DHCP in IPv4 or RFC2462 in IPv6, and L2 address resolution, i.e. ARP (Address Resolution Protocol) in IPv4 or ND (Neighbor Discovery) in IPv6.

The bootstrap sequence described in Section 3.2.3 require the following functions: KDC, NTP, TGT, two TGSs, PSD_{KRB_PRIV}, KINK_I, $\text{PS}_{PUT,\,IPsec}$ and $\text{PS}_{GET,\,IPsec}$ (see Figure 3.2 through Figure 3.5). Figure 3.7 shows the processing time of the sequence. The values without parentheses mean net processing times and The values in parentheses mean waiting times.

Table 3.4. Processing time of each function on the device (msec)

crypto H/W	w/	w/o
KDCD	349 (0.5)	349 (0.5)
NTP	27 (0.5)	27 (0.5)
TGT	74 (23)	106 (23)
TGS	195 (24)	294 (24)
PSD_{KRB_PRIV}	239 (2)	373 (2)
$KINK_I$	263 (112 or 213)	465 (112 or 421)
$KINK_R$	213 (0)	421 (0)
$PS_{PUT,\,IPsec}$	40 (13)	164 (13)
$PS_{GET,\,IPsec}$	51 (17)	362 (17)

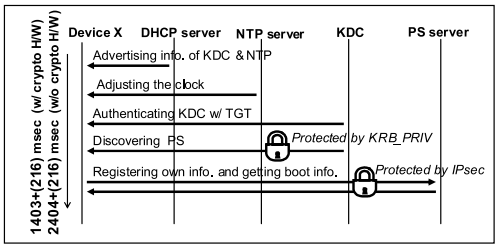

Fig. 3.7. Performance of the Bootstrap Sequence

3.4.3 Performance of the Device-to-Device Communication

As described in Section 3.2.4, the device-to-device communication has the overhead (see Figure 3.6). If the device is an initiator of the communication, the overhead is $PS_{GET,\,IPsec}$, TGS and $KINK_I$. If the device is a responder of the communication, it is only $KINK_R$. Once IPsec is established between devices, the performance of IPsec is one of major factors.

Figure 3.8 shows the overhead and IPsec's throughput. The values without parentheses mean net processing times and the values in parentheses mean waiting times. The throughput of IPsec ESP inbound is omitted because both performances are nearly the same.

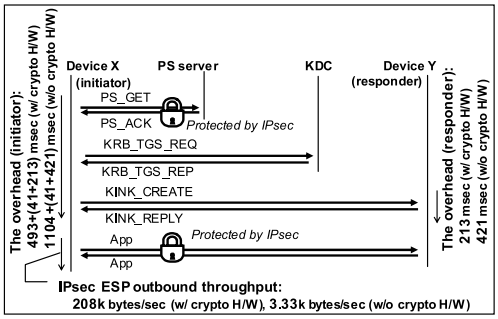

Fig. 3.8. Performance of Device-to-device communication

3.5 Considerations

3.5.1 Object Code Size

The size 400 K bytes (see Table 3.3) is not small enough for small embedded devices. For example, there is not enough room for application programs if using internal 512 KB FLASH ROM of H8/2029 only. IP stack and Kerberos occupies 30% and 40% of the entire code. Hardwired functions can improve the size. One of the popular examples is iReady's Ethernet MAX, which is the hardwired IP stack. It is a further study item to shrink the object code size.

3.5.2 Performance of the Bootstrap Sequence

The sequence takes 1619 msec or 2620 msec with or without cryptographic hardware (see Figure 3.7). It will usually take one or several days to start the entire system if the system is a large one because of starting subsystem by subsystem for making sure. If assuming to start a system described in Section 3.1 with device-by-device manner, i.e. seventeen thousand devices, within 24 hours, every device will have to start within 5 seconds. The actual margin is longer than the above time because of starting a system in a subsystem-by subsystem manner instead of device-by-device usually. Therefore, the

performances of the sequence can be acceptable.

We may have to consider for burst accesses to servers if the system has a large number of devices. For the device side, randomly delayed bootstrap can be a solution. However, the necessity and the validity are future study items. For server side, redundancy can be a solution, i.e. the redundancies of KDCs and PSs. It is not difficult to make KDC redundant[156]. But it is a further study item for PS.

3.5.3 Performance of Device-to-Device Communication

The overhead of the initiator takes 747 msec or 1566 msec with or without cryptographic hardware (see Figure 3.8). The overhead happens when both the device starts and the lifetime of Kerberos's tickets or IPsec SAs is expired. The former case can be acceptable with the reasons in Section 3.5.2. The latter case should be considered because the response time of BA or PA system should usually be on the hundred micro-second order. However, the overhead can be acceptable if the lifetimes are long enough, e.g. days, weeks or months, and are tuned operationally. The overhead of the responder can also be acceptable because it is shorter than the initiator's one.

As an example of IPsec's throughput, the processing time for 1024 bytes payload takes 5 msec or 307 msec with or without cryptographic hardware (see Figure 3.8). Considering the response time, i.e. the hundred micro-second order, the former case is fast enough, but the latter is not. The performance can be improved with AES instead of 3DES if the device cannot introduce cryptographic hardware.

3.6 Related Work

UPnP[13] (Universal Plug and Play)[286] is also a framework of device's auto-configuration. UPnP covers not only devices in home networks but also ones in buildings, e.g. HVAC (Heating, Ventilating and Air-Conditioning), temperature sensors or lighting.

UPnP uses SOAP (Simple Object Access Protocol), HTTP and TCP for control messages and GENA (General Event Notification Architecture), HTTP and TCP for notification messages whereas existing control networks usually use UDP for those purposes. It means that they will have to be changed if introducing UPnP. Therefore, the goal of UPnP is different from our proposed model because one of our goals is to minimize the impact on them when introducing an auto-configuration mechanism.

Public key cryptography is mandated for UPnP. So UPnP's applicability to devices is also different from our proposed model.

3.7 Conclusion

Through implementing the model experimentally, this paper shows the practicability of our proposed model which is intended to solve the issues, i.e. security and configuration complexity, while satisfying the restrictions, i.e. small embedded devices, isolated networks, private name space/naming system and inheriting the property. Secure bootstrap sequence and device-to-device communication using the chain of trust are the points of the model.

The object code size of the prototyped device is not small enough for small embedded devices. It is a further study item to shrink the size. The device's performance of the bootstrap sequence and device-to-device communication can be acceptable if the lifetimes of Kerberos's ticket and IPsec SAs are turned operationally, and if cryptography is implemented reasonably, i.e. cryptographic hardware or faster algorithm than 3DES in software. It is a further study item to make servers redundant.

13　UPnP is a trademark of the UPnP Implementers Corporation.

第4章　まとめ

TACA ワーキンググループでは、以上説明した課題検討と並行して、上記 Bootstrap プロトコルで、各低コストノードの初期認証、セキュリティ鍵交換に使用する Kerberos ベースの IPSEC 鍵交換プロトコル KINK に関し、IETF KINK ワーキンググループで標準化を推進した。

現在、最終ドラフト（Kerberized Internet Negotiation of Keys（KINK）―`draft-ietf-kink-kink-11.txt`、坂根、鎌田が共著）がワーキンググループ Last Call を完了し、Proposed Standard になることが承認されており、RFC 発行待ちの状態である。