Vol.2011-ARC-196 No.6
2011/7/27

PCI Expressを用いた通信リンク PEARLにおけるネットワーク管理機構

金子紘也[†1] 塙敏博[†1,†2] 三浦信一[†2]
朴泰祐[†1,†2] 佐藤三久[†1,†2]

我々はこれまで，PCI Express を直接通信リンクに用いることで，省電力かつ高性能な通信を実現する PEARL を提案してきた．しかし，これまでの PEARL には自律的に経路設定を行う機構がなく，事前にネットワークトポロジに応じて経路表を用意する必要があった．本稿では，PEARL を用いてクラスタを実現する際に必要な，ノードの追加削除の管理や耐故障機構を実現することを目的とする，PEARL Network Manager （PNM）の実装を行った．その結果，PEARL で接続されたクラスタにおいて，PNM が経路制御表を正常に初期化し，各ノードへの最短経路を導出していることを確認することができた．

1. はじめに

近年，計算機を相互結合網で複数台結合したクラスタ計算機の利用が盛んである．クラスタ計算機においては，計算ノードの性能と共に，相互結合網の性能がシステム全体に非常に大きな影響を与えるため、クラスタ向け相互結合網には非常に高いスループットと低いレイテンシが求められる．同時に，多数の計算ノードをからなるクラスタ計算機では，計算ノードの故障が計算中に起きることを現実的に想定する必要がある．現在一般的に利用されている InfiniBand[1] や Ethernet[2] 等の相互結合網は，これらの要求を一定のレベルで実現しているが，これまで高性能クラスタ向けに開発されていたこれらの相互結合網は，消費電力が大きく，近年の省電力化が進むクラスタ計算機において問題になってきている． これを解決するために，我々は，PCI Express[3] を直接通信リンクに用いることで，高い通信性能と耐故障性，省電力を実現する相互結合網，PCI Express Adaptive and Reliable Link (PEARL) の開発を行ってきた[4]．しかしながら，これまでの PEARL では，ネットワーク管理機構が存在せず，あらかじめ経路制御に関連する情報を全て設定しておく必要があった．また，リンク故障時に自動的に迂回経路に切り替えや，省電力化のため一部のリンク速度を低下させることにより最適経路が変更される場合など，自律的にネットワークの状態をモニタして経路情報などを管理する機構が求められる．そこで本研究では，PEARL において柔軟な経路制御及びアドレス管理を行うネットワーク管理機構 PEARL Network Manager (PNM) を提案する．さらに，初期状態におけるノードへのアドレス割当て，経路設定の機能を実装し，PEARL ネットワーク上で実際に動作させて評価を行う．

†1 筑波大学大学院システム情報工学研究科
†2 筑波大学　計算科学研究センター

2. PEARL の概要

PCI Express Adaptive Reliable Link (PEARL) は，PCI Express (PCIe) を直接通信リンクに用いる相互結合網である[4]．また，PEARL を実現するための一種のルータチップとして，PCI Express Adaptive Communication Hub (PEACH) チップを用いる．本節では特にネットワーク管理機構を検討する上で重要となる PEARL における通信機構について記述する．

2.1 PCI Express Adaptive Communication Hub （PEACH)

PEACH は，PCI Express Gen. 2 x4 レーンを 4 ポート，M32R プロセッサ SMP 4 コア，DMA コントローラ，512KB の SRAM 等を搭載する．M32R 上で Linux/M32R を動作させ，デバイスドライバによって PCIe ポート間の DMA 転送を制御することで高速な PCIe ポート間データ転送を実現する．PEARL の接続例を**図 1** に示す．PEACH の各 PCIe ポートには，RootComplex (RC) と EndPoint(EP) を切り替える機構が備わっている．これは，一般的な PCIe 接続の機器はその用途からポートが RC/EP のどちらで動作するかが事前に決定しているが，PEACH の場合は相互に接続するため，接続時に RC/EP を決定する必要があるためである．

各 PCIe ポートは，動作中であっても独立にレーン速度，レーン数を切り替えることができる．これらを適切に変更することで，必要な性能に応じた省電力化が可能になる．

2.2 PEACH における通信機構

PEACH 間のデータ転送は，主に DMA 転送によって行われる．PEACH の各 PCIe ポートに内蔵された DMA コントローラは，PEACH 内部アドレスと PCIe アドレス間で転送を行うデータ転送の際には，PEACH 内蔵 SRAM をパケットバッファとして用いることになる．従って，対向する PCIe リンクに接続された PEACH 同士が通信を行う際には，転

Vol.2011-ARC-196 No.6
2011/7/27

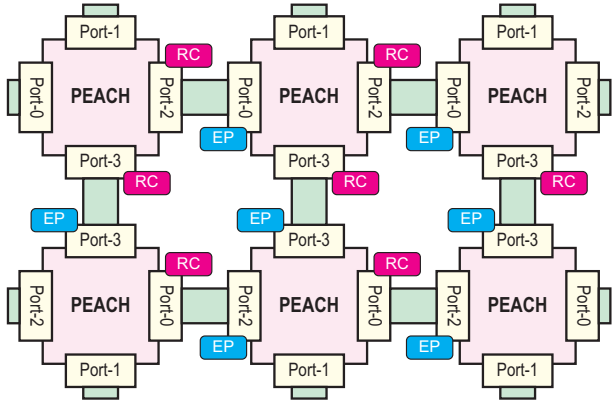


図 1 PEARL 接続例

送相手の PEACH の SRAM に割り当てられた PCIe アドレスに対して DMA 行う．そのため，PEACH の内蔵 SRAM をあらかじめ PCIe アドレス空間にマップする必要がある．PEACH では，RC 側が PEACH の内部アドレスと PCIe アドレスとのマッピングを自分自身 (RC) と対向の EP について設定を行う．一方，EP は RC 側の PEACH 内蔵 SRAM の PCIe アドレスを知る手段がない．

これまで PEACH では，PEACH 間の PCIe リンクはそれぞれが独立した PCIe アドレス空間を持ち，PEACH 中の M32R により中継操作を行っている．そこで，各 PCIe リンクが独立したアドレス空間であることを利用して，全てのリンクについて，SRAM に対し同じ PCIe アドレスを割り当てることができる．

これにより，実際には相手の PEACH 内蔵 SRAM の PCIe アドレスはあらかじめ決められた PCIe アドレスを使用してアクセスしている．

2.3 これまでの PEARL の問題点

一方，PEACH には，イニシエータ機能と呼ばれる，複数ノード間の高速なハードウェア転送機構が備わっている[5]．イニシエータ機能を使用する場合には，PEARL ネットワーク上のすべてのリンクで PCIe アドレス空間を共有する必要がある．そのため，各 PEACH の SRAM アドレスも全て異なる PCIe アドレスに割り当てる必要があり，事前に PEARL のネットワークトポロジを決定し，アドレスマッピングを静的に設定する必要があった．しかしながら，トポロジに従って事前にアドレスを設定する作業は煩雑であり，トポロジも固定されてしまうという問題点がある．

複数の PEACH を経由する際の経路情報に関しても，これまでは事前に設定する必要があった．しかし，リンク故障や省電力制御に伴って，リンク状態が動的に変化した場合には，対応が困難である．従って，リンク状態に応じて自律的に経路制御などを行う仕組みが必要である．

3. PEARL Network Manager (PNM) の提案

本節では，PEARL においてアドレス管理，経路制御を行い，高い耐故障性の実現と省電力制御にも対応する，ネットワーク管理機構 PEARL Network Manager (PNM) を提案する．

3.1 PNM の目的

PNM は，PEARL に対して高速でかつ，高い耐故障性と，省電力機能を実現する．まず，耐故障性に関して，PNM では動的なノードの追加削除や，故障に対応するために，動的なノードアドレスの割り当て及び経路の再構築を行う．これにより，トポロジの変化に対して柔軟に追従していくことが可能になる．同時に，動的に割り当てたノードアドレスと PCIe アドレスとのマッピングを管理することによって，PEARL ネットワークにを利用可能にする．省電力機構に関して，PNM は，PEACH の PCIe ポートのレーン速度，レーン幅を通信状態に応じて制御することで，消費電力の低減を行う．加えて，経路構築においても消費電力の低減を目的として使用するリンクを制限したり，省電力のために転送レートを落とした場合の最適な経路を再計算したりするなどの制御を行う．

3.2 既存のネットワーク管理機構

クラスタ向け相互結合網として広く利用されている InfiniBand や Ethernet では，アドレスの動的な割当てや経路表の構築を行うネットワーク管理機構が存在する．Ethernet 上で IP プロトコルを用いる場合には，代表的なものとして，アドレスの割当てに用いられる Dynamic Host Configuration Protocol (DHCP)[6]，ルーティングプロトコルにはディスタンスベクタ方式の Routing Information Protocol (RIP)[7]，リンクステート方式の Open Shortest Path First (OSPF)[8] などがある．また，InfiniBand においては，ネットワーク内で唯一のマスタノードを選出し，マスタノードがアドレスの配布から経路表の構築までを行う Subnet Manager と呼ばれるネットワーク管理機構が挙げられる．

3.3 ネットワーク管理モデルの検討

PNM が柔軟な経路制御，リンクの状態制御，PCIe アドレス管理を行うために適したネッ

トワーク管理モデルについて検討する．まず，PNM には，耐故障性の観点からデバイス抜差時のトポロジ変化に対する高速な追従性が求められる．そのため，RIP に代表されるディスタンスベクタ方式ではなく，リンクステート方式の経路構築が適していると言える．また，省電力制御を行った場合には，最適な経路を検討するためには，ネットワーク全体の状況に応じた大域的で一貫性のある経路の決定が必要である．そのため，本研究で開発を行う PNM に関しては，InfiniBand における Subnet Manager と似たシングルマスタ-スタンバイノード方式を用いることとした．シングルマスタ-スタンバイノード方式では，ネットワーク上でただ一つのマスタノードが各ノードに関する全ての経路表を構築し，配布を行う．そのため，ネットワーク全体で一貫した経路戦略を取ることができ，省電力制御などを考えた場合に適している．スタンバイノードは，自身が隣接するポートに対するデバイスの抜き差しがあった場合にマスタノードに対して通知を行う．これによって，マスタノードは常にネットワーク全体のトポロジを把握していることとなる．また，スタンバイノードはマスタノードの死活をポーリングによって確認し，応答がない場合は速やかにマスタノードの再選出を行う．これによって，高い耐故障性を実現している．本ネットワーク管理モデルをまとめると以下の通りである．

(1) ネットワーク内に唯一のマスタノードが存在する
(2) トポロジの変化は全てマスタノードに対して通知される
(3) マスタノードはネットワーク全体のトポロジ情報からルーティングテーブルを構築し，配布する

また，本モデルはマスタノードに計算能力が必要であるが，PEARL においては，PEACH に内蔵された M32R コアが十分な計算能力を持っているため，これは問題とはならない．

4. PNM の実装

本節では，本研究で対象とするアドレスの動的な割当て及び経路表の構築を行うアルゴリズム及び実装手法に関して述べる．

4.1 Linux/M32R と PEACH

PEACH チップには M32R コアが内蔵されており，この上で Linux/M32R が動作している．PNM は Linux/M32R 上におけるユーザランドプログラムとして動作する．PEACH ハードウェアへのアクセスは，デバイスドライバが提供する sysfs インタフェースを用いて行う．sysfs インタフェースでは，以下のような情報を得ることができる．

(1) 各 PCIe ポートのリンク情報（リンクアップ・ダウン，レーン速度，レーン数）
(2) PCIe アドレスマッピング情報
(3) ハードウェア ID に関する情報

これら sysfs インタフェースは *epoll()* による監視に対応しているため，linkup/down 等のイベントをユーザランドで動作する PNM から高速に検知することが可能である．

まだ PCIe アドレスが設定されていない PEARL においても，同一 PCIe リンクに接続された PEACH 間で最低限の通信を行う必要がある．そのため，PCI Express で規定されているベンダ定義メッセージを利用したメッセージ通信機構をデバイスドライバ内に実装した．これも sysfs によってユーザ空間に提供され，PNM が隣接する PEACH と通信を行うために利用することができる．なお，メッセージは，PCIe 上で通常用いられるメモリリード・ライトとは別のキューを用いるため，互いに影響を受けない．

4.2 PEARL ネットワーク全体のトポロジの把握と経路構築

PNM は，起動直後の段階ではネットワークトポロジに関する情報を持っていない．本節では，隣接する PEACH 間のみでメッセージ通信が可能な状態から PEARL ネットワークのトポロジを決定し，経路構築を行う手法について述べる．まず，起動直後の PNM は，sysfs を通じて，

(1) 自身のハードウェア ID (以下 HWID)
(2) 自身のポート接続情報（ポートの LinkUp/Down , 接続先 PEACH ノードの HWID)

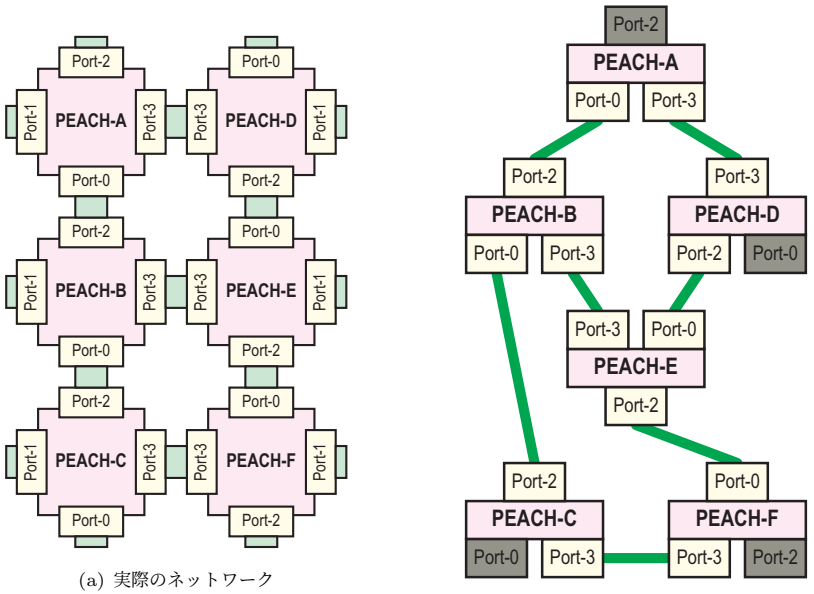
を取得することが出来る．ここで，ハードウェア ID (HWID) とは，PEACH チップがハードウェア的に持つ PEACH チップユニークな ID である．

PEACH チップは PCIe ポートを 4 つ搭載している．そのため，PEARL ネットワークは **図 2** に示すように各 PEACH ノードを根とし，枝が各ノードから 4 本伸びている閉路を含む木として形式化することが出来る．

図 2 を見ると，木として表現された PEARL において，根から任意の葉となるノードまでの経路は，ポート番号のリストによって表現できることがわかる．よって，本節で行うべき経路構築とは，各 PEACH ノードへ最短数で至るポート番号のリストを作ることと考えることが出来る．これは，図の木を幅優先探索していくことで得ることが出来る．

ここで，**図 3** における PEACH A を例に経路構築プロセスを説明する．まず，A が起動した直後，A 上で動作する PNM は，図 3 中の破線で囲まれた，自身に直接隣接するノードまでの経路のみを認識している．ここで， A は B に対して，B の隣接ノードに関する情報を要求する．その結果，PEACH B には A と，E, C が接続されていることがわかる．さらに，新しく発見されたノード C, E について，それぞれ B のポート p0, p3 に接続されてい

Vol.2011-ARC-196 No.6
2011/7/27



(a) 実際のネットワーク

(b) Tree 構造に形式化後

図 2 PEARL ネットワークの形式化

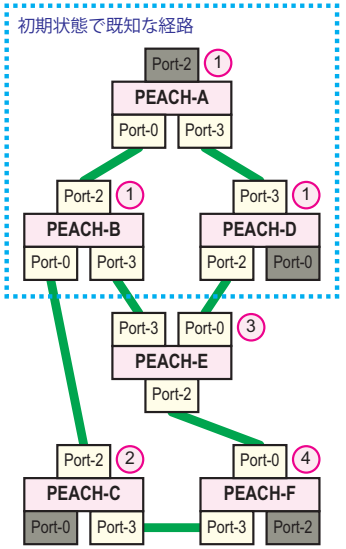


図 3 経路構築プロセス

表 1 探索によって得られる経路

ノード	経由ポート
PEACH A	(自ノード)
PEACH B	p0
PEACH D	p3
PEACH C	p0->p0
PEACH E	p0->p3
PEACH F	p0->p0->p3

ることがわかる．この段階で，PEACH A から B に対する経路はポート p0 に接続されていることが既知であるため， C に対する経路も B への経路の末尾にポート p0 を付け加えた p0→p0 であることがわかる．この操作を，新しい HWID を持つ PEACH ノードが出現しなくなるまで再帰的に行う．その結果，図 3 に示す順序でネットワークの走査が行われ，PEARL ネットワーク内に存在する全 PEACH ノードに対する経路と，全 PEACH ノードの HWID の収集を行うことが出来る．

ここで，本アルゴリズムを行うためには直接隣接しないノードに対してポートの接続情報要求とそれに対する返答を受け取る必要があることがわかる．しかし，PEACH には，現時点で 2 ホップ以上離れたノードに対するメッセージの転送機構は存在しない．そのため，本アルゴリズムを実現するために，遠隔ノードへのデータ送受信を行うポートベースパケット転送機構の実装を行った．

4.3 遠隔ノードとのデータ送受信機構

前節で，遠隔ノードへの経路はポート番号のリストによって表せることを示した．従って，隣接ノードへの通信しか行うことの出来ないベンダ定義メッセージを用いて，遠隔ノードへのデータ送受信を行うためには，ベンダ定義メッセージを何らかの形で中継する機構が必要であることがわかる．

本研究では，これを実現するために，**表 2** に示す構造を持つパケットを送信する．

パケットはヘッダとペイロードからなり，ヘッダにはポート番号の列挙と目的ノードまでのホップ数，送信元までの経路に関する情報が格納され，ペイロードに送信先に対するメッセージを格納する．ベンダ定義メッセージを受信した転送用デーモンは，まず本パケットのヘッダを確認し，自分宛ではなかった場合に，自身の隣接ノードに対して再度ベンダ定義メッセージを用いて送信を行う．パケットが自分宛であった場合は，PNM のメッセージ処理デーモンが監視する FIFO に対してペイロードが格納される．これを検知したメッセージ処理デーモンは，適切な応答を生成し，送信元への情報の返信を行う．これを図示したものが**図 4** である．本機構によって，経路構築で必要とされる遠隔ノードへのポート接続情報要求などを PNM では実現している．

表 2 ポートベースパケット転送ヘッダ

フィールド名	意味	値
RoutingMethod	ルーティング方向	0x00: ポートベースパケット配送
Direction	配送方向	0x00: マスタ検出前 0x01: マスタから他ノード 0x02: 他ノードからマスタ
HopCounter	配送先までのホップ数	N: 配送先まで N 個の PEACH を経由
PathCounter	現在の転送回数	N: 通過した PEACH の個数
ForwardPortPath	行き先ポートリスト	1 Byte 毎にポートの番号を列挙 本フィールドの PathCounter 個目が転送先ポート 例：0x00010200 (Port 0, 1, 2, 0 の順に転送)
BackwardPortPath	通過ポートリスト	1 Byte 毎にポートの番号を列挙 転送中に追記，送信時は無視 例：0x00010200 (Port 0, 1, 2, 0 の順に通過)
PayloadSize	ペイロードサイズ	ヘッダを含まないペイロードサイズ [byte]
Payload	ペイロード本体	転送データ

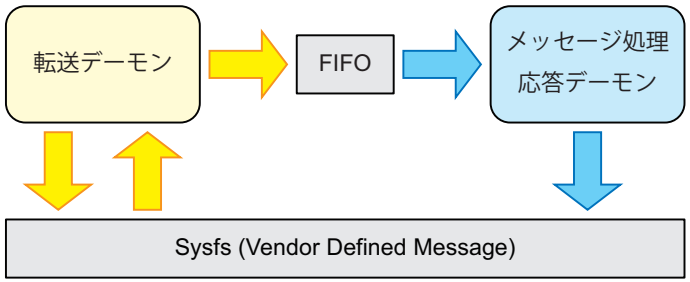


図 4 メッセージ転送

4.4 マスタノードの選出とアドレス配布

マスタノードとは PEARL ネットワーク内に唯一存在し，経路構築や配布などを行うノードである．PNM では，マスタノードの選出を簡単化するために，マスタノードを，PEARL ネットワーク内に存在する PEACH ノードの中で最も HWID の小さなノードと定義する．これによって，経路構築プロセスで収集した HWID と，自身の HWID を比較することによって，各ノードにおいて，自ノードがマスタノードであるかスタンバイノードであるかを自律的に判断することが出来る．マスタノードが決定された後，マスタノードが同一 PEARL ネットワーク内のスタンバイノードに対して，32 bit 整数値で表現される Local ID（LID）と呼ばれる ID の割り当てを行う．割り当ての際には，各ノードに対してマスタノードが割り当てメッセージを送信し，スタンバイノードはメッセージを受信後，自身の LID を更新する．

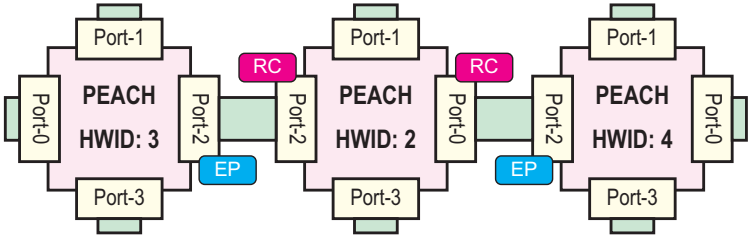


図 5 評価対象としたトポロジ

表 3 探索によって得られる経路

Source node	Destination node		
	2	3	4
2	NA	2	0
3	2	NA	2, 0
4	2	2, 2	NA

5. 検　　証

本節では，実装した PNM について，経路表の構築及びアドレスの割当てが正常に行えることを検証する．

5.1 検証環境

検証には PEACH を搭載した PCI Express アドインボードを用いた．これは，PCIe x4 のカードエッジを一つ，PCIe 外部ケーブルコネクタを 3 つ搭載するボードである．カードエッジと外部ケーブルコネクタはそれぞれ PEACH の PCIe ポートに接続されている．本ボードを PCIe 外部ケーブルで相互接続することで，PEACH 同士の PCIe リンクを確立し，PEARL ネットワークを構築することができる．本検証では，**図 5** に示すトポロジを用いて PEACH ボード 3 枚を直列に接続した．本検証では，カードエッジには接続せず，PEACH 間のみでの検証を行った．本環境において，PNM の検証を行った．

5.2 検証結果

各 PEACH ノードで PNM を動作させ，正常に経路表構築とアドレスの割当てが正常に行われた．取得できた経路表を**表 3** に示す．

決定されたノード種別及び割り当てられた LID を**表 4** に示す．ネットワーク内で最も小さい HWID を持つ HWID:2 のノードが正常にマスターノードとなり，他がスタンバイノー

表 4 ノード種別と割り当てられたアドレス

HWID	種別	LID
2	マスタ	1
3	スタンバイ	3
4	スタンバイ	2

ドとなったことが確認できる．また，正常に構築された経路表を用い，マスタからスタンバイノードへのアドレス割り当てメッセージが正常に送信された．その結果，ネットワーク上で重複のない LID が各ノードに正常に割り振られていることも確認できた．

6. おわりに

本研究では，PEARL において耐故障性，省電力性を実現するネットワーク管理機構 PEARL Network Manager (PNM) を提案した．加えて，PNM を実際の PEARL 上で動作させるための機構の実装を行い，実際に PEACH 上で動作させた．その結果，アドレスの割当て及び動的な経路構築を行うことができた．今後の課題としては，よりノード数の大きな環境での評価，耐故障性，省電力性を考慮した経路表の構築手法の検討などが挙げられる．

謝辞 本研究の一部は, 科学技術振興機構 戦略的創造研究推進事業 (CREST) 研究領域「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」, 研究課題「省電力高信頼組込み並列プラットフォーム」による．

参 考 文 献

1) Infiniband Trade Association: . http://www.infinibandta.org/.
2) ETHERNET WORKING GROUP: IEEE 802.3. http://www.ieee802.org/3/.
3) PCISIG: PCIe Base Spec 2.1. http://www.pcisig.com/specifications/pciexpress/.
4) Hanawa, T., Boku, T., Miura, S., Okamoto, T., Sato, M. and Arimoto, K.: Low-Power and High-Performance Communication Mechanism for Dependable Embedded Systems, *Proceedings of 2008 International Workshop on Innovative Architecture for Future Generation Processors and Systems*, pp.67–73 (2008).
5) Otani, S., Kondo, H., Nonomura, I., Ikeya, A., Uemura, M., Asahina, K., Arimoto, K., Miura, S., Hanawa, T., Boku, T. and Sato, M.: An 80Gb/s Dependable multicore Communication SoC with PCI Express I/F and Intelligent Interrupt Controller, *IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips XIV)*, p.3 pages (2011). PDF.
6) IETF: RFC2131 - Dynamic Host Configuration Protocol. http://tools.ietf.org/html/rfc2131.
7) IETF: RFC2453 - RIP Version 2. http://tools.ietf.org/html/rfc2453.
8) IETF: RFC2740 - OSPF for IPv6. http://tools.ietf.org/html/rfc2740.