Vol.2012-ARC-201 No.3
2012/8/1

PCI Expressネットワーク PEARLにおける耐故障機構

金子 紘也[1,a)]　塙 敏博[2]　児玉 祐悦[1,2]　朴 泰祐[1,2]　佐藤 三久[1,2]

概要：我々はこれまで，PCI Express を直接通信リンクに用いることで，分散メモリ並列環境における省電力かつ高性能な通信を実現する PEARL ネットワークシステムを提案してきた．しかし，これまでの PEARL には PCIe リンクやノードの障害を検知する機構が存在しなかった．本稿では，PEARL における障害の検知及び回復動作を検討し，実際に PEARL におけるネットワーク管理機構である PEARL Network Manager (PNM) において一部実装を行った．その結果，シングルマスタ-スタンバイノードモデルを持つ PEARL において，正常に障害を検知し，回復動作へ移行することが可能であることを確認することができた．

1. はじめに

近年，組込みシステムの高性能化に伴い，プロセッサを相互結合網で複数結合したマルチプロセッサシステムが広く利用され始めている．これら組込みシステムにおいては，性能と同時に，高い耐故障性と省電力性が求められる．現在一般的に利用されている CAN [1] や Flexray [2] はそれら要求を一定のレベルで実現しているが，その通信速度では，マルチメディアなどの大容量データを伝送するためには不十分である．クラスタ計算機で用いられている InfiniBand [3] や Ethernet [4] 等の高速結合網は，高い通信速度を実現するが，消費電力が大きく，省電力化の要求が強い組込みシステムには適さない．これを解決するために，我々は，PCI Express [5] 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 [6,7]. しかしながら，これまで PEARL には耐故障性を実現するためのソフトウェア的な経路制御機構が備わっておらず，あらかじめ設定した経路情報から静的に転送制御を行っていた．そのため，リンクやノードに障害が発生した場合に通信継続を行うことは困難であった．そこで本研究では，PEARL においてリンクやノードの故障を検知し，自動的に経路を再計算することで障害発生時の通信継続を可能とする耐故障機構を検討し，PEARL におけるネットワーク管理機構である PEARL Network Manager (PNM) において一部実装及び評価を行う．

[1] 筑波大学 大学院 システム情報工学研究科
Graduate School of System and Information Engineering, University of Tsukuba
[2] 筑波大学 計算科学研究センター
Center for Computational Sciences, University of Tsukuba
[a)] kaneko@hpcs.cs.tsukuba.ac.jp

2. PEARLの概要

PCI Express Adaptive and 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．PEARL では，PCIe によるネットワークシステム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 PEARL における耐故障機構を考える上で重要となる PEARL の構成及び PEACH チップについて述べる．

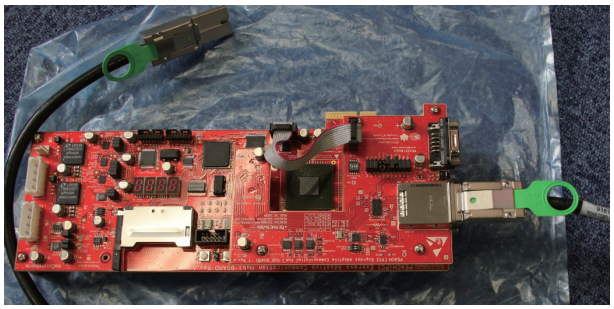


図 1　PEACH ボードと PCIe ケーブル

2.1 PCI Express Adaptive Communication Hub (PEACH)

PEACH チップは，PCI Express Gen2 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 ASIC である [8]. PEACH チップ搭載 PCIe ボードを**図 1** に示す．本ボードをホストに取り付けることにより，ノード間を PCIe リン

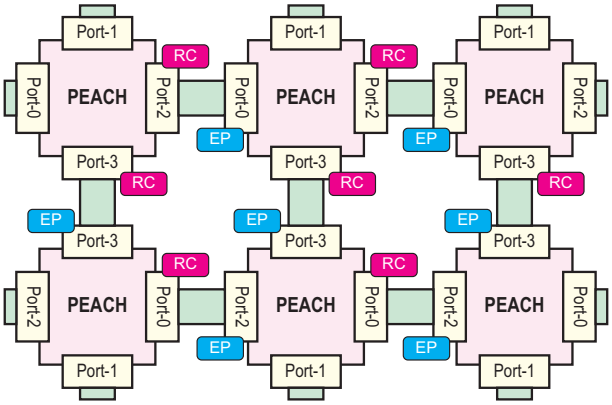


図 2 PEARL 接続例

クで相互接続する．PEACH チップは，搭載された PCIe ポート間のデータ転送を中継することによって，PEARL ネットワークを実現する．PEARL の接続例を**図 2** に示す．PEACH に搭載された各 PCIe ポートは，動作中に独立にレーン速度，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 Linux/M32R と PEACH デバイスドライバ

PEACH に搭載された M32R プロセッサ上では，Linux/M32R が動作しており，任意のソフトウェアをユーザランドで動作させることが出来る．PEACH デバイスの制御は Linux/M32R に組み込まれた PEACH デバイスドライバを通じて行われる．PEACH デバイスドライバを通して，ユーザランドのアプリケーションは各ポートの PCIe リンクのステート（リンクアップ/ダウン，モード，レーン数）の取得/設定を行うことが出来る．また，それらステートに変化があると，即座に M32R プロセッサに対してハードウェア割り込みが発生する．さらにそれらをユーザランドに伝える機構も備えている．

2.3 パケットイニシエータ機構

PEACH は，PCI Express パケットを高速にポート間転送する機構として，パケットイニシエータ機構を備えている [7]．これは，PEACH チップにハードウェア的に実装された機構であり，PEACH デバイスドライバからの転送設定を行うことによって，ソフトウェア制御を行うこと無く，自動的にポート間の高速パケット転送を行うことが可能である．

2.4 PEARL Network Manager (PNM)

PEACH は，前節で述べたようにハードウェア的にポート間のパケット転送を行うことができるが，複数ノードを経由する通信における経路構築や制御機構を搭載していない．これを解決するために，PEARL Network Manager (PNM) と呼ぶネットワーク管理機構を開発している．PNM は Linux/M32R ユーザランド上で動作するデーモンであり PEARL においてアドレスの割当てや経路表の構築，転送の制御を行う．これによって，隣接ノード間の通信をハードウェア的に実現する PEACH チップを用いてマルチホップのホスト間通信を実現する，PNM は各 PEACH チップに搭載された M32R プロセッサ上でそれぞれ独立して動作し，互いにメッセージ通信を行うことで PEARL 上で協調してネットワーク管理を行う．本研究では，PEARL における障害検出及びその回復について，PNM 上に実装を行う．

2.4.1 PNM のネットワーク管理モデル

ネットワーク管理機構では，経路表の構築や転送の制御を行うが，特に経路表の構築については大きく分けて，集中制御と自律分散制御，シングルマスタ制御に分類することが出来る．集中制御とは，OSPF [9] などで用いられている手法であり，ネットワーク全体のトポロジを各ノードが完全に把握し，各々のノードが全体のトポロジを考慮した経路表の構築を行う．一方，RIP [10] などで用いられているのが自律分散制御である．自律分散制御では各ノードは隣接ノード間で経路表の交換を行うことで，ネットワーク全体に経路表を伝搬させていく手法である．最後に，Infiniband [3] で用いられているのがシングルマスタ制御である．これは，集中制御と同様に全体のネットワークトポロジを把握し，経路表の構築を行うが，経路表に関して各ノードが各々計算するのではなく，ネットワーク内で唯一選出されたマスタノードがネットワークにおける N 対 N の経路構築をすべて行い，配布する手法である．マスタノード以外の全てのノードはスタンバイノードと呼ばれ，マスタノードから配布された経路表のとおりにパケット転送を行うと同時に，隣接するリンクにリンクアップ/ダウンが発生したことをマスタノードに通知する役割を持つ．PEARL では，特に省電力を実現する経路構築が求められるため，シングルマスタ制御を採用している．これは，一つのノードにおいてネットワーク内の全ての経路構築を行うため，通信リンクの片寄せなどの省電力制御を行うのに適しているためである．

3. PEARL における障害とその検出

本章では，PEARL における耐故障機構を考える上で重要となる，発生しうる障害の分類とその検出手法について述べる．PEARL における故障とは，何らかのハードウェア/ソフトウェア的な障害によって，通信の継続が行えなくなる状況として定義できる．ここで，PEARL において故障を引き起こす障害を列挙すると

(1) リンクダウンを伴う
　(1-a)PCIe リンクの切断
　(1-b)PEACH ノードにおける電源断

(1-c)ホストノードにおける電源断

(2) リンクダウンを伴わない

(2-a)PNM や PEACH デバイスドライバにおける障害

(2-b)PEACH チップの電源断を伴わない機能停止

が挙げられる．以下，これらの障害を PCIe リンクの切断を伴う障害（以下ハード障害）と PCIe リンクの切断を伴わない障害（以下ソフト障害）に分け，その検出について論じる．

3.1 PCIe リンクの切断を伴う障害（ハード障害）

ハード障害とは，PEACH 間もしくは PEACH-ホストノード間を相互接続する PCIe リンクが何らかの障害によって物理的あるいは論理的に切断された状態を指す．そのうち，(1-a)PCIe リンクの切断の場合は当然であるが，(1-b,1-c)PEACH ノード及びホストノードの障害の場合に関しても，ノードの電源が切断されることによって，結果的に PCIe リンクが切断される．そのため，(1-b,1-c) の障害に関しても，PCIe リンクの切断によってその発生を検出することが可能である．2.2 節で述べたように，PEACH デバイスドライバはリンクダウンに伴うハードウェア割り込みを通知する機構を持つため，リンク障害は全て PEACH デバイスドライバからの通知によって検出することが可能である．

3.2 PCIe リンクの切断を伴わない障害（ソフト障害）

一方，障害発生時にリンクダウンを伴わないソフト障害も存在する．これは，(2-a)PNM や PEACH ファームウェアにおいて，何らかのソフトウェア的な障害によってリンクアップが維持されたまま正常なパケット転送が行えなくなっている状況や，(2-b)PEACH チップが何らかの障害によってリンクを維持したまま正常な応答を返さなくなった状態に該当する．ソフト障害が発生したノードは，PEARL において，正常なパケット転送を阻害する要因となるため，検出及び PEARL からの隔離，ソフトウェアリセットなどによる復旧を試みる必要がある．ソフト障害の検出には，正常にメッセージ通信が行えるかどうかを隣接ノード間で定期的に確認する手法が有効である．PNM においても，定期的に隣接ノード間で応答パケットを要求するメッセージを送信しあう事（Polling）によって，ソフト障害を検出することが可能である．本研究では，隣接ノード間の Polling に加え，マスタノードとの Polling を常に行うことで，よりシステムの安定性を高めている．

3.3 障害検出手法の比較検討

本章では，ここまで PEARL におけるハード/ソフト障害を検出する手法として，デバイスドライバからの PCIe リンクダウン割り込みを利用する方法と，PNM 間において相互に Polling を行う手法が考えられることを述べた．

表 1 障害とその検出手法

障害検出手法	対処可能な障害	応答速度
割り込みによる監視	ハード障害のみ	高速
Polling による監視	全ての障害	送信間隔に依る

ここで，Polling による手法について，検出可能な障害を考えてみると，PNM からのメッセージ送受信を確認するため，ハード障害の検出も可能であることがわかる．しかし，Polling による手法では，その検出タイミングがメッセージ送信間隔に依存するため，割り込みによる手法と比較して遅延が大きくなる．そのため，PNM においては割り込み及び Polling を併用することにした．これによって，ハード障害に対して高速に応答し，同時にソフト障害の検出も可能となる．**表 1** に障害とその検出手法の特性についてまとめた．

4. 障害からの回復

3 章ではリンク障害とソフト障害それぞれについてその検出手法を述べた．本節では，検出した後通信を正常に継続するための障害回復手法について述べる．

4.1 障害回復の流れ

障害が発生した後に通信を継続するためには，マスタノードが障害ノード/リンクを迂回する経路を再計算し，その後ネットワークに属する PNM 全てがその情報を共有する必要がある．障害発生時に通信を継続するためには以下の処理を行う必要がある．

(1) 障害を検出

(2) 検出された障害をマスタノードに報告

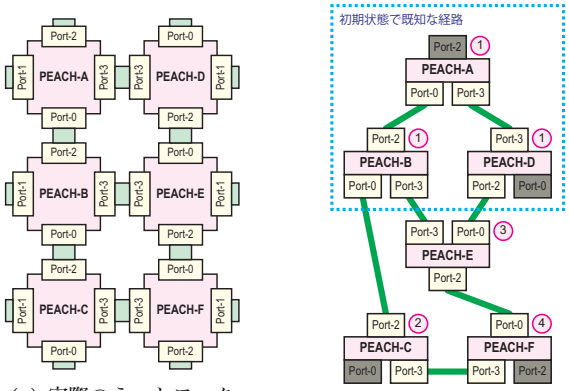
(3) マスタノードはスタンバイノードに対して再計算した経路を配布

以上の回復プロセスを行うことで，障害からの回復が可能である．マスタノードで故障が発生した場合には，これに加え，マスタノードの再選出を行う必要がある．ここからは，特にマスタノードの再選出手法について述べることで，マスタノード障害からの回復動作を概説する．

マスタノードを再選出する手法には，ブロードキャストを利用する手法と，マスタノード隣接ノードがマスタを引き継ぐ手法の 2 つが考えられる．

4.1.1 ブロードキャスト利用によるマスタノード選出

ブロードキャストを利用するマスタノード選出は，PEARL が起動時に行う初期経路及びマスタノード選出と同じ手法である．本手法は，PEARL ネットワークを **図 3** 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．各 PEACH ノードは，木として形式化された PEARL において幅優先探索を行う．これによって，**図 3(b)** に示す順番で PERAL ネットワーク内のノードの走査が行われ，各



(a) 実際のネットワーク　(b) Tree 構造に形式化後

図 3　PEARL ネットワークの形式化

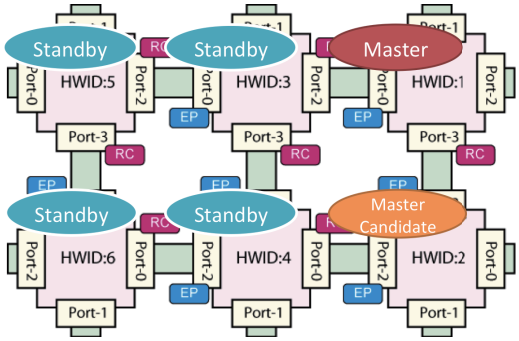


図 4　マスタ候補ノードの導入

ノードへの経路と PEARL ネットワークに属する PEACH チップ全てのハードウェア的な ID(以下 HWID) を収集することが出来る，ここで，PEARL におけるマスタノードをネットワーク内で最も小さい HWID を持つノードと定義することで，各ノードが自律的に自身がマスタノードかどうかを判断する．本手法は，本来 PEARL ネットワーク起動時に各ノードが初期経路の構築とマスタノードの選出を同時に行うための手法であるが障害回復に際しても，ネットワークを一度全て初期化し直すという方針においては，すべてのノードで本手法を再度行うことは有効であるといえる．一方，本手法では，各ノードがそれぞれネットワーク内のすべてのノードに対して通信を行うため，パケットの送信量は $O(n^2)$ のオーダとなり，ネットワーク全体への負荷が増大する可能性がある．

4.1.2　隣接ノードにおける自律的マスタ昇格

一方，ネットワーク全体へのブロードキャストを行うことなくマスタノードの再選出を行う手法として，マスタノードに隣接するノードがマスタを引き継ぐ手法が考えられる．マスタノードが機能不全に陥った時に備え，事前にマスタ権限を移譲すべき隣接ノード（以下マスタ候補ノード）をマスタノードが決定して，通知を行なっておき，自律的にマスタを引き継ぐ．**図 4** に示すように，マスタノード (HWID:1) に隣接するノード (HWID:2) がマスタ候補ノードとして動作し，マスタの障害を検知するとすみやかにマスタに昇格すると同時にネットワーク全体にブロードキャストを行い，現在の PEARL 全体のネットワークトポロジを把握し新しいマスタノードであることを通知する．本手法を用いる利点は，マスタノードに障害が発生した際に，マスタが引き継がれるノードが既に決定しているため，ネットワーク全体に対してブロードキャストを行うノードは常にマスタ候補ノード一つに抑えることが出来る点である．これによって，通信量を削減すると同時に，特にリンク障害発生時に迅速なマスタの再選出を行うことが可能となる．一方，マスタノードからマスタ候補ノードに連鎖的に障害が発生し，本来自律的にマスタ昇格を行うべきマスタ候補ノードが正常にしなかった場合に，PEARL ネットワーク内にマスタが存在しない状態が発生してしまう可能性がある．

4.1.3　ブロードキャスト及び自律昇格の併用

そこで，本研究では 4.1.1 節で述べたブロードキャスト利用によるマスタノード選出及び，4.1.2 節で述べた隣接ノードにおける自律的マスタ昇格を併用する手法を提案する．本手法では，隣接ノードにおける自律的マスタ昇格を基本的な手法とし，それに併用する形でブロードキャストによるマスタノード再選出を行う．マスタ候補ノードがマスタノードを監視するのと同時に，マスタ，マスタ候補ノード以外のノード（以下スタンバイノード）は常にマスタノードへのポーリングによる死活監視を行い，一定の時間内にマスタノードからの応答がなかった場合にはネットワーク全体に対してブロードキャストメッセージによる再選出を要求する．マスタ候補ノードが自律的にマスタを引き継いだ場合には，スタンバイノードに対してマスタが移動したことを通知し，スタンバイノードは監視対象を新しく選出されたマスタノードに変更する．これによって，基本的にはマスタ候補ノードによる自律的なマスタ引き継ぎを行い，何らかの障害によってマスタ候補ノードによる自律的なマスタ昇格が行われなかった場合にも，一定の時間内にマスタの障害を検出し，ネットワーク全体でマスタの再選出を行うことを保証することが可能となる．

5.　検証

本章では，本研究で提案した耐故障機構の実装を行い，実際の PEARL ネットワーク上で検証を行う．特に，本検証では現時点で実装が完了している障害の検出機構及びマスタの再選出機構について実際の PEACH ボードを用いた PEARL ネットワーク上で検証を行った．

5.1　検証環境

検証には PEACH ボードを用いた．これは，2.1 節で述べたように，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．本検証では，**図 5** に示すメッシュトポロジを構成した (**図 6**).

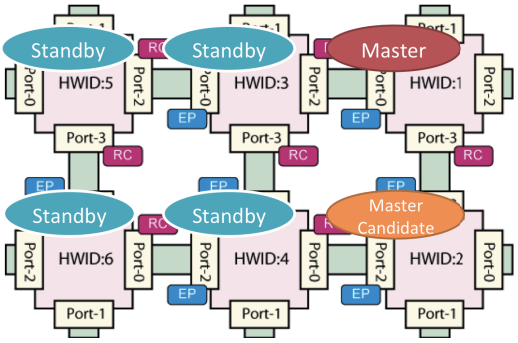


図 5 検証対象とするトポロジ

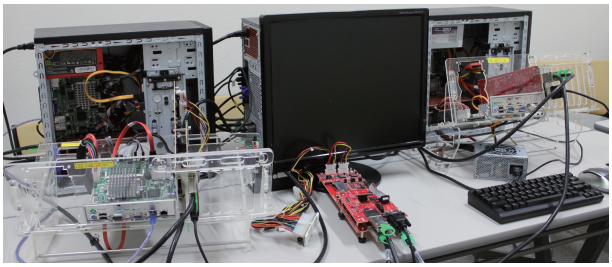


図 6 検証環境

5.2 検証手法

本検証ではまず，マスタノードの再選出手法に関して，ブロードキャストを用いた手法，自律昇格を行う手法のそれぞれを行った場合に，実ネットワークにおいて発生する通信量を測定し，2つの手法に関して通信量の比較を行う．次に，これまで述べた耐故障機構の動作を検証するために，いくつかの故障シナリオを設定し，実環境での障害検出が行えることを検証する．

5.3 再選出手法のコスト評価

本節では，マスタノードの再選出手法の通信コストを評価する．検証にあたっては，その通信量を測定するために，図 5 のトポロジにおいて，マスタノード (HWID:1) に電源断を伴うハード障害が発生したことを想定する．障害が発生した際に，すべてのノードが初期構築プロセスを行った場合（ブロードキャストベース）と，マスタ隣接ノードのみが初期構築プロセスを行った場合（自律昇格ベース）において，PEARL ネットワーク全体で転送されるパケットの転送量を**表 2** に示した．ここで，表 2 における PEACH2 の転送量 1152byte は，図 5 に示すネットワークを幅優先探索で走査するのに必要なパケット転送サイズの総量である．この中には，各ノードにおける HWID だけでなく，ポートの状態を示す情報が含まれている．

表 2 より，自律昇格を行う手法を用いた場合，マスタノードを引き継ぐノードが既に決定しているため，パケット送信を実際に行ったノードが一つに限定されることがわかる．その結果，自律昇格ベースの手法では，再選出プロセスにおけるネットワーク全体のデータ転送量が削減できていることがわかる．ネットワーク内のすべてのノードが他のすべてのノードに対する通信を行う必要があるブロードキャストベースの手法では，ノード数が増加すると全体の通信量増加は $O(n^2)$ のオーダとなる．そのため，ノード数が増えるにつれてより自律昇格を用いた手法が有利になると考えられる．また，表 2 では，各ノードにおいて転送量のばらつきが見られる．これは，各ノードにおける他のノードに対する最短ホップ数の合計が異なるためである．

表 2 マスタ再選出に伴って発生する転送量

送信元ノード	転送量 (BroadCast)	転送量 (自律昇格)
PEACH2	1152byte	1152byte
PEACH3	864byte	0byte
PEACH4	720byte	0byte
PEACH5	1008byte	0byte
PEACH6	864byte	0byte
全ノード計	4608byte	1152byte

5.4 実環境における障害検出

次に，実際に PEARL において発生しうる障害を実際に検出できることを，下に示す 3 つの故障シナリオそれぞれについて検証する．

(1) Case1:HWID5-HWID6 間の PCIe リンク切断
(2) Case2:マスタノードに電源断を伴う障害が発生
(3) Case3:マスタノードにソフト故障が発生

これらのシナリオでは，一般的なスタンバイノード間のリンク故障及び，マスタノードにおけるハード/ソフト故障を想定することで，シングルマスタ-スタンバイノード方式の PNM において適切な障害検出が行えることを検証する．また，各シナリオは図 5 に示したように適切にマスタ，マスタ候補ノード，スタンバイノードが選出され，各スタンバイノードがマスタノードへの Polling を正常に行なっている状態から開始することを前提とする．

Case1:HWID5-HWID6 間の PCIe リンク切断

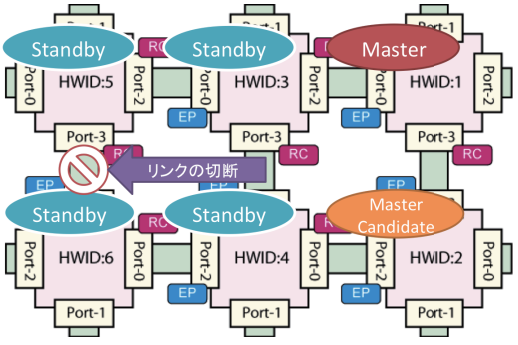


図 7 Case1:スタンバイノード間の PCIe リンク切断

本シナリオでは PCIe ケーブルを物理的に抜くことで HWID5-HWID6 の PCIe リンクを切断した (**図 7**)．リンクが切断された結果，HWID5,HWID6 それぞれのノード

がリンクダウンをデバイスドライバからの割り込みによって検知し，マスタに対してトポロジが変更されたことを通知するメッセージが送信され，マスタノードにおいて経路が正常に再構築された．また，HWID5-HWID6 以外のスタンバイノード間を接続するリンクに関しても，同様の結果が得られた．

Case2:マスタノードに電源断を伴う障害が発生

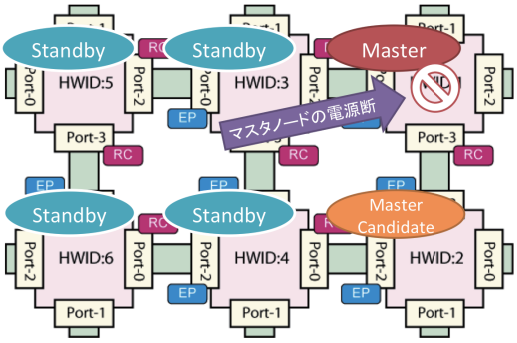


図 8 Case2:マスタノードにおけるハード障害

本シナリオでは，マスタノードへの電源供給を物理的に断つことで電源断を発生させた (**図 8**)．その結果，マスタノードに隣接する HWID2,HWID3 のノードにおいてリンクダウンの割り込みを検出した．それに引き続いて，マスタ候補ノードである HWID2 はマスタノードに昇格した．

Case3:マスタノードにソフト故障が発生

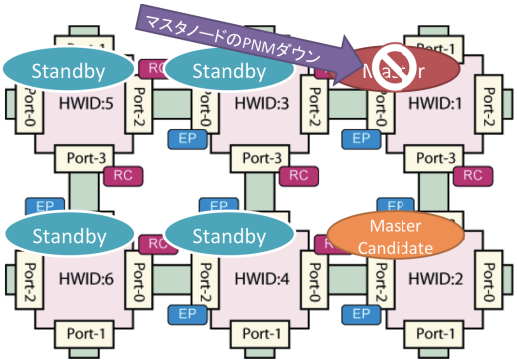


図 9 Case3:マスタノードにおけるソフト障害

本シナリオでは，マスタノード上で動作する PNM を kill することで，ソフト故障を発生させた (**図 9**)．マスタノードでソフト故障が発生した結果，スタンバイノードから送信される Polling に対する応答が途絶え，スタンバイノードがマスタノードの動作不良を検出した．

5.5 考察

以上の検証結果から，本研究で提案したソフトウェアによる Polling 及びハードウェア的なリンクダウン通知を併用することで，PEARL におけるソフト/ハード障害を検出可能であることを示した．同時に，マスタノードの自律昇格を行うことによって，ブロードキャストを全ノードが行う手法と比較してデータ転送量の削減が行えることを示した．この結果より，障害の検出に Polling, 割り込みを併用し，マスタの再選出に自律昇格，ブロードキャストを併用することで，PEARL ネットワークにおいて低コストでかつ検出漏れのない耐故障機構が実現できたといえる．

6. まとめと今後の課題

本研究では PCI Express ネットワーク PEARL における耐故障機構を検討し，一部実装及び検証を行った．検証の結果，マスタノード選出手法に関して，自律昇格を行うことで一定のパケット転送量を削減できることを示した．また，Polling 及び割り込みを用いたノードの障害検出手法を併用することによって，ソフト/ハード障害に対応可能な障害検出機構を実現することができた．今後の課題としては，現時点では実装が完了していない経路の再配布などの障害回復プロセスの実装や，故障回復に要する処理時間の評価などより定量的な評価を行うことが上げられる．特に，PEARL ネットワークの規模に応じた Polling の適切な送出間隔などについて，検証を行なって行きたい．

また，PEARL ネットワークは高い耐故障性とともに高い省電力性を持つネットワークシステムを実現することを目的としている．今後，PEARL において高い省電力性を実現するために，PCIe リンクのモードや速度の通信状況に適応した動的な変更や，ホストノード上で動作するアプリケーションと連携したトポロジの構築などの研究を行なっていく．これによって，状況に応じた電力，性能最適化を実現することの出来るネットワークシステムを実現していきたいと考えている．

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] ISO: ISO 11898:1993. http://www.iso.org/.
[2] FlexRay Consortium: FlexRay Specifications Version 3.0.1. http://www.flexray.com/.
[3] Infiniband Trade Association: . http://www.infinibandta.org/.
[4] ETHERNET WORKING GROUP: IEEE 802.3. http://www.ieee802.org/3/.
[5] PCISIG: PCIe Base Spec 2.1. http://www.pcisig.com/specifications/pciexpress/.
[6] Hanawa, T., Boku, T., Miura, S., Sato, M. and Arimoto, K.: PEARL and PEACH: A Novel PCI Express Direct Link and Its Implementation, *Parallel and Distributed Processing Workshops and PhD Forum, 2011 IEEE International Symposium on*, Vol. 0, pp. 871–879 (online), DOI:

http://doi.ieeecomputersociety.org/10.1109/IPDPS.2011.232 (2011).

[7] Otani, S., Kondo, H., Nonomura, I., Hanawa, T., Miura, S. and Boku, T.: Peach: A Multicore Communication System on Chip with PCI Express, *IEEE Micro*, Vol. 31, pp. 39–50 (online), DOI: http://doi.ieeecomputersociety.org/10.1109/MM.2011.93 (2011).

[8] 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.

[9] IETF: RFC2740 - OSPF for IPv6. `http://tools.ietf.org/html/rfc2740`.

[10] IETF: RFC2453 - RIP Version 2. `http://tools.ietf.org/html/rfc2453`.