(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2010-3257
(P2010-3257A)
(43)公開日　平成22年1月7日(2010.1.7)

(51)Int.Cl.			F I			テーマコード（参考）
G06F	9/50	(2006.01)	G06F	9/46	465Z	
G06F	9/46	(2006.01)	G06F	9/46	350	

審査請求 有　請求項の数 12　ＯＬ　（全 31 頁）

(21)出願番号　特願2008-163686 (P2008-163686)
(22)出願日　平成20年6月23日 (2008.6.23)

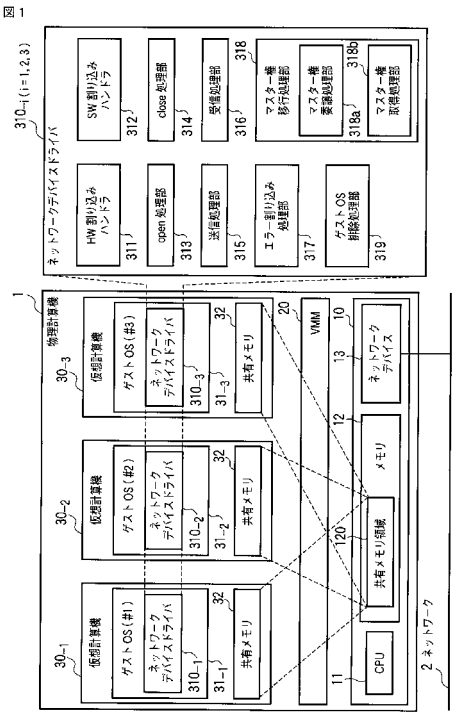
(71)出願人　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(71)出願人　301063496
東芝ソリューション株式会社
東京都港区芝浦一丁目１番１号
(74)代理人　100058479
弁理士　鈴江　武彦
(74)代理人　100108855
弁理士　蔵田　昌俊
(74)代理人　100091351
弁理士　河野　哲
(74)代理人　100088683
弁理士　中村　誠

最終頁に続く

(54)【発明の名称】仮想計算機システム及びネットワークデバイス共有方法

(57)【要約】
【課題】バッファのデータのコピーによるオーバヘッドを招くことなく、複数のゲストＯＳがネットワークデバイスを共有できるようにする。
【解決手段】ＶＭＭ２０は、物理計算機１のメモリ１２に確保される共有メモリ領域１２０を用いて共有メモリ３２を構築する。共有メモリ３２には、ネットワークデバイス１３を介してのデータ転送に用いられるバッファが配置される。ネットワークドライバ３１０-1～３１０-3は、自身が動作するゲストＯＳがマスターゲストＯＳである場合、バッファに格納された受信データを、ネットワークデバイス１３からの受信完了割り込みに応じて宛先のゲストＯＳによって受信させ、マスターゲストＯＳからの送信要求または当該マスターゲストＯＳ以外のゲストＯＳからの送信要求割り込みに応じて、バッファに格納された送信データをネットワークデバイス１３によってネットワーク２へ送信させる。
【選択図】　図１

図1



【特許請求の範囲】
【請求項１】
ＣＰＵ、メモリ及びネットワークデバイスを含む物理計算機上で動作して、複数のゲストＯＳがそれぞれ動作可能な仮想計算機実行環境を構築する仮想計算機マネージャと、
前記複数のゲストＯＳ上でそれぞれ動作する複数のネットワークデバイスドライバであって、自身が動作するゲストＯＳがマスター権を有するマスターゲストＯＳである場合、前記ネットワークデバイスからの割り込みを処理する複数のネットワークデバイスドライバと、
前記メモリに確保される共有メモリ領域を用いて前記仮想計算機マネージャによって構築され、前記複数のゲストＯＳ及び前記複数のネットワークデバイスドライバからアクセス可能な共有メモリであって、前記ネットワークデバイスによってネットワークから受信された受信データを一時格納すると共に、前記複数のゲストＯＳのいずれかから転送された送信データを一時格納するためのバッファが確保された共有メモリとを具備し、
前記複数のネットワークデバイスドライバの各々は、
自身が動作するゲストＯＳが前記マスターゲストＯＳである場合、前記受信データが前記バッファに格納されたことを通知するための前記ネットワークデバイスからの受信完了割り込みに応じて、当該受信データを当該受信データの宛先のゲストＯＳによって受信させる受信処理手段と、
自身が動作するゲストＯＳが前記マスターゲストＯＳである場合、当該マスターゲストＯＳからの送信要求または当該マスターゲストＯＳ以外のゲストＯＳからの送信要求割り込みに応じて、前記バッファに格納された前記送信データを前記ネットワークデバイスによって前記ネットワークへ送信させる送信処理手段とを含む
ことを特徴とする仮想計算機システム。
【請求項２】
前記ネットワークデバイスドライバの前記送信処理手段は、当該ネットワークデバイスドライバが動作するゲストＯＳが前記マスターゲストＯＳである場合には、当該マスターゲストＯＳからの送信要求に応じて、当該マスターゲストＯＳからの送信データを前記バッファに格納し、しかる後に当該バッファに格納された前記送信データを前記ネットワークデバイスによって前記ネットワークへ送信させ、当該ネットワークデバイスドライバが動作するゲストＯＳが前記マスターゲストＯＳとは別のゲストＯＳである場合、当該別のゲストＯＳからの送信要求に応じて、当該別のゲストＯＳからの送信データを前記バッファに格納し、しかる後に前記マスターゲストＯＳに前記送信要求割り込みを発行することを特徴とする請求項１記載の仮想計算機システム。
【請求項３】
前記複数のネットワークデバイスドライバの各々は、自身が動作するゲストＯＳが前記マスターゲストＯＳであって、且つ当該ゲストＯＳから他のゲストＯＳへのマスター権の委譲が必要な場合、前記複数のゲストＯＳの中からマスター権が委譲されるべきゲストＯＳを選択して、当該選択されたゲストＯＳにマスター権を委譲するためのマスター権委譲処理を実行するマスター権委譲手段を含む請求項１または２記載の仮想計算機システム。
【請求項４】
前記マスター権委譲手段は、前記マスターゲストＯＳが前記ネットワークデバイスの使用を終了した場合に前記マスター権委譲処理を実行することを特徴とする請求項３記載の仮想計算機システム。
【請求項５】
前記複数のネットワークデバイスドライバの各々は、前記複数のゲストＯＳのうち自身が動作しているゲストＯＳ以外のゲストＯＳの稼働状況を監視することによって異常なゲストＯＳを検出し、当該検出された異常なゲストＯＳを前記ネットワークデバイスの共有から排除するための排除手段を含み、
前記排除手段は、前記検出された異常なゲストＯＳが前記マスターゲストＯＳである場合、当該排除手段を含む前記ネットワークデバイスドライバが動作するゲストＯＳに前記

マスター権を取得させるためのマスター権取得処理を実行するマスター権取得手段を含むことを特徴とする請求項１乃至４のいずれかに記載の仮想計算機システム。

【請求項６】

前記複数のネットワークデバイスドライバの各々は、前記複数のゲストＯＳの通信回数、または前記複数のゲストＯＳがディスパッチされる時間もしくは回数、またはディスパッチされる時間もしくは回数の指標となる前記複数のゲストＯＳの優先度に基づき、マスターゲストＯＳとして最適なゲストＯＳを決定し、決定されたゲストＯＳが現在のマスターゲストＯＳと異なる場合、当該決定されたゲストＯＳに前記マスター権を移行させるためのマスター権移行処理を実行するマスター権移行手段を含む請求項１または２記載の仮想計算機システム。

【請求項７】

ＣＰＵ、メモリ及びネットワークデバイスを含む物理計算機上で動作する仮想計算機マネージャによって提供される仮想計算機実行環境で複数のゲストＯＳが動作する仮想計算機システムにおいて、前記ネットワークデバイスを前記複数のゲストＯＳで共有させるネットワークデバイス共有方法であって、

前記複数のゲストＯＳ上でそれぞれ動作する複数のネットワークデバイスドライバのうちのいずれかのネットワークデバイスドライバが、当該ネットワークデバイスドライバが動作するゲストＯＳからのオープン要求に応じて、前記メモリに確保される共有メモリ領域が仮想化された、前記複数のゲストＯＳ及び前記複数のネットワークデバイスドライバからアクセス可能な共有メモリであって、前記ネットワークデバイスによってネットワークから受信された受信データを一時格納すると共に、前記複数のゲストＯＳのいずれかから転送された送信データを一時格納するためのバッファが確保された共有メモリを前記仮想計算機マネージャによって構築させるためのステップと、

前記複数のゲストＯＳのうちのマスター権を有するマスターゲストＯＳ上で動作する前記ネットワークデバイスドライバが、前記受信データが前記バッファに格納されたことを通知するための前記ネットワークデバイスからの受信完了割り込みに応じて、当該受信データを当該受信データの宛先のゲストＯＳによって受信させるステップと、

前記マスターゲストＯＳ上で動作する前記ネットワークデバイスドライバが、当該マスターゲストＯＳからの送信要求または当該マスターゲストＯＳ以外のゲストＯＳからの送信要求割り込みに応じて、前記バッファに格納された前記送信データを前記ネットワークデバイスによって前記ネットワークへ送信させるステップと

を具備することを特徴とするネットワークデバイス共有方法。

【請求項８】

前記複数のネットワークデバイスドライバのうち、自身が動作するゲストＯＳから前記送信要求を受けたネットワークデバイスドライバが、当該ゲストＯＳからの送信データを前記バッファに格納するステップと、

前記送信要求を受けたネットワークデバイスドライバが、送信要求元が前記マスターゲストＯＳであるかを判定するステップと、

前記送信要求元が前記マスターゲストＯＳでない場合、前記送信要求を受けたネットワークデバイスドライバが、前記マスターゲストＯＳに前記送信要求割り込みを発行するステップと

を更に具備することを特徴とする請求項７記載のネットワークデバイス共有方法。

【請求項９】

前記複数のネットワークデバイスドライバのうち、前記マスターゲストＯＳ上で動作するネットワークデバイスドライバが、前記複数のゲストＯＳの中からマスター権が委譲されるべきゲストＯＳを選択して、当該選択されたゲストＯＳにマスター権を委譲するステップを更に具備することを特徴とする請求項８記載のネットワークデバイス共有方法。

【請求項１０】

前記複数のネットワークデバイスドライバの各々が、前記複数のゲストＯＳのうち自身が動作しているゲストＯＳ以外のゲストＯＳの稼働状況を監視することによって異常なゲ

ストＯＳを検出するステップと、
前記検出された異常なゲストＯＳが前記ゲストＯＳである場合、前記異常なゲストＯＳを検出したネットワークデバイスドライバが、自身が動作しているゲストＯＳに前記マスター権を取得させるステップと、
前記異常なゲストＯＳを検出したネットワークデバイスドライバが、当該異常なゲストＯＳを前記ネットワークデバイスの共有から排除するステップと
を更に具備することを特徴とする請求項７乃至９のいずれかに記載のネットワークデバイス共有方法。
【請求項１１】
前記複数のネットワークデバイスドライバの各々が、前記複数のゲストＯＳの通信回数、または前記複数のゲストＯＳがディスパッチされる時間もしくは回数、またはディスパッチされる時間もしくは回数の指標となる前記複数のゲストＯＳの優先度に基づき、マスターゲストＯＳとして最適なゲストＯＳを決定し、決定されたゲストＯＳが現在のマスターゲストＯＳと異なる場合、当該決定されたゲストＯＳに前記マスター権を移行させるステップを更に具備することを特徴とする請求項７または８記載のネットワークデバイス共有方法。
【請求項１２】
前記ネットワークデバイスドライバが前記共有メモリを構築させるためのステップは、
当該ネットワークデバイスドライバが動作するゲストＯＳからのオープン要求に応じて、前記共有メモリが既に構築されているかを当該ネットワークデバイスドライバが判定するステップと、
前記共有メモリが構築されていない場合、前記ネットワークデバイスドライバが前記共有メモリを前記仮想計算機マネージャによって構築させるステップと、
前記共有メモリを構築させた前記ネットワークデバイスドライバが、当該ネットワークデバイスドライバが動作するゲストＯＳが前記マスターゲストＯＳであることを示すマスターゲストＯＳ情報を前記共有メモリに登録するステップと
を含むことを特徴とする請求項７記載のネットワークデバイス共有方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ＣＰＵ及びメモリを含む物理計算機上で動作する仮想計算機マネージャによって提供される仮想計算機実行環境で複数のゲストＯＳが動作する仮想計算機システムに係り、特に、単一のネットワークデバイスを当該複数のゲストＯＳから共有するのに好適な、仮想計算機システム及びネットワークデバイス共有方法に関する
【背景技術】
【０００２】
近年、パーソナルコンピュータのような計算機システムにおいて、仮想計算機（Virtual Machine：ＶＭ）の技術を応用した研究や開発が行われている。また、仮想計算機システムを実現するための商用のアプリケーションプログラムが実際に広く利用されている。また、メインフレームにおいて古くからハードウェア（ＨＷ）で構成された仮想計算機支援機構を用いた仮想計算機システムも存在する。
【０００３】
一般に、パーソナルコンピュータのような計算機（物理計算機、実計算機）は、ＣＰＵ（実ＣＰＵ）及びメモリ（実メモリ）を含むＨＷ（実ＨＷ）で構成されている。この実ＨＷで構成される物理計算機（物理環境）上でハイパバイザＯＳ（オペレーティングシステム）である仮想計算機マネージャ（Virtual Machine Manager：ＶＭＭ）が動作する。そして、以下に述べるように、ＶＭＭの提供する仮想計算機環境のもとで複数のゲストＯＳが動作をする（特許文献１参照）。
【０００４】
ＶＭＭは、実ＣＰＵ、実メモリ及び物理計算機に付属した各種の実入出力デバイス（デ

ィスクドライブ等）のような物理資源の管理を行い、それらを仮想化して仮想計算機（ＶＭ）を構築する。つまりＶＭＭは、物理計算機が有する実ＣＰＵ、実メモリ及び実入出力デバイスを時間的、空間（領域）的に、複数の仮想計算機のそれぞれ仮想ＣＰＵ、仮想メモリ及び仮想入出力デバイスとして割り当てることによって、当該複数の仮想計算機が同時に動作可能な環境（仮想計算機環境）を構築する。

【０００５】

複数の仮想計算機（が動作する仮想計算機環境）にはそれぞれゲストＯＳがロードされる。各仮想計算機にロードされたゲストＯＳは、当該仮想計算機内の仮想ＣＰＵによって実行される。つまり、各仮想計算機（が動作する仮想計算機環境）では、ゲストＯＳが動作する。このようにＶＭＭは、実ＣＰＵ、実メモリ及び実入出力装置のような物理資源を各ゲストＯＳに割り当てることによって、当該各ゲストＯＳが同時に動作できる環境を提供し、仮想計算機システムの管理を行う。

【０００６】

さて、上記実入出力装置の１つとしてネットワークデバイス（実ネットワークデバイス）が知られている。また、ゲストＯＳがネットワークデバイスを使用する仕組みとして、以下に挙げる仕組みが従来から知られている。

【０００７】

第１は、ＶＭＭが仮想ネットワークデバイスを構築して仮想計算機環境に提供することにより、実ネットワークデバイスの管理をＶＭＭが完全に行う仕組み（第１の仕組み）である

第２は、１つのゲストＯＳがマスターとなって実ネットワークデバイスを管理し、他のゲストＯＳと外部との通信が、マスターとなるゲストＯＳを経由して行われる仕組み（第２の仕組み）である。

【０００８】

第３は、ゲストＯＳが自身のデバイスドライバによって直接に実ネットワークデバイスを管理する仕組み（第３の仕組み）である。

【特許文献１】特開２００６－０３９７６３号公報

【発明の開示】

【発明が解決しようとする課題】

【０００９】

しかしながら、上記第１乃至第３の仕組みは、それぞれ次のような問題を有している。

【００１０】

第１の仕組みの問題は、仮想ネットワークデバイスを提供するためにＶＭＭ内にドライバを実装しなければならない点である。第１の仕組みの問題はまた、ＶＭＭが複雑になり実装のコストも増大する点にもある。また第１の仕組みの問題は更に、状況によっては、ネットワークデバイスの故障に引きずられてＶＭＭの動作環境が機能しなくなり、加えて、本来は影響を受けないはずの仮想計算機環境までが機能しなくなり、結果としてＶＭＭ自体の信頼性が低下する点にもある。

【００１１】

第２の仕組みの問題は、ネットワークデバイスを介してのデータ転送に用いられるバッファ（ＤＭＡバッファ領域）のデータを、マスターゲストＯＳ及びその他のゲストＯＳの仮想メモリ空間相互間でコピーする必要があるため、オーバヘッドを招いて十分なスループットが得られない点にある。また、第２の仕組みの問題は更に、マスターゲストＯＳが動作不能に陥れば、他のゲストＯＳが通信することができなくなる点にもある。

【００１２】

第３の仕組みの問題は、デバイスドライバが、他のゲストＯＳが同じネットワークデバイスを使用することを考慮して作られていないことに起因する。つまり第３の仕組みの問題は、複数のゲストＯＳ（で動作する第３の仕組み）のそれぞれでデバイスドライバが動作するために、複数のゲストＯＳ間の排他の仕組みがなく複数のゲストＯＳでネットワークデバイスを共有するのが困難である点にある。そのため、一般には１つのネットワーク

デバイスは１つのゲストＯＳしか使用することができない。また、ＶＭＭは、どのゲストＯＳにネットワークデバイスを管理させるか（見せるか）を決定する機能を持たなければならない。

【００１３】

本発明は上記事情を考慮してなされたものでその目的は、ネットワークデバイスを介してのデータ転送に用いられるバッファのデータのコピーによるオーバヘッドを招くことなく、複数のゲストＯＳがネットワークデバイスを共有できる仮想計算機システム及びネットワークデバイス共有方法を提供することにある。

【課題を解決するための手段】

【００１４】

本発明の１つの観点によれば、ＣＰＵ、メモリ及びネットワークデバイスを含む物理計算機上で動作する仮想計算機マネージャによって提供される仮想計算機実行環境で複数のゲストＯＳが動作する仮想計算機システムが提供される。この仮想計算機システムは、前記複数のゲストＯＳ上でそれぞれ動作する複数のネットワークデバイスドライバであって、自身が動作するゲストＯＳがマスター権を有するマスターゲストＯＳである場合、前記ネットワークデバイスからの割り込みを処理する複数のネットワークデバイスドライバと、前記メモリに確保される共有メモリ領域を用いて前記仮想計算機マネージャによって構築され、前記複数のゲストＯＳ及び前記複数のネットワークデバイスドライバからアクセス可能な共有メモリであって、前記ネットワークデバイスによってネットワークから受信された受信データを一時格納すると共に、前記複数のゲストＯＳのいずれかから転送された送信データを一時格納するためのバッファが確保された共有メモリとを具備し、前記複数のネットワークデバイスドライバの各々は、自身が動作するゲストＯＳが前記マスターゲストＯＳである場合、前記受信データが前記バッファに格納されたことを通知するための前記ネットワークデバイスからの受信完了割り込みに応じて、当該受信データを当該受信データの宛先のゲストＯＳによって受信させる受信処理手段と、自身が動作するゲストＯＳが前記マスターゲストＯＳである場合、当該マスターゲストＯＳからの送信要求または当該マスターゲストＯＳ以外のゲストＯＳからの送信要求割り込みに応じて、前記バッファに格納された前記送信データを前記ネットワークデバイスによって前記ネットワークへ送信させる送信処理手段とを含むことを特徴とする。

【発明の効果】

【００１５】

本発明によれば、マスターゲストＯＳとその他のゲストＯＳとが協調することによってネットワークデバイスを共有する仕組みを適用しながら、メモリに確保される共有メモリ領域を用いて構築される、各ゲストＯＳ及び各ネットワークデバイスドライバからアクセス可能な共有メモリに、ネットワークデバイスを介して送受信されるデータを一時格納するためのバッファを配置することにより、当該バッファのデータのコピーによるオーバヘッドを招くことなく、各ゲストＯＳがネットワークデバイスを共有できる。

【発明を実施するための最良の形態】

【００１６】

以下、本発明の実施の形態につき図面を参照して説明する。

図１は本発明の一実施形態に係る仮想計算機システムの構成を示すブロック図である。図１に示す仮想計算機システムは、物理計算機（実計算機）１を用いて実現される。物理計算機１は、仮想計算機環境を提供するのに用いられるハードウェア（ＨＷ）１０を備えている。ＨＷ１０は、ＨＷ資源（物理資源）であるＣＰＵ（実ＣＰＵ）１１、メモリ（実メモリ）１２及びネットワークデバイス（実ネットワークデバイス）１３を含む。なお、図１では、ディスクドライブのような、物理計算機１に付属する、ネットワークデバイス１３以外の入出力デバイスは省略されている。

【００１７】

ネットワークデバイス１３は、物理計算機１の外部のネットワーク２に接続されている。ネットワークデバイス１３は、ネットワーク２を介して物理計算機１と外部の物理デバ

イスとの間の通信を行う。

【００１８】

物理計算機１（のＨＷ１０）上では、仮想計算機マネージャ（ＶＭＭ）２０が動作する。ＶＭＭ２０は、仮想計算機システムの管理を行い、仮想計算機が動作する環境（仮想計算機環境）を構築する。更に具体的に述べるならば、ＶＭＭ２０は、物理計算機１のＨＷ１０を構成するＣＰＵ１１、メモリ１２及び、ネットワークデバイス１３を含む各種の入出力デバイスのような物理資源（ハードウェアリソース）を制御すると共に当該物理資源の時間的、領域的な配分を管理することにより、複数の仮想計算機、例えば3台の仮想計算機３０-1，３０-2，３０-3が動作する仮想計算機環境（仮想計算機実行環境）を構築する。

【００１９】

仮想計算機３０-1，３０-2，３０-3上では、それぞれゲストＯＳ３１-1（＃１），３１-2（＃２），３１-3（＃３）が動作する。更に具体的に述べるならば、ゲストＯＳ３１-1～３１-3は、それぞれ仮想計算機３０-1～３０-3にロードされて、当該仮想計算機３０-1～３０-3内の仮想ＣＰＵによって実行される。ゲストＯＳ３１-1，３１-2，３１-3は固有の識別子（ゲストＯＳ識別子）を有する。本実施形態においてゲストＯＳ３１-1，３１-2，３１-3のゲストＯＳ識別子は、それぞれ、１，２，３であるものとする。

【００２０】

ＶＭＭ２０はまた、ゲストＯＳ３１-1～３１-3（の後述するドライバ３１０-1～３１０-3）からアクセス（参照／書き込み）可能な共有メモリ３２を構築する。共有メモリ３２は、ＨＷ１０に含まれているメモリ１２の記憶領域の一部（共有メモリ領域）３２を用いて構築される仮想化されたメモリである。ＶＭＭ２０は、ゲストＯＳ３１-1～３１-3の間の通知のために割り込みを発生させるインタフェース（ゲストＯＳ間割り込み配信インタフェース）を当該ゲストＯＳ３１-1～３１-3に提供する。

【００２１】

ゲストＯＳ３１-1，３１-2，３１-3には、それぞれネットワークデバイスドライバ（以下、ドライバと略称する）３１０-1，３１０-2，３１０-3が付加されている。ドライバ３１０-1～３１０-3は、それぞれ、ネットワークデバイス１３をゲストＯＳ３１-1～３１-3が使用可能とするためのデバイスドライバであり、当該ゲストＯＳ３１-1～３１-3の一部として当該ゲストＯＳ３１-1～３１-3上で動作する。つまりゲストＯＳ３１-1～３１-3は、それぞれドライバ３１０-1～３１０-3を含む。

【００２２】

ゲストＯＳ３１-1～３１-3は、それぞれがマスター（マスターゲストＯＳ）となる機能を持つ。ドライバ３１０-1～３１０-3は、それぞれゲストＯＳ３１-1～３１-3がマスターとして動作する場合、自身もマスターとして動作する。本実施形態において、ドライバ３１０-1，３１０-2，３１０-3は同等の機能を持つ。但し、ドライバ３１０-1～３１０-3の間で本発明に直接関係しない一部の機能が異なっていても構わない。例えば、ドライバ３１０-1～３１０-3のうちのいずれかで、マスターとなる機能が制限されていても構わない。

【００２３】

ドライバ３１０-i（ｉ＝１，２，３）は、ＨＷ（ハードウェア）割り込みハンドラ３１１及びＳＷ（ソフトウェア）割り込みハンドラ３１２の２種類の割り込みハンドラを有する。ＨＷ割り込みハンドラ３１１は、ネットワークデバイス１３によって発生される割り込み（ＨＷ割り込み）を処理する。ＳＷ割り込みハンドラ３１２は、ゲストＯＳ間割り込み配信インタフェースによって発生される割り込み（ゲストＯＳ３１-1～３１-3の間の通知のための割り込み）を処理する。

【００２４】

ドライバ３１０-iは更に、ｏｐｅｎ（オープン）処理部３１３、ｃｌｏｓｅ（クローズ）処理部３１４、送信処理部３１５、受信処理部３１６、エラー割り込み処理部３１７、マスター権移行処理部３１８、ゲストＯＳ排除処理部３１９を有する。マスター権移行処

理部３１８は、マスター権委譲処理部３１８ａ及びマスター権取得処理部３１８ｂを含む。これらの各処理部の機能については後述する。

【００２５】

図２は、図１に示される共有メモリ３２における領域の割り当て例を示す。共有メモリ３２は、物理レジスタマップ領域３２１、特殊処理ステータス領域３２２、ゲストＯＳ情報領域３２３及びＤＭＡバッファ領域３２４を含む。共有メモリ３２はまた、送信ディスクリプタチェーン領域３２５、受信ディスクリプタチェーン領域３２６及び他共有情報領域３２７を含む。

【００２６】

物理レジスタマップ領域３２１は、ネットワークデバイス１３に所属する物理レジスタの群をマップした領域である。物理レジスタマップ領域３２１によってマップされる物理レジスタの群は、初期化処理で設定が必要なレジスタ、及び割り込み要因レジスタを含む。物理レジスタマップ領域３２１にアクセスすることによって物理レジスタがアクセスされる。

【００２７】

特殊処理ステータス領域３２２は、ドライバ３１０-iが行うべき特殊な処理（特殊処理）が発生しているかを示すステータスを格納するのに用いられる。本実施形態において、特殊処理ステータス領域３２２を用いてステータスが管理される特殊処理は、共有メモリ３２を初期化する処理、ネットワークの物理的切断（ネットワークケーブル抜けなどへの対応）処理、ネットワークの物理的接続処理を含む。これらの特殊処理は、共有メモリ３２内の情報を使用して各ドライバ３１０-iで進めることができる処理であり、各ドライバ３１０-iで重複して実行されても問題の生じない処理である。本実施形態において、特殊処理ステータス領域３２２は、予め定められた特殊処理毎に、その処理が発生しているかをビットの状態で示すビットマップを格納する。ドライバ３１０-iは、ある特殊処理が発生しているかを確認するには、特殊処理ステータス領域３２２（に格納されているビットマップ）内の当該特殊処理に対応するビットを参照すればよい。

【００２８】

ゲストＯＳ情報領域３２３は、ネットワークデバイス１３を使用している全てのゲストＯＳの情報と、マスターゲストＯＳ情報とを格納するのに用いられる。ゲストＯＳの情報（ゲストＯＳ情報）は、ゲストＯＳ識別子、ＳＷ割り込み番号及びＳＷ割り込み要因情報を含む。

【００２９】

ゲストＯＳ識別子は、ネットワークデバイス１３を現在使用しているゲストＯＳの識別子を示す。

ＳＷ割り込み番号は、ゲストＯＳ識別子で示されるゲストＯＳに対して他のゲストＯＳから、ゲストＯＳ間の通知機能（ゲストＯＳ間割り込み配信インタフェース）によって割り込みを発生させる場合の割り込み配信先のゲストＯＳのＳＷ割り込みハンドラ３１２を示す。

【００３０】

ＳＷ割り込み要因情報は、上記割り込みを発生させる場合の通知内容を表す情報。ＳＷ割り込み要因情報は例えばビットマップから構成される。配信先のゲストＯＳは、配信元のゲストＯＳ（ゲストＯＳ識別子で示されるゲストＯＳ）のゲストＯＳ間割り込み配信インタフェースから送信されたビットマップ形式のＳＷ割り込み要因情報（の各ビットの状態）を参照することにより通知内容を判断する。

【００３１】

ゲストＯＳ情報は更に、付加的な情報として、タイムスタンプ、通信回数統計情報、ゲストＯＳ優先度及びネットワークアドレスを含む。

タイムスタンプは、ゲストＯＳ識別子で示されるゲストＯＳの最終動作時刻を示す。

【００３２】

通信回数統計情報は、直近の通信回数を示す統計情報である。本実施形態において、通

信回数統計情報には、１ディスクリプタチェーンサイクルの期間にゲストＯＳの使用したディスクリプタの数が用いられる。ディスクリプタチェーンサイクルについては後述する。

【００３３】

ゲストＯＳ優先度は、ゲストＯＳ識別子で示されるゲストＯＳの実行優先度を示す。つまりゲストＯＳ優先度は、ＶＭＭ２０が、ディスパッチされるべきゲストＯＳを選択する際に選定の基準（指標）とする値である。本実施形態では、ゲストＯＳ優先度の値が大きいゲストＯＳほど、優先的にＣＰＵ１１（ＣＰＵ時間）が割り当てられる。

【００３４】

ネットワークアドレスは、ネットワーク２を介しての通信のためにゲストＯＳ識別子で示されるゲストＯＳに割り当てられる、ＩＰ（Internet Protocol）アドレスのようなネットワークアドレスである。

【００３５】

マスターゲストＯＳ情報は、マスタとして動作している（つまりマスター権を有している）ゲストＯＳ（マスターゲストＯＳ）を示す。ここでは、マスターゲストＯＳ情報として、マスターゲストＯＳのゲストＯＳ識別子が用いられる。

【００３６】

なお、図２に示されるゲストＯＳ情報領域３２３には、便宜的に、ゲストＯＳ識別子が「１」のゲストＯＳ（つまりゲストＯＳ３１-1）のゲストＯＳ情報のみが格納されている状態が示されている。しかし、ゲストＯＳ情報領域３２３には、ネットワークデバイス１３を共有する他のゲストＯＳ３１-2，３１-3のゲストＯＳ情報も格納されているものとする。

【００３７】

ＤＭＡバッファ領域３２４は、ネットワークデバイス１３が送受信データのＤＭＡ（Direct Memory Access）転送に使用するデータバッファ（以下、ＤＭＡバッファと称する）３２４ａのための領域である。

【００３８】

送信ディスクリプタチェーン領域３２５は、送信ディスクリプタチェーン３２５ａを格納するのに用いられる。送信ディスクリプタチェーン３２５ａは、ＤＭＡバッファ３２４ａ内の領域（へのリンク）及びＨＷ仕様に基づいた送信ディスクリプタを管理するためのディスクリプタ管理構造を有する。送信ディスクリプタチェーン３２５ａに含まれる各送信ディスクリプタは、チェーンでリンクされている順にサイクリックに利用される。送信ディスクリプタチェーン３２５ａを一巡するサイクルをディスクリプタチェーンサイクルと呼ぶ。

【００３９】

図３は、送信ディスクリプタチェーン３２５ａの一例を示す。図３の例では、送信ディスクリプタチェーン３２５ａは、送信ディスクリプタ３２５-1～３２５-10を含む。本実施形態において送信ディスクリプタ３２５-1～３２５-10はリング状のチェーンによりリンクされている。送信ディスクリプタ３２５-j（ｊ＝１～１０）は、ステータス部３２５１、ポインタ部３２５２及び利用ゲストＯＳ情報部（図示せず）を含む。

【００４０】

ポインタ部３２５２は、送信されるべきデータが設定（格納）されているＤＭＡバッファ３２４ａ内の領域を指し示すポインタを格納する。利用ゲストＯＳ情報部は、当該利用ゲストＯＳ情報部を含む送信ディスクリプタ３２５-jが、いずれのゲストＯＳによって参照（使用）中であるかを示す情報（例えばゲストＯＳ識別子）を格納する。

【００４１】

ステータス部３２５１は、送信ディスクリプタ３２５-jのステータスを示す。送信ディスクリプタ３２５-jの取り得るステータスは、例えば、「ｅｍｐｔｙ」、「ｎｏｔ　ｒｅａｄｙ」、「ｒｅａｄｙ」及び「ｄｍａ」の４種類である。

【００４２】

「ｅｍｐｔｙ」は、送信ディスクリプタ３２５-jが未使用の状態にあることを示す。
「ｎｏｔ　ｒｅａｄｙ」は、送信ディスクリプタ３２５-jがデータ送信のために使用予定である（使用が予約されている）が、まだ送信されるべきデータの設定が終わっていない状態にあることを示す。

【００４３】
「ｒｅａｄｙ」は、送信ディスクリプタ３２５-jがデータ送信のために使用予定であり、且つ送信されるべきデータの設定も終了している状態にあることを示す。

【００４４】
「ｄｍａ」は送信ディスクリプタ３２５-jのポインタ部３２５２で指定されるデータの送信のためのＤＭＡ転送が実行中であることを示す。

【００４５】
送信ディスクリプタチェーン３２５ａは、送信中先頭ポインタ、送信前先頭ポインタ及び空き先頭ポインタの３種類のポインタで管理される。これらのポインタは、任意の時点において、

送信中先頭ポインタ≦送信前先頭ポインタ≦空き先頭ポインタ

のような不等号で示される順番の位置関係（大きいほうが進行方向）にある送信ディスクリプタを指し示す。

【００４６】
送信中先頭ポインタの指し示す位置から送信前先頭ポインタの指し示す位置の直前までには、ステータスが「ｄｍａ」の送信ディスクリプタのみが存在する。図３の例では、送信ディスクリプタ３２５-3，３２５-4が、これに該当する。

【００４７】
送信前先頭ポインタの指し示す位置から空き先頭ポインタの指し示す位置の直前までには、ステータスが「ｒｅａｄｙ」の送信ディスクリプタとステータスが「ｎｏｔ　ｒｅａｄｙ」の送信ディスクリプタとが混在し得る。図３の例では、送信ディスクリプタ３２５-5～３２５-9が、これに該当する。

【００４８】
空き先頭ポインタから先には、ステータスが「ｅｍｐｔｙ」の送信ディスクリプタのみが存在する。図３の例では、送信ディスクリプタ３２５-10，３２５-1，３２５-2が、これに該当する。
これら３種類のポインタは、ネットワークデバイス１３によって操作される。

【００４９】
再び図２を参照すると、受信ディスクリプタチェーン領域３２６は、受信ディスクリプタチェーンを格納するのに用いられる。受信ディスクリプタチェーンは、ＤＭＡバッファ３２４ａ内の領域（へのリンク）及びＨＷ仕様に基づいた受信ディスクリプタを管理するためのディスクリプタ管理構造を有する。受信ディスクリプタチェーンに含まれる各受信ディスクリプタは、送信ディスクリプタチェーン３２５ａに含まれる上記各送信ディスクリプタと同様に、チェーンでリンクされている順にサイクリックに利用される。この受信ディスクリプタチェーンを一巡するサイクルもディスクリプタチェーンサイクルと呼ばれる。

【００５０】
図４は、受信ディスクリプタチェーンの一例を示す。図４の例では、受信ディスクリプタチェーンは、受信ディスクリプタ３２６-1～３２６-10を含む。本実施形態において受信ディスクリプタ３２６-1～３２６-10はリング状のチェーンによりリンクされている。受信ディスクリプタ３２６-j（ｊ＝１～１０）は、送信ディスクリプタ３２５-jと同様に、ステータス部３２６１、ポインタ部３２６２及び利用ゲストＯＳ情報部（図示せず）を含む。

【００５１】
ポインタ部３２６２は、受信されるべきデータが設定されているＤＭＡバッファ３２４ａ内の領域を指し示すポインタを格納する。利用ゲストＯＳ情報部は、当該利用ゲストＯ

Ｓ情報部を含む受信ディスクリプタ３２６-jが、いずれのゲストＯＳによって参照（使用）中であるかを示す情報（ゲストＯＳ識別子）を格納する。

【００５２】

ステータス部３２６１は、受信ディスクリプタ３２６-jのステータスを示す。受信ディスクリプタ３２６-jの取り得るステータスは、例えば、「ｅｍｐｔｙ」、「ｒｅｃｅｉｖｅｄ」及び「ｄｍａ」の３種類である。

【００５３】

「ｅｍｐｔｙ」は、受信ディスクリプタ３２６-jが未使用の状態にあることを示す。

「ｒｅｃｅｉｖｅｄ」は、受信ディスクリプタ３２６-jのポインタ部３２６２で指定されるＤＭＡバッファ３２４ａ内の領域にデータを受信済みであるが、その旨をゲストＯＳが認識していない状態にあることを示す。

【００５４】

「ｄｍａ」は、受信ディスクリプタ３２６-jのポインタ部３２６２で指定されるＤＭＡバッファ３２４ａ内の領域に受信データを一時格納するためのＤＭＡ転送が実行中であることを示す。

【００５５】

受信ディスクリプタチェーンは、受信中先頭ポインタ、ＤＭＡ中先頭ポインタ及び空き先頭ポインタの３種類のポインタで管理される。これらのポインタは、任意の時点において、

受信中先頭ポインタ≦ＤＭＡ中先頭ポインタ≦空き先頭ポインタ

のような不等号で示される順番の位置関係（大きいほうが進行方向）にある受信ディスクリプタを指し示す。受信中先頭ポインタの指し示す位置からＤＭＡ中先頭ポインタの指し示す位置の直前までには、ステータスが「ｒｅｃｅｉｖｅｄ」の受信ディスクリプタと「ｅｍｐｔｙ」の受信ディスクリプタとが混在する。図４の例では、受信ディスクリプタ３２６-3，３２６-4が、これに該当する。

【００５６】

ＤＭＡ中先頭ポインタの指し示す位置から空き先頭ポインタの指し示す位置の直前までには、ステータスが「ｄｍａ」の受信ディスクリプタのみが存在する。図４の例では、受信ディスクリプタ３２６-5，３２６-6が、これに該当する。

【００５７】

空き先頭ポインタから先には、ステータスが「ｅｍｐｔｙ」の受信ディスクリプタのみが存在する。図４の例では、受信ディスクリプタ３２６-7～３２６-10，３２６-1，３２６-2が、これに該当する。

これら３種類のポインタは、ネットワークデバイス１３によって操作される。

【００５８】

他共有情報領域３２７は、各ゲストＯＳ３１-i（ｉ＝１，２，３）のドライバ３１０-i間で共有すべき情報、例えば特殊処理で使用する必要がある情報を格納するのに用いられる。

【００５９】

次に、上述の構成の仮想計算機システムの特徴について説明する。

【００６０】

まず、図１の仮想計算機システムでは、ネットワークデバイス１３を共有するゲストＯＳ３１-1～３１-3上でそれぞれドライバ３１０-1～３１０-3が動作する。ドライバ３１０-1～３１０-3の１つはマスターとして動作して、ＤＭＡ処理や、ＤＭＡ処理のためのネットワークデバイス１３からの割り込みに対する処理など、当該ネットワークデバイス１３の管理の主要な部分を取り扱う。

【００６１】

図１の仮想計算機システムの特徴は、ゲストＯＳ３１-1～３１-3上でそれぞれ動作するドライバ３１０-1～３１０-3に共通して必要となる情報、及びゲストＯＳ３１-1～３１-3に固有のゲストＯＳ情報のうち必要な部分を相互参照できるようにした点にある。この相

互参照を可能とするために、ＶＭＭ２０によってゲストＯＳ３１-1～３１-3に提供される共有メモリ３２が使用される。ゲストＯＳ３１-1～３１-3のそれぞれドライバ３１０-1～３１０-3は共有メモリ３２にアクセスする。

【００６２】

共有メモリ３２は、ＨＷ１０に含まれているメモリ１２に確保された共有メモリ領域１２０を用いて構築される仮想化されたメモリである。このため、共有メモリ３２へのアクセスは、物理的にはメモリ１２内の共有メモリ領域１２０へのアクセスによって実現される。

【００６３】

本実施形態ではまた、ＤＭＡバッファ３２４ａも共有メモリ３２に配置される。これにより、ＤＭＡバッファ３２４ａも各ゲストＯＳ３１-1～３１-3から直接参照（アクセス）することが可能となり、従来は大きな問題であったデータコピーに伴うオーバヘッドを後述するように削減することができる。

【００６４】

また本実施形態では、マスターとなるゲストＯＳを変更可能とする仕組み、つまりマスターゲストＯＳが機能しなくなっても他のゲストＯＳがネットワークデバイス１３を使用し続けることを可能とする仕組みが提供される。本実施形態では更に、効率的なネットワークデバイス１３の使用を支援する仕組みも提供される。

【００６５】

以下、上述の特徴を有する仮想計算機システムにおけるゲストＯＳのドライバを中心とする動作について、「ｏｐｅｎ（オープン）処理」、「送信処理」、「受信処理」及び「エラー割り込み処理」を例に順次説明する。

【００６６】

［ｏｐｅｎ処理］

まず、ゲストＯＳのドライバ（のｏｐｅｎ処理部３１３）がネットワークデバイス１３の使用を開始する際に実行する「ｏｐｅｎ処理（ドライバのｏｐｅｎ処理）」の手順について、当該ドライバがゲストＯＳ３１-1のドライバ３１０-1である場合を例に、図５のフローチャートを参照して説明する。

【００６７】

今、ゲストＯＳ３１-1から当該ゲストＯＳ３１-1のドライバ３１０-1に対して、「ｏｐｅｎ要求」が与えられたものとする。するとドライバ３１０-1は、共有メモリ３２の領域を探して、当該共有メモリ３２の領域が存在するかを判定する（ステップＳ１）。

【００６８】

もし、ステップＳ１の判定がＹｅｓであるならば、ドライバ３１０-1は、マスターとなっている別のゲストＯＳのドライバからの要求によって共有メモリ３２の領域が確保されているとしてステップＳ８に進む。

【００６９】

これに対し、ステップＳ１の判定がＮｏであるならば、ドライバ３１０-1はＶＭＭ２０に依頼して、共有メモリ３２の領域を確保する（ステップＳ２）。つまりドライバ３１０-1は、ＶＭＭ２０により、ＨＷ１０に含まれているメモリ１２の記憶領域の一部を共有メモリ領域１２０として確保させることにより、共有メモリ３２（仮想化された共有メモリメモリ）を構築させる。

【００７０】

次にドライバ３１０-1は、共有メモリ３２を初期化するための初期化処理を行う（ステップＳ３）。この初期化処理は、共有メモリ３２における領域３２１～３２７の割り当て、物理レジスタマップ領域３２１に物理レジスタの群（ネットワークデバイス１３に所属する物理レジスタの群）をマップする処理を含む。また、上記初期化処理は、特殊処理ステータス領域３２２に当該初期化処理が進行中であることを示すステータスを記録する処理を含む。ここでは、初期化処理が進行中であるかを示すビットがオンされる。以降、ドライバ３１０-1は、初期化処理の進捗状況を、例えば予め定められたタイミング毎に記録

する。
【００７１】
　またドライバ３１０-1は、ネットワークデバイス１３を初期化する（ステップＳ４）。次にドライバ３１０-1は、ネットワークデバイス１３からの割り込み（ＨＷ割り込み）の配信先を自身に設定する（ステップＳ５）。
【００７２】
　次にドライバ３１０-1は、共有メモリ３２のゲストＯＳ情報領域３２３に、自身を含む（自身が動作している）ゲストＯＳ３１-1のゲストＯＳ情報を書き込む（ステップＳ６）。即ちドライバ３１０-1は、ゲストＯＳ３１-1のゲストＯＳ識別子、ＳＷ割り込みハンドラ３１２が使用するＳＷ割り込み番号等を含む、ゲストＯＳ３１-1のゲストＯＳ情報を、ゲストＯＳ情報領域３２３に書き込む。また、ステップＳ６においてドライバ３１０-1は、マスターゲストＯＳがゲストＯＳ３１-1であることを示すマスターゲストＯＳ情報をゲストＯＳ情報領域３２３に登録する。そしてドライバ３１０-1は、共有メモリ３２の特殊処理ステータス領域３２２に初期化処理が終了したことを記録して（ステップＳ７）、「ｏｐｅｎ処理」を終了する。
【００７３】
　一方、ステップＳ１の判定がＹｅｓの場合、ドライバ３１０-1は、マスターとなっている別のゲストＯＳのドライバからの要求によって構築された共有メモリ３２の特殊処理ステータス領域３２２を参照して当該別のゲストＯＳのドライバによる初期化処理（ステップＳ２～Ｓ７）の進捗状況を確認し、当該初期化処理が継続していれば、当該初期化処理が終了するまで待つ（ステップＳ８）。そして初期化処理が終了すると、ドライバ３１０-1は、共有メモリ３２のゲストＯＳ情報領域３２３にゲストＯＳ３１-1に固有のゲストＯＳ情報（自身のゲストＯＳ情報）を書き込んで（ステップＳ９）、「ｏｐｅｎ処理」を終了する。
【００７４】
　［送信処理］
　次に、ゲストＯＳのドライバ（の送信処理部３１５）がネットワークデバイス１３を通じてパケットをネットワーク２に送信する際に実行される「送信処理」の手順について、当該ドライバがゲストＯＳ３１-1のドライバ３１０-1である場合を例に、図６のフローチャートを参照して説明する。この「送信処理」において実行されるネットワークデバイス１３へのパケットデータの転送には、ＤＭＡが用いられる。
【００７５】
　今、ゲストＯＳ３１-1から当該ゲストＯＳ３１-1のドライバ３１０-1に対して、「送信要求」が与えられたものとする。するとドライバ３１０-1は、共有メモリ３２の送信ディスクリプタチェーン領域３２５における送信ディスクリプタチェーン３２５ａから、空き先頭ポインタの指し示す送信ディスクリプタを先頭とする、送信に必要な数の送信ディスクリプタ３２５-j（図３の例では、送信ディスクリプタ３２５-jは送信ディスクリプタ３２５-10を含む）を確保する（ステップＳ１１）。
【００７６】
　次にドライバ３１０-1は、確保した送信ディスクリプタ３２５-jのステータス（ステータス部３２５１に設定されているステータス情報の示すステータス）を「ｅｍｐｔｙ」から「ｎｏｔ　ｒｅａｄｙ」に変更すると共に、確保した送信ディスクリプタ３２５-jの数だけ空き先頭ポインタを更新する（ステップＳ１２）。つまりドライバ３１０-1は、確保した送信ディスクリプタ３２５-jの数だけ空き先頭ポインタの指し示す位置を移動する。
【００７７】
　次にドライバ３１０-1は、確保した送信ディスクリプタ３２５-jを（当該ドライバ３１０-1を含む）ゲストＯＳ３１-1が使用していることを示す情報を、当該ディスクリプタ３２５-jに追加する。上記ステップＳ１１～Ｓ１３は、各ゲストＯＳのドライバ間で排他的に実行される必要がある。この制御には、共有メモリを使用する際に従来から用いられる一般的な排他制御手法が適用可能である。この排他制御については、以降の手順でも同様

の条件が必要となる箇所があるが、説明を省略する。

【００７８】

次にドライバ３１０-1は、ゲストＯＳ３１-1から要求されたパケットの送信のために、次のように送信ディスクリプタ３２５-jの準備を行う（ステップＳ１４）。まずドライバ３１０-1は、送信ディスクリプタ３２５-jのポインタ部３２５２に格納されているポインタに基づいて共有メモリ３２上のＤＭＡバッファ３２４ａ内の領域に直接アクセスすることにより、送信されるべきパケットのデータを書き込む。ドライバ３１０-1は、このデータの書き込み（つまり送信の準備）が完了した送信ディスクリプタ３２５-jのステータスを「ｎｏｔ　ｒｅａｄｙ」から「ｒｅａｄｙ」に変更する。

【００７９】

本実施形態おいて、ＤＭＡ転送はマスターゲストＯＳが管理する。そこでドライバ３１０-1は、当該ドライバ３１０-1を含む（当該ドライバ３１０-1動作している）ゲストＯＳ３１-1がマスターゲストＯＳであるかを判定する（ステップＳ１５）。

【００８０】

もし、ゲストＯＳ３１-1がマスターゲストＯＳでないならば（ステップＳ１５がＮｏ）、ドライバ３１０-1はマスターゲストＯＳ（のドライバ）にＳＷ割り込みによる通知を行って、送信のためのＤＭＡ転送を要求する（ステップＳ１６）。即ちドライバ３１０-1はマスターゲストＯＳ（のドライバ）に、送信のためのＤＭＡ要求の割り込み（送信ＤＭＡ要求割り込み）を行う。マスターゲストＯＳ（のドライバ）への割り込み配信先は、共有メモリ３２のゲストＯＳ情報領域３２３を参照することによって知ることができる。ドライバ３１０-1は、ステップＳ１６を実行すると、送信処理を終了する。

【００８１】

上述のように、本実施形態では、ゲストＯＳ間の通知にＳＷ割り込みが使用される。しかし、ＶＭＭ２０が提供するゲストＯＳ間の通知手段であれば、ＳＷ割り込み以外の手段を用いても構わない。本実施形態では、複数の用途でゲストＯＳ間の通知にＳＷ割り込みが使用される。そこで本実施形態では、割り込み配信先が用途別に設定され、また、割り込み要因を示す仮想の割り込み要因レジスタが共有メモリ３２内に配置される構成とすることによって、割り込み要因の種別を対象のゲストＯＳが認識できるようになっているものとする。

【００８２】

一方、ステップＳ１５で、ドライバ３１０-1を含むゲストＯＳ３１-1がマスターゲストＯＳであると判定された場合、つまり送信要求元のゲストＯＳ３１-1がマスターゲストＯＳである場合（ステップＳ１５がＹｅｓ）、ドライバ３１０-1はステップＳ１７に進む。また、ゲストＯＳ３１-1がマスターゲストＯＳであり、且つ当該ゲストＯＳ３１-1に、他のゲストＯＳからの送信要求に従って当該他のゲストＯＳのドライバから送信ＤＭＡ要求割り込み（送信要求割り込み）が入った場合にも、ドライバ３１０-1はステップＳ１７に進む。

【００８３】

ステップＳ１７においてドライバ３１０-1は、ネットワークデバイス１３に対するＤＭＡ処理を開始する。具体的には、ドライバ３１０-1は、送信前先頭ポインタで指し示される送信ディスクリプタチェーン３２５ａ内の位置からステータスが「ｒｅａｄｙ」である連続する送信ディスクリプタの分だけ、ネットワークデバイス１３に対してＤＭＡ要求を発行する。ここで、複数のゲストＯＳが送信ディスクリプタを確保して送信のための準備中である可能性もあるため、送信前先頭ポインタの指し示す位置から空き先頭ポインタの指し示す位置の直前までには、ステータスが「ｎｏｔ　ｒｅａｄｙ」の送信ディスクリプタとステータスが「ｒｅａｄｙ」の送信ディスクリプタとが混在している可能性がある。ステップＳ１７においてドライバ３１０-1は、ＤＭＡ処理が開始された送信ディスクリプタのステータスを「ｄｍａ」に変更し、送信前先頭ポインタを当該ＤＭＡ処理が開始された送信ディスクリプタの数の分だけ進める。

【００８４】

ドライバ３１０-1からのＤＭＡ要求に応じてネットワークデバイス１３によって実行されるＤＭＡ転送が終了すると、当該ネットワークデバイス１３からドライバ３１０-1（マスターゲストＯＳ３１-1のドライバ３１０-1）に、ＤＭＡ転送終了（ＤＭＡ終了）がＨＷ割り込みによって通知される。ドライバ３１０-1は、このＤＭＡ終了のＨＷ割り込みに応じて、送信中先頭ポインタの指し示す位置から始まる送信終了した送信ディスクリプタのステータスを「ｅｍｐｔｙ」に変更し、変更の対象となった送信ディスクリプタの数の分だけ、当該送信中先頭ポインタを更新する（ステップＳ１８）。このステップＳ１８においてドライバ３１０-1は、変更の対象となった各送信ディスクリプタの利用ゲストＯＳ情報部に格納されている、当該ディスクリプタを参照（使用）中のゲストＯＳを示す情報をクリアする。

【００８５】

ドライバ３１０-1（マスターゲストＯＳ３１-1のドライバ３１０-1）は、ステップＳ１８を実行すると、空き先頭ポインタと送信前先頭ポインタとが一致しているかを判定する（ステップＳ１９）。もし、空き先頭ポインタと送信前先頭ポインタとが一致していなければ（ステップＳ１９がＮｏ）、ドライバ３１０-1は、ＤＭＡ要求が未発行の送信ディスクリプタがあるはずであると判断する。この場合、ドライバ３１０-1は、ネットワークデバイス１３に対して再びＤＭＡ要求を発行するために、ステップＳ１７に戻る。

【００８６】

これに対し、空き先頭ポインタと送信前先頭ポインタとが一致しているならば（ステップＳ１９がＹｅｓ）、ドライバ３１０-1は、ＤＭＡ要求が未発行の送信ディスクリプタがないとして送信処理を終了する。

【００８７】

［受信処理］

次に、ゲストＯＳのドライバ（の受信処理部３１６）がネットワークデバイス１３を通じてパケットを受信する際に実行される「受信処理」の手順について、当該ドライバがゲストＯＳ３１-1のドライバ３１０-1である場合を例に、図７のフローチャートを参照して説明する。

【００８８】

ネットワークデバイス１３は、ネットワーク２から自身が受信すべきパケットを検知すると、空きの受信ディスクリプタ３２６-jのポインタ部３２５２で指定されるＤＭＡバッファ３２４ａ内の領域へ検知されたパケットをＤＭＡ転送する動作（受信ＤＭＡ動作）を開始する（ステップＳ２１）。ＤＭＡバッファ３２４ａ内の領域の使用順序は例えばＨＷを用いた設定によって予め指定することができる。本実施形態では、ＤＭＡバッファ３２４ａ内の領域の使用順序は、受信ディスクリプタチェーンの順番通りになるよう設定されている。

【００８９】

さて、ネットワークデバイス１３による上述の受信ＤＭＡ動作（ステップＳ２１）が完了したものとする。即ち、ネットワークデバイス１３がネットワーク２からパケットを受信し、その受信されたパケットを、受信ディスクリプタチェーンに従って空きの受信ディスクリプタ３２６-jのポインタ部３２５２で指定されるＤＭＡバッファ３２４ａ内の領域へＤＭＡ転送する受信ＤＭＡ動作（ステップＳ２１）が完了したものとする。この場合、ネットワークデバイス１３はマスターゲストＯＳにＨＷ割り込みを発行する（ステップＳ２２）。ここでは、マスターゲストＯＳはゲストＯＳ３１-1であるものとする。

【００９０】

マスターゲストＯＳ３１-1のドライバ３１０-1は、ネットワークデバイス１３からのＨＷ割り込みを受け付けると、ＤＭＡ中先頭ポインタの指し示す受信ディスクリプタチェーンの位置から、受信ＤＭＡ（受信データを一時格納するためのＤＭＡ転送）が完了した受信ディスクリプタ３２６-jを調べる（ステップＳ２３）。このステップＳ２３においてドライバ３１０-1は、受信ＤＭＡが完了した受信ディスクリプタ３２６-jのステータスを「ｅｍｐｔｙ」から「ｒｅｃｅｉｖｅｄ」に更新すると共に、ＤＭＡ中先頭ポインタを更新

する。また、ステップＳ２３においてドライバ３１０-1は、受信ＤＭＡが完了した受信ディスクリプタ３２６-jの内容（受信ディスクリプタ３２６-jで示される受信データ）を必要としているゲストＯＳを特定する。即ちドライバ３１０-1は、ゲストＯＳ情報領域３２３に格納されている各ゲストＯＳのゲストＯＳ情報の中から、受信パケットに含まれている宛先ネットワークアドレスに一致するネットワークアドレスを含むゲストＯＳ情報を選択し、当該選択されたゲストＯＳ情報に含まれているゲストＯＳ識別子から上述の受信ディスクリプタ３２６-jの内容を必要とするゲストＯＳを特定する。

【００９１】

ドライバ３１０-1は、受信ＤＭＡが完了した受信ディスクリプタ３２６-jをいずれのゲストＯＳが参照すべきかを示す情報を、当該受信ディスクリプタ３２６-jの利用ゲストＯＳ情報部に設定する。もし、ネットワークデバイス１３によってネットワーク２から受信されたパケットがブロードキャストパケットのような、複数のゲストＯＳによって参照されるべきパケットである場合、当該複数のゲストＯＳの情報が、該当する受信ディスクリプタ３２６-jの利用ゲストＯＳ情報部に設定される。

【００９２】

ゲストＯＳ３１-1のドライバ３１０-1は、受信ディスクリプタ３２６-jの情報を必要とするゲストＯＳに対して、受信ＤＭＡの完了をＳＷ割り込みによって通知する（ステップＳ２４）。ここで、ドライバ３１０-1からのＳＷ割り込みを受けたゲストＯＳがゲストＯＳ３１-2及び３１-3であるものとする。

【００９３】

ゲストＯＳ３１-1からのＳＷ割り込みを受けたゲストＯＳ３１-2及び３１-3のそれぞれドライバ３１０-2及び３１０-3は、自身に割り当てられた受信ディスクリプタ３２６-jを参照して受信に必要な処理を行う（ステップＳ２５）。この受信に必要な処理は、受信ディスクリプタ３２６-jのポインタ部３２６２によって指し示されるＤＭＡバッファ３２４ａ内の領域のデータを、ゲストＯＳ３１-2及び及び３１-3に固有の記憶領域に転送する処理を含む。ドライバ３１０-2及び３１０-3は、受信に必要な処理を完了して、受信ディスクリプタ３２６-jを参照する必要がなくなると、当該ディスクリプタ３２６-jからそれぞれゲストＯＳ３１-2及び３１-3に関する情報を削除する。

【００９４】

マスターゲストＯＳ３１-1のドライバ３１０-1は、受信ディスクリプタ３２６-jから当該ディスクリプタ３２６-jを参照すべきゲストＯＳ３１-2及び３１-3に関する情報が全て削除されると、当該ディスクリプタ３２６-jのステータスを「ｒｅｃｅｉｖｅｄ」から「ｅｍｐｔｙ」に更新する（ステップＳ２６）。つまりドライバ３１０-1は、受信ディスクリプタ３２６-jを参照する必要のあった全てのゲストＯＳにとって、もはや当該ディスクリプタ３２６-jを参照する必要がなくなると、当該ディスクリプタ３２６-jのステータスを「ｅｍｐｔｙ」に更新する。ステップＳ２６においてドライバ３１０-1は、受信中先頭ポインタの指し示す受信ディスクリプタ３２６-jのステータスが「ｅｍｐｔｙ」に更新されたならば、当該受信中先頭ポインタを１受信ディスクリプタ分だけ進める。

【００９５】

［エラー割り込み処理］

次に、ネットワークデバイス１３での動作でエラーが発生した場合に実行される「エラー割り込み処理」の手順について、ゲストＯＳ３１-1がマスターゲストＯＳである場合を例に、図８のフローチャートを参照して説明する。

【００９６】

今、ネットワークデバイス１３での動作でエラーが発生したものとする。この場合、ネットワークデバイス１３はマスターゲストＯＳ３１-1のドライバ３１０-1にエラー発生を通知するためのＨＷ割り込みを発行する（ステップＳ３０）。

【００９７】

ドライバ３１０-1（のＨＷ割り込みハンドラ３１１）は、ネットワークデバイス１３からＨＷ割り込み（ステップＳ３０）を受けると、共有メモリ３２の物理レジスタマップ領

域３２１にマップされた、当該ＨＷ割り込みの要因を示す割り込み要因レジスタ（例えば当該ネットワークデバイス１３のエラーステータスを示すエラーステータスレジスタ）を参照することによって当該ＨＷ割り込みの要因となったエラーの種類を識別する（ステップＳ３１）。このステップＳ３１においてドライバ３１０-1（のエラー割り込み処理部３１７）は、識別されたエラーに対応する処理（エラー処理）が必要ならば、共有メモリ３２の特殊処理ステータス領域３２２に格納されているビットマップ中の当該エラー処理に対応するビットを、エラー処理が開始されていることを示す状態に設定する。つまりドライバ３１０-1（マスターゲストＯＳ３１-1のドライバ３１０-1）は特殊処理ステータス領域３２２に、識別されたエラーに対応する処理（エラー処理）が開始されていることを示すステータス（特殊処理ステータス）を設定する（ステップＳ３１）。

【００９８】

次にマスターゲストＯＳ３１-1のドライバ３１０-1（のエラー割り込み処理部３１７）は、他のゲストＯＳ３１-2及び３１-3のそれぞれドライバ３１０-2及び３１０-3に対し、エラー処理を開始したことを、ＳＷ割り込みを使用して通知する（ステップＳ３２）。その後、マスターゲストＯＳ３１-1のドライバ３１０-1は、エラー処理が終了すると、共有メモリ３２の特殊処理ステータス領域３２２に格納されているビットマップ中の当該処理に対応するビットの状態を、処理が行われていないことを示す状態に切り替える。つまりドライバ３１０-1は、特殊処理ステータス領域３２２に設定されている、エラー処理が開始されていることを示すステータスを解除する（ステップＳ３３）。

【００９９】

一方、マスターゲストＯＳ３１-1以外のゲストＯＳ３１-2及び３１-3のそれぞれドライバ３１０-2及び３１０-3（のエラー割り込み処理部３１７）は、ドライバ３１０-1からエラー処理の開始を通知するＳＷ割り込み（ステップＳ３２）を受けると、現在実行中の処理を中断または中止するために必要な処理を、当該エラーの種類に応じて行う（ステップＳ４１）。

【０１００】

次にドライバ３１０-2及び３１０-3は、それぞれ共有メモリ３２の特殊処理ステータス領域３２２のステータスを監視し、エラー処理が開始されていることを示すステータスが解除されていることを確認したならば、つまりエラー処理が終了したことを確認したならば、通常の処理に復帰する（ステップＳ４２）。

【０１０１】

以上、本実施形態において適用される、ゲストＯＳ３１-i（ｉ＝１，２，３）のドライバ３１０-i（に設けられたｏｐｅｎ処理部３１３、送信処理部３１５、受信処理部３１６及びエラー割り込み処理部３１７）が有する４つの基本的な機能、つまり「ｏｐｅｎ処理機能」、「送信処理機能」、「受信処理機能」及び「エラー割り込み処理機能」について詳述した。これらの機能、特に「送信処理機能」及び「受信処理機能」により、ＤＭＡバッファのデータのコピーによるオーバヘッドを除くことができる。

【０１０２】

さて本実施形態では、ドライバ３１０-iに、「マスター権の委譲処理機能」及び「マスター権の取得処理機能」（をそれぞれ有するマスター権委譲処理部３１８ａ及びマスター権取得処理部３１８ｂ）が追加されている。以下、「マスター権の委譲処理」及び「マスター権の取得処理」について順次説明する。

【０１０３】

［マスター権の委譲処理］

「マスター権の委譲処理」とは、マスターゲストＯＳが自律的に、マスター権を他のゲストＯＳへと移すための処理である。以下、「マスター権の委譲処理」について、ゲストＯＳ３１-1がマスターゲストＯＳである場合を例に、図９のフローチャートを参照して説明する。

【０１０４】

マスターゲストＯＳ３１-1のドライバ３１０-1（のマスター権委譲処理部３１８ａ）は

、マスター権の委譲が必要となった場合、現在ネットワークデバイス１３を共有している他のゲストＯＳの中から、マスター権が委譲されるべきゲストＯＳを１つ選択する（ステップＳ５１）。ここでは、ゲストＯＳ３１-2が選択されたものとする。ステップＳ５１において、マスターゲストＯＳ３１-1のドライバ３１０-1は、ＳＷ割り込みを使用して、マスターゲストＯＳ３１-1以外の全てのゲストＯＳ（ここでは、ゲストＯＳ３１-2及び３２-3）に、ゲストＯＳ３１-1からゲストＯＳ３１-2へのマスターゲストＯＳ変更を通知する。

【０１０５】

ＳＷ割り込みによってマスターゲストＯＳ変更の通知を受けたゲストＯＳのうち、マスター権が委譲されるゲストＯＳ３１-2のドライバ３１０-2は、共有メモリ３２のゲストＯＳ情報領域３２３のマスターゲストＯＳに関連する情報を当該ゲストＯＳ３１-2を示すように更新する（ステップＳ５２）。

【０１０６】

マスターゲストＯＳ変更の通知を受けたゲストＯＳのうち、マスター権が委譲されるゲストＯＳ（新マスターゲストＯＳ）３１-2のドライバ３１０-2は、ネットワークデバイス１３からのＨＷ割り込み先がゲストＯＳ３１-2（ドライバ３１０-2）自身になるように設定を変更する（ステップＳ６１）。もし、マスターゲストＯＳ変更を通知したゲストＯＳ（旧マスターゲストＯＳ）３１-1が特殊処理を実行中であったならば、新マスターゲストＯＳ３１-2は、共有メモリ３２の特殊処理ステータス領域３２２を参照して当該特殊処理を途中から引き継ぐ（ステップＳ６２）。

【０１０７】

［マスター権の取得処理］

「マスター権の取得処理」とは、マスター以外のゲストＯＳが自身へマスター権を移すための処理、つまりマスター以外のゲストＯＳがマスター権を取得するための処理である。以下、「マスター権の取得処理」について、ゲストＯＳ３１-1がマスターゲストＯＳであり、ゲストＯＳ３１-2がマスター権を取得しようとするゲストＯＳである場合を例に、図１０のフローチャートを参照して説明する。

【０１０８】

ゲストＯＳ３１-2のドライバ３１０-2（のマスター権取得処理部３１８ｂ）は、当該ゲストＯＳ３１-2がマスター権を取得しようとする場合、その旨を当該ゲストＯＳ３１-2以外の、マスターゲストＯＳ３１-1を含む全てのゲストＯＳ（ここでは、ゲストＯＳ３１-1及び３２-3）に対してＳＷ割り込みを使用して通知する（ステップＳ７１）。この通知により、ゲストＯＳ３１-2以外のゲストＯＳでのマスターゲストＯＳに関する処理が抑止され、ゲストＯＳ３１-2はマスター権を排他的に取得することが可能となる。

【０１０９】

ゲストＯＳ３１-2のドライバ３１０-2はステップＳ７１を実行すると、共有メモリ３２のゲストＯＳ情報領域３２３のマスターゲストＯＳ情報を当該ゲストＯＳ３１-2を示すように更新する（ステップＳ７２）。これによりゲストＯＳ３１-2はマスター権を取得して、新たにマスターゲストＯＳ（新マスターゲストＯＳ）となる。

【０１１０】

ゲストＯＳ３１-2のドライバ３１０-2は、ネットワークデバイス１３からのＨＷ割り込み先がゲストＯＳ３１-2自身になるように設定を変更する（ステップＳ７３）。

【０１１１】

もし、ゲストＯＳ（新マスターゲストＯＳ）３１-2がマスター権を取得するまでマスターであったゲストＯＳ（旧マスターゲストＯＳ）３１-1が特殊処理を実行中であったならば、新マスターゲストＯＳ３１-2は、共有メモリ３２の特殊処理ステータス領域３２２を参照して当該特殊処理を途中から引き継ぐ（ステップＳ７４）。

【０１１２】

［ｃｌｏｓｅ処理］

本実施形態では、ゲストＯＳのドライバ（のｃｌｏｓｅ処理部３１４）がネットワーク

デバイス１３の使用を終了する際に実行される「ｃｌｏｓｅ処理（ドライバのｃｌｏｓｅ処理）」で、上述の「マスター権の委譲処理」が利用される。以下、この「ｃｌｏｓｅ（クローズ）処理」の手順について、ゲストＯＳ３１-1のドライバ３１０-1がネットワークデバイス１３の使用を終了する場合を例に、図１１のフローチャートを参照して説明する。

【０１１３】

今、ゲストＯＳ３１-1から当該ゲストＯＳ３１-1のドライバ３１０-1に対して、「ｃｌｏｓｅ要求」が与えられたものとする。するとドライバ３１０-1（のｃｌｏｓｅ処理部３１４）は、当該ドライバ３１０-1を含むゲストＯＳ３１-1がマスターゲストＯＳであるかを判定する（ステップＳ８１）。もし、ゲストＯＳ３１-1がマスターゲストＯＳでないならば（ステップＳ８１がＮｏ）、ドライバ３１０-1は後述するステップＳ８４に進む。

【０１１４】

一方、ゲストＯＳ３１-1がマスターゲストＯＳであるならば（ステップＳ８１がＹｅｓ）、ドライバ３１０-1は、他にネットワークデバイス１３を共有しているゲストＯＳ（他のゲストＯＳ）が存在するかを判定する（ステップＳ８２）。

【０１１５】

もし、ステップＳ８２の判定がＹｅｓであるならば、即ちゲストＯＳ３１-1がマスターゲストＯＳで、且つ他にネットワークデバイス１３を共有しているゲストＯＳがあるならば、当該ゲストＯＳ３１-1のドライバ３１０-1（のｃｌｏｓｅ処理部３１４）は（マスター権委譲処理部３１８ａを用いて）上述の「マスター権の委譲処理」を行う（ステップＳ８３）。そしてドライバ３１０-1はステップＳ８４に進む。

【０１１６】

ステップＳ８４においてドライバ３１０-1（旧マスターゲストＯＳ３１-1のドライバ３１０-1）は、共有メモリ３２から当該ドライバ３１０-1を含むゲストＯＳ（旧マスターゲストＯＳ）３１-1に関連する情報を削除する。このステップＳ８４での処理は、ドライバ３１０-1が確保・参照していた各ディスクリプタの解放を含む。ドライバ３１０-1はステップＳ８４を実行すると、「ｃｌｏｓｅ処理」を終了する。

【０１１７】

一方、ステップＳ８２の判定がＮｏであるならば、即ちゲストＯＳ３１-1がマスターゲストＯＳで、且つ他にネットワークデバイス１３を共有しているゲストＯＳがないならば、当該ゲストＯＳ３１-1のドライバ３１０-1は、ネットワークデバイス１３を停止させるための処理（ネットワークデバイス終了処理）を行う（ステップＳ８５）。そしてドライバ３１０-1は、ＶＭＭ２０により共有メモリ３２（共有メモリ領域１２０）を解放させて（ステップＳ８６）、「ｃｌｏｓｅ処理」を終了する。

【０１１８】

このような「ｃｌｏｓｅ処理」により、たとえマスターゲストＯＳのドライバがｃｌｏｓｅされても、つまりマスターゲストＯＳがネットワークデバイス１３の使用を終了しても、他のゲストＯＳで運用を続けることが可能となる。

【０１１９】

＜相互監視に基づいたゲストＯＳの排除＞

本実施形態のように複数のゲストＯＳのドライバ（ネットワークデバイスドライバ）が協調して動作する場合、次のような問題が生じる可能性がある。例えば、あるゲストＯＳが「ｐａｎｉｃ」或いは「ｈａｎｇ」と呼ばれるような動作不能な状態に陥った場合に、ネットワークデバイス１３のＨＷ仕様によってはそのゲストＯＳが使用しているディスクリプタの処理が進まなくなって、結果として送受信の処理を進めなくなる可能性がある。また、マスターゲストＯＳが動作不能になった場合は、通信自体を行うことができなくなってしまう。

【０１２０】

そこで本実施形態では、ゲストＯＳ３１-1～３１-3が正常に動作しているかを、当該ゲストＯＳ３１-1～３１-3のドライバ３１０-1～３１０-3が相互に監視する仕組みと、正常

に動作していないと判断されたゲストＯＳをネットワークデバイス１３の共有状態から強制的に排除する仕組みとが用意される。この２つの仕組みにより、ゲストＯＳ（例えばマスターゲストＯＳ）が動作不能になっても、残りのゲストＯＳによって運用が続けられる。

【０１２１】

正常に動作していないゲストＯＳを検出する手法は種々考えられる。ここでは以下に示す簡単な検出手法が適用されるものとする。

まず、共有メモリ３２（共有メモリ領域１２０）に、ゲストＯＳ３１-1～３１-3に共通のカウンタ（カウンタ領域）が用意される。

【０１２２】

ゲストＯＳ３１-1～３１-3のドライバ３１０-1～３１０-3は、それぞれ特定のタイミングで上記共通のカウンタをカウントアップし、そのカウントアップ後の当該カウンタの値（カウンタ値）を、共有メモリ３２のゲストＯＳ情報領域３２３に格納される自身のゲストＯＳ情報内に保持する。なお、上記特定のタイミングを、ゲストＯＳ内のタイマで管理しても良いし、通信時における特定の処理の機会としても良い。

【０１２３】

ゲストＯＳ３１-1～３１-3のドライバ３１０-1～３１０-3は、それぞれ他のゲストＯＳによって当該他のゲストＯＳのゲストＯＳ情報に保持されているカウンタ値（つまり他のゲストＯＳのカウンタ値）を監視する。そしてドライバ３１０-1～３１０-3は、他のゲストＯＳのカウンタ値と共通のカウンタの値との差が閾値を超えていたなら、当該他のゲストＯＳにＳＷ割り込みを配信する。なお、上記の監視のタイミングを、ゲストＯＳ内のタイマで管理しても良いし、通信時における特定の処理の機会としても良い。

【０１２４】

ドライバ３１０-1～３１０-3は、ＳＷ割り込みの配信先のゲストＯＳが当該ＳＷ割り込みに反応（例えば上記共通のカウンタのカウントアップ）できなければ、そのゲストＯＳは正常に動作していない、異常ゲストＯＳであると判断する。

【０１２５】

［ゲストＯＳ排除処理］

以下、正常に動作していないと判断された異常ゲストＯＳを（ネットワークデバイス１３を共有する状態から）排除するための「ゲストＯＳ排除処理」の手順について、図１２のフローチャートを参照して説明する。

【０１２６】

今、ゲストＯＳ３１-2のドライバ３１０-2（のゲストＯＳ排除処理部３１９）が、例えばゲストＯＳ３１-1を、正常に動作していない異常ゲストＯＳ（つまり排除されるべきゲストＯＳ）として検出したものとする（ステップＳ９１）。異常ゲストＯＳ３１-1を検出したドライバ３１０-2は、当該異常ゲストＯＳ３１-1がマスターゲストＯＳであるかを当該異常ゲストＯＳ３１-1のゲストＯＳ情報に基づいて判定する（ステップＳ９２）。

【０１２７】

もし、異常ゲストＯＳ３１-1がマスターゲストＯＳでないならば（ステップＳ９２がＮｏ）、当該異常ゲストＯＳ３１-1を検出したドライバ３１０-2は後述するステップＳ９４に進む。これに対して異常ゲストＯＳ３１-1がマスターゲストＯＳであるならば（ステップＳ９２がＹｅｓ）、当該異常ゲストＯＳ３１-1を検出したドライバ３１０-2が（マスター権取得処理部３１８ｂを用いて）上述の「マスター権の取得処理」を行う（ステップＳ９３）。ドライバ３１０-2はステップＳ９３を実行するとステップＳ９４に進む。

【０１２８】

ステップＳ９４においてドライバ３１０-2は、排除されるべき異常ゲストＯＳ３１-1に、その旨をＳＷ割り込みを使用して通知する（ステップＳ９４）。この通知は、異常ゲストＯＳ３１-1（正常に動作していないと判断された異常ゲストＯＳ）が動作不能状態から復帰した場合に、ネットワークデバイス１３を共有する状態から排除されているにもかかわらずに自身が当該ネットワークデバイス１３を使用しているものとして誤動作しないよ

うに知らせるために行われる。
【０１２９】
次にドライバ３１０-2は、共有メモリ３２から異常ゲストＯＳ３１-1に関連する情報を削除する（ステップＳ９５）。このステップＳ９５での処理は、異常ゲストＯＳ３１-1のドライバ３１０-1が確保・参照していた各ディスクリプタの解放を含む。ドライバ３１０-2は、ステップＳ９５の実行により、異常ゲストＯＳ３１-1をネットワークデバイス１３を共有する状態から強制的に排除すると、「ゲストＯＳ排除処理」を終了する。
【０１３０】
さて、排除されたゲストＯＳ３１-1のドライバ３１０-1が復帰したものとする。そして、復帰したドライバ３１０-1が、ドライバ３１０-2からのＳＷ割り込み（ステップＳ９４）によって当該ドライバ３１０-1を含むゲストＯＳ３１-1が排除されたことが認識できたものとする。この場合、ドライバ３１０-1は、ゲストＯＳ３１-1の排除に対応した後処理、例えばゲストＯＳ３１-1をエラー扱いとする処理を行う（ステップＳ１００）。
【０１３１】
＜動的なマスター権の移行＞
上述の説明では、マスター権の移行は、マスターゲストＯＳの終了や、マスターゲストＯＳに異常が発生した際に特別に行われる。しかし、マスターゲストＯＳが正常稼動中でも、動作効率の向上のために、動的にマスター権を移行することも可能である。このような動作効率向上のためのマスター権移行の例を以下に列挙する。
【０１３２】
（１）最も通信回数の多いゲストＯＳがマスターとなる。この場合、通信の際のマスターゲストＯＳとその他のゲストＯＳとの間の通知やディスパッチによるオーバヘッドを軽減することができる。通信回数に関しては、共有メモリ３２-1～３２１-2のゲストＯＳ情報領域３２３に格納されるゲストＯＳ情報に含まれている通信回数統計情報を利用すれば良い。
【０１３３】
（２）ディスパッチ機会（時間）の最も多いゲストＯＳ（つまりディスパッチ機会の最も多いゲストＯＳ）がマスターとなる。各ゲストＯＳのディスパッチ機会（時間）の情報を取得するには、当該各ゲストＯＳのディスパッチポリシ（スケジュール）を当該ゲストＯＳ自身が把握すれば良い。本実施形態においてＶＭＭ２０は、ゲストＯＳ３１-1～３１-3のディスパッチポリシを提供する手段を有している。ゲストＯＳ３１-1～３１-3のそれぞれドライバ３１０-1～３１０-3（のマスター権移行処理部３１８）は、この手段を利用して、ゲストＯＳ３１-1～３１-3のディスパッチポリシを把握する。ディスパッチポリシを提供する手段は、例えばゲストＯＳ３１-1～３１-3にＣＰＵ１１（ＣＰＵ時間）を割り当てるための、ＶＭＭ２０に設けられたスケジューラまたはディスパッチャに含まれているものとする。ここでは、ディスパッチポリシを提供する手段は、ゲストＯＳ３１-1～３１-3のディスパッチ時間の比率を指定するディスパッチ指定手段である。
【０１３４】
（３）各ゲストＯＳに予め優先度（ゲストＯＳ優先度）を付与し、稼働中のゲストＯＳの中で最も優先度が高いゲストＯＳがマスターとなる。各ゲストＯＳの優先度は、例えば、システムの構築者のようなユーザの指定によって、ネットワーク２に接続されたクライアント端末から付与されるものとする。各ゲストＯＳの優先度は、前述のように、共有メモリ３２-1～３２１-2のゲストＯＳ情報領域３２３に格納されるゲストＯＳ情報に含まれている。
【０１３５】
［通信回数の多いゲストＯＳへのマスター権の移行処理］
次に上記（１）に対応した「通信回数の多いゲストＯＳへのマスター権の移行処理」について説明する。
【０１３６】
本実施形態では、マスターゲストＯＳ以外のゲストＯＳが通信を行う度に、マスターゲ

ストＯＳ自身が通信を行う場合と比較して、マスターゲストＯＳへの割り込み通知とディスパッチによるオーバヘッドが発生する。したがって、より通信回数の多いゲストＯＳがマスターゲストＯＳとなる方が、システム全体のオーバヘッドが少なくなり、通信のスループットが良くなることが期待される。

【０１３７】

本実施形態では、通信回数の判断に、各ゲストＯＳが使用したディスクリプタの数が使用される。ここでは通信回数の判断に、送信ディスクリプタチェーン３２５ａ（における送信ディスクリプタ３２５-j）が使用されるものとする。しかし通信回数の判断に、受信ディスクリプタチェーン（における受信ディスクリプタ３２６-j）を用いても良い。また、通信回数をより高精度に判断するために、両ディスクリプタチェーン（におけるディスクリプタ３２５-j及び３２６-j）を用いても構わない。

【０１３８】

マスター権を移行するかを判定するのに、「最も多くディスクリプタを使用したゲストＯＳをマスターゲストＯＳとする」ような簡単な手法が考えられる。この手法を適用した場合、マスター権の移行処理が頻発する可能性がある。また、マスター権の移行処理自体がオーバヘッドとなる可能性もある。

【０１３９】

そこで本実施形態では、マスター権の移行処理が頻発するのを防ぐために、以下に挙げる２つの条件に基づきマスター権を移行するかが判定される。

１ａ）マスターゲストＯＳ以外のある１つのゲストＯＳの通信回数が著しく多い場合：

ディスクリプタチェーンに含まれるディスクリプタ数の一定割合以上（例えば１／２以上）を、ある１つのゲストＯＳが使用していた場合が、これに該当する。

【０１４０】

１ｂ）マスターゲストＯＳの通信回数が著しく少ない場合：

Ｎ個のゲストＯＳがネットワークデバイス１３を共有しているものとすると、ディスクリプタチェーンに含まれるディスクリプタの数の一定割合未満、例えば１／（２Ｎ）個未満しか、マスターゲストＯＳが使用していなかった場合が、これに該当する。

【０１４１】

以下、上述の２つの条件に基づいて行われる「通信回数の多いゲストＯＳへのマスター権の移行処理」の手順について、ゲストＯＳ３１-1がマスターゲストＯＳである場合を例に、図１３のフローチャートを参照して説明する。

【０１４２】

マスターゲストＯＳ３１-1のドライバ３１０-1（のマスター権移行処理部３１８）は、当該マスターゲストＯＳ３１-1からの送信要求、または他のゲストＯＳ（ゲストＯＳ３１-2または３１-3）からのＤＭＡ要求割り込みに応じて送信処理が行われる際に、共有メモリ３２内にある該当するゲストＯＳ（送信要求元またはＤＭＡ要求割り込み元のゲストＯＳ）の通信回数の統計情報を更新する（ステップＳ１１１）。

【０１４３】

次にマスターゲストＯＳ３１-1のドライバ３１０-1は、通信回数の統計情報によって示されるゲストＯＳ３１-1～３１-3の通信回数の合計ｚが、予め定められた一定個数、例えば１２００個に達したかを判定する（ステップＳ１１２）。もし、上記ｚが１２００個に達したならば（ステップＳ１１２がＹｅｓ）、ドライバ３１０-1は通信回数の調査を次のように行う。なお、調査を行うタイミングが、例えば一定時間毎のように、ｚの数以外の別のタイミングであっても構わない。

【０１４４】

まずマスターゲストＯＳ３１-1のドライバ３１０-1は、マスターゲストＯＳ３１-1以外のゲストＯＳ、つまりゲストＯＳ３１-2及び３１-3の通信回数の統計情報を調査し、当該ゲストＯＳ３１-2及び３１-3の中に、ディスクリプタ使用数（通信回数）ｂが上記一定個数（１２００個）の一定割合以上、例えば１／２以上（つまり６００個以上）のゲストＯＳが存在するかを判定する（ステップＳ１１３）。もし、ｂが６００個以上のゲストＯＳ

が見つかれば（ステップＳ１１３がＹｅｓ）、ドライバ３１０-1（のマスター権移行処理部３１８に含まれているマスター権委譲処理部３１８ａ）は、そのゲストＯＳにマスター権を委譲するための前述の「マスター権委譲処理」を行う（ステップＳ１１４）。そしてドライバ３１０-1は、後述するステップＳ１１６に進む。

【０１４５】

一方、ｂが６００個以上のゲストＯＳが見つからないならば（ステップＳ１１３がＮｏ）、マスターゲストＯＳ３１-1のドライバ３１０-1は、当該マスターゲストＯＳ３１-1の通信回数の統計情報を調査し、当該マスターゲストＯＳ３１-1のディスクリプタ使用数（通信回数）ａが上記一定個数（１２００個）の一定割合未満、例えば１／（２Ｎ）未満（Ｎが３の場合、２００個未満）であるかを判定する（ステップＳ１１５）。

【０１４６】

もし、マスターゲストＯＳ３１-1のディスクリプタ使用数が１２００の１／（２Ｎ）未満であれば（ステップＳ１１５がＹｅｓ）、当該マスターゲストＯＳ３１-1のドライバ３１０-1はステップＳ１１４に進む。このステップＳ１１４においてマスターゲストＯＳ３１-1のドライバ３１０-1（のマスター権移行処理部３１８に含まれているマスター権委譲処理部３１８ａ）は、当該ゲストＯＳ３１-1を除くゲストＯＳ３１-2及び３１-3のうち、最もディスクリプタ使用数の多いゲストＯＳにマスター権を委譲するための前述の「マスター権委譲処理」を行う（ステップＳ１１４）。そしてドライバ３１０-1は、ステップＳ１１６に進む。

【０１４７】

ステップＳ１１６においてドライバ３１０-1は、共有メモリ３２内にあるゲストＯＳ３１-1～３１-3の通信回数（ディスクリプタ使用数）の統計情報をクリアする。

【０１４８】

次に、上記「通信回数の多いゲストＯＳへのマスター権の移行処理」の具体例（例１乃至３）について説明する。ここでは、ゲストＯＳ３１-1、ゲストＯＳ３１-2及びゲストＯＳ３１-3を、それぞれゲストＯＳ＃１、ゲストＯＳ＃２及びゲストＯＳ＃３と表現する。また、ステップＳ１１２が実行される時点において、ゲストＯＳ＃１がマスターゲストＯＳであり、送信ディスクリプタチェーン３２５ａのディスクリプタ数の合計が１２００であるものとする。

【０１４９】

（例１）

ゲストＯＳ＃１のディスクリプタ使用数：４００
ゲストＯＳ＃２のディスクリプタ使用数：７００
ゲストＯＳ＃３のディスクリプタ使用数：１００

例１では、ステップＳ１１３の判定条件にゲストＯＳ＃２が合致する。この場合、ゲストＯＳ＃１からゲストＯＳ＃２にマスター権が移行される。

【０１５０】

（例２）

ゲストＯＳ＃１のディスクリプタ使用数：１００
ゲストＯＳ＃２のディスクリプタ使用数：５００
ゲストＯＳ＃３のディスクリプタ使用数：６００

例２では、ステップＳ１１３の判定条件に合致するゲストＯＳは存在しない。ここでは、マスターゲストＯＳ＃１のディスクリプタ使用数は、１２００／（３×２）、つまり２００よりも少ない。したがって、マスターゲストＯＳ＃１のディスクリプタ使用数（通信数）は、ステップＳ１１５の判定条件に合致する。この場合、残りのゲストＯＳのうち、最もディスクリプタ使用数の多いゲストＯＳ＃３にマスター権が移行される。

【０１５１】

（例３）

ゲストＯＳ＃１のディスクリプタ使用数：３００
ゲストＯＳ＃２のディスクリプタ使用数：５００

ゲストＯＳ＃３のディスクリプタ使用数：４００
例３では、ステップＳ１１３の判定条件に合致するゲストＯＳは存在しない。また、マスターゲストＯＳ＃１のディスクリプタ使用数は、ステップＳ１１５の判定条件に合致しない。この場合、ゲストＯＳ＃２のディスクリプタ使用数は最も多いものの、当該ゲストＯＳ＃２へのマスター権の移行は行われない。これにより、マスター権の移行処理が頻発するのが防止される。

【０１５２】

［ディスパッチ機会の多いゲストＯＳへマスター権の移行処理］

次に、上記（２）に対応した「ディスパッチ機会の多いゲストＯＳへのマスター権の移行処理」について説明する。

【０１５３】

ディスパッチされる機会（または時間）が最も多いゲストＯＳがマスター権を持つと、その分だけネットワークデバイス１３へのアクセスや特殊処理を行う機会が増える。このため、結果としてスループットが向上する可能性がある。

【０１５４】

そこで本実施形態では、ゲストＯＳ３１-1～３１-3のそれぞれドライバ３１０-1～３１０-3は、ＶＭＭ２０が有するディスパッチ指定手段を利用して、ゲストＯＳ３１-1～３１-3のディスパッチ時間の比率を把握する。

【０１５５】

ゲストＯＳ３１-1～３１-3のドライバ３１０-1～３１０-3は、当該ゲストＯＳ３１-1～３１-3のディスパッチ時間の比率に基づき、当該ドライバのｏｐｅｎ時における「マスター権の取得処理」及び当該ドライバのｃｌｏｓｅ時における「マスター権の委譲処理」を次のように実行する。なお、マスターゲストＯＳはゲストＯＳ３１-1であるものとする。またマスターゲストＯＳ以外のゲストＯＳ３１-2またはゲストＯＳ３１-3をゲストＯＳ３１-ｋ（ｋは２または３）と表現するものとする。

【０１５６】

（ドライバのｏｐｅｎ時）

ゲストＯＳ３１-kのドライバ３１０-kのｏｐｅｎ時において、マスターゲストＯＳ３１-1よりも当該ゲストＯＳ３１-kのディスパッチ時間の比率の設定が大きいものとする。この場合、ゲストＯＳ３１-kのドライバ３１０-kは、マスター権の取得処理を行う。

【０１５７】

（ドライバのｃｌｏｓｅ時）

マスターゲストＯＳ３１-1のドライバ３１０-1のｃｌｏｓｅ時において、他のゲストＯＳの中でゲストＯＳ３１-kのディスパッチ時間の比率の設定が最も大きいものとする。この場合、マスターゲストＯＳ３１-1のドライバ３１０-1は、自身のｏｐｅｎ時に、ゲストＯＳ３１-kへマスター権を委譲する。

なお、ディスパッチ時間に代えて、ディスパッチ機会（回数）を用いても良い。

【０１５８】

［優先度の高いゲストＯＳへのマスター権の移行処理］

次に上記（３）に対応した「優先度の高いゲストＯＳへのマスター権の移行処理」について説明する。

【０１５９】

前述のように、ゲストＯＳ３１-1～３１-3の優先度（ゲストＯＳ優先度）は、ＶＭＭ２０が、ディスパッチされるべきゲストＯＳを選択する際に選定の基準とする値である。このため優先度が高いゲストＯＳほど、当該ゲストＯＳへのディスパッチの機会が多く、ＣＰＵ割当時間が多くなると想定される。したがって、ゲストＯＳ３１-1～３１-3の優先度は、当該ゲストＯＳ３１-1～３１-3のディスパッチのされやすさの指標（つまりディスパッチされる時間もしくは回数の指標）であるといえる。このため、ゲストＯＳ３１-1～３１-3のディスパッチ時間の比率に代えて、当該ゲストＯＳ３１-1～３１-3の優先度を用いることも可能である。

【０１６０】
そこでゲストＯＳ３１-1～３１-3のドライバ３１０-1～３１０-3は、共有メモリ３２のゲストＯＳ情報領域３２３に格納されているゲストＯＳ情報に含まれているゲストＯＳ優先度に基づいて、当該ドライバのｏｐｅｎ時における「マスター権の取得処理」及び当該ドライバのｃｌｏｓｅ時における「マスター権の委譲処理」を次のように実行する。なお、マスターゲストＯＳはゲストＯＳ３１-1であるものとする。またマスターゲストＯＳ以外のゲストＯＳ３１-2またはゲストＯＳ３１-3をゲストＯＳ３１-ｋ（ｋは２または３）と表現するものとする。
【０１６１】
（ドライバのｏｐｅｎ時）
ゲストＯＳ３１-kのドライバ３１０-kのｏｐｅｎ時において、マスターゲストＯＳ３１-1よりも当該ゲストＯＳ３１-kの優先度が高いものとする。この場合、ゲストＯＳ３１-kのドライバ３１０-kは、マスター権の取得処理を行う。
【０１６２】
（ドライバのｃｌｏｓｅ時）
マスターゲストＯＳ３１-1のドライバ３１０-1のｃｌｏｓｅ時において、他のゲストＯＳの中でゲストＯＳ３１-kの優先度が最も高いものとする。この場合、マスターゲストＯＳ３１-1のドライバ３１０-1は、自身のｏｐｅｎ時に、ゲストＯＳ３１-kへマスター権を委譲する。
【０１６３】
［ゲストＯＳの優先度の変更時おけるマスター権の移行処理］
ゲストＯＳ３１-1～３１-3の優先度は、例えばＶＭＭ２０によって変更され、その変更がＶＭＭ２０からゲストＯＳ３１-1～３１-3に通知されるものとする。またゲストＯＳ３１-1～３１-3の優先度は、当該ゲストＯＳ３１-1～３１-3からＶＭＭ２０への要求によっても変更されるものとする。
【０１６４】
以下、「ゲストＯＳの優先度の変更時おける（優先度の高いゲストＯＳへの）マスター権の移行処理」について、ゲストＯＳ３１-1の優先度が変更された場合を例に、図１４のフローチャートを参照して説明する。
【０１６５】
今、ゲストＯＳ３１-1のドライバ３１０-1（のマスター権移行処理部３１８）が、当該ゲストＯＳ３１-1の優先度（ゲストＯＳ優先度）がＶＭＭ２０によって変更されたことを、当該ＶＭＭ２０からの通知によって検出したものとする。するとゲストＯＳ３１-1のドライバ３１０-1は、共有メモリ３２のゲストＯＳ情報領域３２３に格納されているゲストＯＳ情報に含まれている当該ゲストＯＳ３１-1のゲストＯＳ優先度を当該変更後の優先度に更新する（ステップＳ１２１）。なお、ゲストＯＳ３１-1のドライバ３１０-1が上記ゲストＯＳ情報領域３２３に格納されているゲストＯＳ情報を例えば定期的に参照することにより、当該ゲストＯＳ３１-1のゲストＯＳ優先度が変更されたことを検出しても良い。
【０１６６】
次にゲストＯＳ３１-1のドライバ３１０-1は、当該ゲストＯＳ３１-1がマスターゲストＯＳであるかを判定する（ステップＳ１２２）。もし、ゲストＯＳ３１-1がマスターゲストＯＳであるならば（ステップＳ１２２がＹｅｓ）、当該ゲストＯＳ３１-1のドライバ３１０-1は、当該ゲストＯＳ３１-1の優先度が下がったかを判定する（ステップＳ１２３）。
【０１６７】
もし、ゲストＯＳ３１-1の優先度が下がってないならば（ステップＳ１２３がＮｏ）、当該ゲストＯＳ３１-1のドライバ３１０-1は何もせずにマスター権の移行処理を終了する。
【０１６８】
これに対し、ゲストＯＳ３１-1の優先度が下がったならば（ステップＳ１２３がＹｅｓ

）、当該ゲストＯＳ３１-1のドライバ３１０-1は、共有メモリ３２のゲストＯＳ情報領域３２３に格納されているゲストＯＳ情報に基づき、当該ゲストＯＳ３１-1（つまりマスターゲストＯＳ）よりも優先度の高いゲストＯＳが存在するかを判定する（ステップＳ１２４）

もし、ゲストＯＳ３１-1（マスターゲストＯＳ）よりも優先度の高いゲストＯＳが存在しないならば（ステップＳ１２４がＮｏ）、当該ゲストＯＳ３１-1のドライバ３１０-1は何もせずにマスター権の移行処理を終了する。

【０１６９】

これに対し、ゲストＯＳ３１-1（マスターゲストＯＳ）よりも優先度の高いゲストＯＳが存在するならば（ステップＳ１２４がＹｅｓ）、当該ゲストＯＳ３１-1のドライバ３１０-1（のマスター権移行処理部３１８に含まれているマスター権委譲処理部３１８ａ）は、当該ゲストＯＳ３１-1（マスターゲストＯＳ）よりも優先度の高いゲストＯＳのうち最も優先度の高いゲストＯＳにマスター権を委譲して（ステップＳ１２５）、マスター権の移行処理を終了する。即ちゲストＯＳ３１-1のドライバ３１０-1は、当該ゲストＯＳ３１-1がマスターゲストＯＳであり、且つ当該ゲストＯＳ３１-1の優先度が下がって他のゲストＯＳの優先度のほうが高くなっていれば、「マスター権委譲処理」を行う。

【０１７０】

一方、ゲストＯＳ３１-1がマスターゲストＯＳでない場合（ステップＳ１２２がＮｏ）にも、ドライバ３１０-1は、当該ゲストＯＳ３１-1の優先度が下がったかを判定する（ステップＳ１２６）。

【０１７１】

もし、ゲストＯＳ３１-1の優先度が下がったならば（ステップＳ１２６がＹｅｓ）、当該ゲストＯＳ３１-1のドライバ３１０-1は何もせずにマスター権の移行処理を終了する。

【０１７２】

これに対し、ゲストＯＳ３１-1の優先度が下がってないならば（ステップＳ１２６がＮｏ）、当該ゲストＯＳ３１-1のドライバ３１０-1は、当該ゲストＯＳ３１-1の優先度が上がってマスターゲストＯＳの優先度よりも高くなったかを判定する（ステップＳ１２７）。

【０１７３】

もし、ゲストＯＳ３１-1の優先度がマスターゲストＯＳの優先度よりも依然として低いならば（ステップＳ１２７のＮｏ）、当該ゲストＯＳ３１-1のドライバ３１０-1は何もせずにマスター権の移行処理を終了する。

【０１７４】

これに対し、ゲストＯＳ３１-1の優先度がマスターゲストＯＳの優先度よりも高くなったならば（ステップＳ１２７のＹｅｓ）、当該ゲストＯＳ３１-1のドライバ３１０-1（のマスター権移行処理部３１８に含まれているマスター権取得処理部３１８ｂ）はマスター権を取得して（ステップＳ１２８）、マスター権の移行処理を終了する。即ちゲストＯＳ３１-1のドライバ３１０-1は、当該ゲストＯＳ３１-1がマスターゲストＯＳでなく、且つ当該ゲストＯＳ３１-1の優先度が上がってマスターゲストＯＳの優先度よりも高くなっていれば、「マスター権取得処理」を行う。

【０１７５】

なお、本発明は、上記実施形態そのままに限定されるものではなく、実施段階ではその要旨を逸脱しない範囲で構成要素を変形して具体化できる。また、上記実施形態に開示されている複数の構成要素の適宜な組み合わせにより種々の発明を形成できる。例えば、実施形態に示される全構成要素から幾つかの構成要素を削除してもよい。

【図面の簡単な説明】

【０１７６】

【図１】本発明の一実施形態に係る仮想計算機システムの構成を示すブロック図。

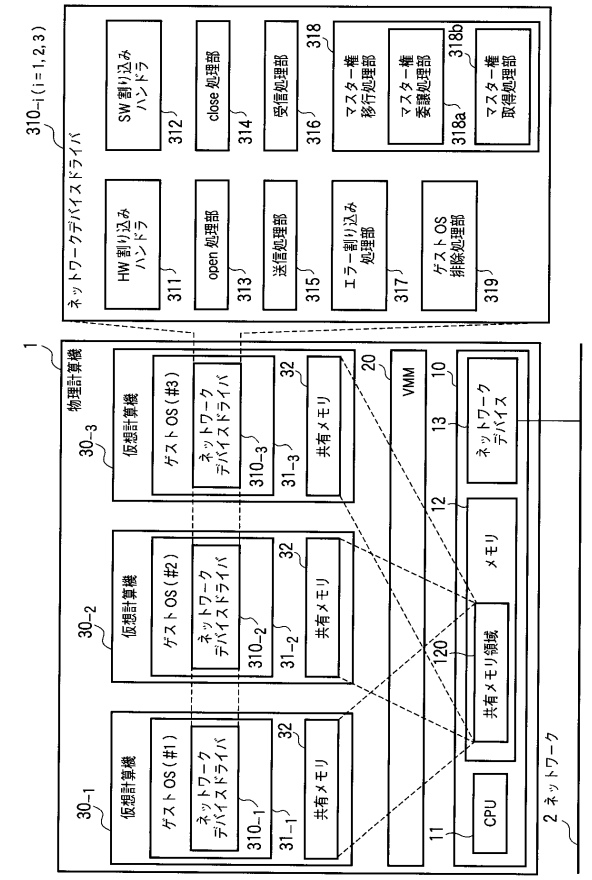
【図２】図１に示される共有メモリにおける領域の割り当て例を示す図。

【図３】図２に示される送信ディスクリプタチェーン領域に格納される送信ディスクリプ

タチェーンの一例を示す図。
【図４】図２に示される受信ディスクリプタチェーン領域に格納される受信ディスクリプタチェーンの一例を示す図。
【図５】同実施形態における「ｏｐｅｎ処理」の手順を示すフローチャート。
【図６】同実施形態における「送信処理」の手順を示すフローチャート。
【図７】同実施形態における「受信処理」の手順を示すフローチャート。
【図８】同実施形態における「エラー割り込み処理」の手順を示すフローチャート。
【図９】同実施形態における「マスター権の委譲処理」の手順を示すフローチャート。
【図１０】同実施形態における「マスター権の取得処理」の手順を示すフローチャート。
【図１１】同実施形態における「ｃｌｏｓｅ処理」の手順を示すフローチャート。
【図１２】同実施形態における「ゲストＯＳ排除処理」の手順を示すフローチャート。
【図１３】同実施形態における「通信回数の多いゲストＯＳへのマスター権の移行処理」の手順を示すフローチャート。
【図１４】同実施形態における「ゲストＯＳの優先度の変更時おけるマスター権の移行処理」の手順を示すフローチャート。
【符号の説明】
【０１７７】
１…物理計算機、２…ネットワーク、１０…ＨＷ（ハードウェア）、１１…ＣＰＵ、１２…メモリ、１３…ネットワークデバイス、２０…ＶＭＭ（仮想計算機マネージャ）、３０-1～３０-3…仮想計算機、３１-1～３１-3…ゲストＯＳ、３２…共有メモリ、１２０…共有メモリ領域、３１０-1～３１０-3…ネットワークデバイスドライバ、３１１…ＨＷ割り込みハンドラ、３１２…ＳＷ割り込みハンドラ、３１３…ｏｐｅｎ処理部、３１４…ｃｌｏｓｅ処理部、３１５…送信処理部、３１６…受信処理部、３１７…エラー割り込み処理部、３１８…マスター権移行処理部、３１８ａ…マスター権委譲処理部、３１８ｂ…マスター権取得処理部、３１９…ゲストＯＳ排除処理部、３２１…物理レジスタマップ領域、３２２…特殊処理ステータス領域、３２３…ゲストＯＳ情報領域、３２４…ＤＭＡバッファ領域、３２４ａ…ＤＭＡバッファ、３２５…送信ディスクリプタチェーン領域、３２５ａ…送信ディスクリプタチェーン、３２５-1～３２５-10…送信ディスクリプタ、３２６…受信ディスクリプタチェーン領域、３２６ａ…受信ディスクリプタチェーン、３２６-1～３２６-10…受信ディスクリプタ、３２７…他共有情報領域。

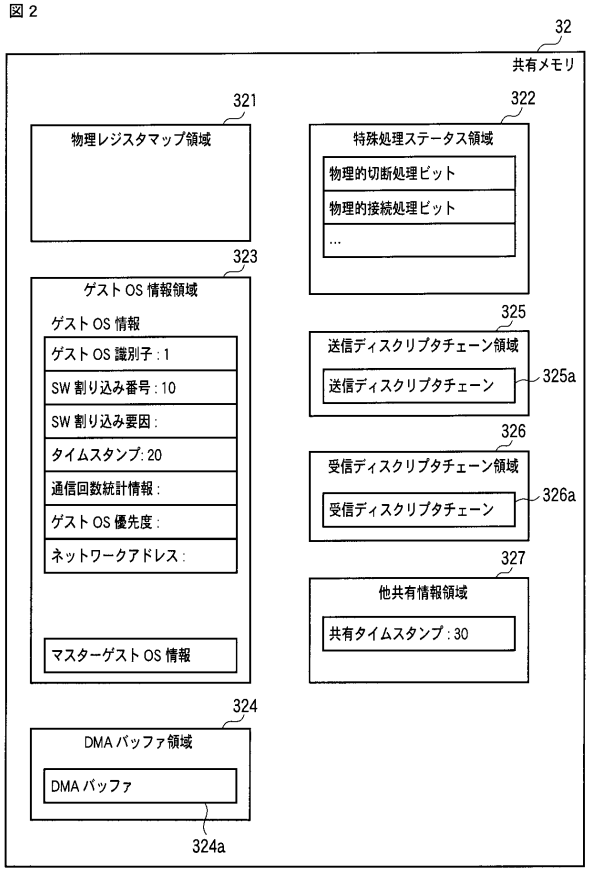
【図１】

図１



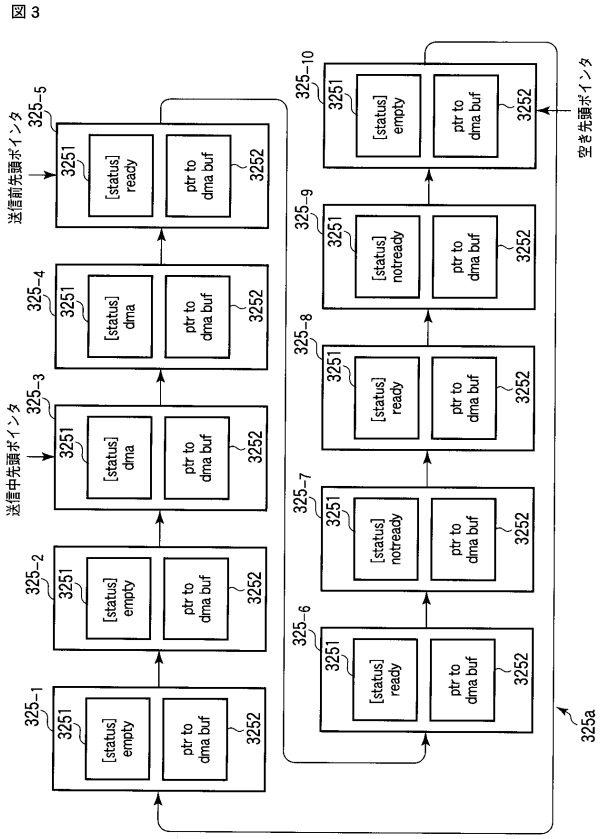
【図２】

図２



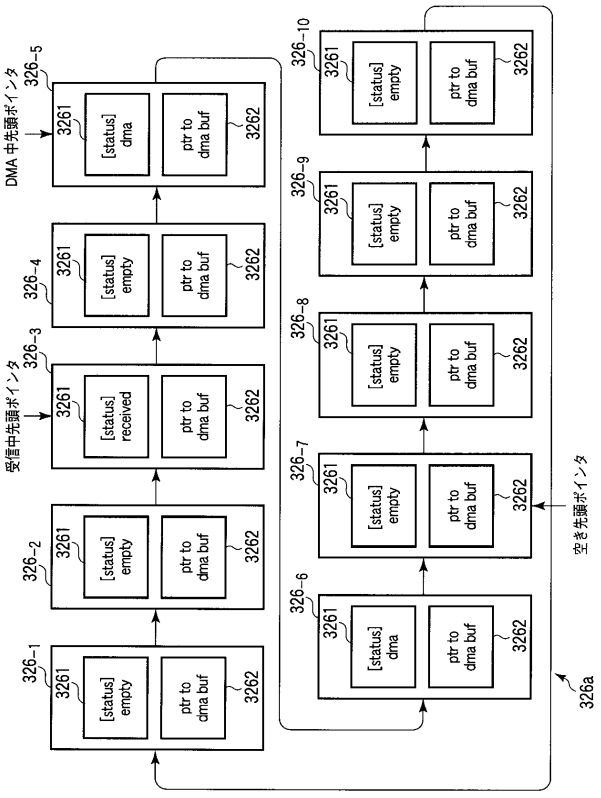
【図３】

図３



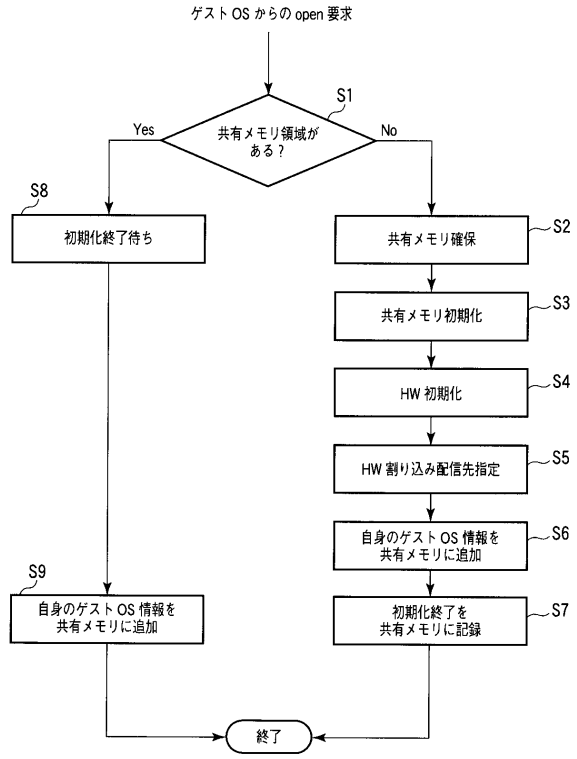
【図４】

図４



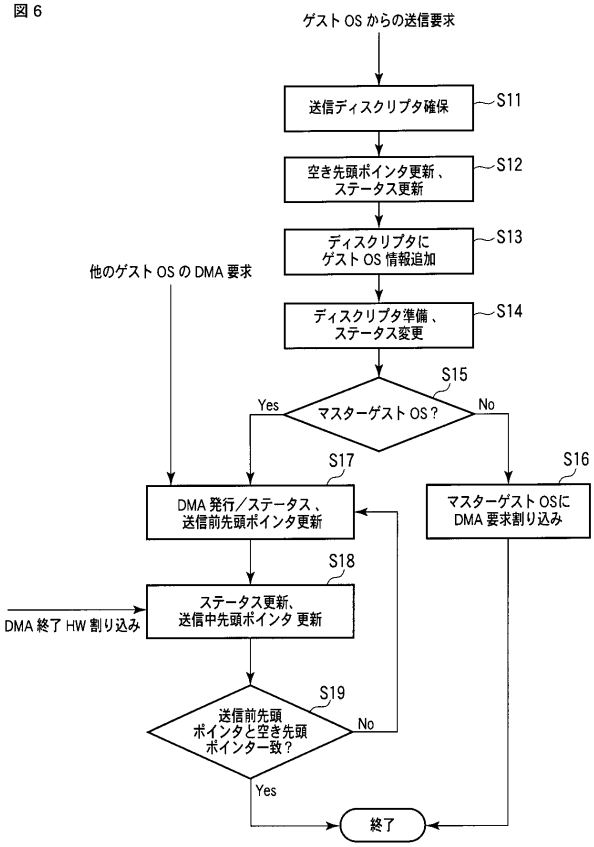
【図５】

図５



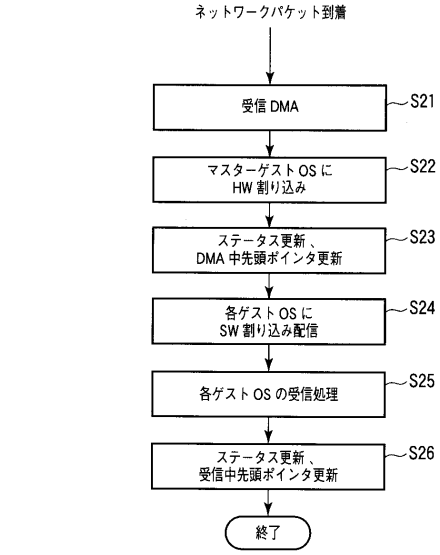
【図６】

図６



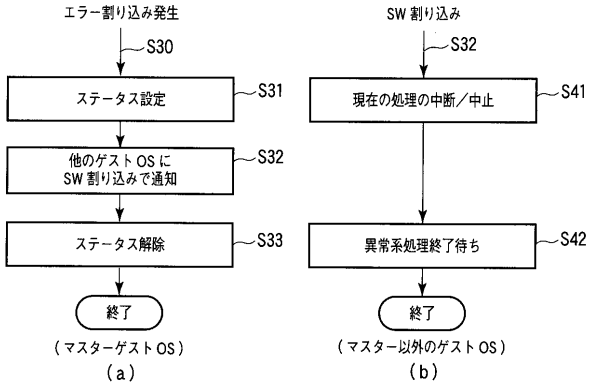
【図７】

図７



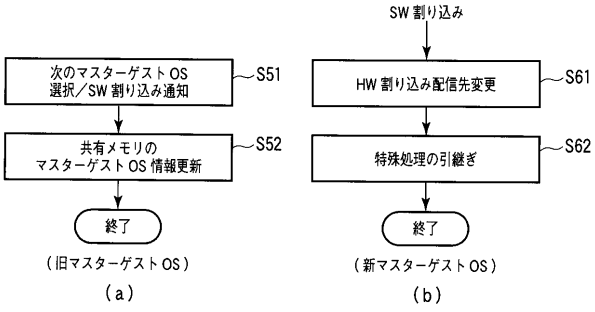
【図８】

図８



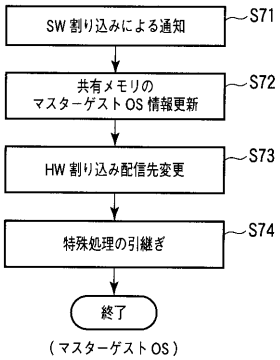
【図９】

図９



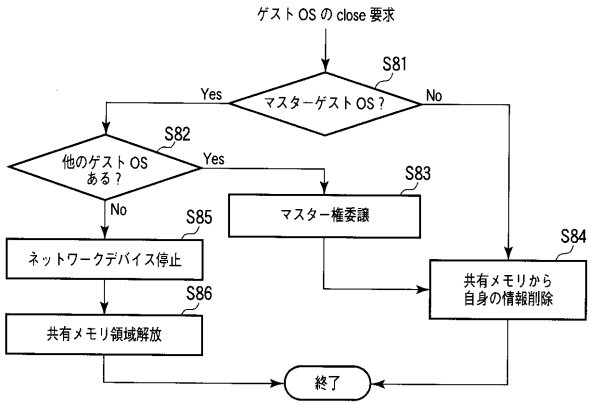
【図１０】

図 10



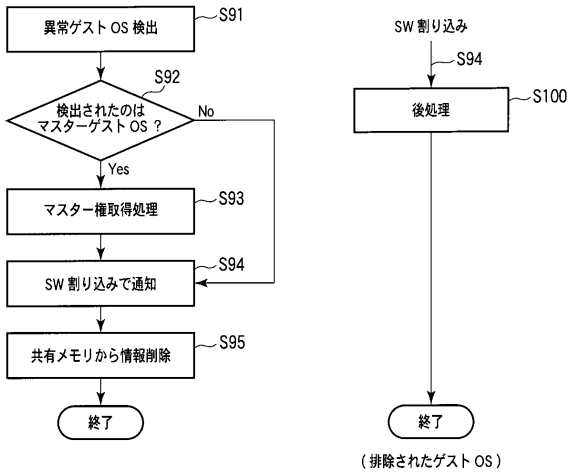
【図１１】

図 11



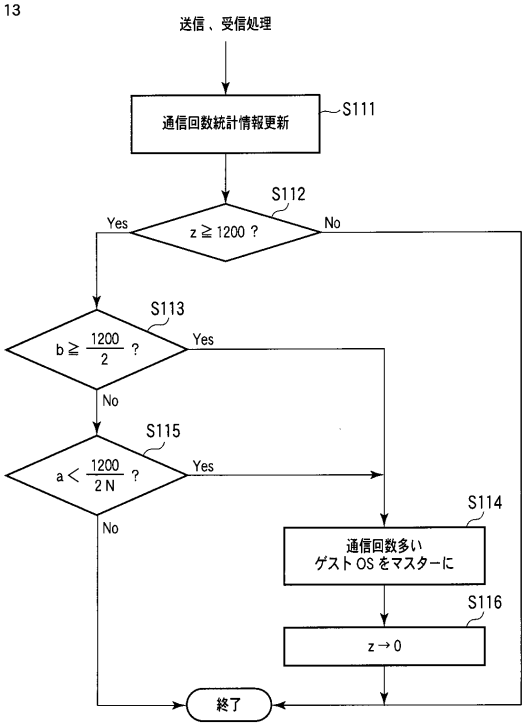
【図１２】

図 12



【図１３】

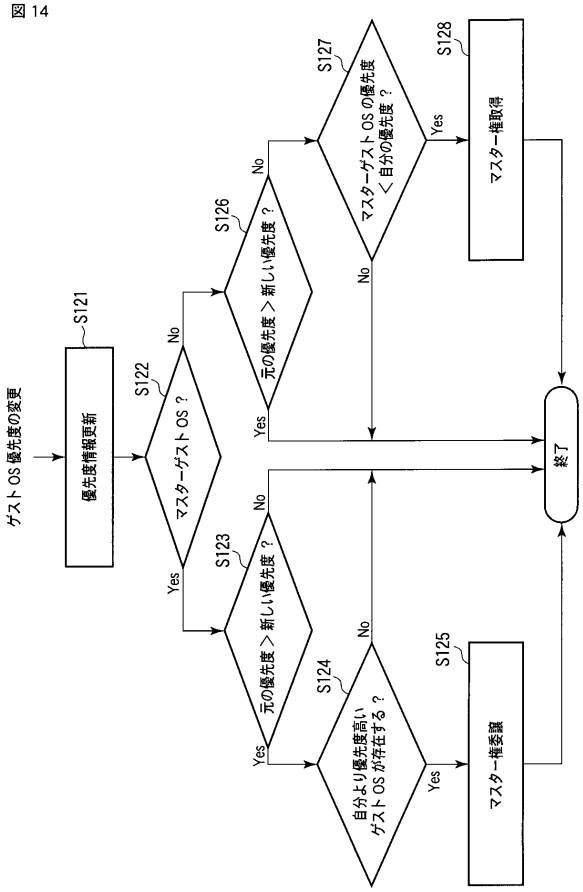
図 13



z：各ゲスト OS の通信回数の合計
N：ネットワークデバイスを共有するゲスト OS の数
a：マスターゲスト OS の通信回数
b：その他ゲスト OS の通信回数のうちの最大値

【図１４】

図 14



フロントページの続き

(74)代理人 100109830
弁理士 福原 淑弘
(74)代理人 100075672
弁理士 峰 隆司
(74)代理人 100095441
弁理士 白根 俊郎
(74)代理人 100084618
弁理士 村松 貞男
(74)代理人 100103034
弁理士 野河 信久
(74)代理人 100119976
弁理士 幸長 保次郎
(74)代理人 100153051
弁理士 河野 直樹
(74)代理人 100140176
弁理士 砂川 克
(74)代理人 100100952
弁理士 風間 鉄也
(74)代理人 100101812
弁理士 勝村 紘
(74)代理人 100070437
弁理士 河井 将次
(74)代理人 100124394
弁理士 佐藤 立志
(74)代理人 100112807
弁理士 岡田 貴志
(74)代理人 100111073
弁理士 堀内 美保子
(74)代理人 100134290
弁理士 竹内 将訓
(74)代理人 100127144
弁理士 市原 卓三
(74)代理人 100141933
弁理士 山下 元
(72)発明者 高山 雅陽
東京都港区芝浦一丁目１番１号 東芝ソリューション株式会社内