第Ⅴ章　ローカルエリアネットワーク
（Ｌ　Ａ　Ｎ）

第Ⅴ章　ローカルエリアネットワーク（LAN）

学習目標

1．LANを構成するための基本的な技術を理解させる。
2．LANの基本的な利用技術を理解させる。

内容のあらまし

節　・　項	内　容
1．LANの概要	
（1）LANとは	LANというデータ通信システムの概要
（2）通信網の発達	
（3）コンピュータシステムの発達	
（4）WAN化への過程	
（5）LAN導入による効果	
2．LANの基本的な技術	
（1）基本的な対話（通信）のスタイル	LANを構成するための基本的な技術
（2）LANと通信プロトコル	
（3）ネットワーク通信のプロトコル群（例）	
（4）トポロジーと物理形状	
（5）メディアアクセス方式	
（6）LANの構成要素	
（7）LANの代表的な応用例	
3．LAN間接続とインターネットワーキング	
（1）インターネットワークとは	LAN間接続に関わる基本的な技術と実用化されている
（2）LAN間接続装置とインターネットワーク	代表的なLAN間接続の通信プロトコルの概要
（3）TCP／IPによるインターネットワーキング	
（4）ルーターによるインターネットワーキング	
4．ネットワーグ・プログラミングとAPI	
（1）ソケットインタフェース	実用化されている代表的なLAN間接続通信プロトコル
（2）ソケットインタフェース関連システムコール	（TCP／IP）の基本的な技術とその実例
（3）ネットワークプログラムの動作環境	
（4）ネットワークプログラミング例	

1．LANの概要

（1）LANとは

LAN：Local Area Network

日本では「企業内・地域情報通信網」と訳されている。

L（Local）　とは・・・ある特定の限られた。

A（Area）　とは・・・地域で

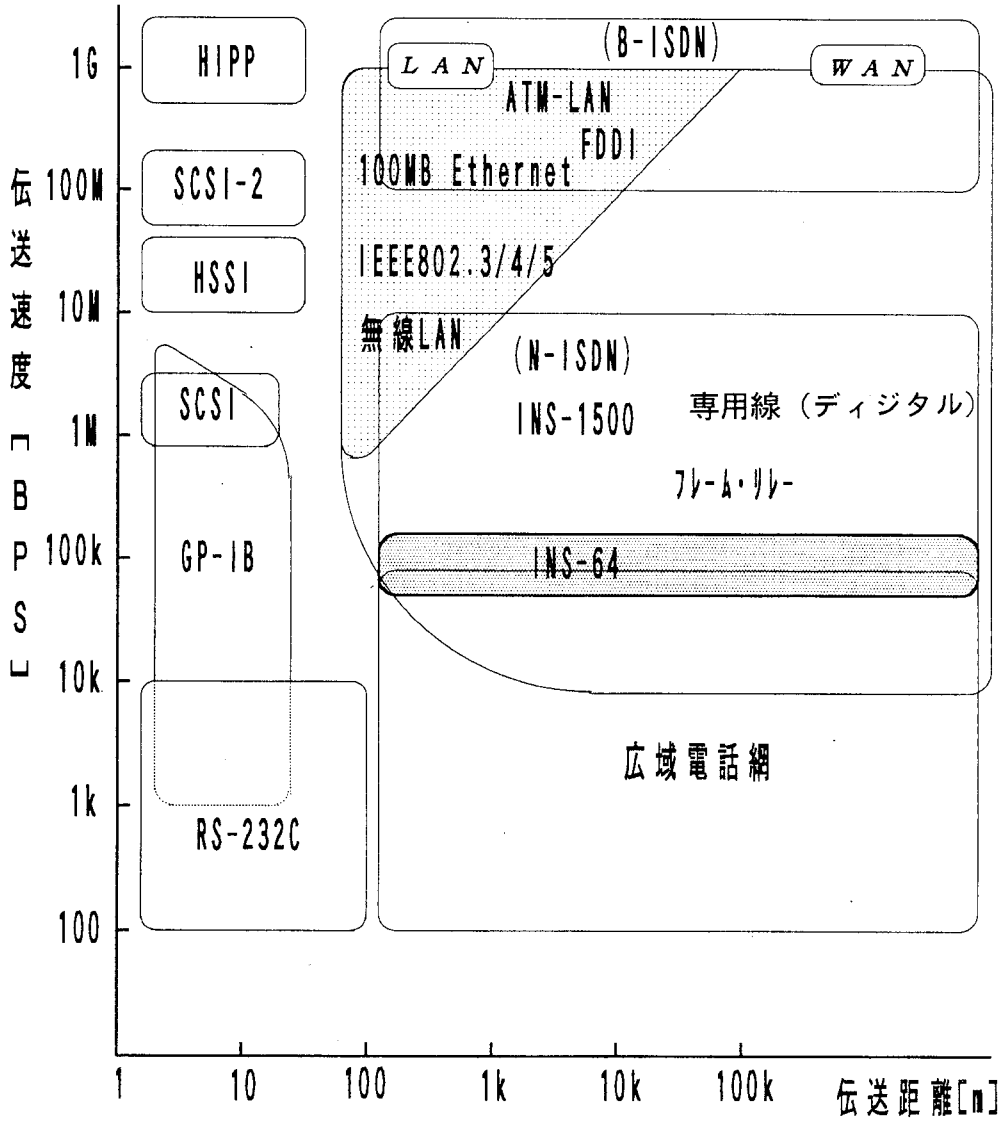
N（Network）とは・・・データなどを相互に通信する網

これに対して

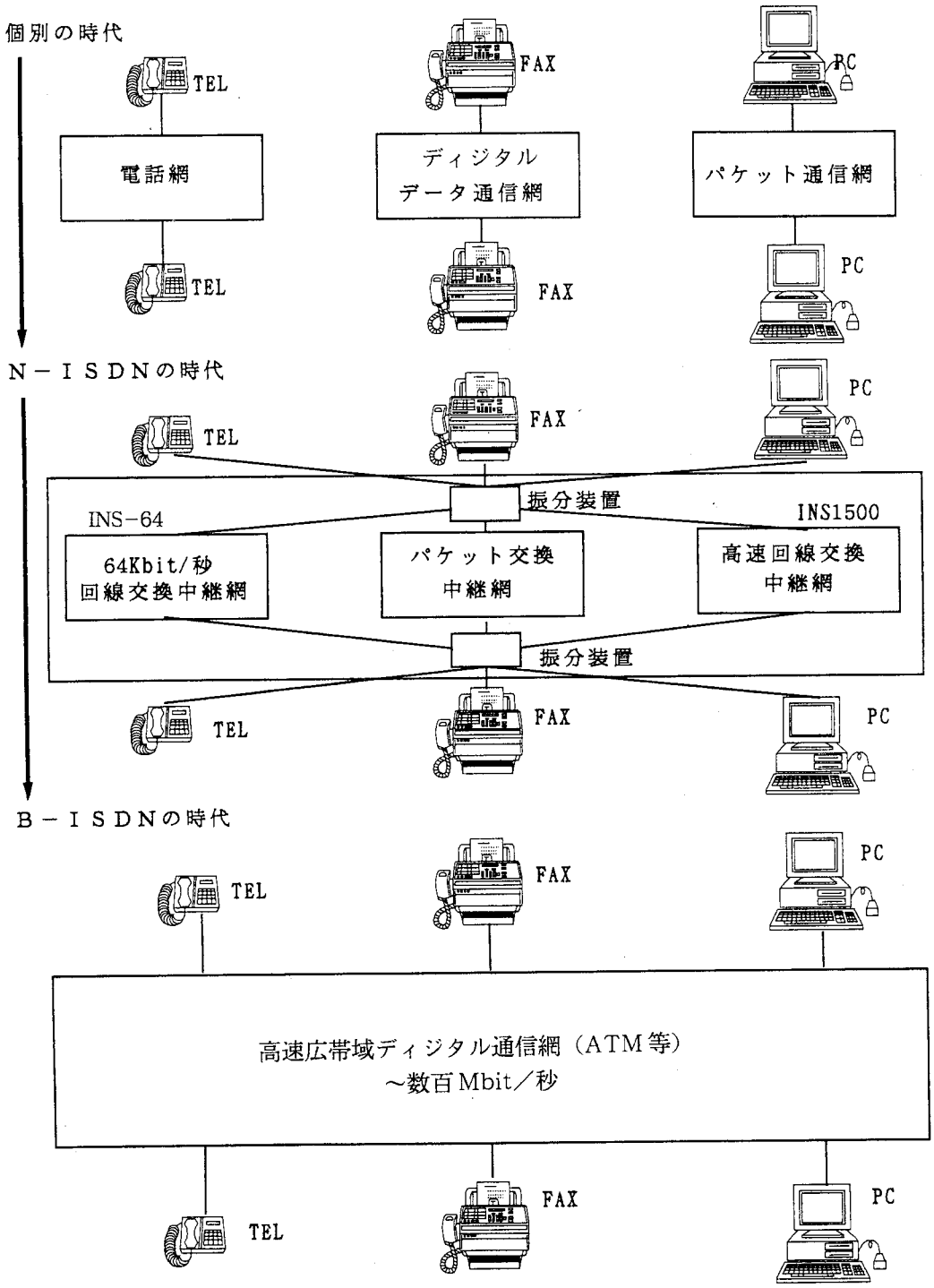
広範囲な地域（地域を限らない。）でデータ（情報）などを相互に通信する網

WAN：Wide Area Network

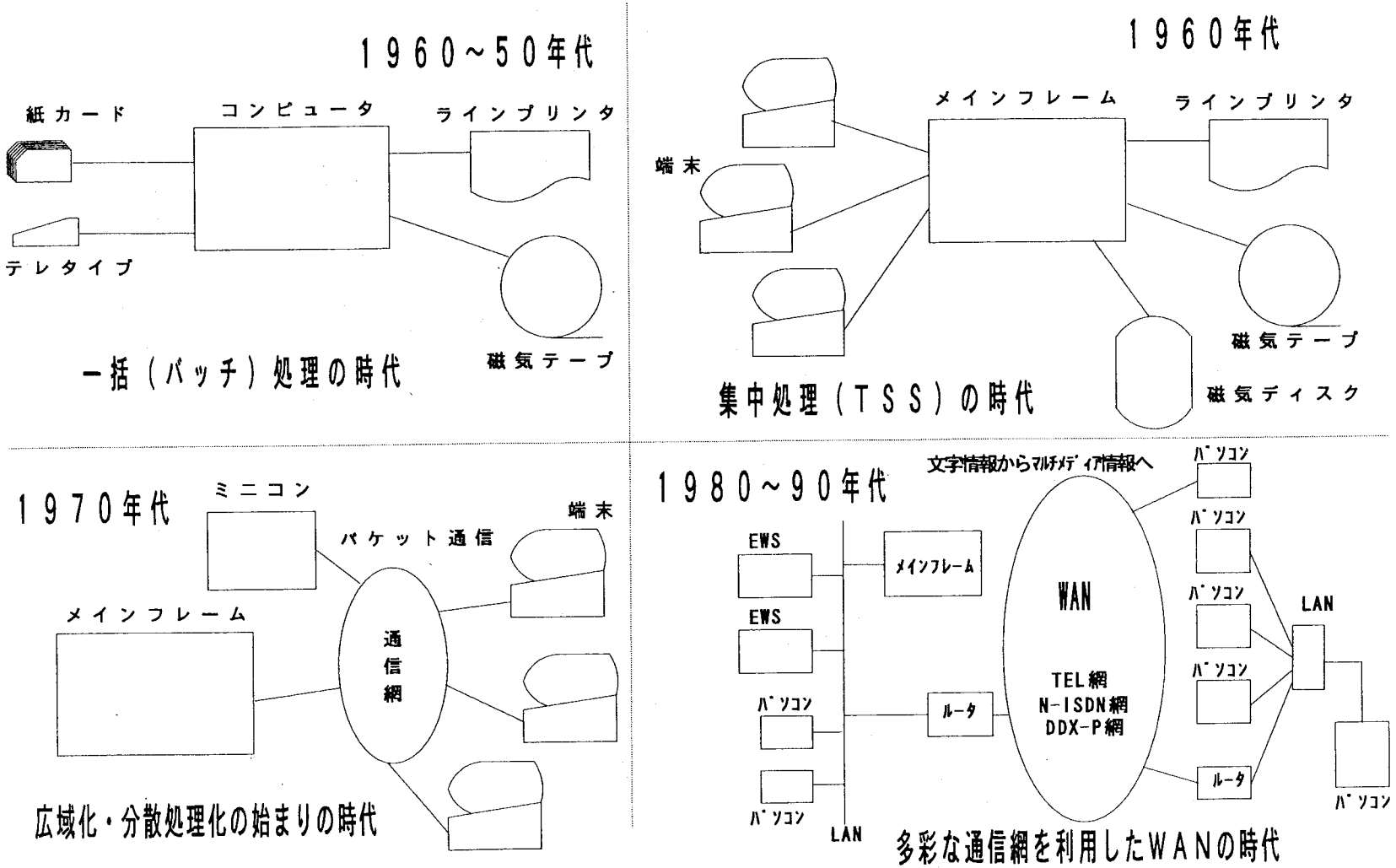
通信媒体（網）とLAN／WAN



（２） 通信網の発達


個別の時代
TEL
FAX
PC
電話網
ディジタル
データ通信網
パケット通信網
TEL
FAX
PC
N－ＩＳＤＮの時代
TEL
FAX
PC
振分装置
INS−64
INS1500
64Kbit/秒
回線交換中継網
パケット交換
中継網
高速回線交換
中継網
振分装置
TEL
FAX
PC
Ｂ－ＩＳＤＮの時代
TEL
FAX
PC
高速広帯域ディジタル通信網（ATM等）
～数百Mbit／秒
TEL
FAX
PC

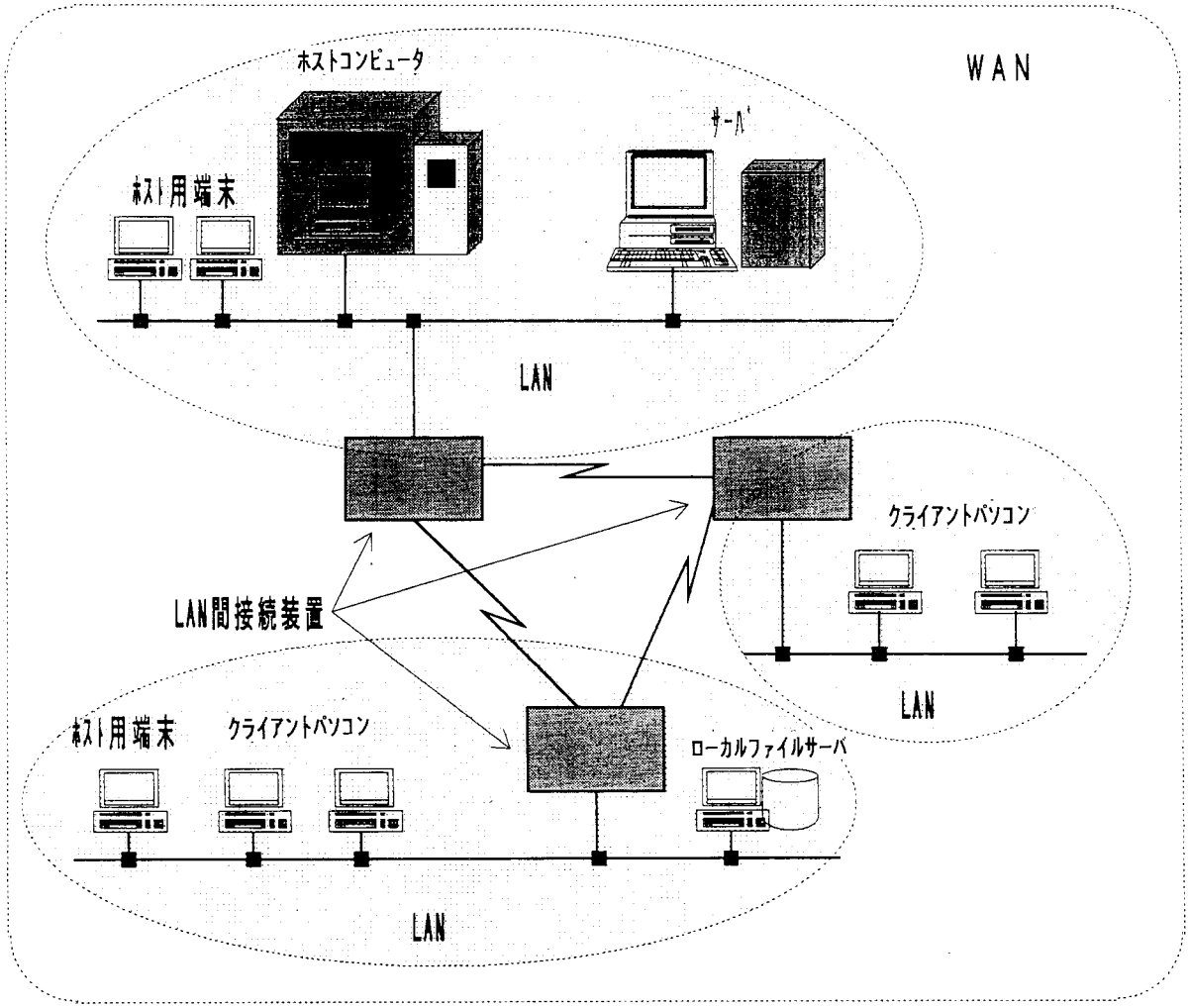
(3)コンピュータシステムの発達



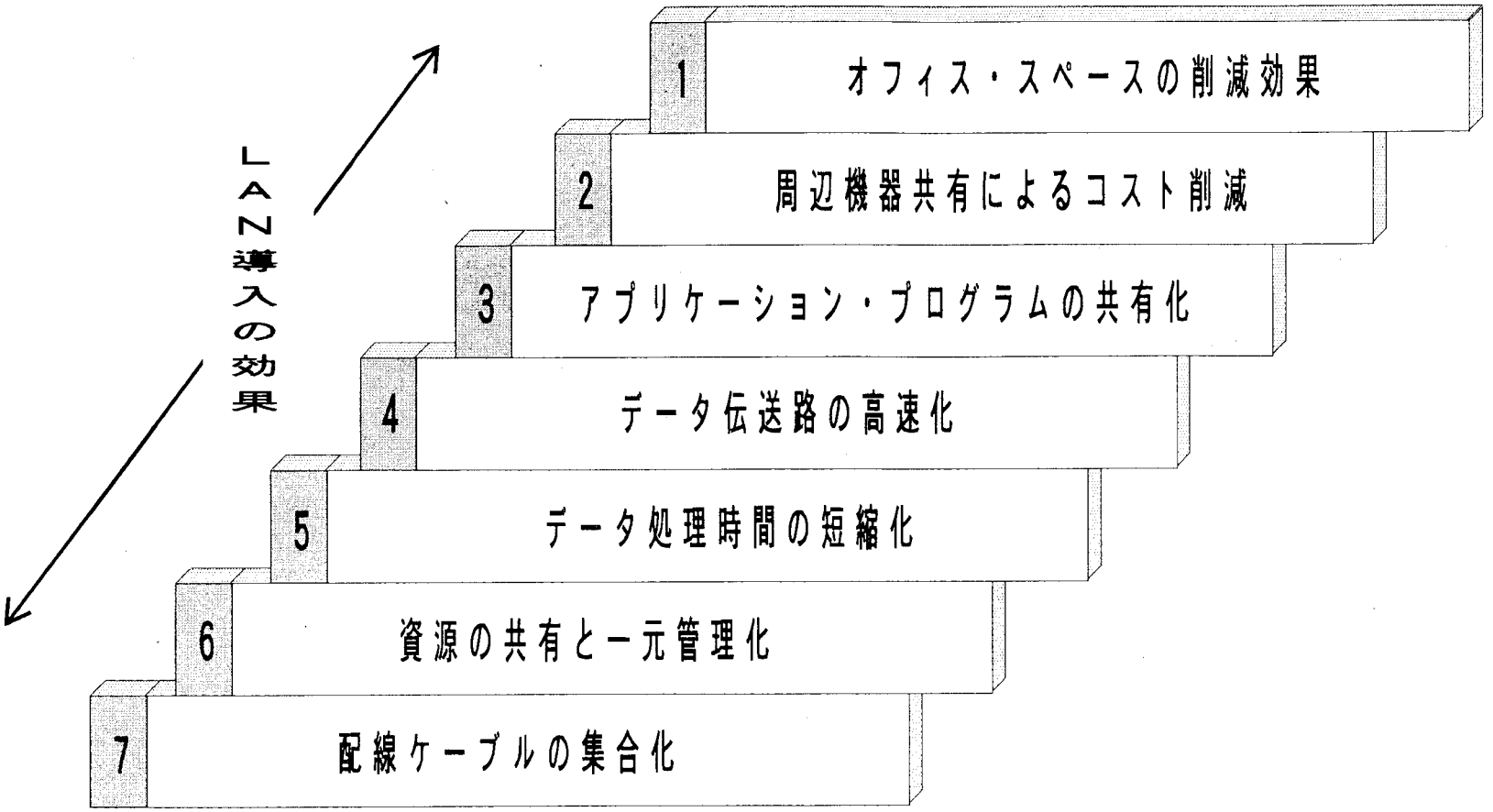
（４） WAN 化への過程

● 拠点レベルで構築したＬＡＮを統合（ＬＡＮサーバの共有化）

● ホストコンピュータと端末の配線統合の結果として

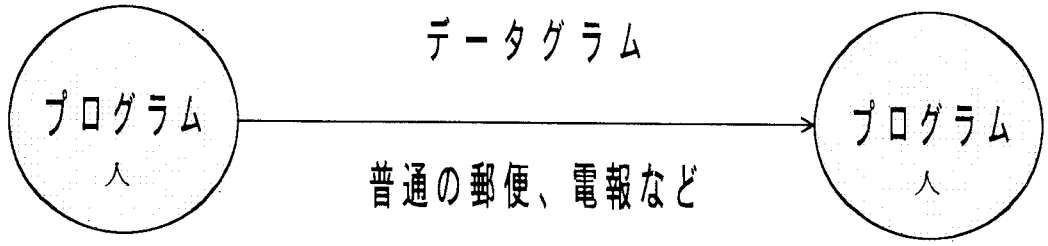


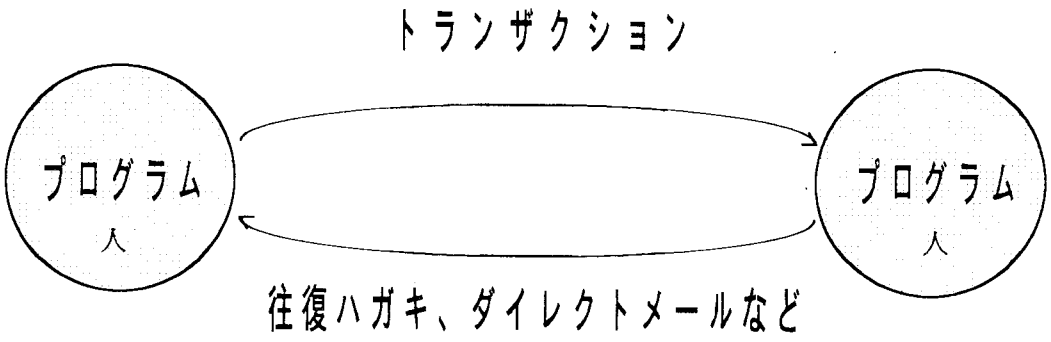
（５）ＬＡＮ導入による効果

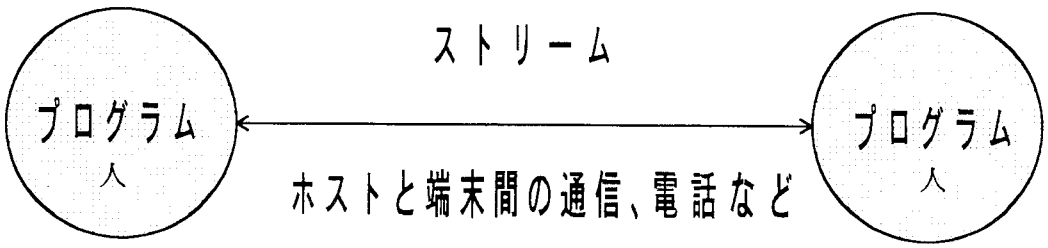


2．ＬＡＮの基本的な技術

（1）基本的な対話（通信）のスタイル

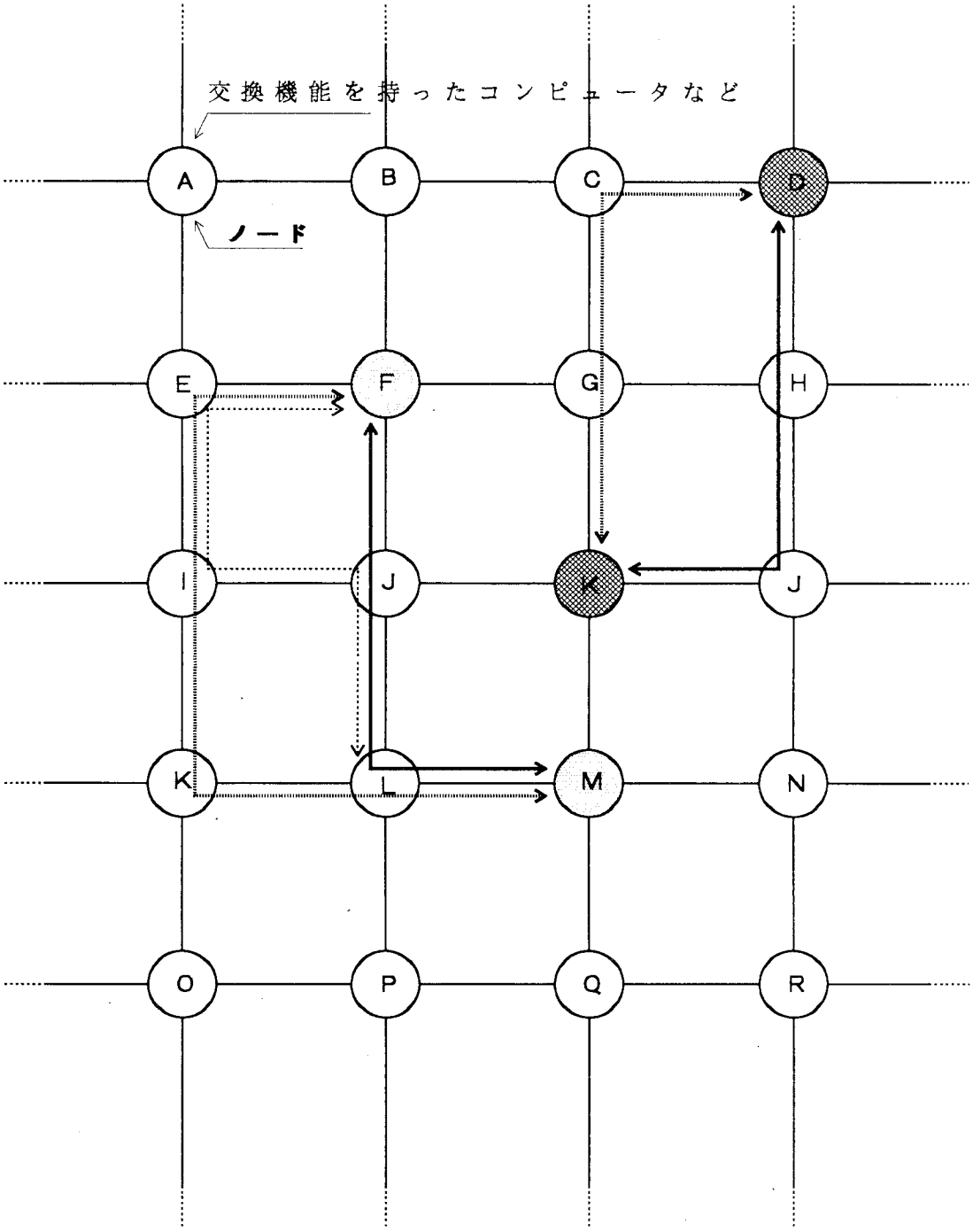




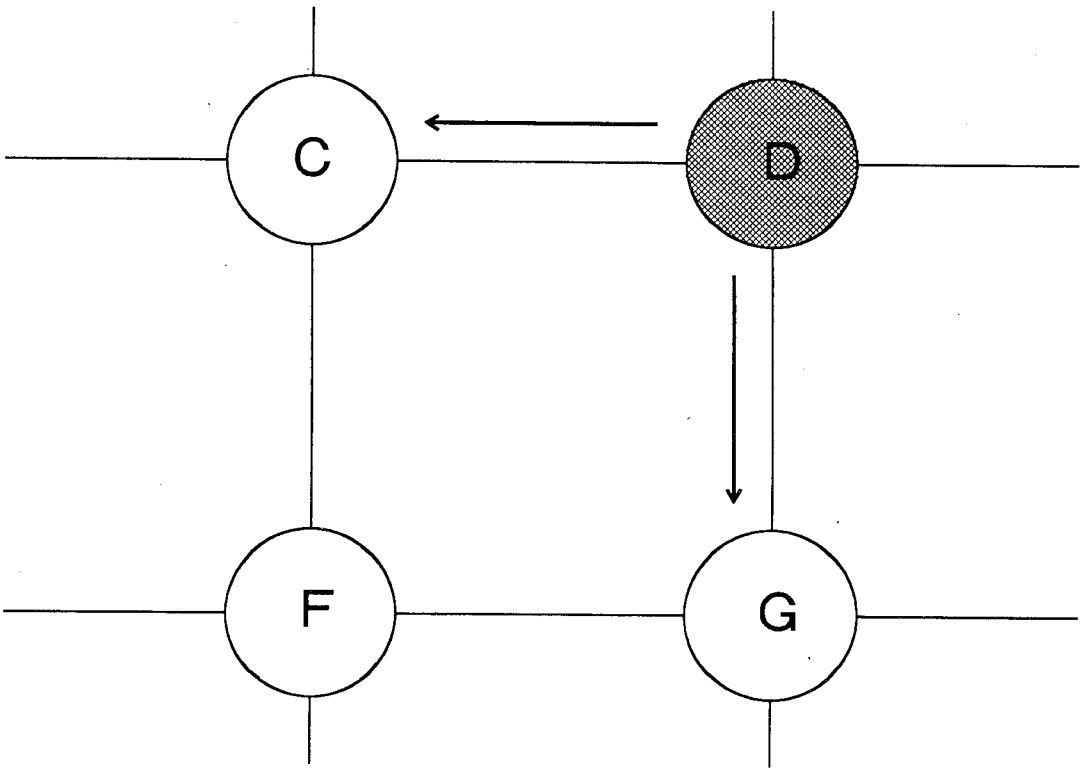


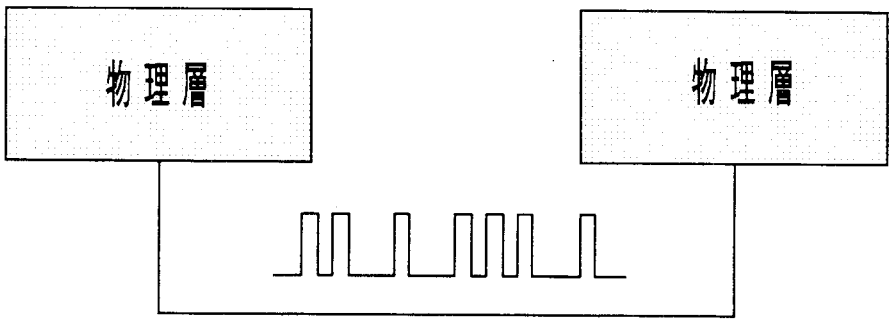
（2） LANと通信プロトコル

ネットワーク上に配置されたコンピュータ間で相互通信するためには通信通プロトコルを断層化してEND－END間の通信を行う。



① 隣接するノード間のデータ伝送を確実に行う。
①－１ビット列の送受信を実現する。

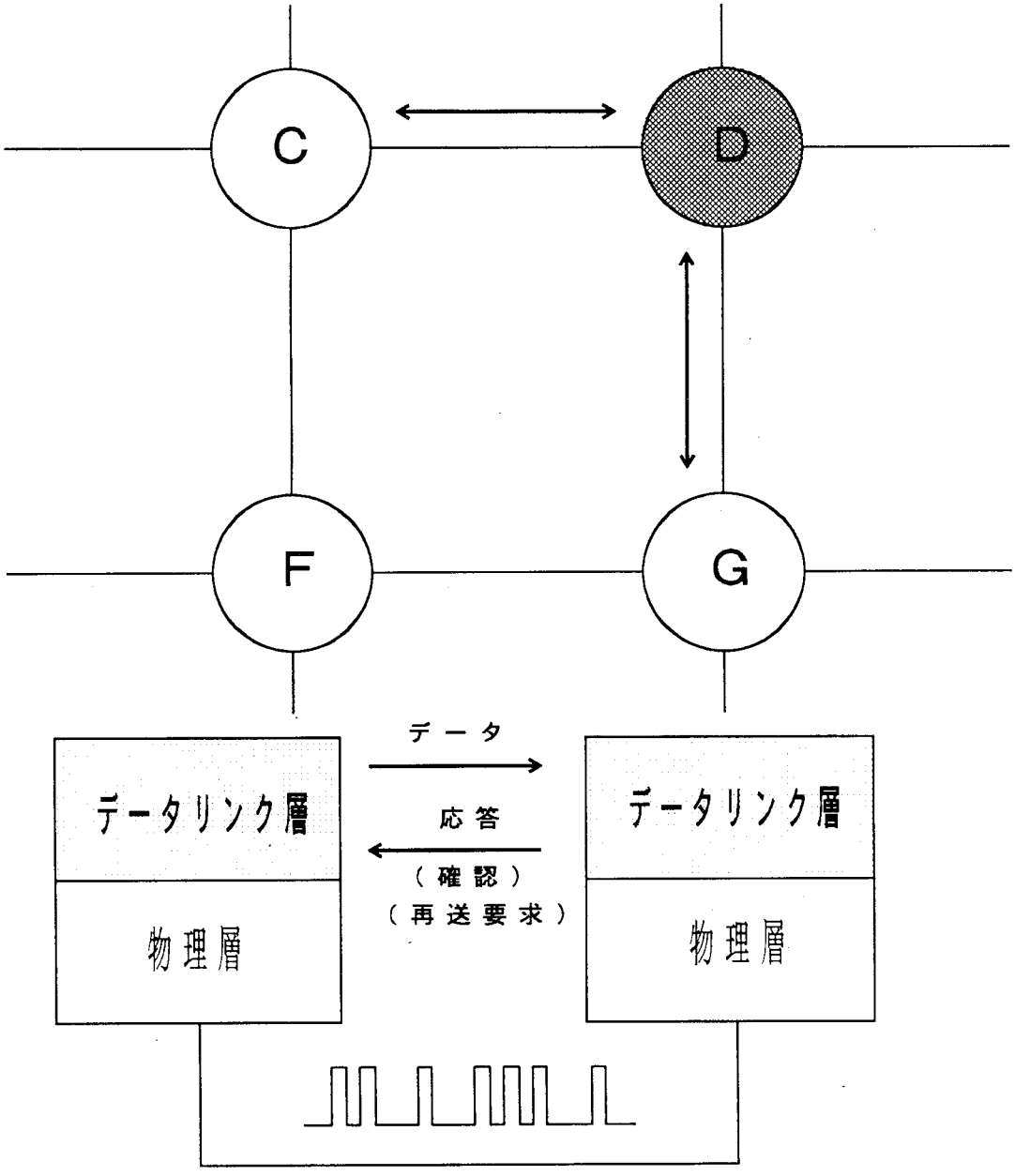




電気的・物理的な仕様を合わせる。
最低限のデータ通信を可能にする。

①－②　隣接するノード間のデータ伝送を確実に行う。

ビット列の正確な伝送を実現する。

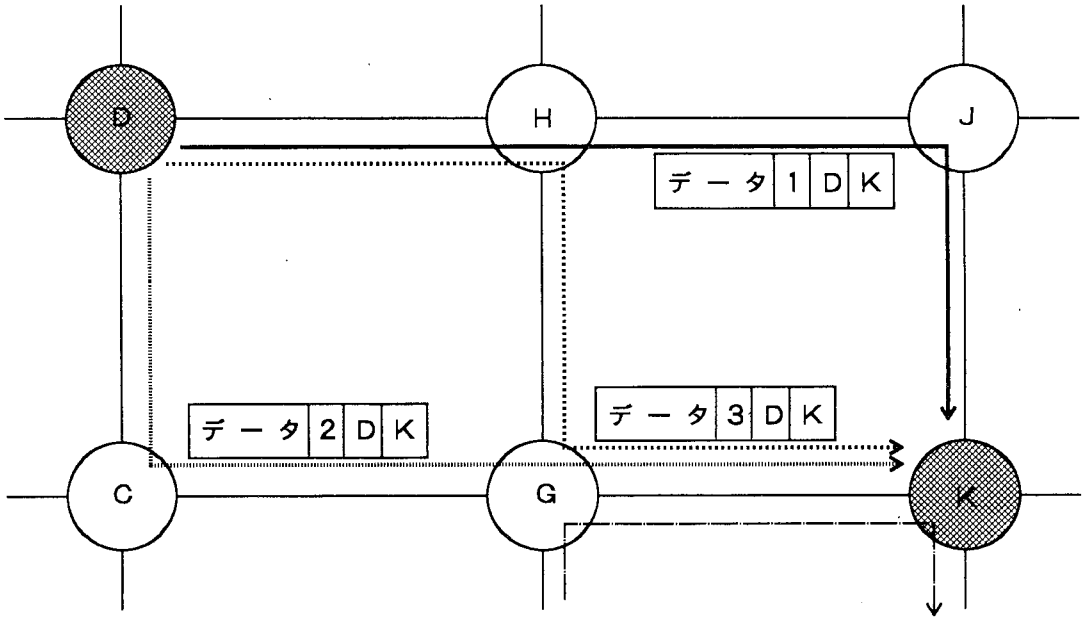


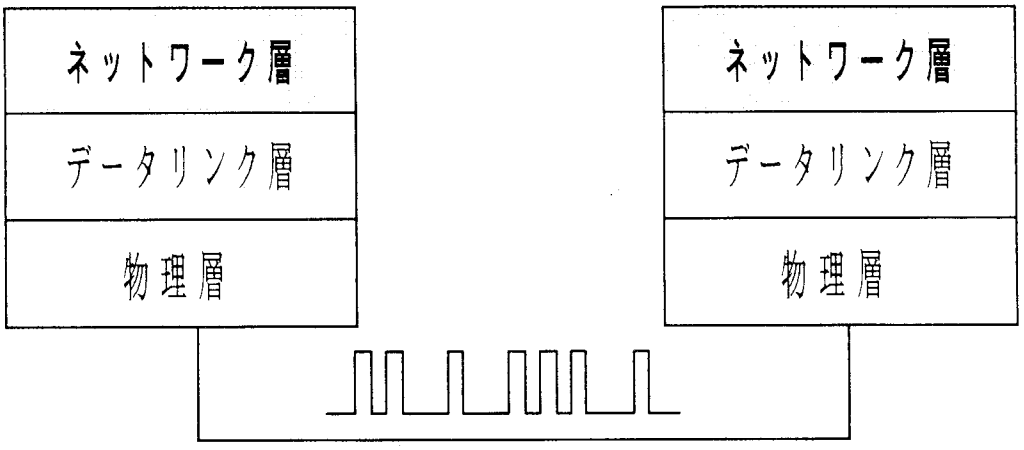
伝送誤りを回復しつつ、より正確な伝送を行う。

送達確認機能、再送機能、ビジー制御機能等

② END to ENDでの相互通信を実現する。

②－1 ネットワークを経由した伝送を実現する。

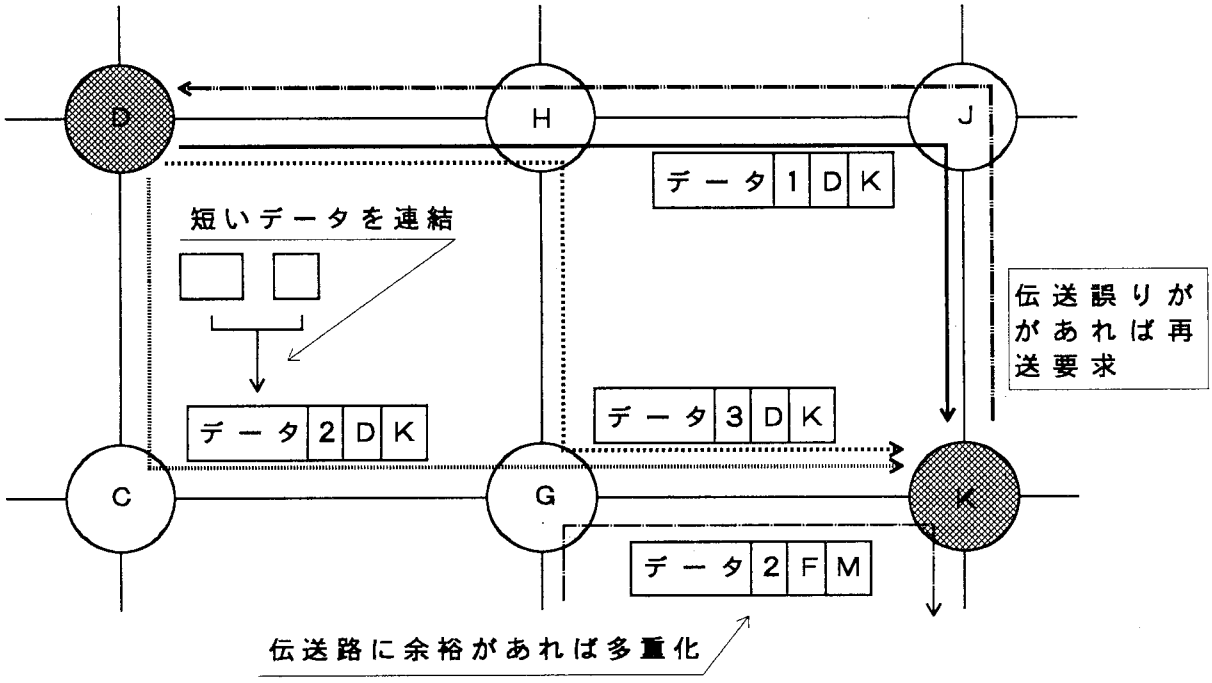


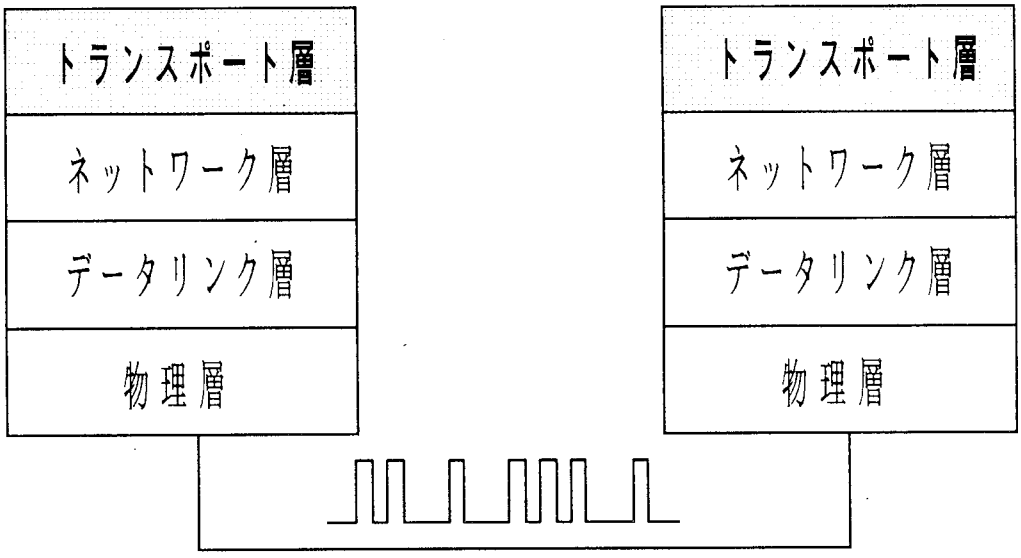


ルート制御機能、順序制御機能

フロー制御機能・END間の送達確認機能

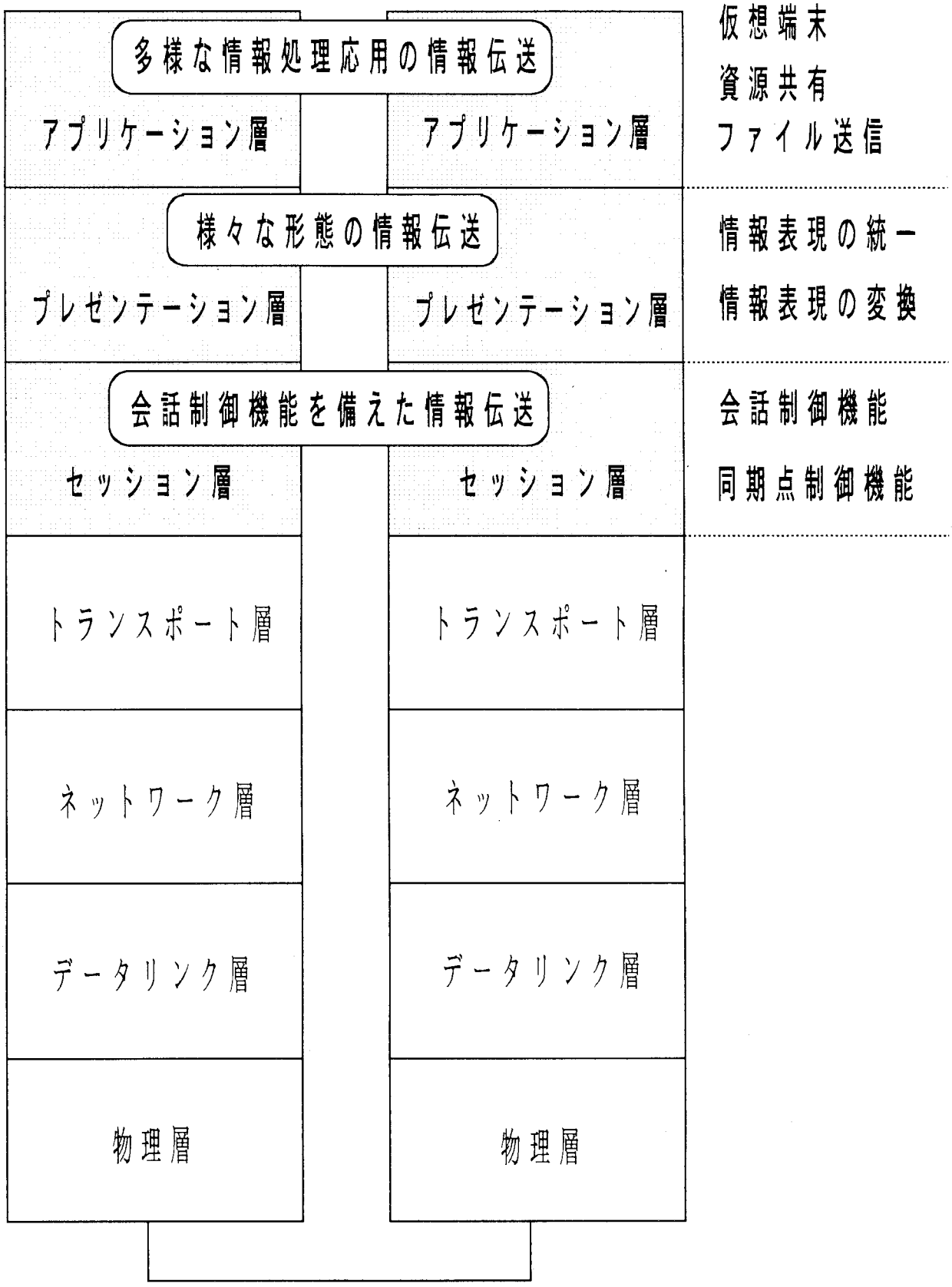
②－２　利用者の要求にかなう伝送品質・性能を実現する。





多重化・分流制御機能、連結制御機能
送達確認・再送制御認機能

③　利用者間レベルのメッセージ伝送や資源の共用等を実現する。
OSI参照モデルにおいては、通信プロトコルを７階層に分けている。



<table>
<tr><th></th><th colspan="4">OSI</th><th colspan="3">TCP／IP</th><th colspan="2">ONA（Sun）</th><th colspan="3">DECnet</th><th>SNA</th></tr>
<tr><td rowspan="3">ｱﾌﾟﾘｹｰｼｮﾝ層</td><td colspan="4">FTM/VT/TP
OTAM/MHS(X.400)/
ﾃﾞｨﾚｸﾄﾘ(X.500)/JTM/
ODA-ODIF-CMIS/CMP</td><td colspan="3" rowspan="3">FTP/
TELNET/
SMTP/
DNS/SNMP/

Berkeleyｻｰﾋﾞｽ
rcp,rlogin,rsh,
rexec

NFS</td><td colspan="2" rowspan="3">Sunﾈｯﾄﾜｰｸ･ｻｰﾋﾞｽ:
NFS,NLM,REX,NIS,
Automounter等

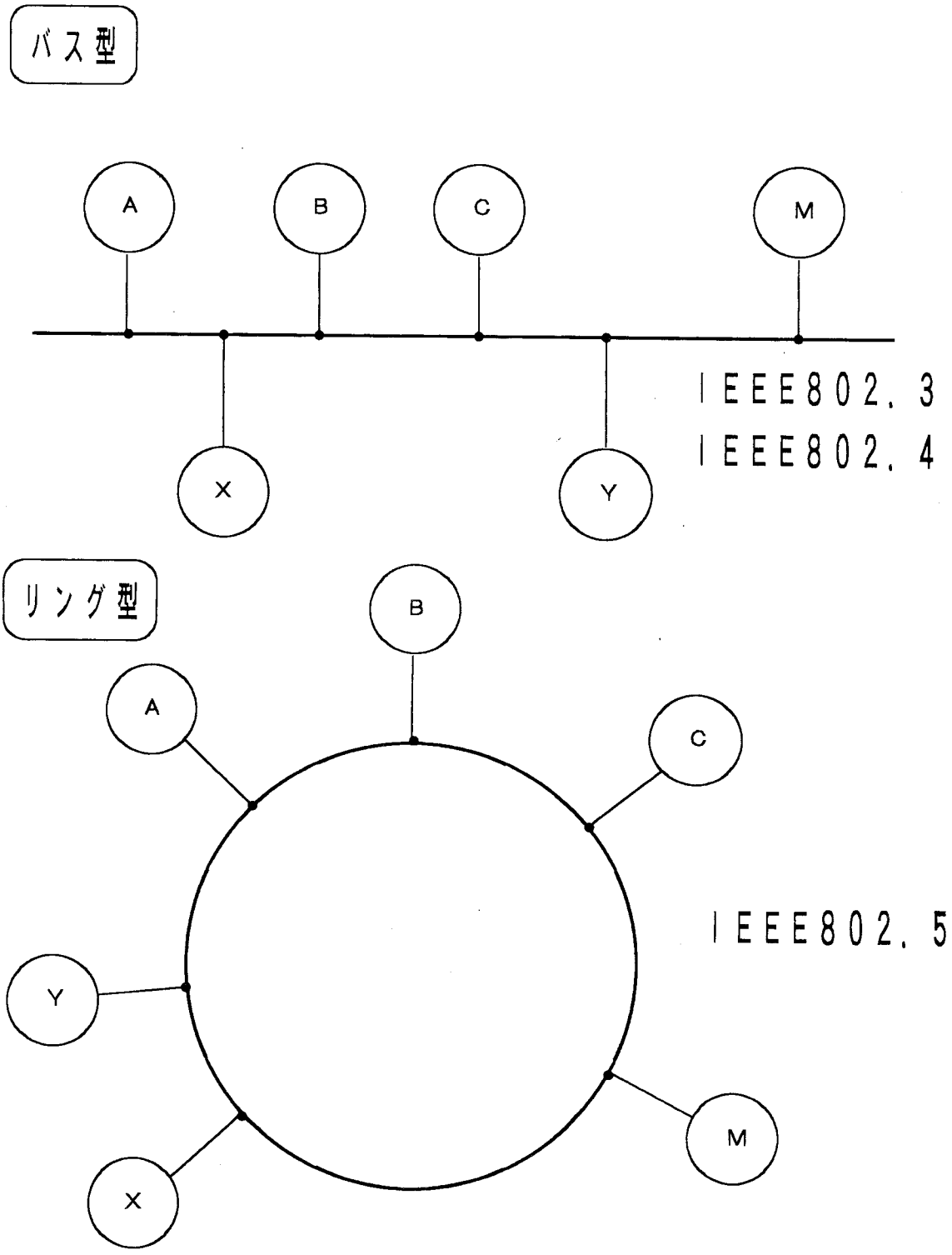
Berkeleyｻｰﾋﾞｽ
rcp,rlogin,rsh,</td><td colspan="3" rowspan="3">DECnetｱﾌﾟﾘｹｰｼｮﾝ:
ﾌｧｲﾙ転送,仮想端末,
ﾘﾓｰﾄ･ﾌｧｲﾙ･ｱｸｾｽ等

OSIｱﾌﾟﾘｹｰｼｮﾝ:
FTAM,MHS,CMIP等

TCP/IPｱﾌﾟﾘｹｰｼｮﾝ:
FTP,TELNET,NFS,
SNMP等</td><td>ﾄﾗﾝｻﾞｸｼｮﾝ･ｻｰﾋﾞｽ</td></tr>
<tr><td colspan="4">RTS/ROS/CCR</td><td></td></tr>
<tr><td colspan="4">ACSE</td><td></td></tr>
<tr><td>ﾌﾟﾚｾﾞﾝﾃｰｼｮﾝ層</td><td colspan="2">ASN.1
ｺﾈｸｼｮﾝﾚｽ型</td><td colspan="2">符号化規則
ｺﾈｸｼｮﾝ型</td><td colspan="2">XDR</td><td>ﾌﾟﾚｾﾞﾝﾃｰｼｮﾝ･ｻｰﾋﾞｽ</td></tr>
<tr><td>ｾｯｼｮﾝ層</td><td colspan="2">ｺﾈｸｼｮﾝﾚｽ型</td><td colspan="2">ｺﾈｸｼｮﾝ型</td><td colspan="2">NETBIOS</td><td>ｿｹｯﾄ･ｲﾝﾀｰﾌｪｰｽ</td><td colspan="2">RCP</td><td colspan="2">ﾈｰﾐﾝｸﾞ･ｻｰﾋﾞｽ</td><td>ﾃﾞｰﾀﾌﾛｰ制御</td></tr>
<tr><td>ﾄﾗﾝｽﾎﾟｰﾄ層</td><td colspan="2">CLTP
ｺﾈｸｼｮﾝﾚｽ型</td><td colspan="2">COTP
ｺﾈｸｼｮﾝ型</td><td colspan="2">TCP</td><td>UDP</td><td>TCP</td><td>UDP</td><td colspan="3">XTI(X/Open Transport Interface)</td><td rowspan="2">伝送制御</td></tr>
<tr><td>ﾈｯﾄﾜｰｸ層</td><td>X.25
PLP
ﾊﾟｹｯﾄ
制御手順</td><td>Dﾁｬﾝﾈﾙ
(ISDN
用)</td><td colspan="2">CLNP
ｺﾈｸｼｮﾝﾚｽ型</td><td>IP</td><td>ARP</td><td>RARP</td><td colspan="2">IP</td><td>DECnet</td><td>OSI
TP:ｸﾗｽ
0,2,4
IS-IS
ES-IS</td><td>TCP/IP</td><td>経路制御</td></tr>
<tr><td>ﾃﾞｰﾀﾘﾝｸ層</td><td>LAPB
/HDLC</td><td>LAPD
(ISDN
用)</td><td colspan="2">LLC

MAC</td><td colspan="3" rowspan="2">Ethernet
IEEE802標準
FDDI
ISDN
X.25 他</td><td colspan="2" rowspan="2">Ethernet
IEEE802.3
IEEE802.4
FDDI 他</td><td rowspan="2">DCMP</td><td colspan="2" rowspan="2">Ethernet
IEEE802.3
FDDI
X.25
ﾌﾚｰﾑﾘﾚｰ等</td><td>ﾃﾞｰﾀﾘﾝｸ制御</td></tr>
<tr><td>物理層</td><td>電話網</td><td>ISDN</td><td colspan="2">LAN</td><td>物理制御</td></tr>
</table>

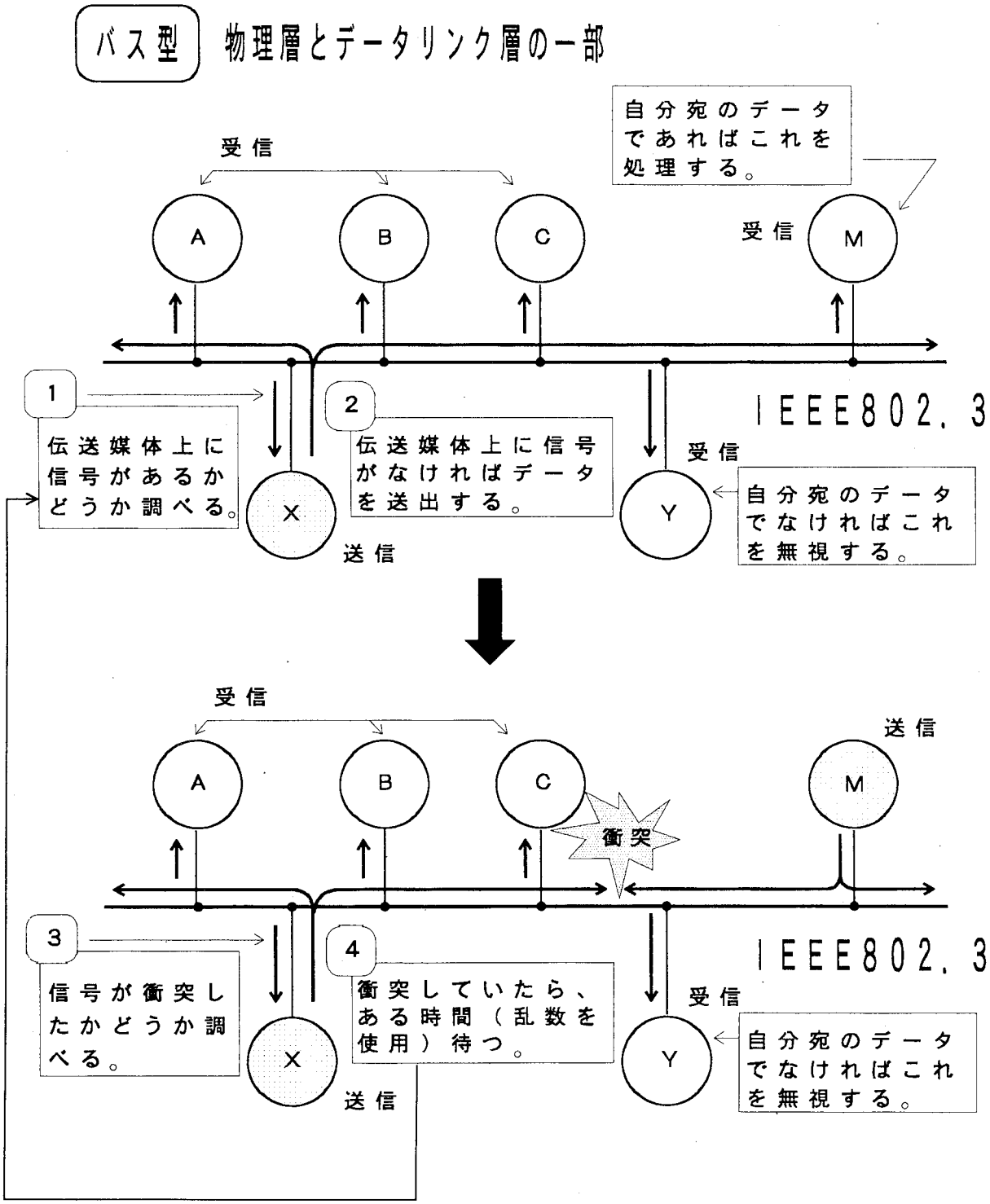
（4） トポロジーと物理形状



（5） メディアアクセス方式

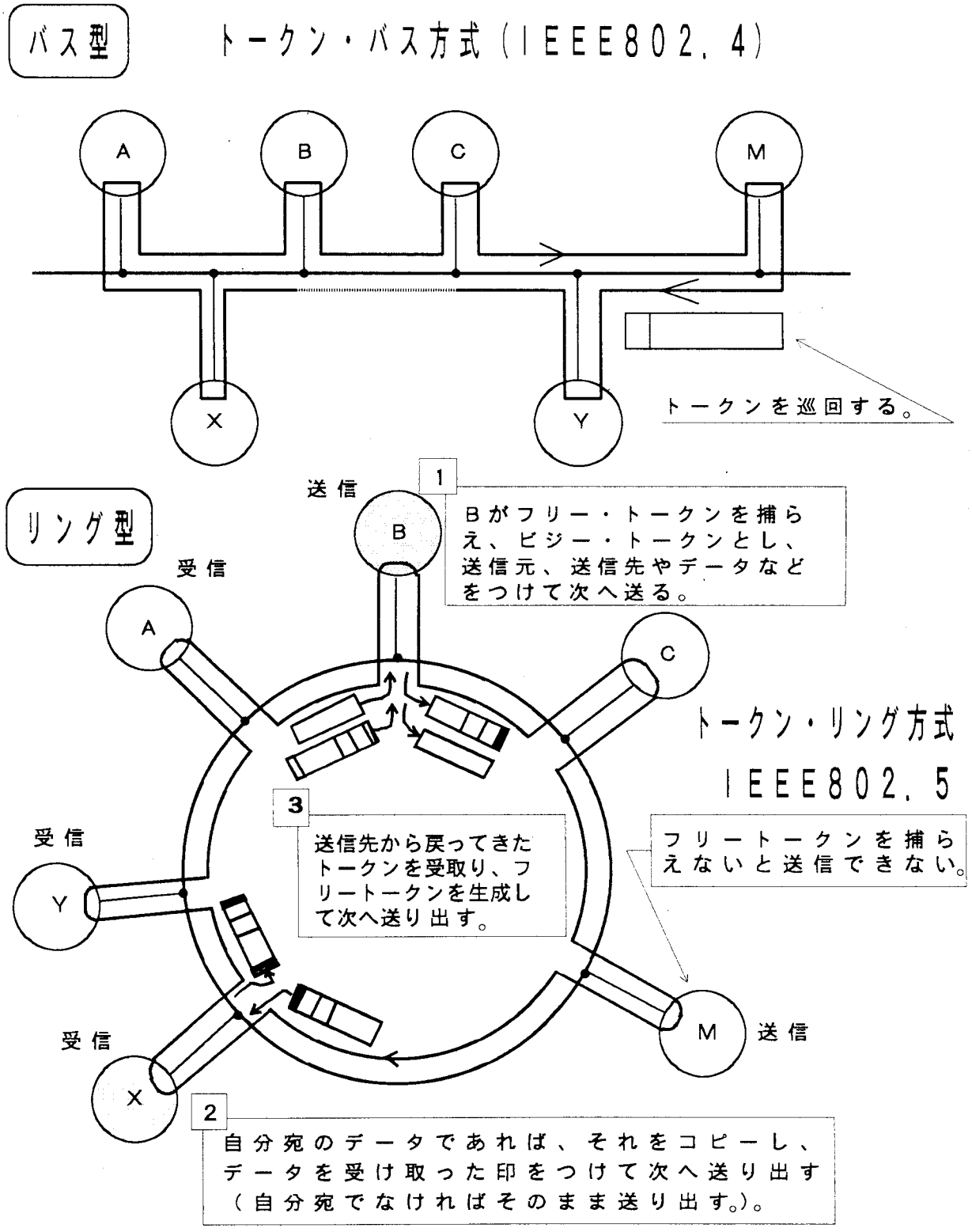
伝送媒体（メディア）アクセス方式（その1）

CSMA／CD

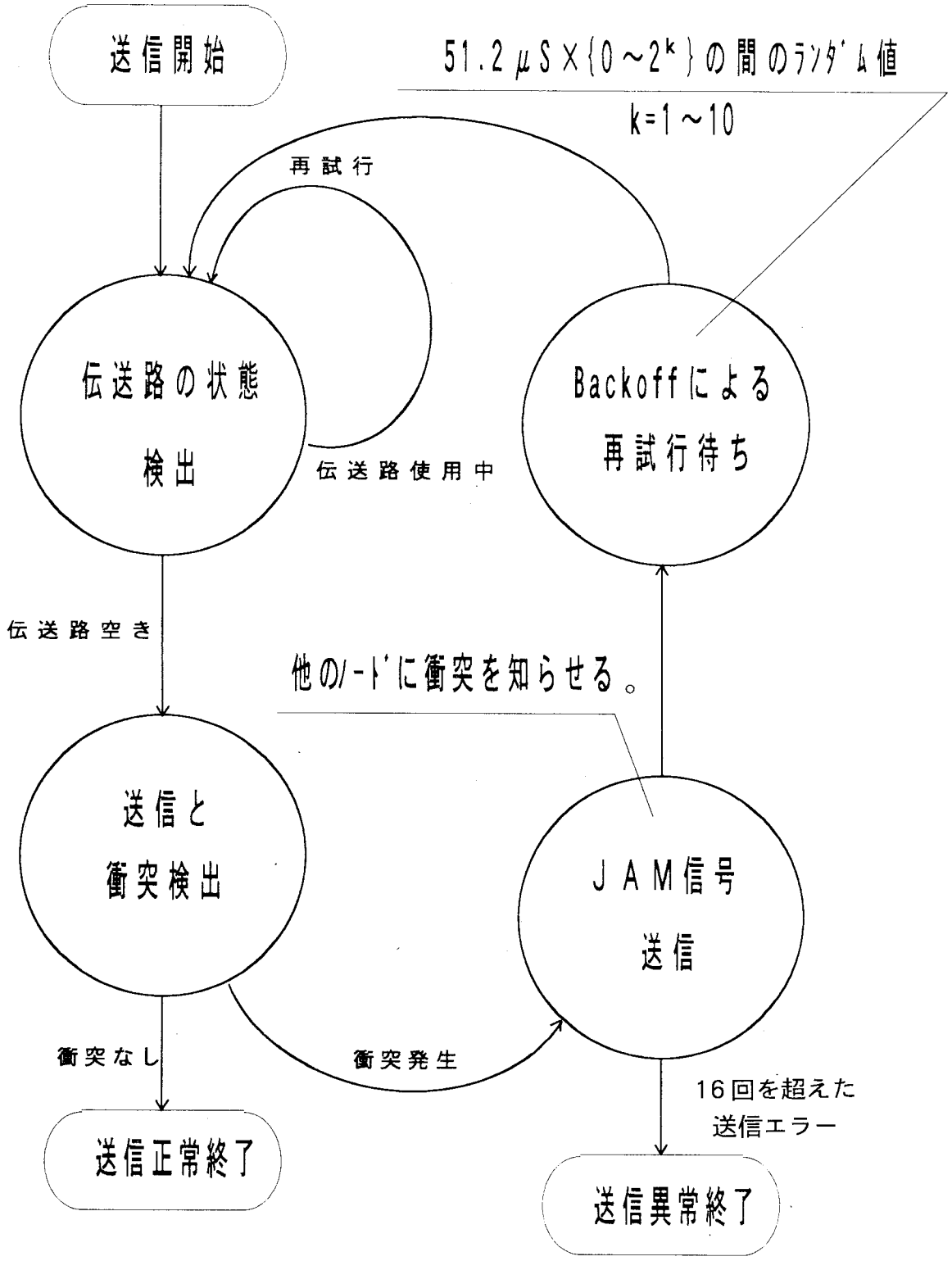


CSMA／CD :Carrier Sense Multiple Access with Collision Detection

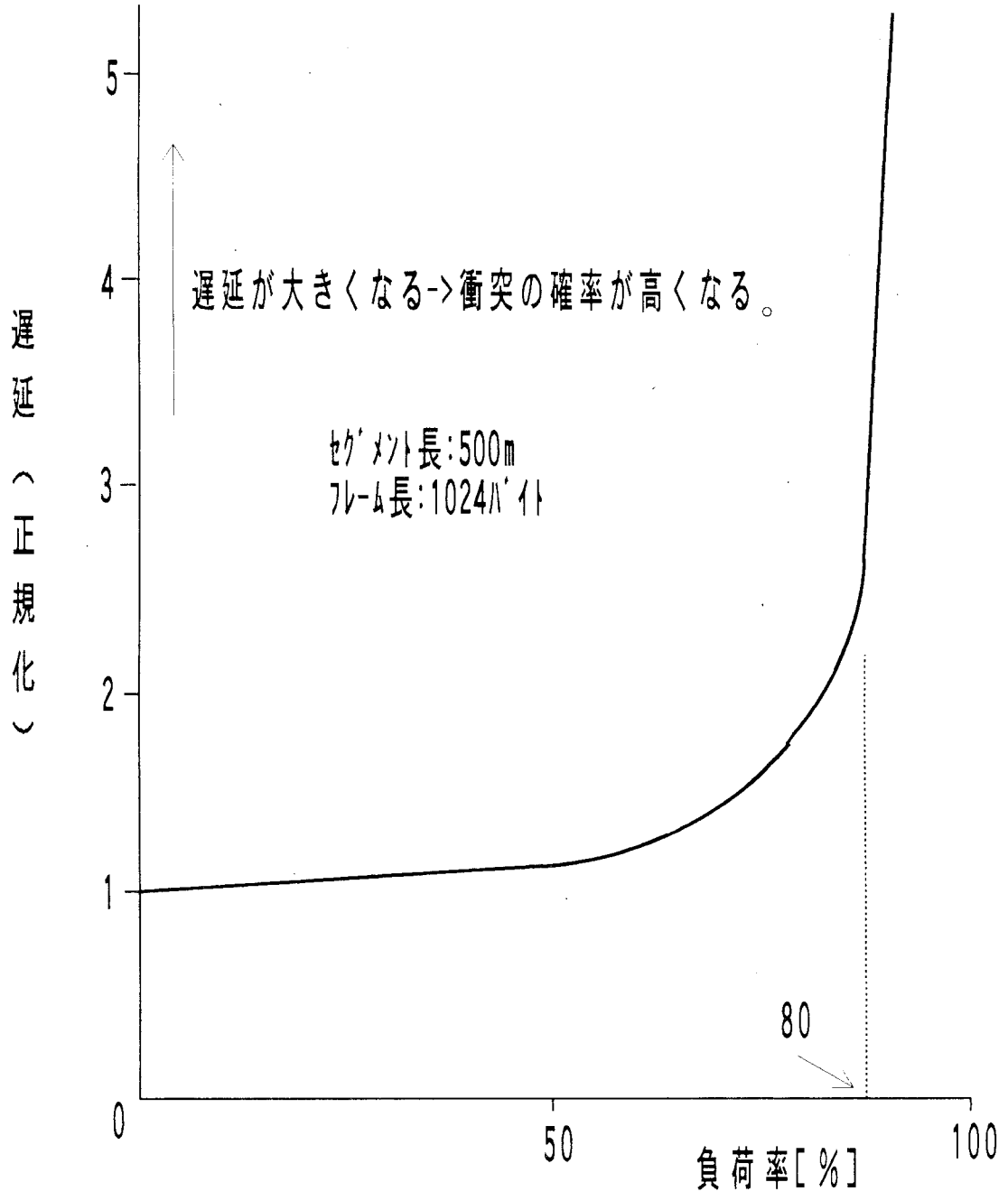
メディアアクセス方式　（その2）



CSMA／CDの動作フロー

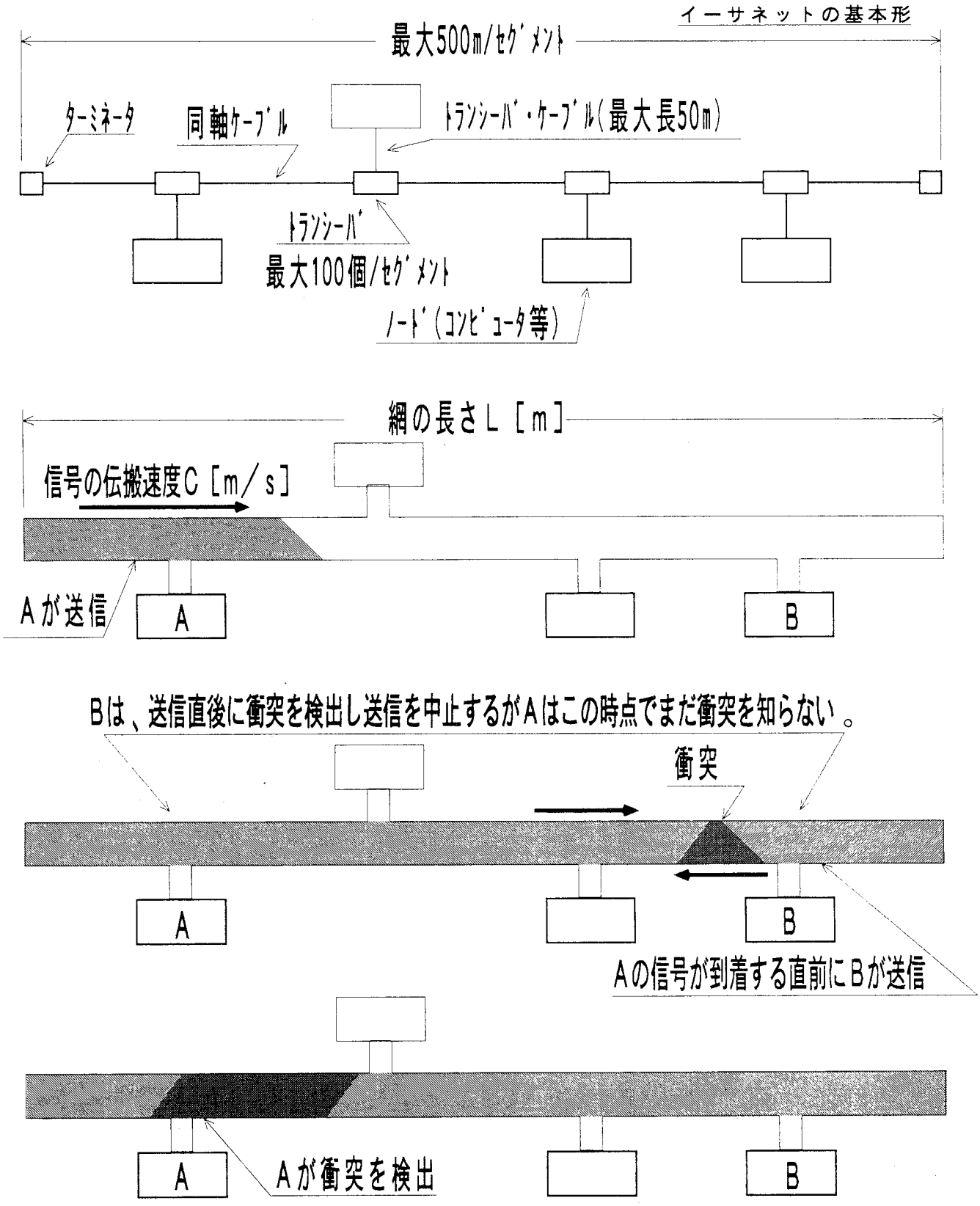


イーサネットの負荷と遅延特性（例）



約80％以上の負荷状態では実質上なデータ伝送ができなくなる。

最もたちの悪い衝突について

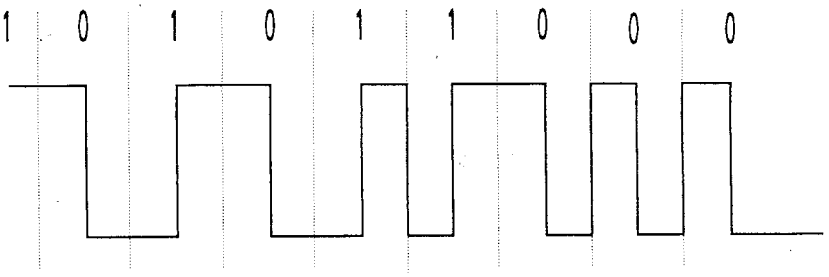


ラウンドトリップ遅延時間Ｒ＝２Ｌ／Ｃ［Ｓ］（≒４５μＳ）

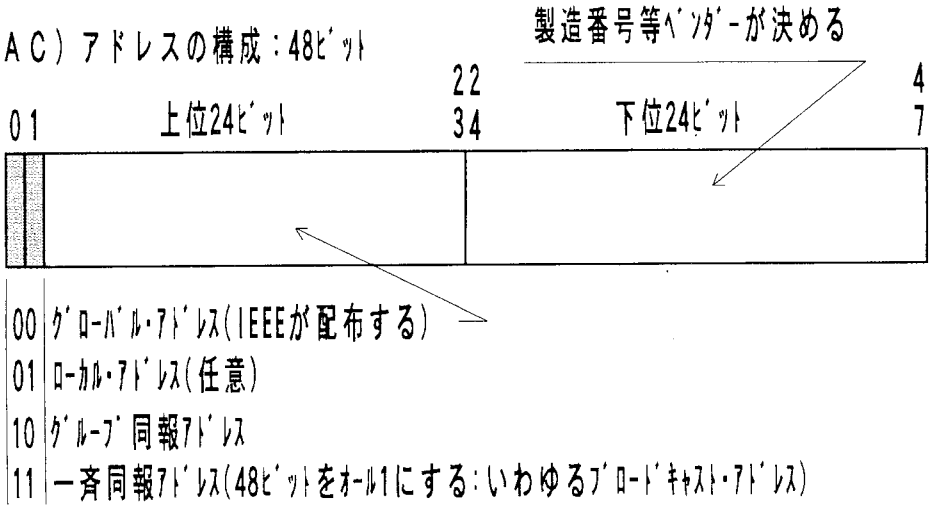
イーサネットの物理層とフレームについて

●物理層の符合：マンチェスタ符合

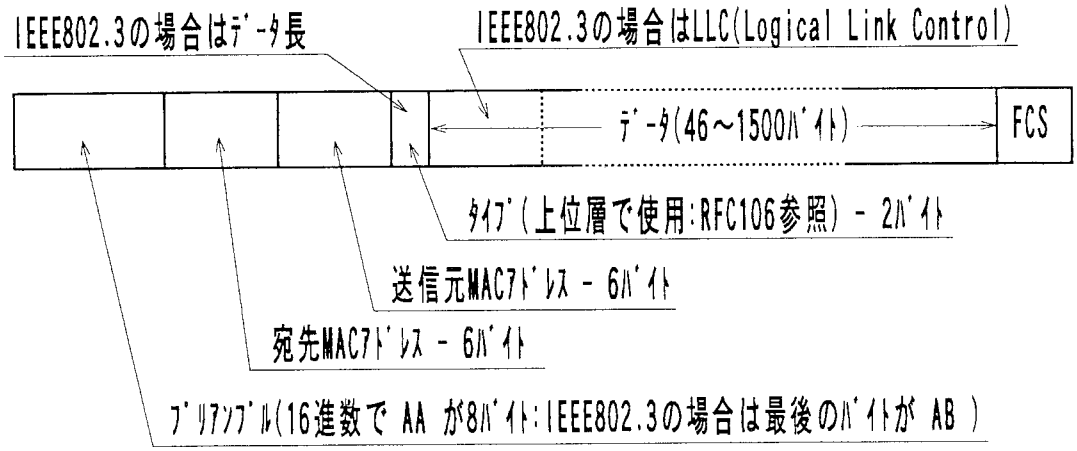
位相の違いによって0／1を符合化し、直流によって伝送するベースバンド伝送



●物理（ＭＡＣ）アドレスの構成：48ﾋﾞｯﾄ



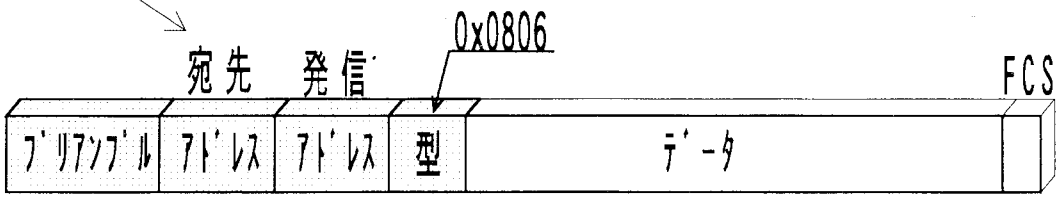
● フレーム･フォーマット



ARPについて（イーサネット：IEEE802.3の場合）

：ネットワーク層（IP層）のアドレスからデータリンク層のアドレスを得る方法

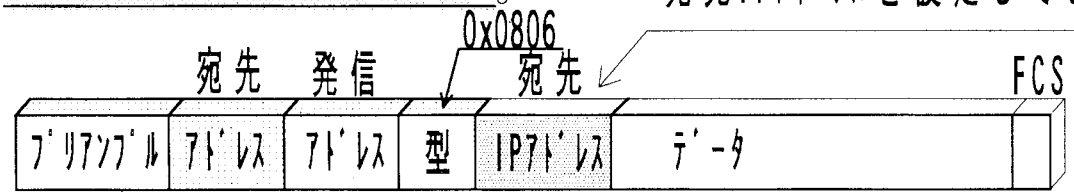
宛先の物理ｱﾄﾞﾚｽ（MACｱﾄﾞﾚｽ）が分からないと通信できない。



ARPを用いて宛先ｱﾄﾞﾚｽを得る。

ARPﾊﾟｹｯﾄをﾈｯﾄﾜｰｸに送出する。

宛先IPｱﾄﾞﾚｽを設定しておく。



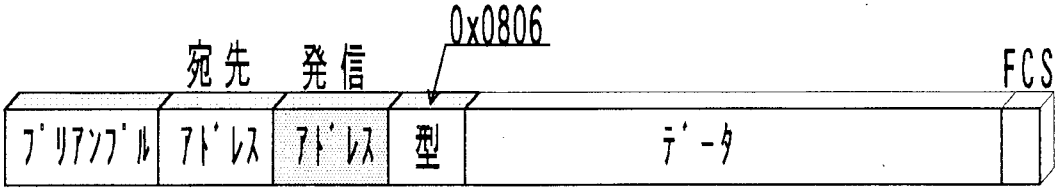
宛先ｱﾄﾞﾚｽ:FFFFFFFFFFFFh（全ﾋﾞｯﾄがすべて1）

ﾌﾞﾛｰﾄﾞｷｬｽﾄ・ｱﾄﾞﾚｽという。

全ﾉｰﾄﾞ（ﾎｽﾄ）がこれを受信する。

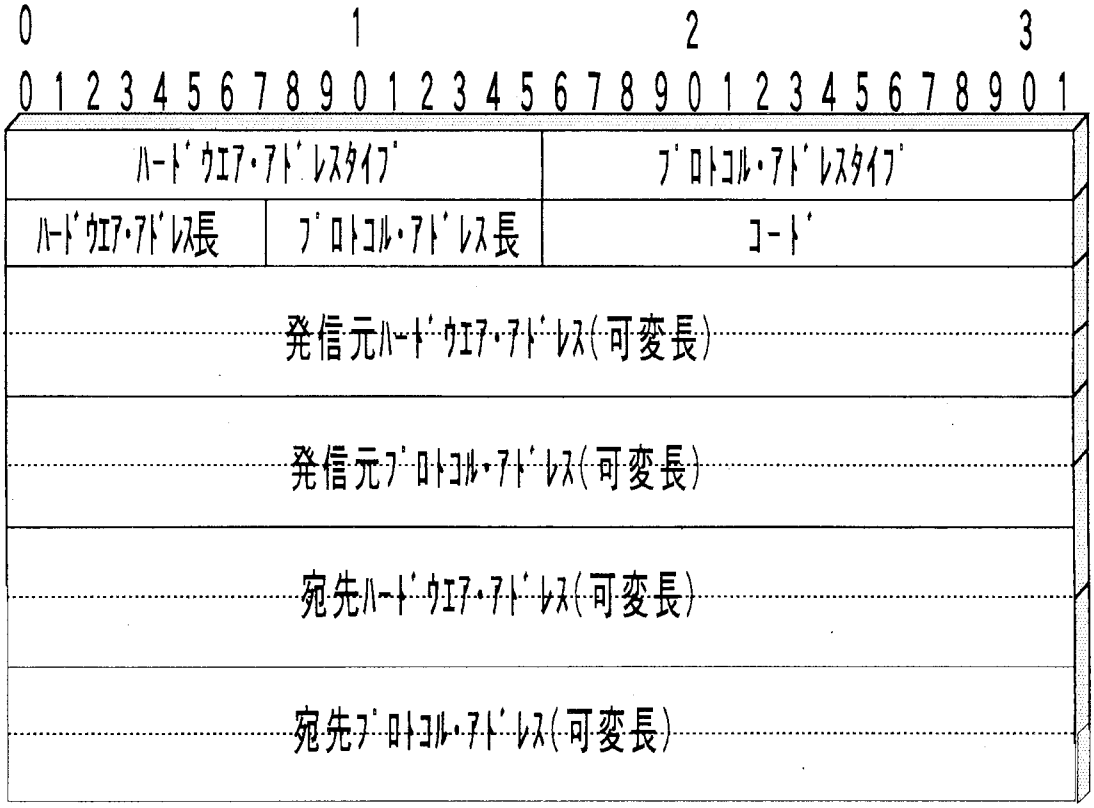
該当するﾉｰﾄﾞがそのMACｱﾄﾞﾚｽをﾈｯﾄﾜｰｸに送信する。

これをARPを発信したﾉｰﾄﾞが受信する（他のﾉｰﾄﾞも受信するが無視する。）。



発信ｱﾄﾞﾚｽ:ここに相手のMACｱﾄﾞﾚｽが設定されている。

ＡＲＰパケット・フォーマットについて



ＡＲＰパケットは種々のデータリンク層で使用される。

ﾊｰﾄﾞｳｴｱ･ｱﾄﾞﾚｽﾀｲﾌﾟ：ｲｰｻﾈｯﾄの場合は１、IEEE802.3の場合は６

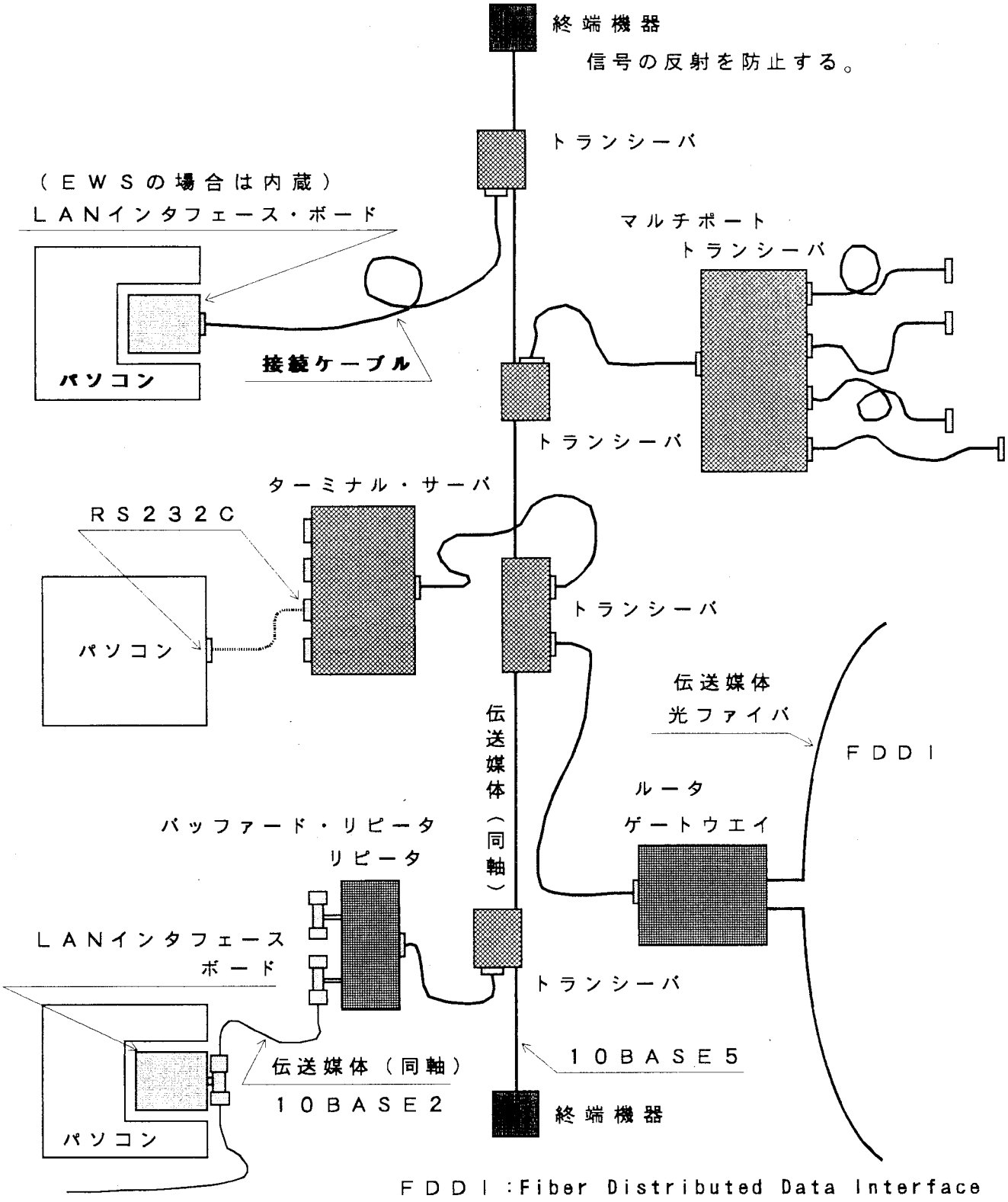
ﾌﾟﾛﾄｺﾙ･ｱﾄﾞﾚｽﾀｲﾌﾟ：IPﾌﾟﾛﾄｺﾙの場合は0x800,AppleTalkの場合は0x809b

ﾊｰﾄﾞｳｴｱ･ｱﾄﾞﾚｽ：ｲｰｻﾈｯﾄやIEEE802.3の場合は48ﾋﾞｯﾄ(6ﾊﾞｲﾄ)

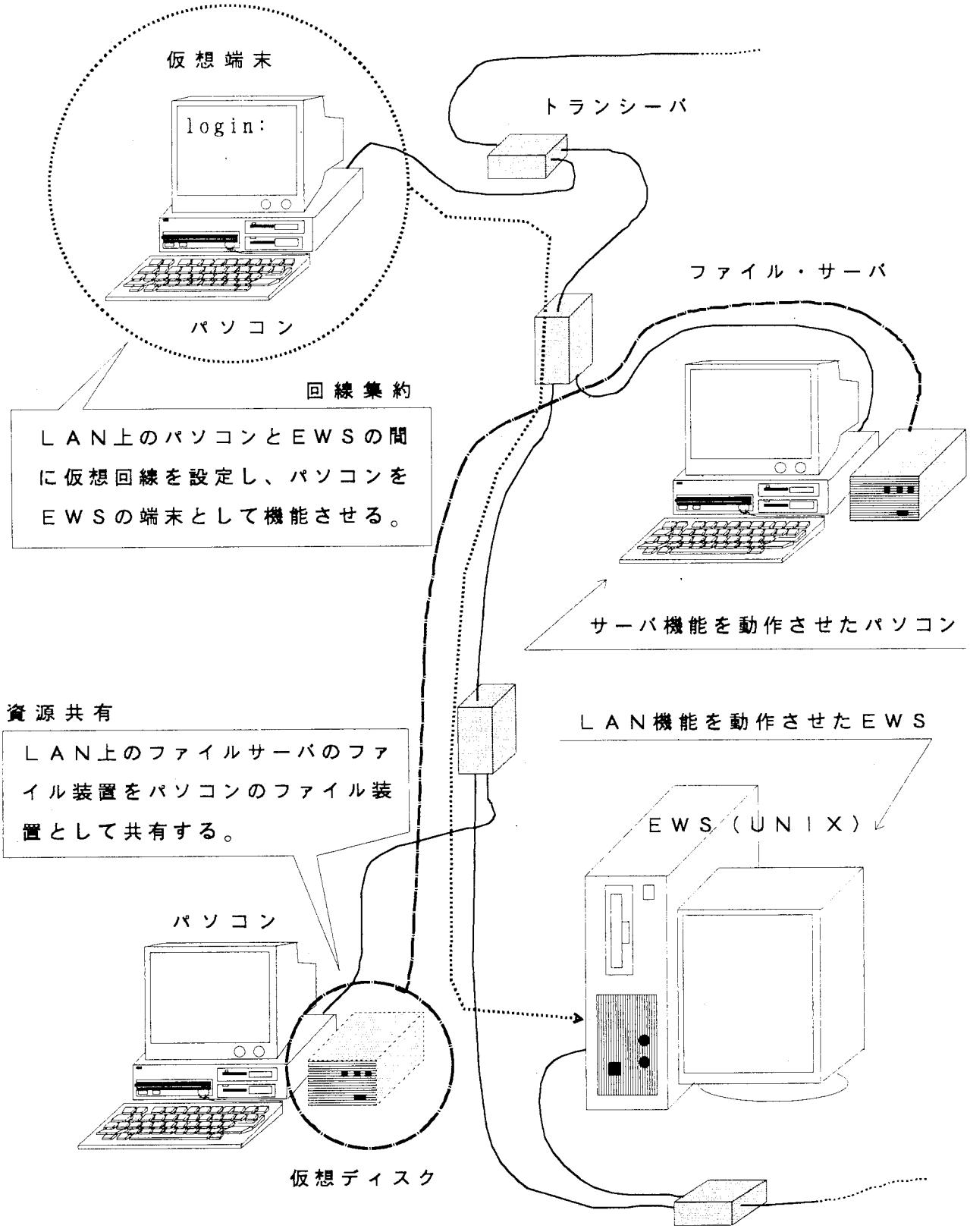
ﾌﾟﾛﾄｺﾙ･ｱﾄﾞﾚｽ：TCP/IPの場合は現在32ﾋﾞｯﾄ(4ﾊﾞｲﾄ)

（6） LANの構成要素

１０ＢＡＳＥ５を中核にしたＬＡＮの構成例



（７） ＬＡＮの代表的な応用例



3．ＬＡＮ間接続とインターネットワーキング

（1） インターネットワークとは

ＬＡＮ：Ｌｏｃａｌ Ａｒｅａ Ｎｅｔｗｏｒｋ とは

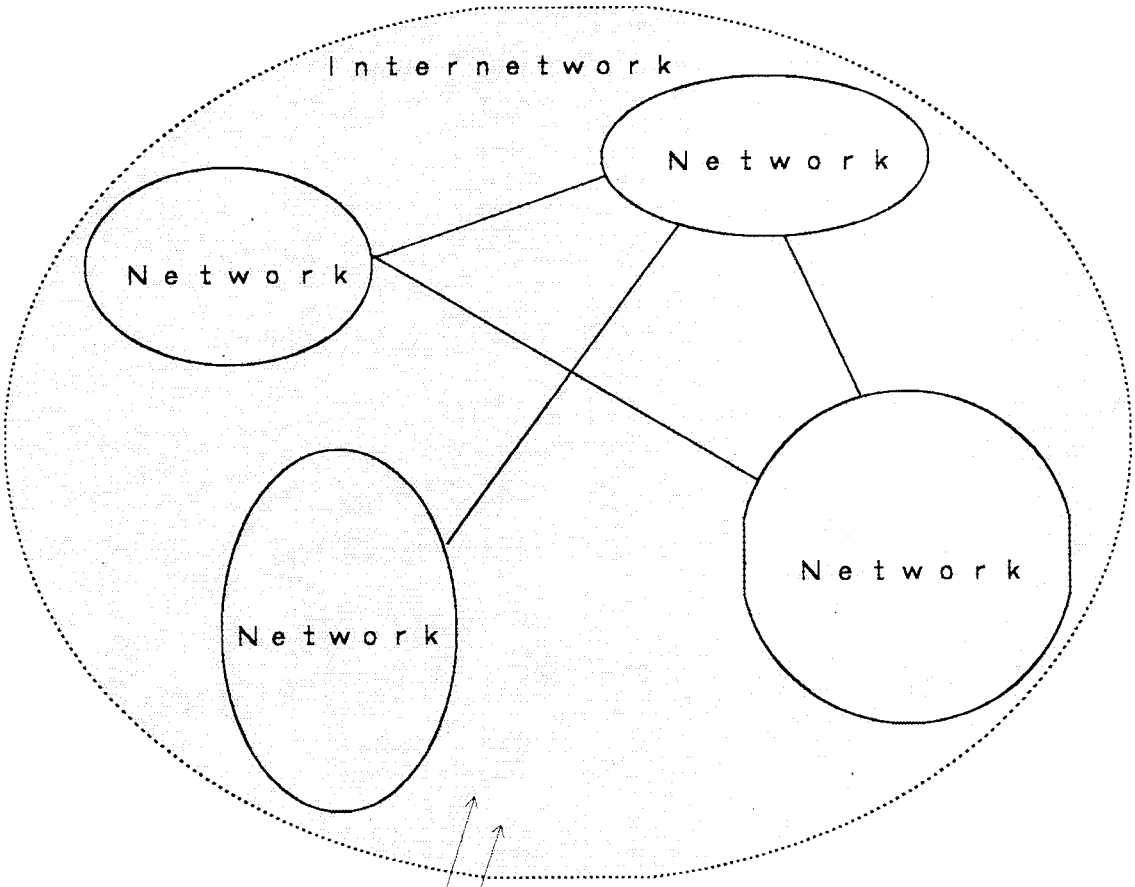
ある特定の地域内でデータなどを相互に通信する網

ＷＡＮ：Ｗｉｄｅ Ａｒｅａ Ｎｅｔｗｏｒｋ とは

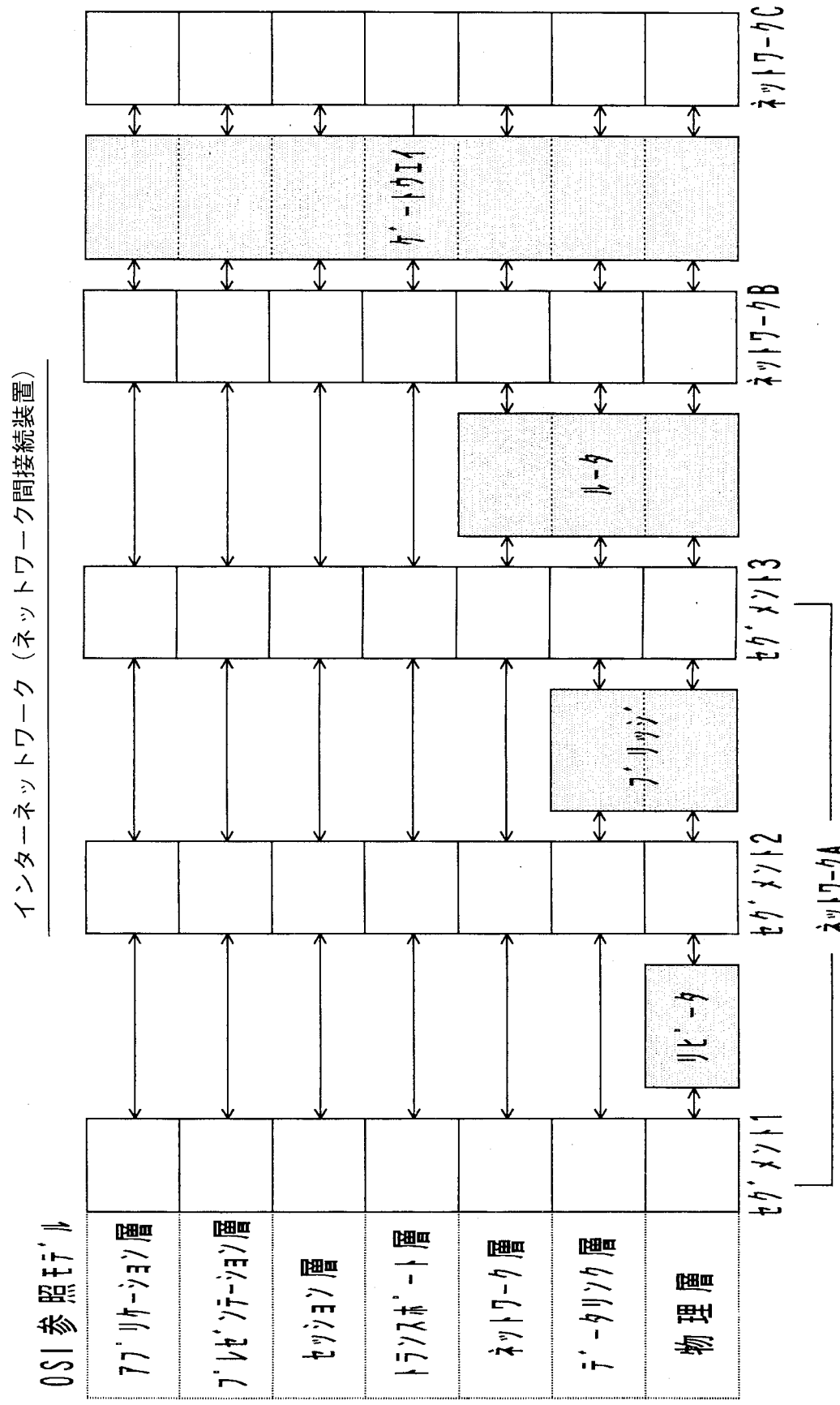
広範囲な地域（地域を限定しない）でデータなどを相互に通信する。

Ｉｎｔｅｒｎｅｔｗｏｒｋ とは

ＮｅｔｗｏｒｋとＮｅｔｗｏｒｋを相互接続したもの。



（2） LAN 間接続装置とインターネットワーク



（3） TCP／IPによるインターネットワーキング

① TCP／IPの概要

TCP/IP:Transmission Control Protocol/Internet Protocol

1）マルチベンダのコンピュータ間通信に利用できる通信プロトコル

2）米国の研究者用ネットワークＡＲＰＡＮＥＴ用に開発され、政府や軍の請負契約のためにＤＯＤ（Department of Defense:米国防総省）が採用

3）ＡＲＰＡＮＥＴは米国政府の支援を受けて発展した

4）初めての階層構造を持つ通信プロトコル

5）ＩＳＯのＯＳＩプロトコルより１０年近く先行

ＴＣＰとＩＰの目的は

1）ＴＣＰは信頼性の低いサブネットワーク上で高い信頼性を実現する。

2）ＩＰは異なるネットワークを介しての情報伝送を実現する。

ＴＣＰ／ＩＰ発展の理由は

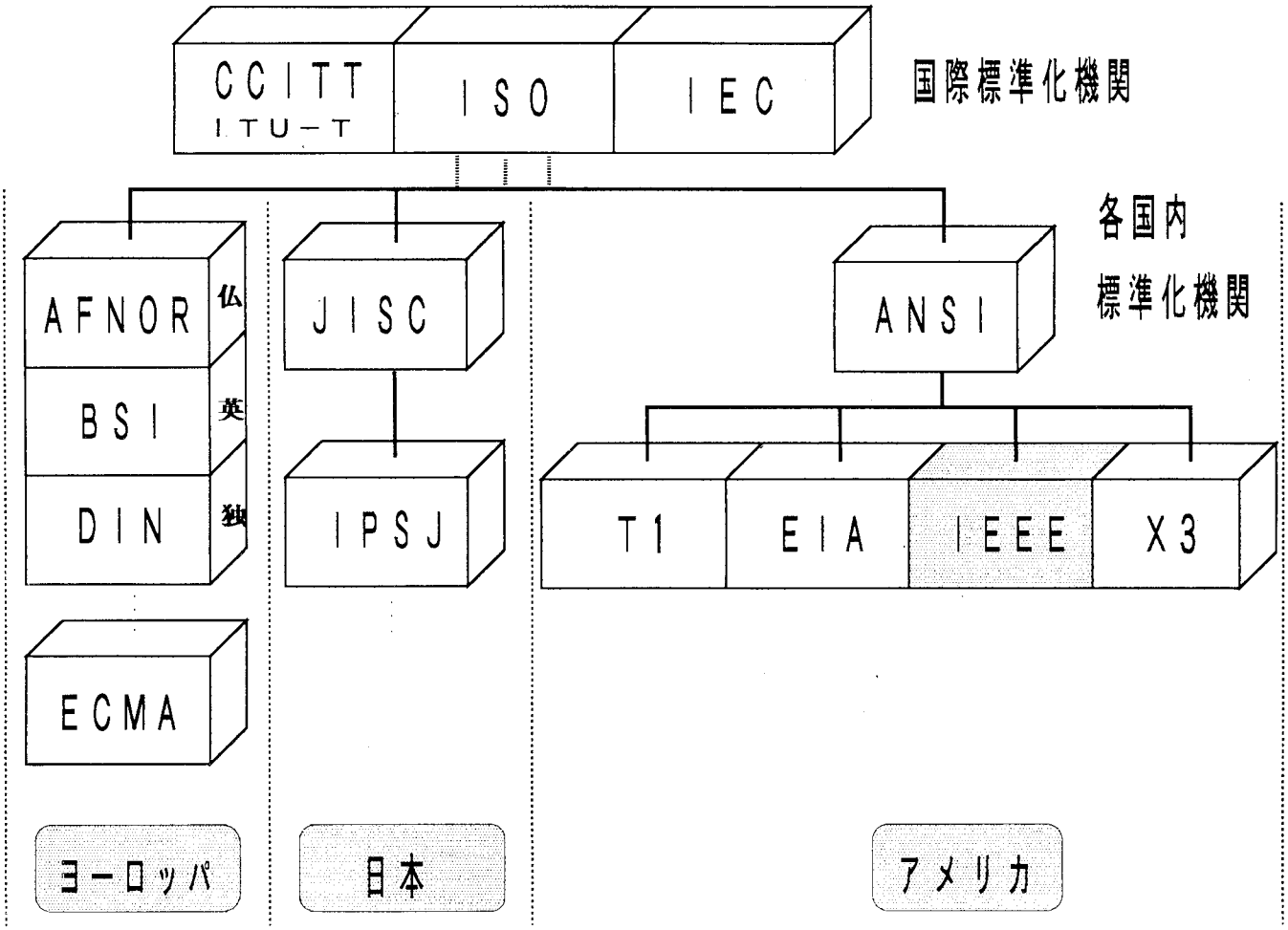
1）ＵＮＩＸマシーンに広く採用されている。

2）非ＵＮＩＸマシーン向けのＴＣＰ／ＩＰツールも多くある。

3）複数のＬＡＮで構成されたシステムにも十分対応できる。

4）現在中心的なＬＡＮのアクセス方式であるＣＳＭＡ／ＣＤと似ているため、ＬＡＮとの親和性が高い。

② インターネットワークに関わる国際標準化機構と各国の標準化機関



ＣＣＩＴＴ：国際電信電話諮問委員会(International Telegraph and Telephone Consultative Committee)
ＩＳＯ：国際標準化機構(International Organization for Standardizaion)
ＩＥＣ：国際電気標準化会議(International Electrotechnical Commission)
ＡＮＳＩ：アメリカ規格協会(American National Standards Institute)
Ｘ３：ＡＮＳＩの中で、コンピュータと情報処理分野を担当する組織
ＩＥＥＥ：アメリカ電気・電子技術者協会(The Institute of Electrical and Electronics Engineers)
ＥＩＡ：アメリカ電子工業会(Electronic Industries Association)
Ｔ１：アメリカ電気通信標準化（Ｔ１）委員会
ＪＩＳＣ：日本工業標準化調査会(Japan Industrial Standards Committee)
ＩＰＳＪ：情報処理学会(Information Processing Society of Japan)
ＡＦＮＯＲ：フランス規格協会(Association Francaise de Normalisation)
ＢＳＩ：イギリス規格協会(British Standards Institution)
ＤＩＮ：ドイツ規格協会（Deutches Institut fur Normung)
ＥＣＭＡ：ヨーロッパ・コンピュータ製造業者協会(European Computer Manufactures' Association)

③　インターネットワーク・プロトコルの標準化について

ＴＣＰ／ＩＰはＩｎｔｅｒｎｅｔｗｏｒｋを意図したプロトコル

ＴＣＰ／ＩＰはＯＳＩ参照モデルのトランスポート層と
ネットワーク層にそれぞれ対応したプロトコル

ＴＣＰ／ＩＰは、現在、Ｉｎｔｅｒｎｅｔｗｏｒｋの
デファクト・スタンダードになっている。

インターネット（Ｉｎｔｅｒｎｅｔ）は、ＴＣＰ／ＩＰを用いて
全世界をカーバーするＷＡＮを構成している。

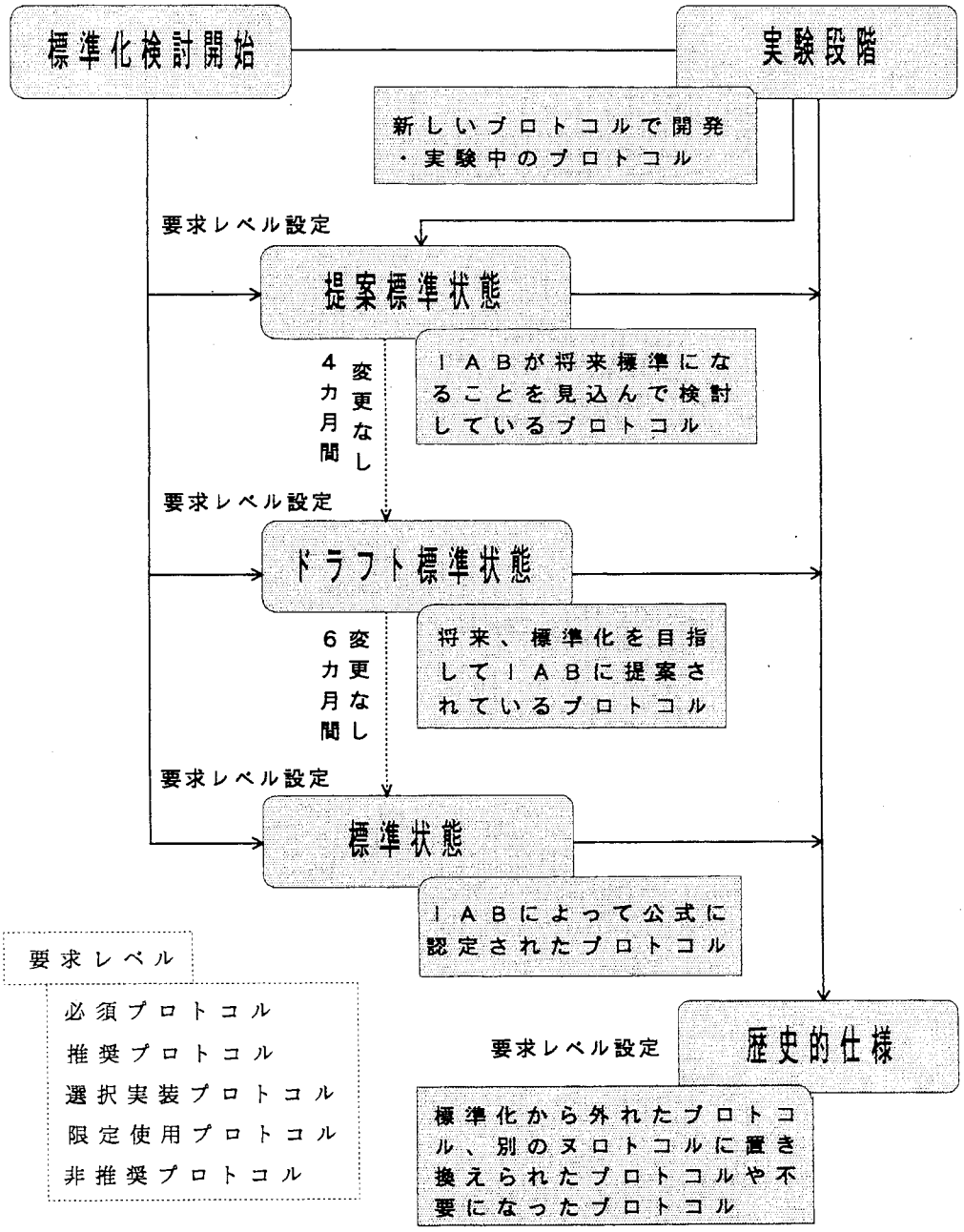
インターネット上では、現在も、新たなネットワーク・プロトコルや
ネットワーク・アプリケーションが将来に向けて研究・開発されている。

インターネット上でのプロトコルの標準化は、ＩＡＢが担っている。
実作業は、ＩＥＴＦ，ＩＲＴＦなどが行なっている。

IAB :Internet Achitecture Board
IETF:Internet Enginering Task Force
IRTF:Internet Research Task Force

標準化の対象となるプロトコル仕様は、IABやその下部組織が検討し開発したものもあるが、企業、大学や研究機関において開発されたプロトコルを対象にすることも多くある。

a．プロトコルの標準化手順



b．RFC（Request For Comments）とは？

●インターネット標準化の内容を記述したドキュメントの集まり。

●ＲＦＣは登録した順番にしたがって番号が付与される。

●ＲＦＣのうち、いくかのドキュメントはインターネットでの標準を規定している。

●ＲＦＣは（紙、電子メール、ＣＤ－ＲＯＭなどで）再配布できる。

●一旦、番号が決められたＲＦＣの内容は変更できない。

ＯＳＩのプロトコルとＴＣＰ／ＩＰの違いは？

ＯＳＩのプロトコルは、先に理論的なプロトコルの検討を行い、標準化されてから実装がはじまる。

ＴＣＰ／ＩＰは、プロトコルの検証を行なった結果に基づいて標準化される。

c．ＲＦＣの例

Network Working Group
Request for Comments: 1311

Internet Activities Board
J. Postel, Editor
March 1992

Introduction to the STD Notes

Status of this Memo

This RFC describes a new sub-series of RFCs, called STDs (Standards). Distribution of this memo is unlimited.

1. Introduction

The STDs are a subseries of notes within the RFC series that are the Internet standards. The intent is to identify clearly for the Internet community those RFCs which document Internet standards.

2. The Assignment of STD Numbers

There is a need to be very clear about which specifications have completed the full process of standardization in the Internet. To do this an STD number will be assigned to a specification when it reaches the Standard maturity level. Note that specifications may be either Technical Specifications (TS) or Applicability Statements (AS).

When a specification reaches the final stage of the standardization process and the IAB has designated it a standard for the Internet, an STD number will be assigned to that specification.

The existing standards have been assigned STD numbers (see Appendix).

The standard for a particular protocol will always have the same STD number.

If at some future time a protocol is reworked and a new document is produced as the specification of that standard and the new specification is designated by the IAB as a standard for the Internet, then the new document will be labeled with the same STD number (of course, that new document will have a new RFC number).

Multiple Documents for One Standard:

A STD number identifies a standard not a document. A document is identified by its RFC number. If the specification of a standard is spread over several documents they will each carry the same STD number.

For example, the Domain Name System (DNS) is currently specified by the combination of RFCs 1034 and 1035. Both of these documents are now labeled STD-13.

To be completely clear the DNS "Concepts and Facilities" document can be referenced as "STD-13/RFC-1034".

In such cases, whenever possible, the set of documents defining a particular standard will cross reference each other.

One Standard or Multiple Standards:

One difficult decision is deciding whether a set of documents describe one standard or multiple standards. In the Appendix, one can see that there are several cases in which one STD applies to multiple RFCs (see STDs 5, 13, and 20). There is one case in which a family of specifications has multiple STD numbers; that is the Telnet Options.

The general rule is that a separate STD number is used when the specification is logically separable. That is, logically separable options are assigned distinct STD numbers while amendments and non-optional extensions use the same STD number as the base specification.

Multiple Versions or Editions of a Standard:

It may occur that the documentation of a standard is updated or replaced with a new document. In such cases, the same STD number will be used to label the standard. No version numbers will be attached to STD numbers. There need be no confusion about having the up-to-date document about STD-9 since each version of the document will have a distinct RFC number (and of course a different date).

The complete identification of a specification and its document is the combination of the STD and the RFC. For example, "STD-13/RFC-1035" completely identifies the current version of the second part of the Domain Name System specification.

To completely identify all of the DNS standard the citation would be "STD-13/RFC-1034/RFC-1035".

One way to think of this is that an acronym (like TCP) refers to a concept, which is called a protocol. An RFC number (like RFC-793) indicates the specific version of the protocol specification. An STD number (like STD-7) designates the status of the protocol.

2. Why an RFC Subseries ?

There are several reasons why the STDs are part of the larger RFC series of notes.

The foremost reason is that the distribution mechanisms for RFCs are tried and true. Anyone who can get an RFC, can automatically get a STD. More important, anyone who knows of the RFC series can easily find the STDs.

Another reason for making STDs part of the RFC series is that the maintenance mechanisms for RFCs are already in place. It makes sense to maintain similar documents is a similar way.

3. Format Rules

Since the STDs are a part of the RFC series, they must conform to "Request for Comments on Request for Comments: Instructions to RFC Authors" (RFC-1111) with respect to format.

3.1 Status Statement

Each STD RFC must include on its first page the "Status of this Memo" section which contains a paragraph describing the intention of the RFC. This section is meant to convey the status approved by the Internet Activities Board (IAB).

3.2. Distribution Statement

Each STD RFC will also include a "distribution statement". As the purpose of the STD series is to disseminate information, there is no reason for the distribution to be anything other than "unlimited".

Typically, the distribution statement will simply be the sentence "Distribution of this memo is unlimited." appended to the "Status of this Memo" section.

3.3. Security Considerations

All STD RFCs must contain a section that discusses the security considerations of the procedures that are the main topic of the RFC.

3.4. Author's Address

Each STD RFC must have at the very end a section giving the author's address, including the name and postal address, the telephone number, and the Internet email address.

In the case of multiple authors, each of the authors will be listed. In the case of a document produced by a group, the editor of the document will be listed and optionally the chair of the group may be listed.

4. The STD Publication

New documents can only become STD RFCs through an action of the IAB. The publication of STDs will be performed by the RFC Editor.

5. STD Announcements

New STD RFCs are announced to the RFC distribution list maintained by the Network Information Center (NIC). Contact the NIC to be added or deleted from this mailing list by sending an email message to RFC-REQUEST@NIC.DDN.MIL.

6. Obtaining STDs

STD RFCs may be obtained in the same way as any RFC.

Details on obtaining RFCs via FTP or EMAIL may be obtained by sending an EMAIL message to "rfc-info@ISI.EDU" with the message body "help: ways_to_get_rfcs". For example:

```
To: rfc-info@ISI.EDU
Subject: getting rfcs

help: ways_to_get_rfcs
```

The current standards are listed in the "IAB Official Protocol Standards" (which is STD-1), whose current edition is RFC-1280.

Security Considerations

Security issues are not discussed in this memo.

Author's Address

Jon Postel
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

Phone: 310-822-1511
Fax: 310-823-6714

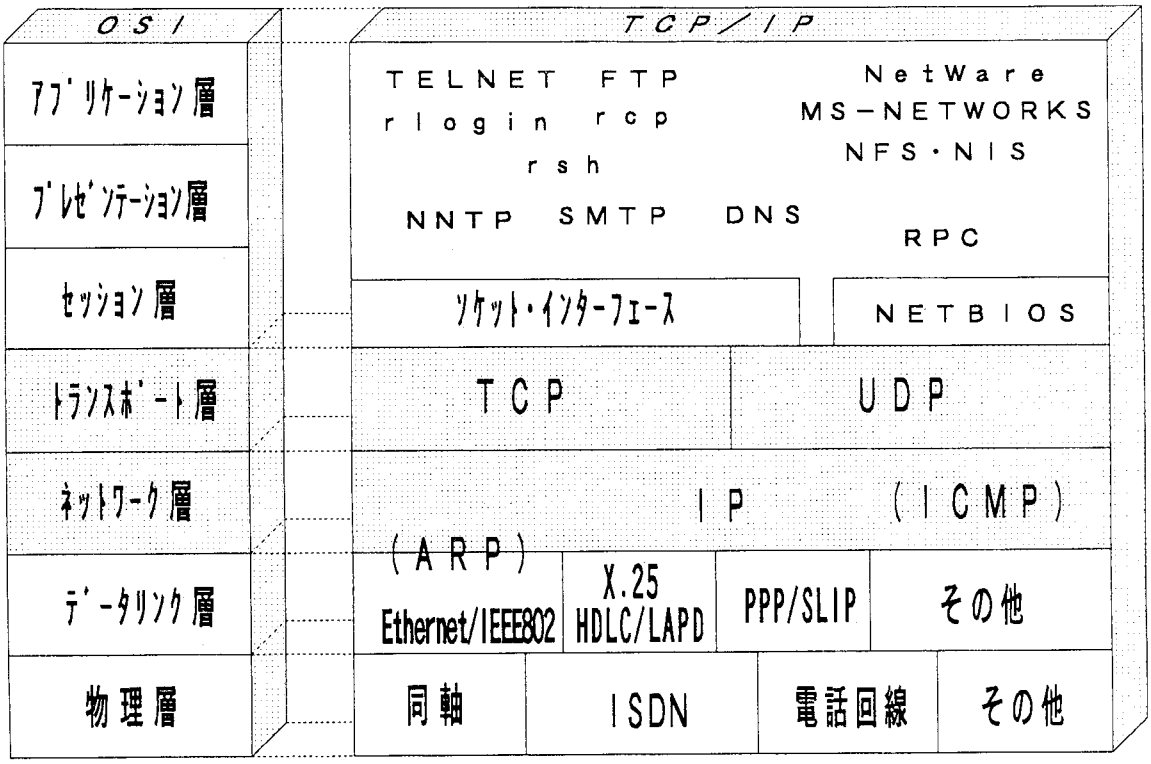
Email: Postel@ISI.EDU

APPENDIX -- The Grandfathered STDs

Protocol	Name	Status	RFC	STD
--------	IAB Official Protocol Standards	Req	1280	1
--------	Assigned Numbers	Req	1060	2
--------	Host Requirements	Req	1122, 1123	3
--------	Gateway Requirements	Req	1009	4
IP	Internet Protocol	Req	791	5
	as amended by:			
--------	IP Subnet Extension	Req	950	5
--------	IP Broadcast Datagrams	Req	919	5
--------	IP Broadcast Datagrams with Subnets	Req	922	5
ICMP	Internet Control Message Protocol	Req	792	5
IGMP	Internet Group Multicast Protocol	Rec	1112	5
UDP	User Datagram Protocol	Rec	768	6
TCP	Transmission Control Protocol	Rec	793	7
TELNET	Telnet Protocol	Rec	854, 855	8
FTP	File Transfer Protocol	Rec	959	9
SMTP	Simple Mail Transfer Protocol	Rec	821	10
MAIL	Format of Electronic Mail Messages	Rec	822	11
CONTENT	Content Type Header Field	Rec	1049	11
NTP	Network Time Protocol	Rec	1119	12
DOMAIN	Domain Name System	Rec	1034, 1035	13
DNS-MX	Mail Routing and the Domain System	Rec	974	14
SNMP	Simple Network Management Protocol	Rec	1157	15
SMI	Structure of Management Information	Rec	1155	16
MIB-II	Management Information Base-II	Rec	1213	17
EGP	Exterior Gateway Protocol	Rec	904	18
NETBIOS	NetBIOS Service Protocols	Ele	1001, 1002	19
ECHO	Echo Protocol	Rec	862	20
DISCARD	Discard Protocol	Ele	863	21
CHARGEN	Character Generator Protocol	Ele	864	22
QUOTE	Quote of the Day Protocol	Ele	865	23
USERS	Active Users Protocol	Ele	866	24
DAYTIME	Daytime Protocol	Ele	867	25
TIME	Time Server Protocol	Ele	868	26

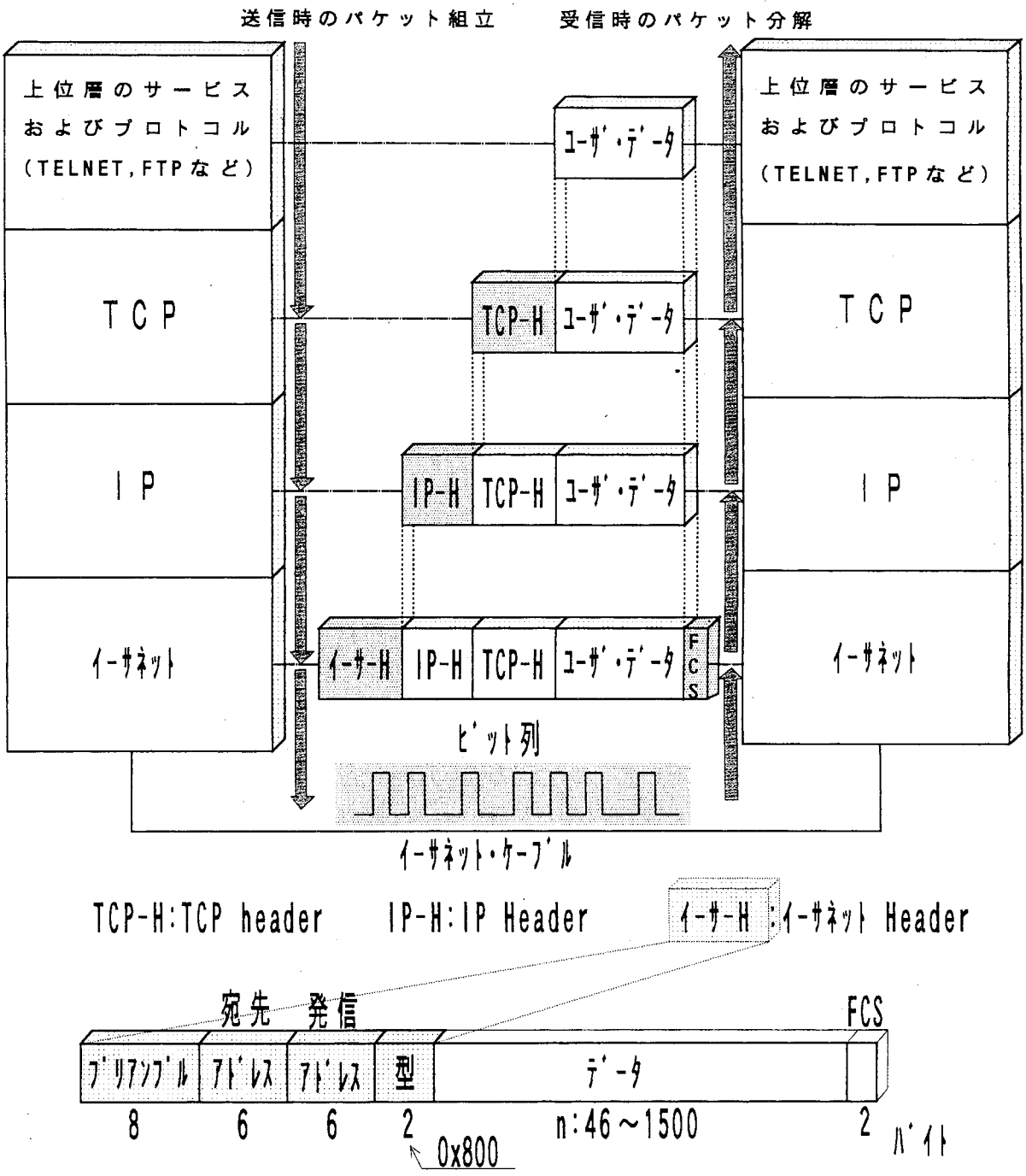
Telnet Options		Option	Status	RFC	STD
TOPT-BIN	Binary Transmission	0	Rec	856	27
TOPT-ECHO	Echo	1	Rec	857	28
TOPT-SUPP	Suppress Go Ahead	3	Rec	858	29
TOPT-STAT	Status	5	Rec	859	30
TOPT-TIM	Timing Mark	6	Rec	860	31
TOPT-EXTOP	Extended-Options-List	255	Rec	861	32

d．ＯＳＩ参照モデルとＴＣＰ／ＩＰ



OSI :Open System Interconnection
SMTP :Simple Mail Transfer Protocol
FTP :File Transfer Protocol
TELNET :仮想端末ｻｰﾋﾞｽ
rcp :remote copy
rlogin :remote login
ARP :Address Resolution Protocol
NFS :Network File System
NETBIOS:Network Basic Input Output System
TCP :Transmission Control Protocol
UDP :User Datagram Protocol
IP :Internet Protocol
ICMP :Internet Control Message Protocol
RPC :Remote Pprocedure Call
ARP :Address Resolution Protocol

e．ＴＣＰ／ＩＰ通信の層（レイヤ）間インタフェース

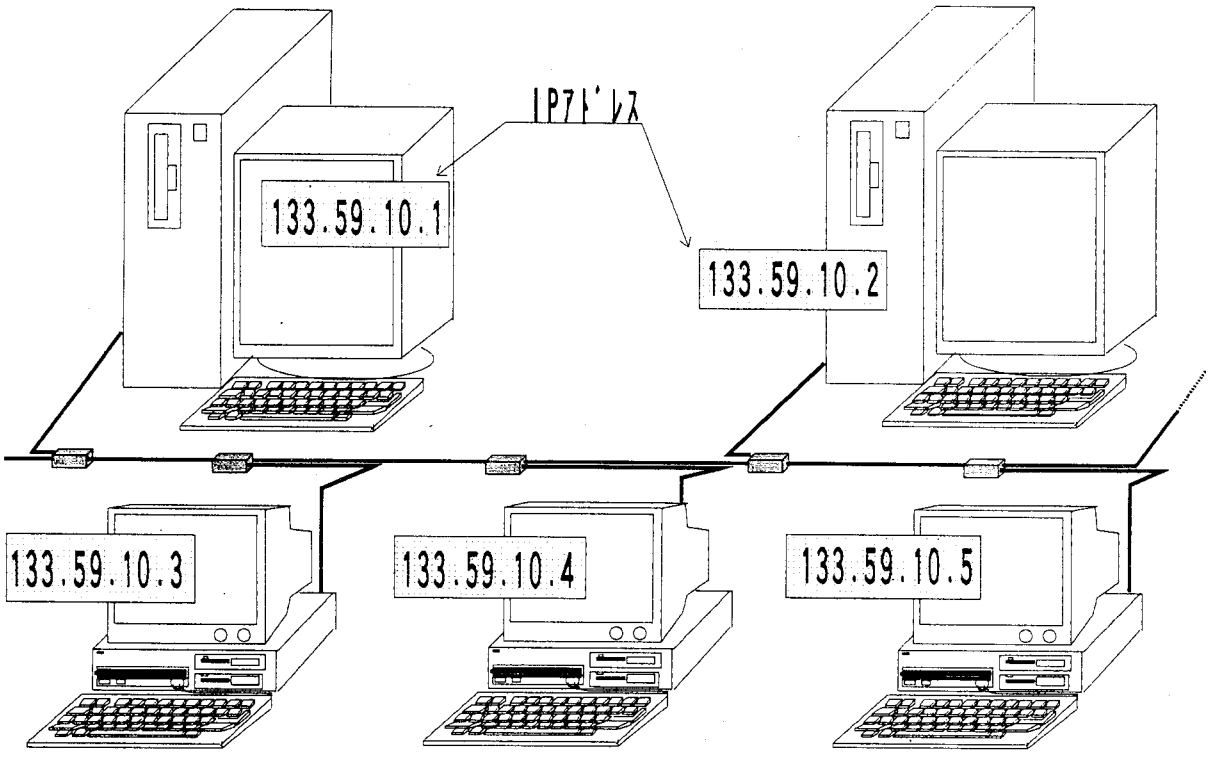


f．ＩＰ（Internet Protocol）について

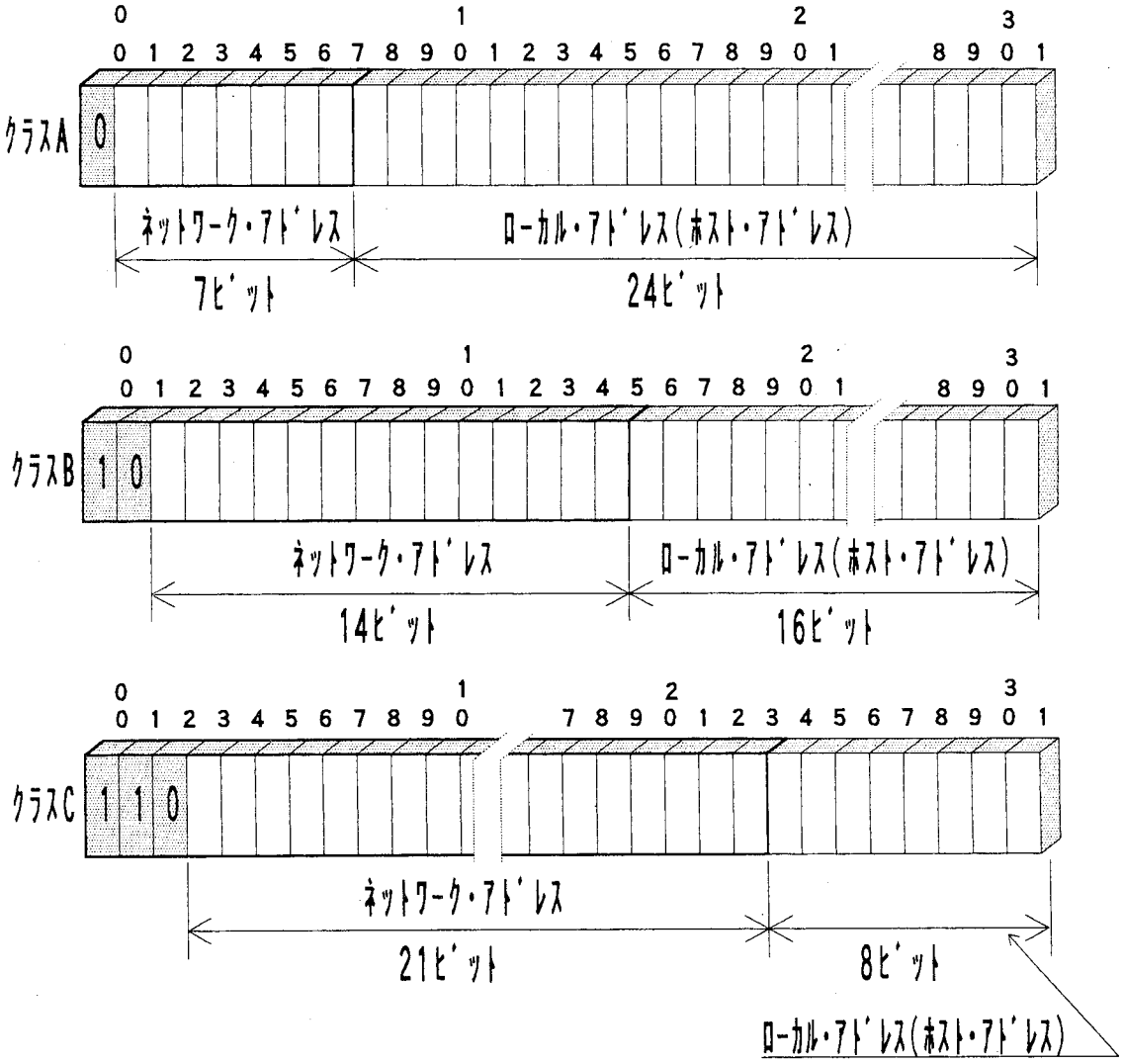
ＩＰは、ＯＳＩ参照モデルのネットワーク層に相当し、伝送経路の確立やネットワーク・アドレスとホスト・アドレスの定義によるネットワークの論理的管理などを担当する。
ネットワーク層のプロトコルとしてはＩＰの他にＡＲＰ、ＲＡＲＰ、ＩＣＭＰなどがある。

ＩＰアドレス

ＴＣＰ／ＩＰ通信を使用する場合は、ネットワークに接続される機器にそれぞれ個別のアドレスを設定する必要がある。



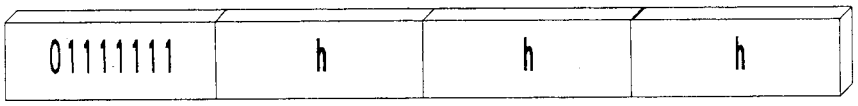
ＩＰアドレスは、論理的なネットワークの大きさにより、
クラスＡ、Ｂ、Ｃの３種類がある。



クラスA	0.H.H.H - 127.H.H.H	どのクラスにおいても、ホスト・アドレスの部分(Ｈ)のビットをすべて0や1にしたＩＰアドレスを、ノードのアドレスとして設定することはできない。
クラスB	128.0.H.H - 191.255.H.H	
クラスC	192.0.0.H - 233.255.255.H	

特別な意味をもつＩＰアドレス（１）

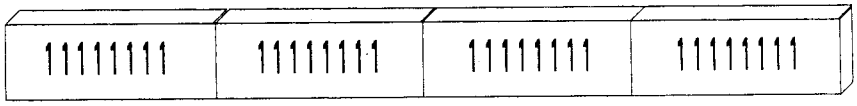
(1)ﾙｰﾌﾟﾊﾞｯｸ・ｱﾄﾞﾚｽ（自分宛のｱﾄﾞﾚｽに用いる。）



127 . h . h . h

（一般的には、127.0.0.1 を使用する。）

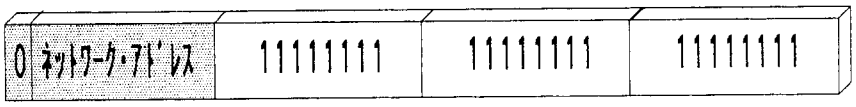
(2)ﾈｯﾄﾜｰｸ内限定用ﾌﾞﾛｰﾄﾞｷｬｽﾄ・ｱﾄﾞﾚｽ



255 . 255 . 255 . 255

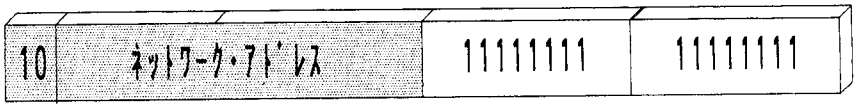
（ルータを超えないブロードキャスト）

(3)ｲﾝﾀｰﾈｯﾄﾜｰｸ・ﾌﾞﾛｰﾄﾞｷｬｽﾄ



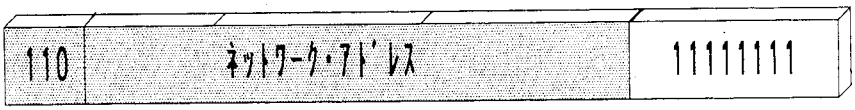
CLASS A

1～127 . 255 . 255 . 255



CLASS B

128～191 . 0～255 . 255 . 255

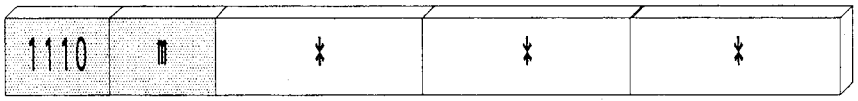


CLASS C

192～223 . 0～255 . 0～255 . 255

特別な意味をもつIPアドレス（２）

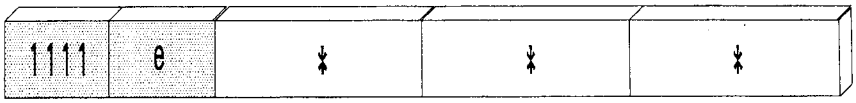
(4)ﾏﾙﾁｷｬｽﾄ･ｱﾄﾞﾚｽ



244～239 . * . * . *

（詳細はRFC1060参照）

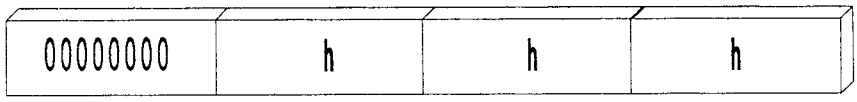
(2)予備のｱﾄﾞﾚｽ



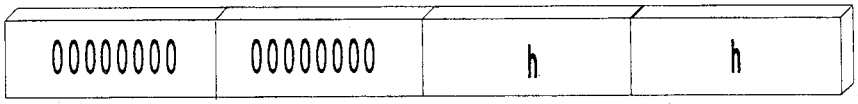
240～255 . * . * . *

（現在、実験用に用いられている。）

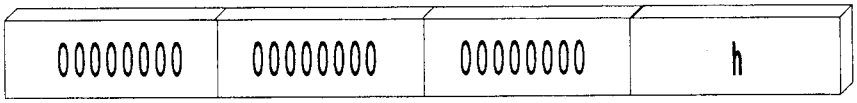
(3)自ﾈｯﾄﾜｰｸのﾈｯﾄﾜｰｸｱﾄﾞﾚｽが不明の場合



0 . h . h . h



0 . 0 . h . h

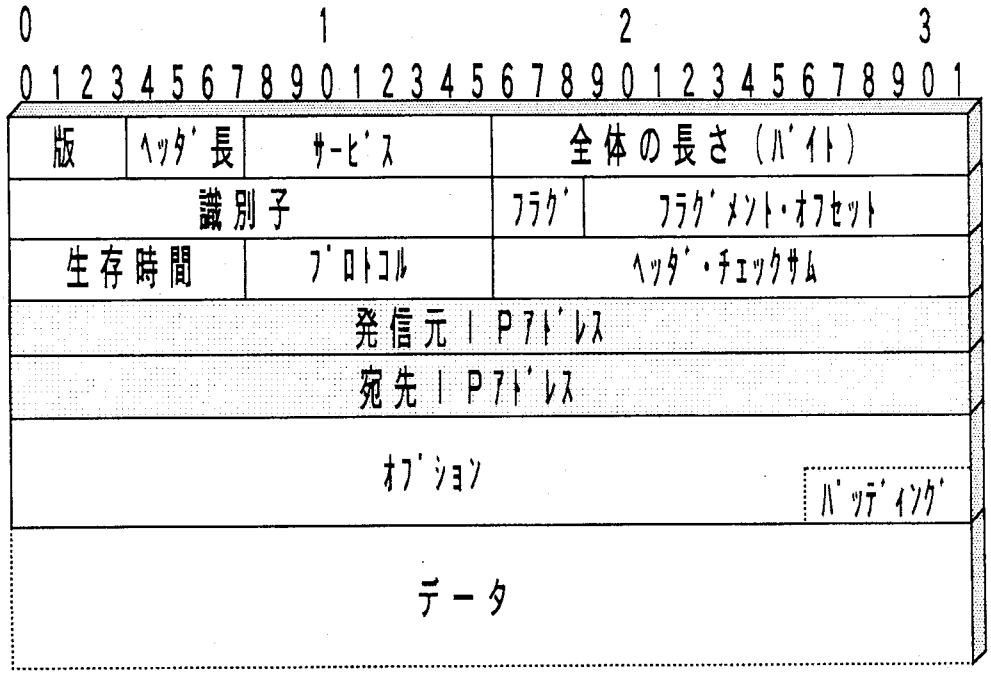


0 . 0 . 0 . h

0.0.0.0 は、このﾈｯﾄﾜｰｸのこのﾎｽﾄ(つまり自分)を表す。

0.0.0.0 は、ﾌﾞﾛｰﾄﾞｷｬｽﾄ･ｱﾄﾞﾚｽを表わすことがあるので注意！

ＩＰヘッダ



版　　：ＩＰパケットのバージョン（現在は４）

ヘッダ長　：ＩＰヘッダの長さ(ｵﾌﾟｼｮﾝなしの場合は5:5×4=20ﾊﾞｲﾄ)

サービス　：Type Of Servicesでﾙｰﾀ上での取扱い方法を示す(RFC795参照)。

トータル長：ＩＰパケットのトータル長

生存時間　：Time To LiveでＩＰパケットの寿命を表す(ﾙｰﾀを通過すると-1される。)。

ﾌﾟﾛﾄｺﾙ番号：ＩＰパケットが運ぶ上位のプロトコルを表す。
ICMP: 1 ,TCP: 6 ,UDP: 17

識別子　　：ＩＰパケットを識別するための番号（16ﾋﾞｯﾄの数値）

フラッグ　：ＩＰパケットの分割や分割されたパケットの位置関係を示す。
ＩＰパケットが分割された場合は識別子は同じ番号になる。

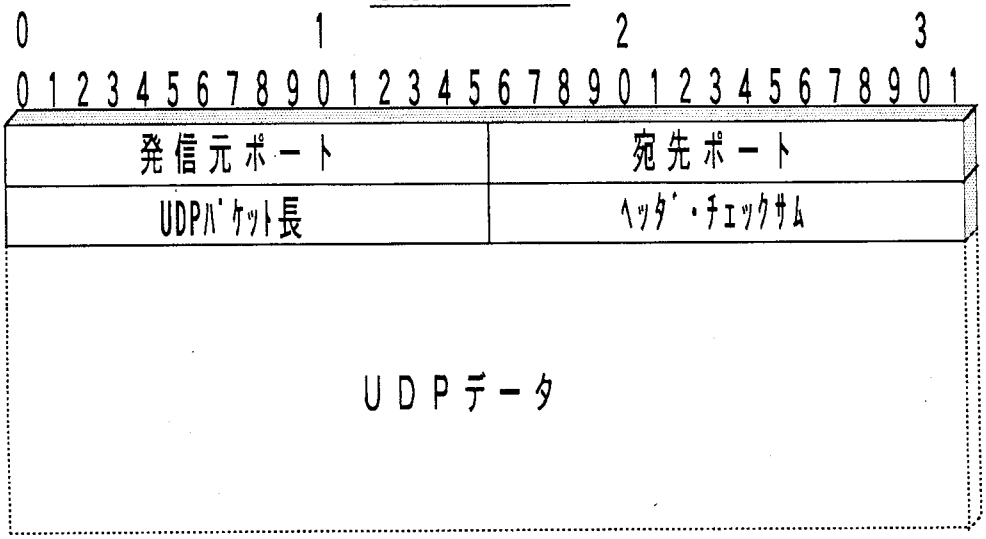
ﾍｯﾀﾞ・ﾁｪｯｸｻﾑ：ＩＰヘッダの16ﾋﾞｯﾄ単位の和の１の補数

発信元ｱﾄﾞﾚｽ：32ﾋﾞｯﾄの発信元ｱﾄﾞﾚｽの値
ﾌﾞﾛｰﾄﾞｷｬｽﾄ・ｱﾄﾞﾚｽやﾏﾙﾁｷｬｽﾄ・ｱﾄﾞﾚｽは指定できない。

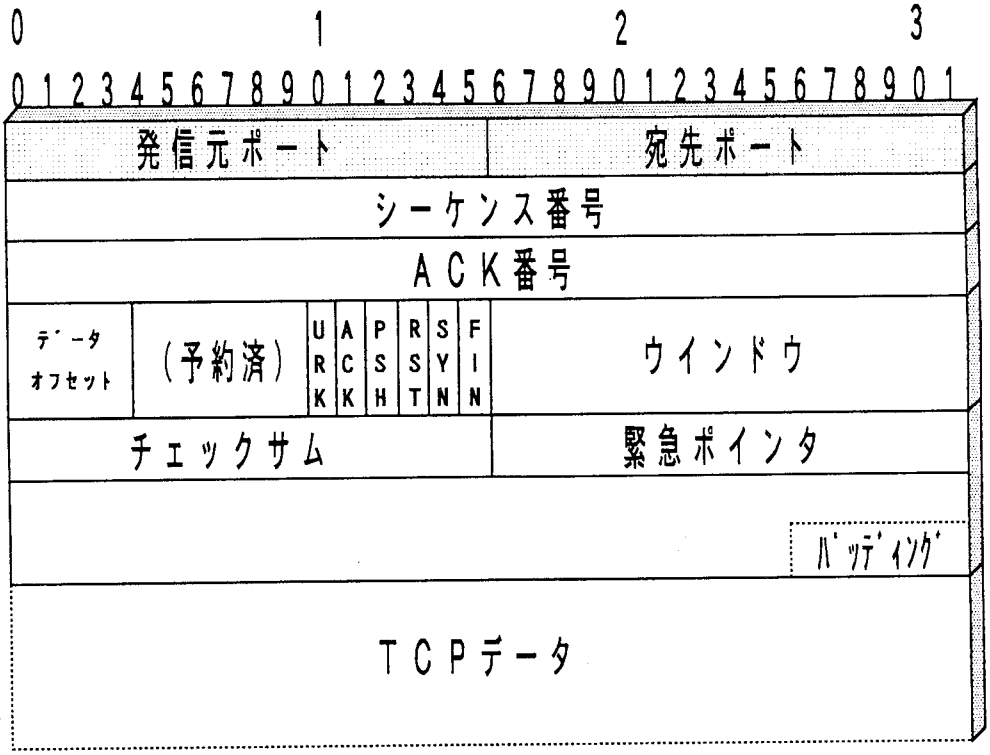
宛先ｱﾄﾞﾚｽ　：32ﾋﾞｯﾄの宛先ｱﾄﾞﾚｽの値

g．トランスポート層のヘッダ

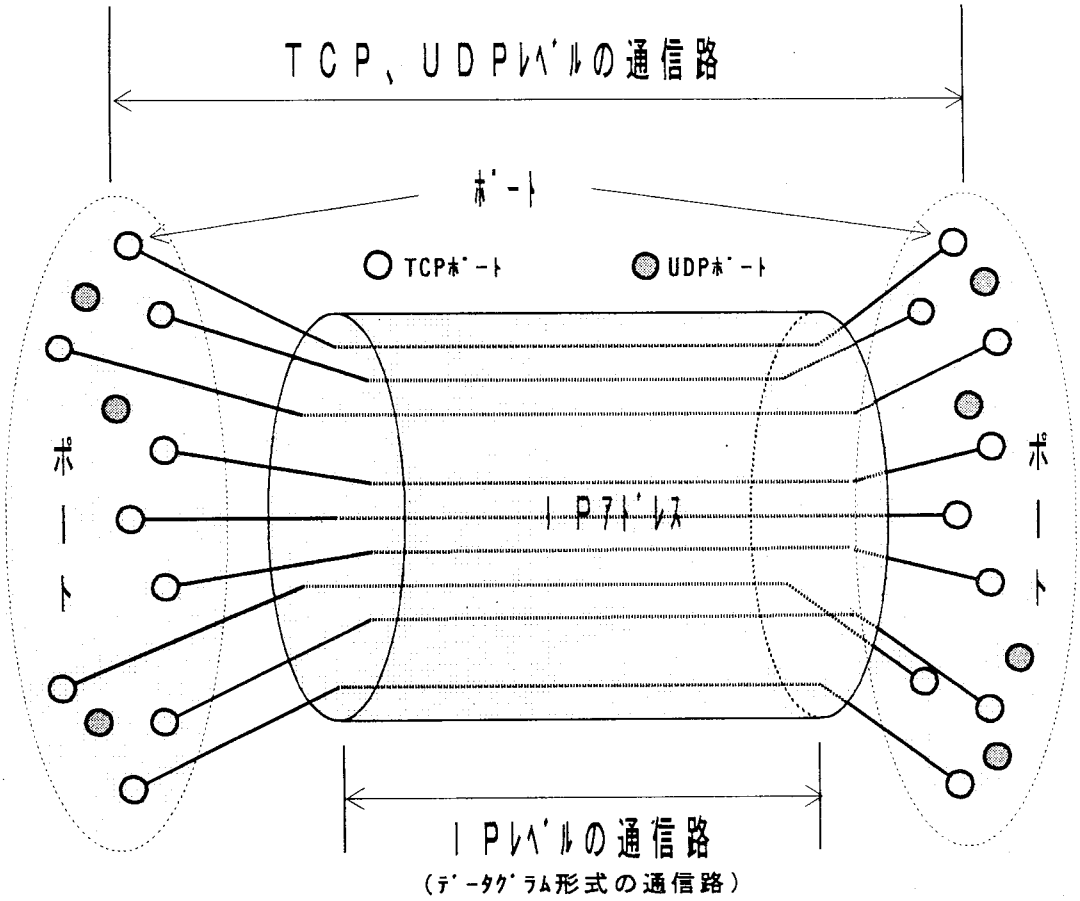
ＵＤＰヘッダ



ＴＣＰヘッダ



h.　ＩＰ層とＴＣＰ層の通信イメージ



ＩＰは、ノード間（ＩＰアドレス間）で通信路を設定する。

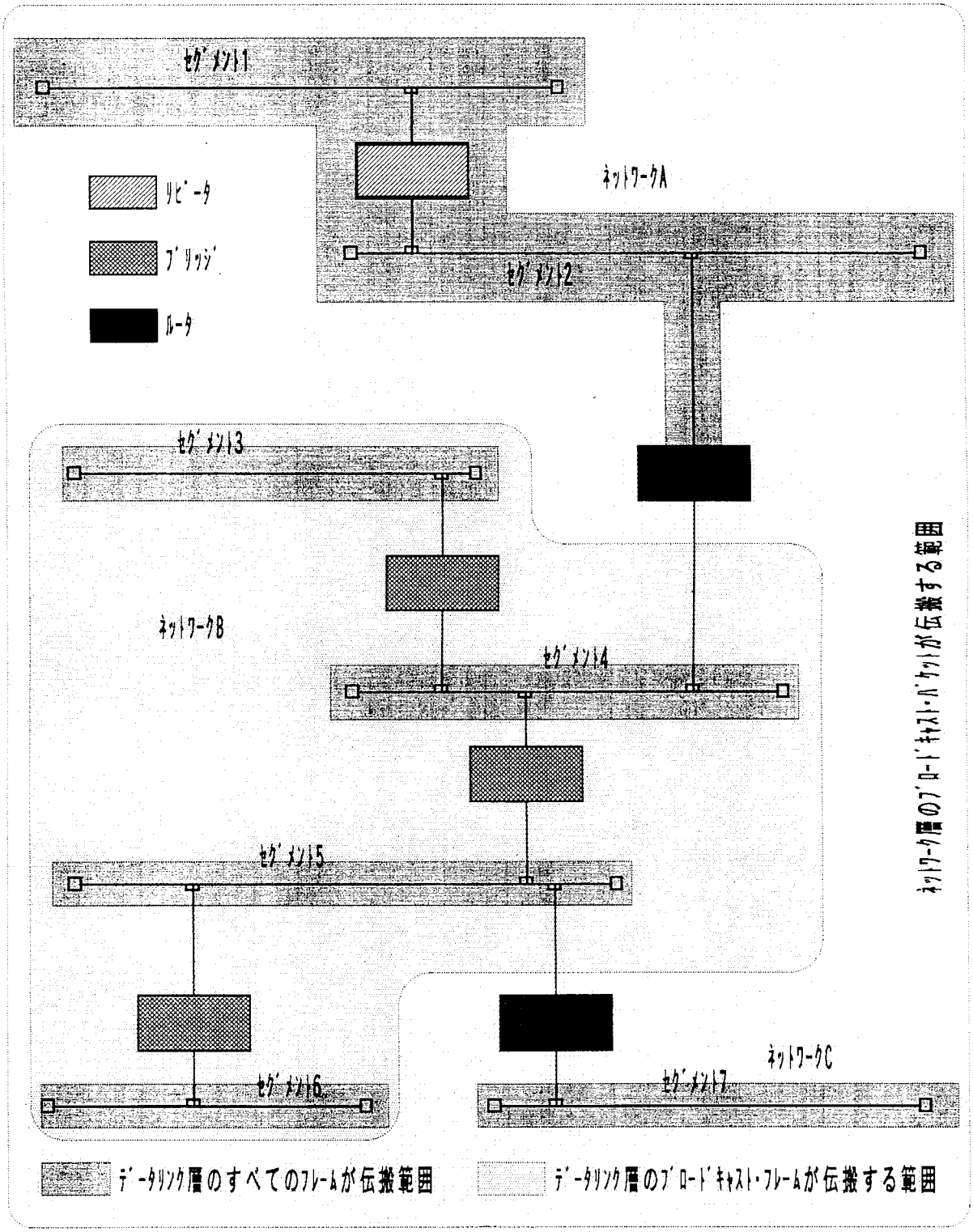
ＩＰは、パケットの送出までは保証するがその後は何の保証もしない。

ＴＣＰ、ＵＤＰはポート番号間で通信路を設定する。

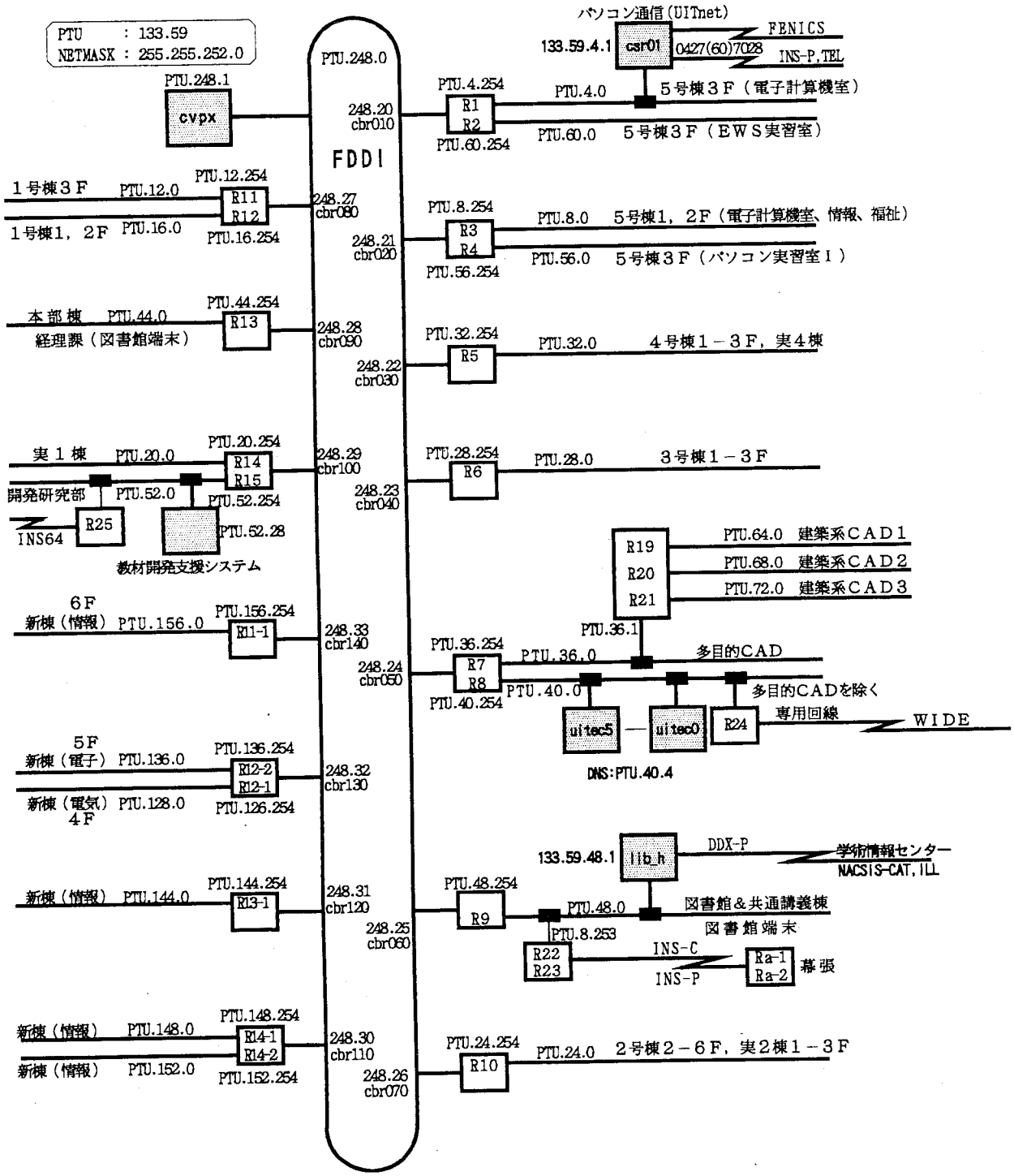
ＵＤＰは、ＩＰと同等な通信を行う（データグラム形式）。

ＴＣＰは、ノード間で確実な通信をする（ストリーム形式）。

ｉ．ＬＡＮ（ＷＡＮ）のフレームやパケットの伝搬範囲

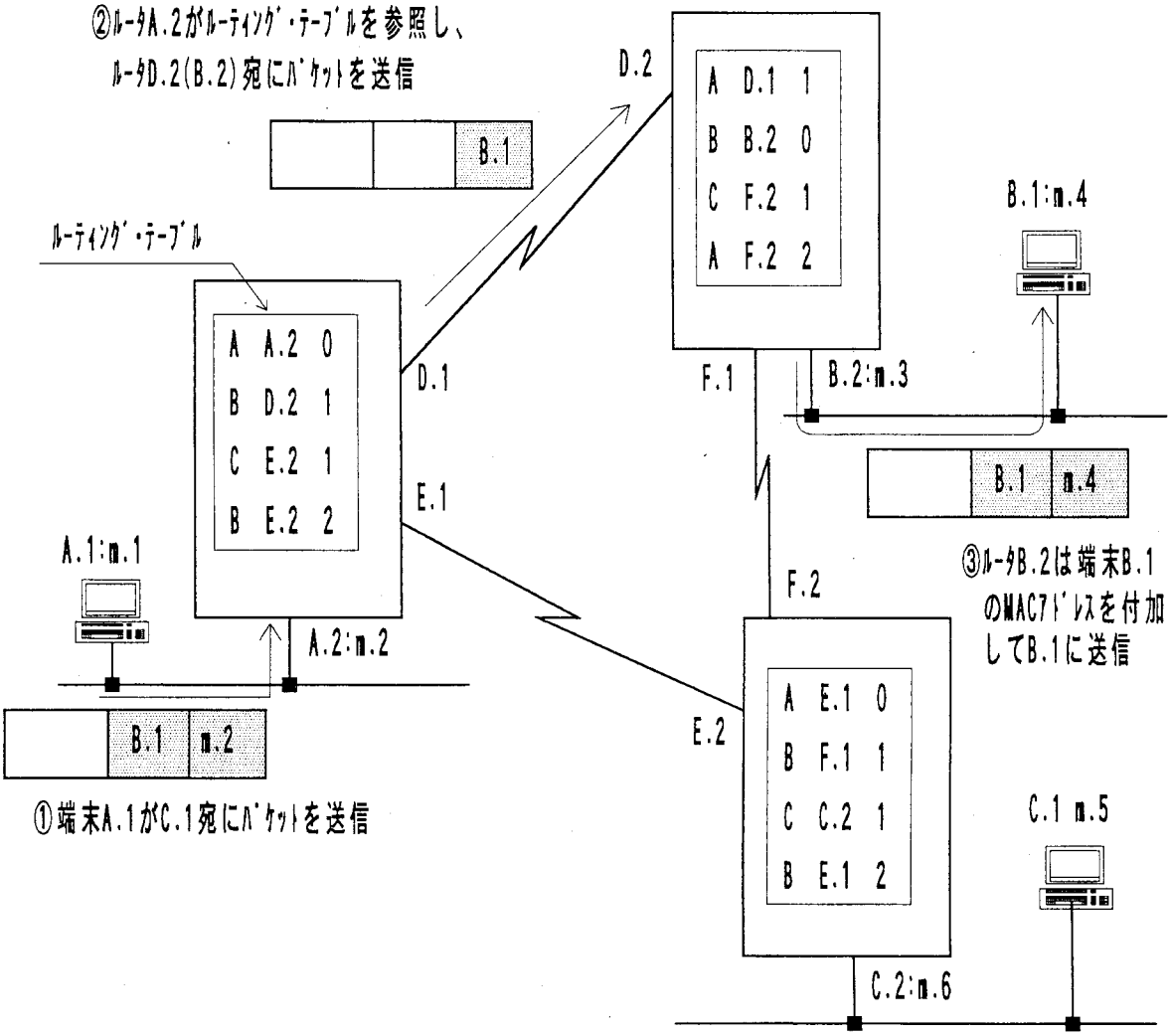


j．ＬＡＮ／ＷＡＮ構成例


PTU : 133.59
NETMASK : 255.255.252.0
PTU.248.0
FDDI
PTU.248.1
cvpx
248.20
cbr010
PTU.4.254
R1
R2
PTU.60.254
パソコン通信(UITnet)
133.59.4.1
csr01
FENICS
0427(60)7028
INS-P,TEL
PTU.4.0
５号棟３Ｆ（電子計算機室）
PTU.60.0
５号棟３Ｆ（ＥＷＳ実習室）
１号棟３Ｆ
PTU.12.0
PTU.12.254
R11
R12
PTU.16.254
１号棟１，２Ｆ
PTU.16.0
248.27
cbr080
PTU.8.254
R3
R4
PTU.56.254
248.21
cbr020
PTU.8.0
５号棟１，２Ｆ（電子計算機室、情報、福祉）
PTU.56.0
５号棟３Ｆ（パソコン実習室Ｉ）
本部棟
経理課（図書館端末）
PTU.44.0
PTU.44.254
R13
248.28
cbr090
PTU.32.254
R5
248.22
cbr030
PTU.32.0
４号棟１－３Ｆ，実４棟
実１棟
PTU.20.0
PTU.20.254
R14
R15
248.29
cbr100
開発研究部
PTU.52.0
PTU.52.254
INS64
R25
PTU.52.28
教材開発支援システム
248.23
cbr040
PTU.28.254
R6
PTU.28.0
３号棟１－３Ｆ
R19
R20
R21
PTU.64.0 建築系ＣＡＤ１
PTU.68.0 建築系ＣＡＤ２
PTU.72.0 建築系ＣＡＤ３
PTU.36.1
6Ｆ
新棟（情報）
PTU.156.0
PTU.156.254
R11-1
248.33
cbr140
PTU.36.254
R7
R8
PTU.40.254
248.24
cbr050
PTU.36.0
多目的ＣＡＤ
PTU.40.0
多目的ＣＡＤを除く
専用回線
WIDE
uitec5
uitec0
R24
DNS:PTU.40.4
5Ｆ
新棟（電子）
PTU.136.0
PTU.136.254
R12-2
R12-1
PTU.126.254
新棟（電気）
4Ｆ
PTU.128.0
248.32
cbr130
133.59.48.1
lib_h
DDX-P
学術情報センター
NACSIS-CAT,ILL
新棟（情報）
PTU.144.0
PTU.144.254
R13-1
248.31
cbr120
PTU.48.254
R9
PTU.48.0
図書館＆共通講義棟
図書館端末
PTU.8.253
248.25
cbr060
R22
R23
INS-C
INS-P
Ra-1
Ra-2
幕張
新棟（情報）
PTU.148.0
PTU.148.254
R14-1
R14-2
新棟（情報）
PTU.152.0
PTU.152.254
248.30
cbr110
PTU.24.254
R10
248.26
cbr070
PTU.24.0
２号棟２－６Ｆ，実２棟１－３Ｆ

（４） ルータによるインターネットワーキング

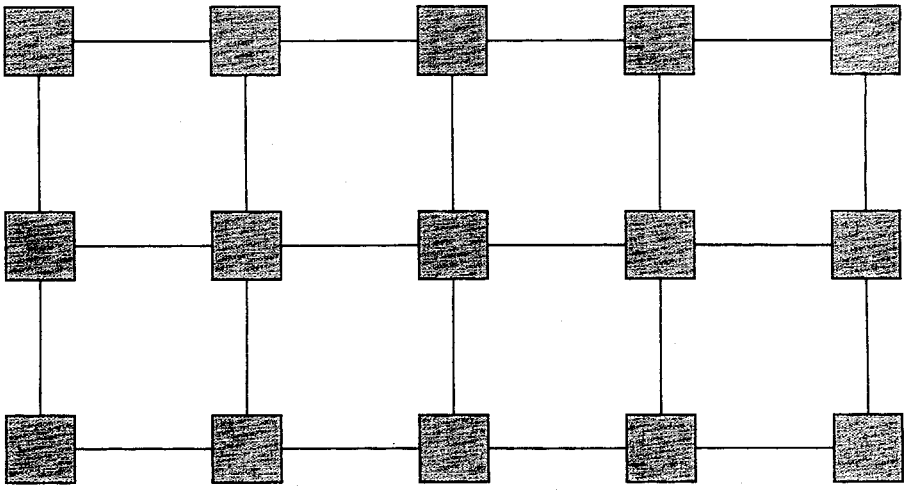
① パケット中継の仕組み



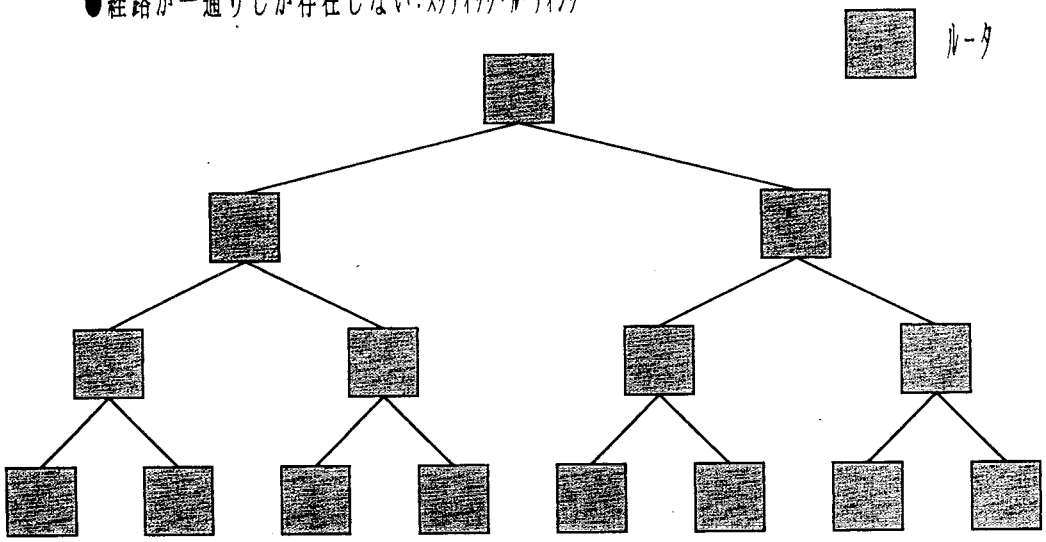
② ダイナミック・ルーティングとスタティック・ルーティング

●複数の経路が存在するｲﾝﾀｰﾈｯﾄ･ﾜｰｸ場合:ﾀﾞｲﾅﾐｯｸ･ﾙｰﾃｨﾝｸﾞ

-> ﾙｰﾀ間で経路情報を交換するﾌﾟﾛﾄｺﾙが必要となる：ＲＩＰ



●経路が一通りしか存在しない:ｽﾀﾃｨｯｸ･ﾙｰﾃｨﾝｸﾞ



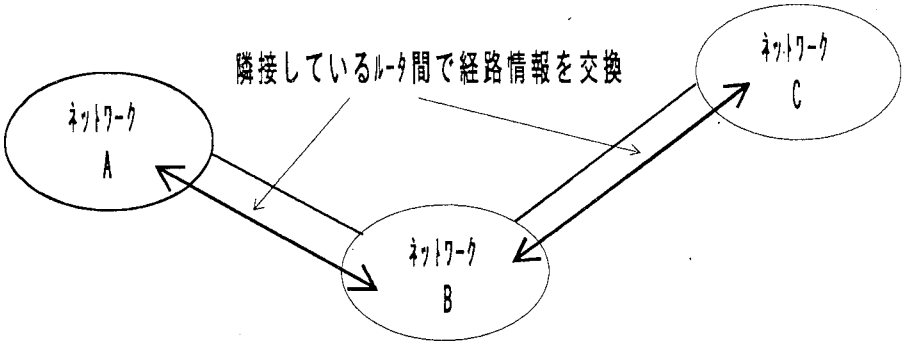
ＲＩＰ(Routing Information Protocol)について

ＲＩＰは、ＩＧＰ(Interior Gateway Protocols)として、現在最も広く使用されている。

●ネットワークを構成する全ルータはすべて対等な関係にある。

-> ネットワークを構成する上で物理的制約がない。
ただし、ルータを16個超えたネットワークはルーティング・テーブルに登録されない。

●ＲＩＰはディスタントベクター方式を採用している。



経路情報として、通信可能なﾈｯﾄﾜｰｸ･ｱﾄﾞﾚｽ、
そこに至るﾎｯﾌﾟ数、
そして、次に渡すﾙｰﾀのｱﾄﾞﾚｽを保持している。

●ＲＩＰは自分の持っている経路情報をそのまま配布する。

●経路情報の配布にＵＤＰのブロードキャストを用いる。

-> これを周期的に(３０秒に１回)行うことで、信頼性を確保している。

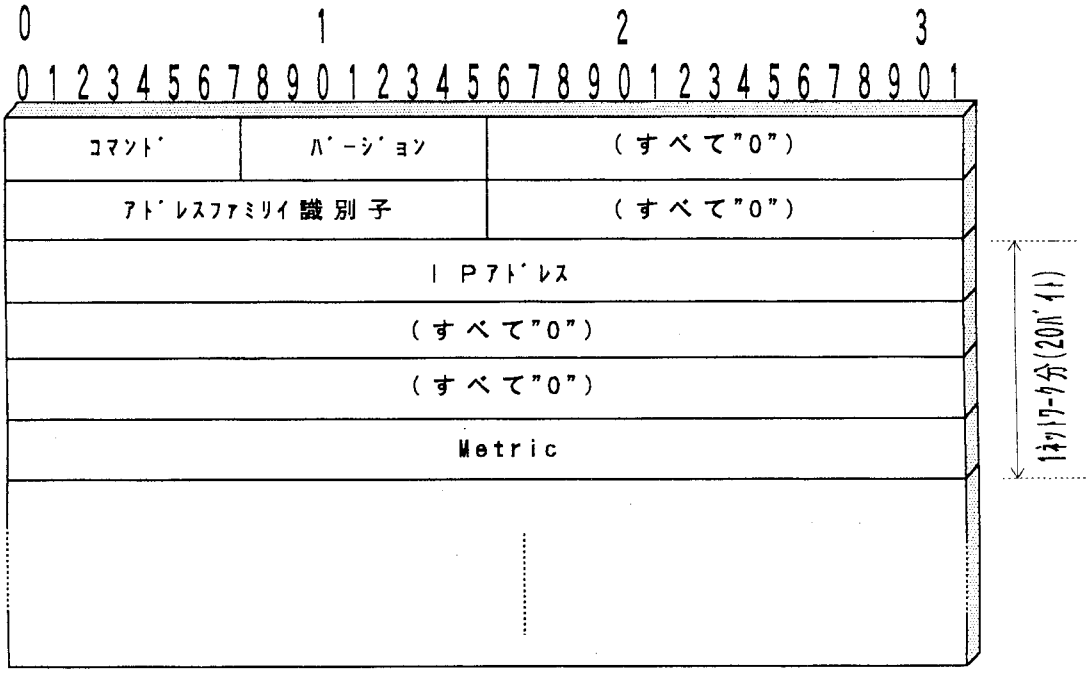
ＲＩＰの欠点

●経路情報が伝搬するのに時間がかかる。

●ネットワーク間の物理的な構造がループになっていると、ルーティング・ループが発生する可能性がある。

●隣接ネットワークに至るのに複数のルートが存在する場合、回線速度や通信速度て選択できない。

ＲＩＰのパケットフォーマット



コマンド：Request＝1, Reaponse＝2　　バージョン：現在は1

アドレス・ファミリイ識別子：2　　Ipアドレス：IPネットワーク番号

Metric　：IPアドレスに至るメトリック

●RIPの最大パケットサイズは、最大512バイト（1回の配布で最大25ネットワーク）

●Requestには、ブロードキャストを用いるがResponseはユニキャストを用いる。

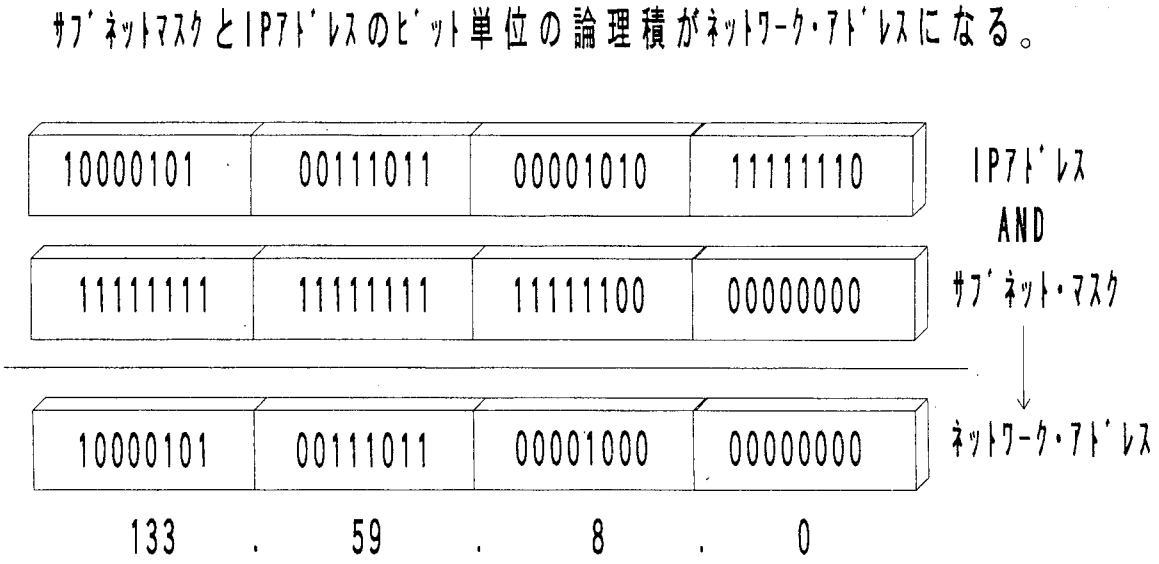
●RIPのパケットには、サブネットマスクの情報がない。-> 別ネットワークにはネットワークの情報のみを配布する。

●ルーティング情報を受信したインタフェースには、その情報（貰った経路情報）を流さない（Split Horizon）。

-> 隣接したルータが互いにNEXTルータとなるようなことがあるのを防止する。

●ルータは、TTLの値を1減じ、その結果TTLが0になったらパケットを破棄する。

③　サブネットマスク（ネットマスク）について

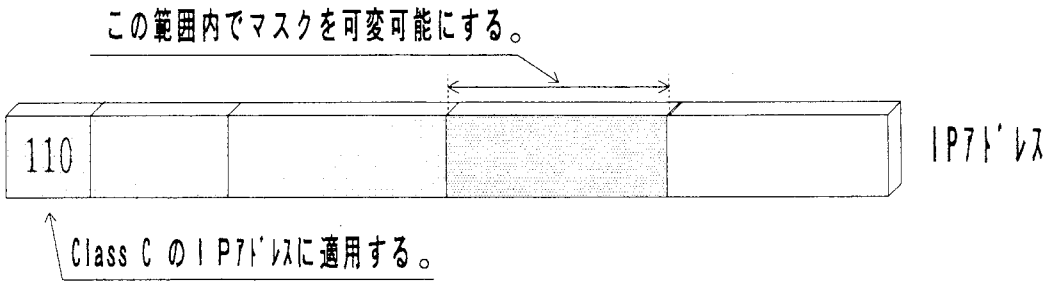


上記のIPアドレスはClassBのアドレスであるが、ホスト部のアドレスの一部をネットワーク・アドレスとして使用している。

通常（インターネット）は、ClassBのサブネットマスクに255.255.0.0を用いるが、上記の例のように1つのネットワークアドレスを複数のネットワーク・アドレスとして用いることができる。上記の場合は、64のネットワークのインターネットワークが可能である。

しかし、1つのネットワーク内のホストの数は、最大1022となる。

④ CIDR（Classless Inter－Domain Routing）について

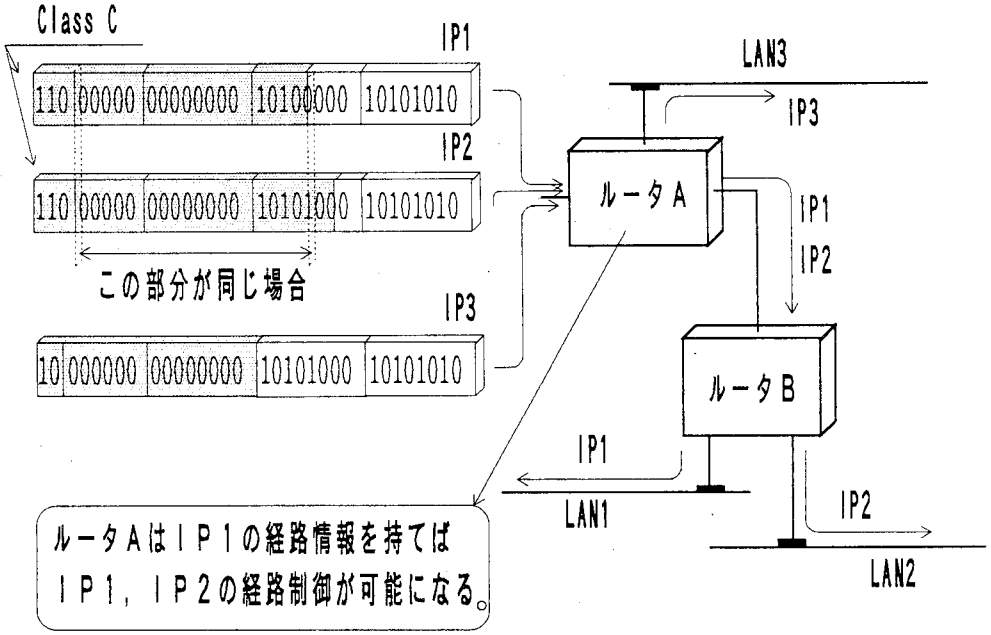


「ＣＩＤＲ化の目的」

●グローバル・アドレスの不足への対応

-> 主として、Class B の枯渇状態への対応

●ルーティング・テーブルの爆発への対応

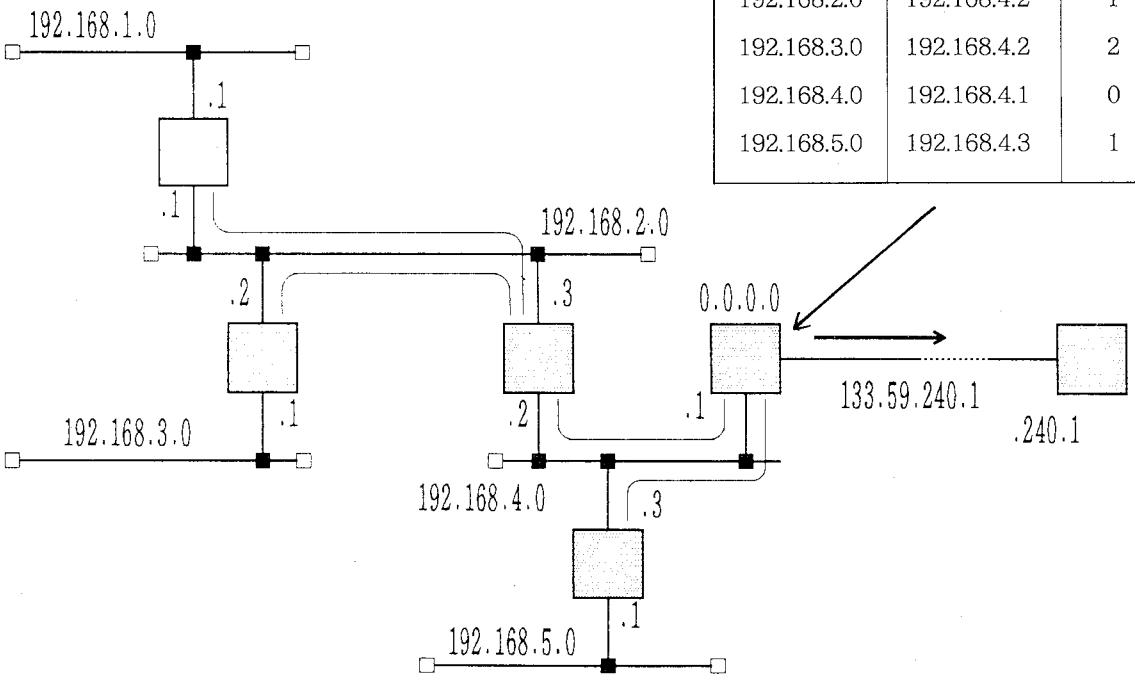


⑤　デフォルト・ルーティングについて

ルータはIPパケットを受信すると、そのパケットを転送するインタフェースをルーティング・テーブルから検索し、そのパケットを該当するインタフェースから送出する。もし、登録されていなければそのパケットを破棄する。

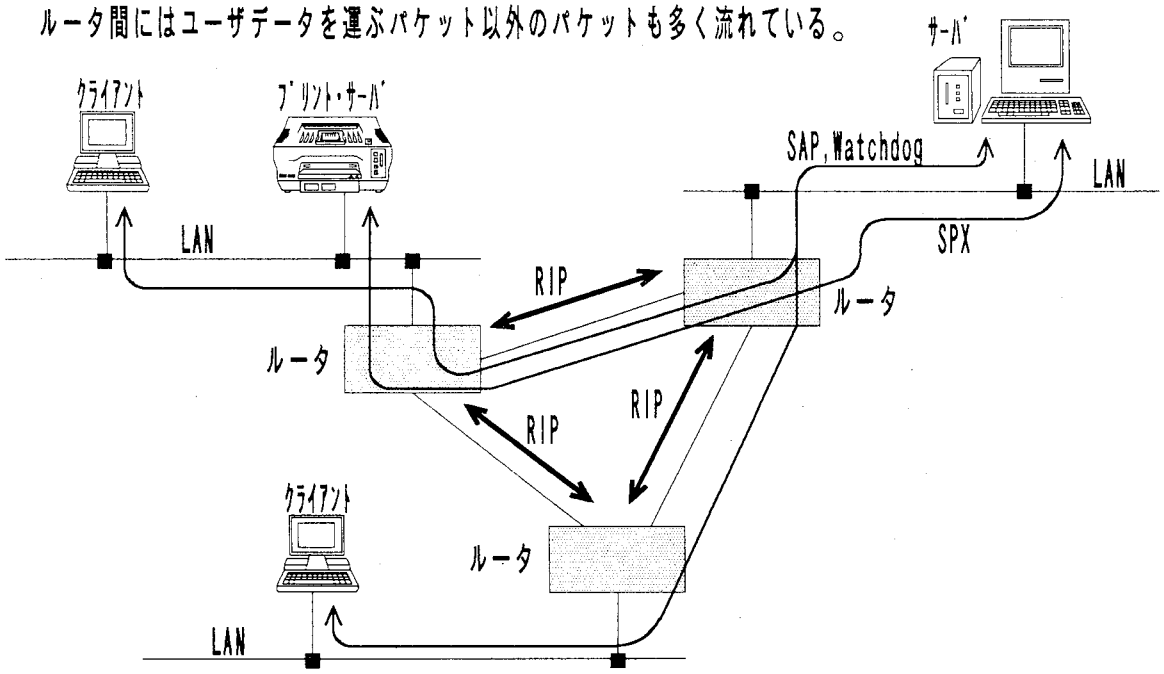
デフォルト・ルートとは、ルーティング・テーブルにないIPパケットの経路を示すものである。相手先ネットワークとして0.0.0.0を登録（管理者がスタティックな経路として登録）することによっている。デフォルト・ルートの場合のホップ数（Metric）は、デフォルト・ルートとして指定したルータまでのホップ数（通常ルーティングのホップ数は到達ネットワークまでの数）である。そして、このデフォルト・ルートの情報はRIPによって、インターネットワーク内に伝搬していく。-> ルーティング・テーブルへの登録数を減らす効果がある。

ネットワーク	Next Hop	Metric
0.0.0.0	133.59.240.1	1
192.168.1.1	192.168.4.2	2
192.168.2.0	192.168.4.2	1
192.168.3.0	192.168.4.2	2
192.168.4.0	192.168.4.1	0
192.168.5.0	192.168.4.3	1



⑥　ルータ間を流れるパケット（例：Netwareの場合）

ルータ間にはユーザデータを運ぶパケット以外のパケットも多く流れている。



●ＲＩＰ：ルータ間で受け渡す経路情報（６０秒に１回：ＩＰの場合は３０秒に１回）

●ＳＡＰ：サーバが稼働中であることをクライアントに知らせる（６０秒に１回）。

●Watchdog：一定時間以上サーバにアクセスしないクライアントに送信（６０秒に１回）

●プリント・サーバがサーバにプリント出力の有無を確認する（数秒に１回）。

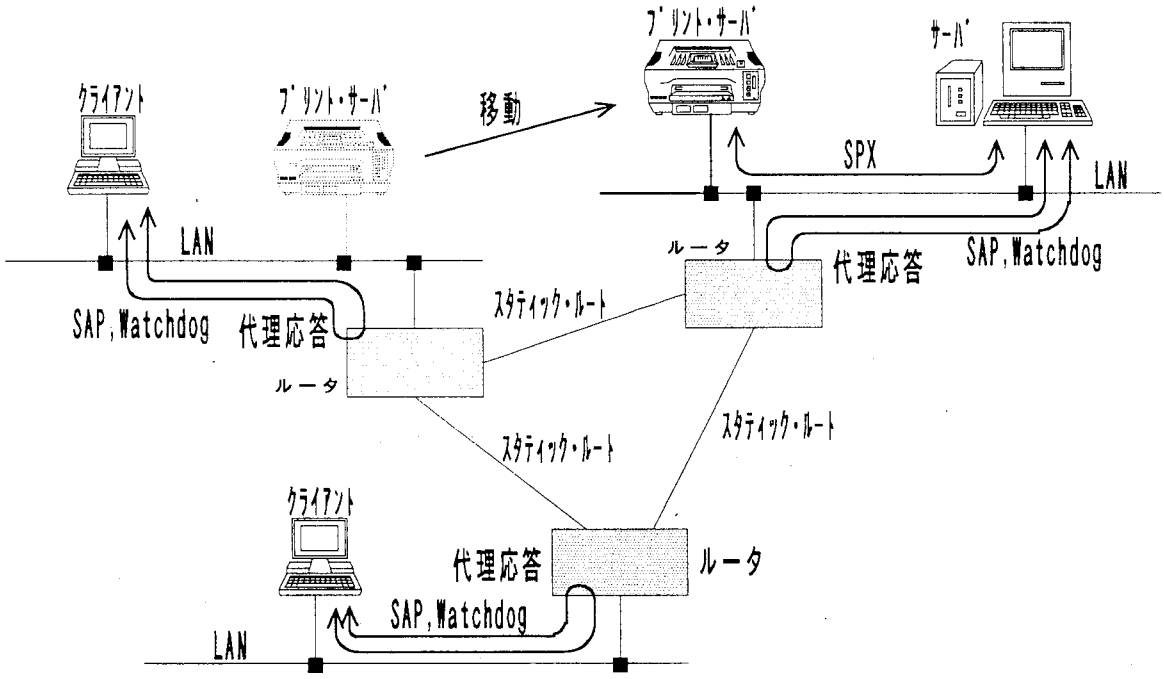
ルータ間の通信路に広域通信網（電話網、ＩＳＤＮ網、ＤＤＸ－Ｐ網等）を利用すると、

{ 回線交換の場合　：回線が切れない。
{ ﾊﾟｹｯﾄ交換の場合：ユーザパケット以外のパケット（上記のＲＩＰ等）で課金される。

「対策」

ユーザパケット以外のパケットはルータ間の通信路に流さない。

ルータ間を流れるパケットの抑制（例：Netwareの場合）



●ＲＩＰ：ルータがフィルタ（代理応答）をする(ＩＰの場合はｽﾀﾃｨｯｸ･ﾙｰﾃｨﾝｸﾞにする。)。

●ＳＡＰ：ルータがフィルタし代理応答をする。

●Watchdog：ルータがフィルタし代理応答をする。

●プリント・サーバがサーバにプリント出力の有無を確認する（数秒に１回）。
ＳＰＸを使用しているため、フィルタや代理応答ができない。
-> サーバとプリント・サーバを同一のＬＡＮに配置する。

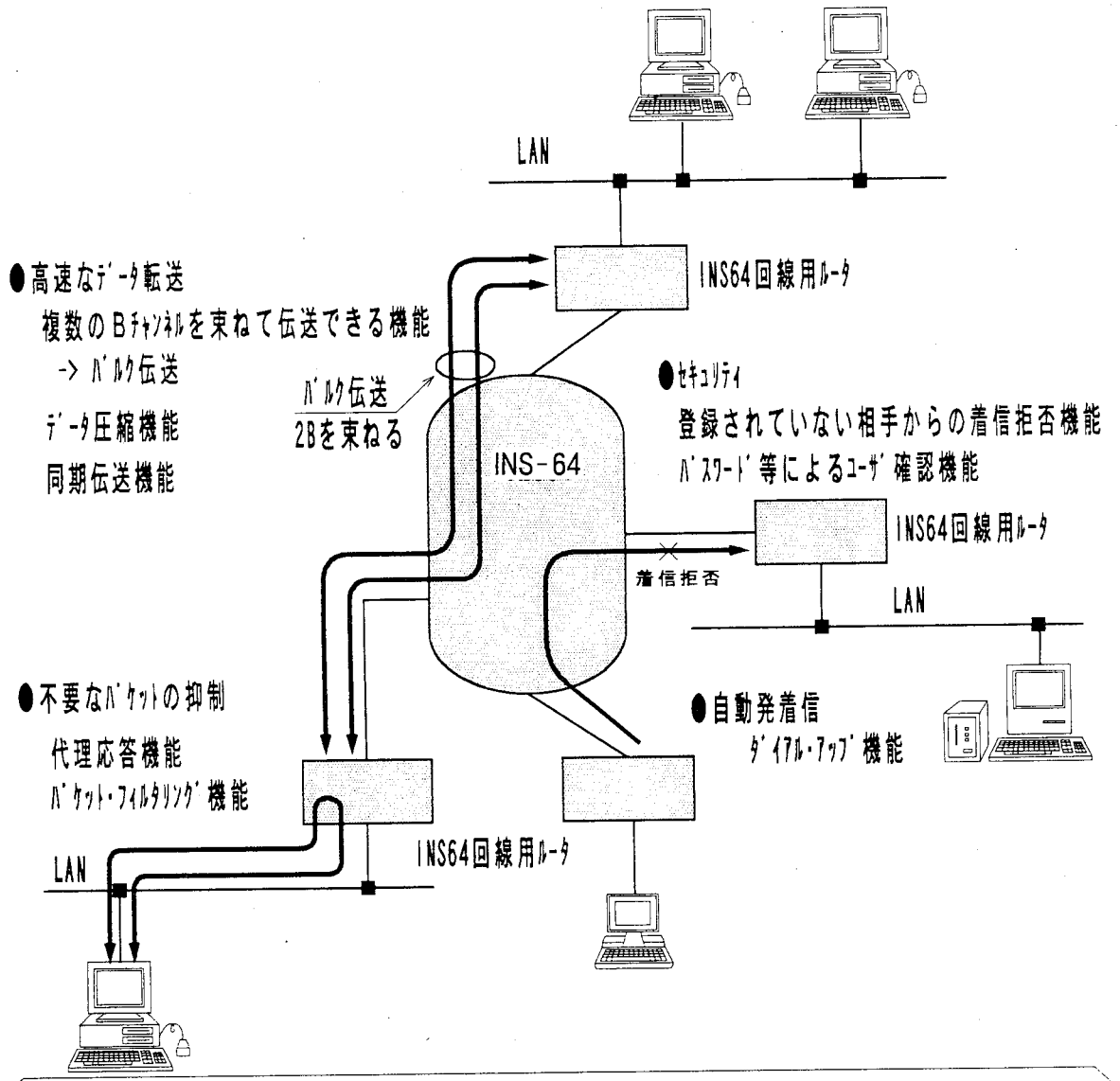
●ツールやデータの配置を工夫する。
例えば、SYSCON、PCONSOLE、NETXなど使用頻度は高いが頻繁に更新しないファイルをクライアント側に置く。

●クライアント・シェル(NETX.COM等)の起動とログイン

NetwareなどのＬＡＮ用ソフトウエアは、常時使用できる高速の通信路（ネットワーク）が前提で作られている。広域の公衆網を通信路に用いるには、現在のところ限界がある。

⑦　ISDNを用いたLAN間接続で考慮すべき機能

主としてINS-64のBチャンネルを用いる場合



「異機種間接続に当たって」

異機種間の相互接続に当たって、下記の例で示すように、現在のところ限界がある。

△ データ圧縮に関しては、種々の方式がある。

△ バルク伝送に関しては、ハンドシェイクの方式やまとめ方の方法が異なる場合がある。

△ スキュリティに関しても種々の方法（発信者番号通知機能や接続時のＩＤ確認など）がある。

LAN／WANに関する演習問題

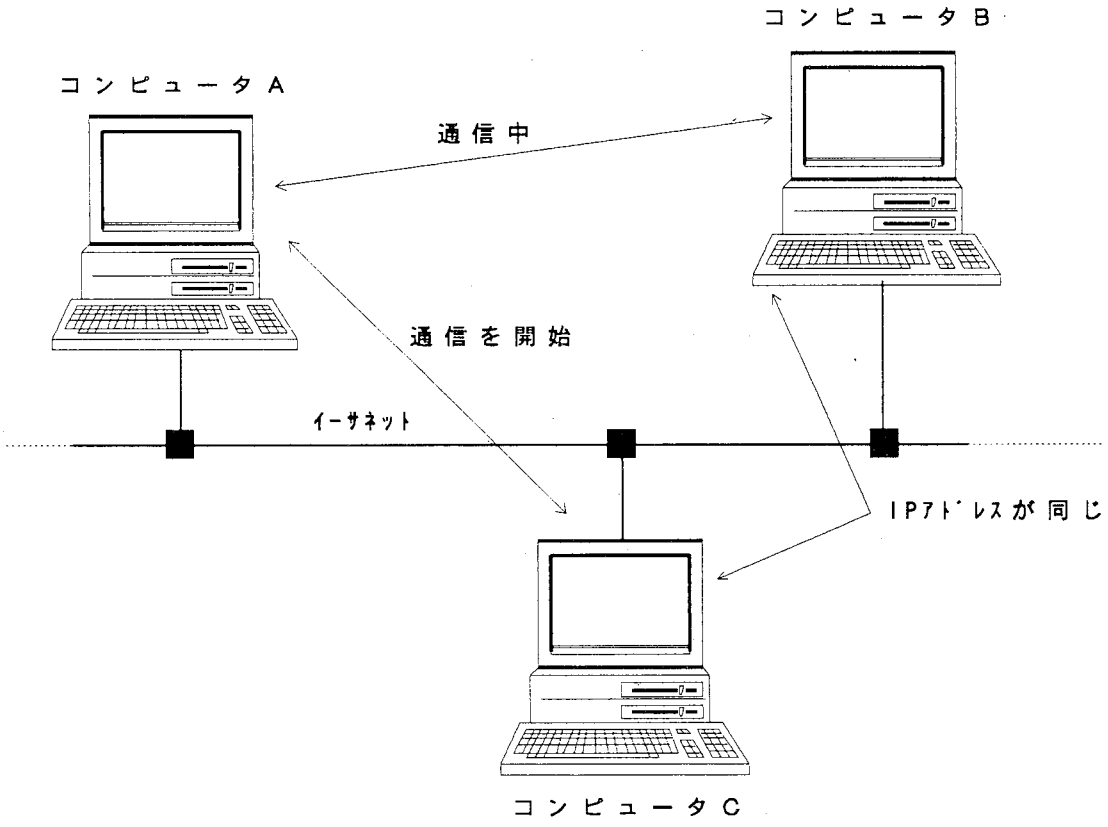
問1　下図において、以下の条件で各コンピュータを動作させた場合、どの様な現象が起きるでしょうか、また、その現象に至る過程について示しなさい。なお、コンピュータＡは多重処理が可能であるとする。

〈条件〉

（1）　コンピュータＢとＣのIPアドレスが同じ。

（2）　コンピュータＡとＢがLANを介して通信を行なっている。

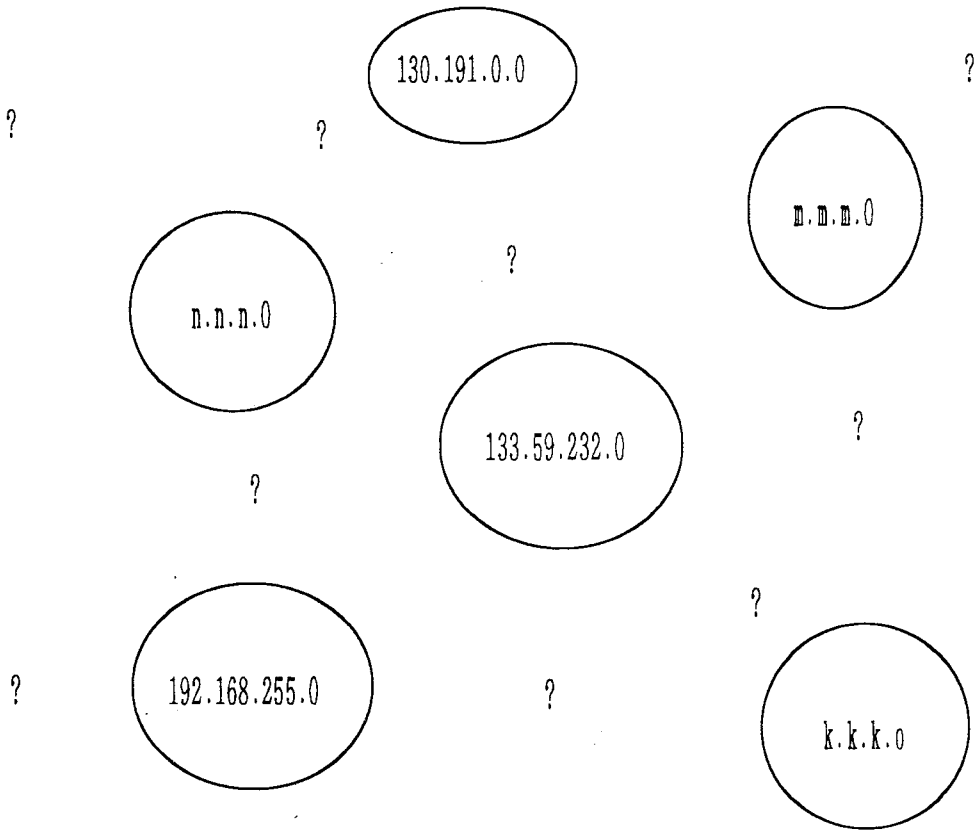
（3）　コンピュータＣの電源を投入し、コンピュータＡとの通信を開始した。



問2　参考資料に例を示したルーティング・テーブル（その1からその3）から推定できるインターネットのトポロジイを図を用いて示して下さい。また、これらのルーティング・テーブルにおいて矛盾があればご指摘下さい。

なお、各ルーティング・テーブルの最下行の文字列は、ルータやホスト（コンピュータ）のニックネイムである。また、メトリックの情報がないテーブルがホストである（ネットワーク・インターフェースを2個以上持っているホストはルータとしても機能している。）。

さらに、133.59.x.xのIPネットワーク・アドレスは、ホスト・アドレスの最上位桁から6ビットをネットワーク・アドレスの一部（サブネット化）にしている。



ルーティング・テーブルの例　その１

Destination	Mtr	Next Hop	T/P	Age	IF
130.191.0.0	1	130.191.254.254	D/L	112605	2
133.59.8.0	2	133.59.232.254	R/R	11	1
133.59.40.0	1	133.59.232.254	R/L	112610	1
133.59.48.0	2	133.59.232.254	R/R	14	1
133.59.232.0	1	133.59.232.253	D/L	112611	1
133.59.240.0	2	133.59.232.254	R/R	17	1
133.59.248.0	4	133.59.232.254	R/R	19	1
192.168.10.0	8	130.191.200.200	R/R	15	2
192.168.20.0	5	130.191.200.200	R/R	16	2
192.168.30.0	2	130.191.200.200	R/R	18	2
192.168.31.0	5	130.191.200.200	R/R	19	2
192.168.40.0	6	130.191.200.200	R/R	21	2
192.168.50.0	10	130.191.200.200	R/R	22	2
192.168.60.0	11	130.191.200.200	R/R	24	2
192.168.61.0	11	130.191.200.200	R/R	25	2
192.168.100.0	2	130.191.200.200	R/R	27	2
192.168.200.0	3	130.191.200.200	R/R	28	2
192.168.255.0	2	133.59.232.2	R/R	18	1

ROUTER7F004:

Destination	Mtr	Next Hop	T/P	Age	IF
130.191.0.0	2	133.59.232.253	R/R	16	1
133.59.8.0	1	133.59.240.254	R/L	112553	2
133.59.40.0	1	133.59.240.254	R/L	112554	2
133.59.48.0	1	133.59.240.254	R/L	112556	2
133.59.232.0	1	133.59.232.254	D/L	112552	1
133.59.240.0	1	133.59.240.253	D/L	112552	2
133.59.248.0	3	133.59.240.254	R/L	112559	2
192.168.10.0	9	133.59.232.253	R/R	25	1
192.168.20.0	6	133.59.232.253	R/R	26	1
192.168.30.0	3	133.59.232.253	R/R	27	1
192.168.31.0	6	133.59.232.253	R/R	29	1
192.168.40.0	7	133.59.232.253	R/R	0	1
192.168.50.0	11	133.59.232.253	R/R	1	1
192.168.60.0	12	133.59.232.253	R/R	2	1
192.168.61.0	12	133.59.232.253	R/R	3	1
192.168.100.0	3	133.59.232.253	R/R	5	1
192.168.200.0	4	133.59.232.253	R/R	6	1
192.168.255.0	2	133.59.232.2	R/R	18	1

ROUTER7F006:

ルーティング・テーブルの例　その２

```
Destination      Gateway          Flags     Refs     Use  Interface
127.0.0.1        127.0.0.1        UH           4    6331  lo0
133.59.248       133.59.232.254   UG           0       0  lan0
133.59.240       133.59.232.254   UG           0     375  lan0
133.59.48        133.59.232.254   UG           0       0  lan0
133.59.40        133.59.232.254   UG           0       0  lan0
133.59.8         133.59.232.254   UG           1     531  lan0
192.168.200      133.59.232.253   UG           0    3443  lan0
192.168.40       133.59.232.253   UG           0     951  lan0
133.59.232       133.59.232.1     U           10   18763  lan0
192.168.50       133.59.232.253   UG           0    1068  lan0
192.168.10       133.59.232.253   UG           0    5971  lan0
192.168.100      133.59.232.253   UG           0    2260  lan0
192.168.60       133.59.232.253   UG           0     377  lan0
192.168.20       133.59.232.253   UG           0    1129  lan0
192.168.61       133.59.232.253   UG           0    1490  lan0
192.168.30       133.59.232.253   UG           1    9068  lan0
192.168.255      133.59.232.2     UG           0     377  lan0
192.168.31       133.59.232.253   UG           0    1530  lan0
130.191          133.59.232.253   UG           0   11902  lan0
hp7f002 26:
```

```
Destination      Gateway          Flags     Refs     Use  Interface
127.0.0.1        127.0.0.1        UH           0      28  lo0
192.168.200      133.59.232.253   UG           0       0  lan0
192.168.40       133.59.232.253   UG           0       0  lan0
133.59.248       133.59.232.254   UG           0       0  lan0
133.59.240       133.59.232.254   UG           0       0  lan0
133.59.48        133.59.232.254   UG           0       0  lan0
133.59.40        133.59.232.254   UG           0       0  lan0
133.59.8         133.59.232.254   UG           1      78  lan0
133.59.232       133.59.232.2     U            0   16935  lan0
192.168.50       133.59.232.253   UG           0       0  lan0
192.168.10       133.59.232.253   UG           0       0  lan0
192.168.100      133.59.232.253   UG           0       0  lan0
192.168.60       133.59.232.253   UG           0       0  lan0
192.168.20       133.59.232.253   UG           0       0  lan0
192.168.61       133.59.232.253   UG           0       0  lan0
192.168.30       133.59.232.253   UG           0       0  lan0
192.168.31       133.59.232.253   UG           0       0  lan0
130.191          133.59.232.253   UG           0   16383  lan0
192.168.255      192.168.255.3    U            0       0  lan1
sv7f001 23:
```

ルーティング・テーブルの例　その３

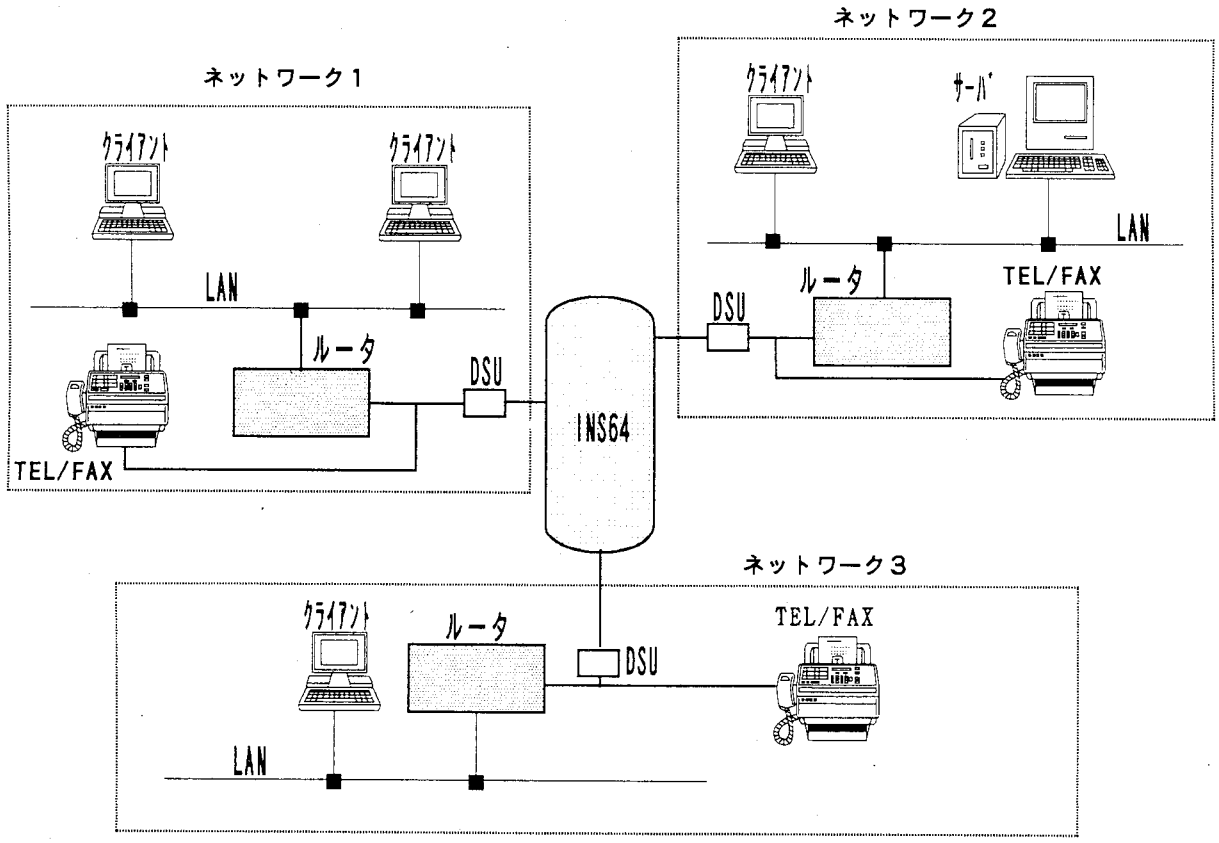
Destination	Mtr	Next Hop	T/P	Age	IF
130.191.0.0	1	133.59.240.253	R/L	1499420	3
133.4.0.0	5	133.59.48.254	R/R	11	1
133.59.4.0	3	133.59.48.254	R/R	12	1
133.59.8.0	3	133.59.48.254	R/R	13	1
133.59.12.0	3	133.59.48.254	R/R	15	1
133.59.16.0	3	133.59.48.254	R/R	16	1
133.59.20.0	3	133.59.48.254	R/R	17	1
133.59.24.0	3	133.59.48.254	R/R	18	1
133.59.28.0	3	133.59.48.254	R/R	20	1
133.59.32.0	3	133.59.48.254	R/R	21	1
133.59.36.0	3	133.59.48.254	R/R	22	1
133.59.40.0	3	133.59.48.254	R/R	23	1
133.59.44.0	3	133.59.48.254	R/R	25	1
133.59.48.0	1	133.59.48.253	D/L	1499430	1
133.59.52.0	3	133.59.48.254	R/R	27	1
133.59.56.0	3	133.59.48.254	R/R	28	1
133.59.60.0	3	133.59.48.254	R/R	0	1
133.59.64.0	4	133.59.48.254	R/R	1	1
133.59.68.0	4	133.59.48.254	R/R	2	1
133.59.72.0	4	133.59.48.254	R/R	5	1
133.59.232.0	1	133.59.240.253	R/L	1499476	3
133.59.240.0	1	133.59.240.254	D/L	1499469	3
133.59.244.0	1	133.59.244.254	D/L	1499465	2
133.59.248.0	2	133.59.48.254	R/R	10	1
192.168.10.0	4	133.59.48.254	R/R	11	1
192.168.255.0	1	133.59.244.252	R/L	1499482	2
SAGAMIHARA:					

問3　下図のインターネットワーク上で、Aさんがネットワーク1のクライアントからネットワーク2のサーバにログインした直後、Aさんに来客があり、Aさんはサーバにログインしたまま10分程度来客に応対した。

その後、サーバ上にある数キロバイト程度のファイルをクライアント側にコピーしようとしたが、いくら待ってもコピーが終わらなかった。このような状況に陥った原因としてどのようなことが考えられるでしょうか。また、その原因に至る根拠は何ですか。

ただし、INS64回線は1回線で、ダイヤルインになっていて、Bチャンネルの1つは常にFAXや電話用に用いられている。つまり、ルータ間の通信路には1つのBチャンネルのみが使用可能であるとする。

また、全ルータとも、無線通信時間が60分以上の場合、回線を切断する設定になっていて、さらに、RIP, SAP等の制御パケットは、ルータがフィルタリングや代理応答する設定になっている(ユーザパケット以外はルータ間に流れない。)。



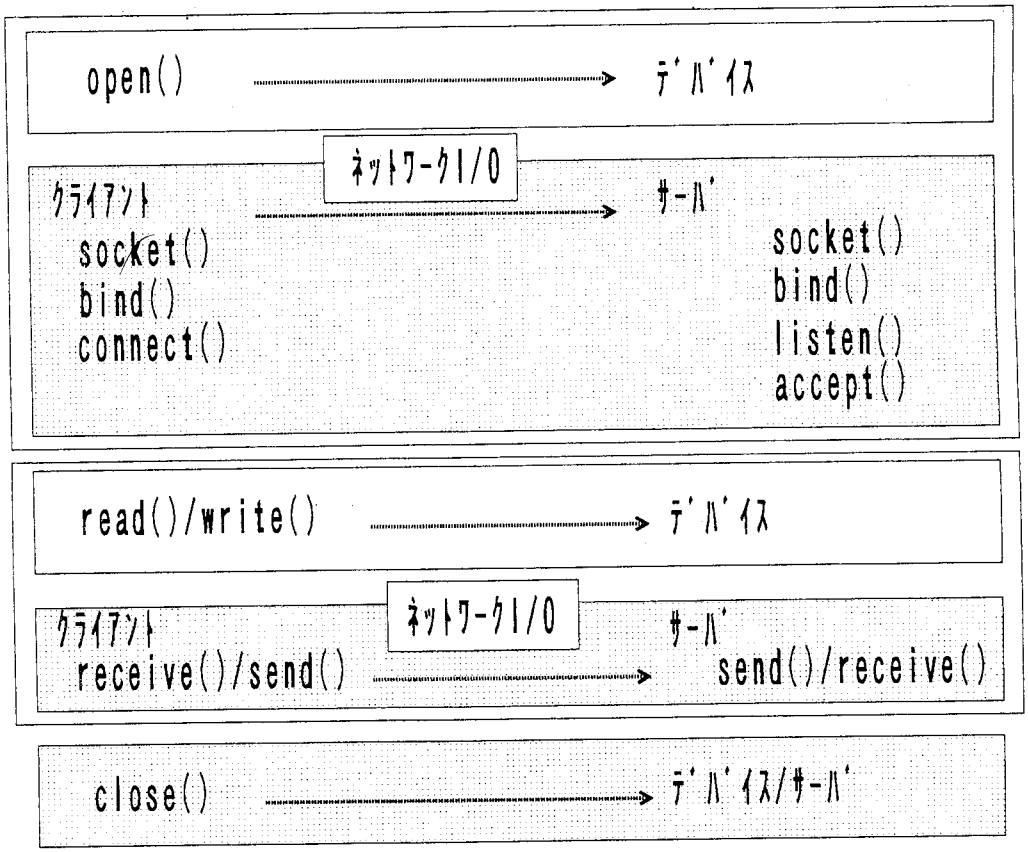
4．ネットワークプログラミングとＡＰＩ

（1）ソケット（socket）インタフェース

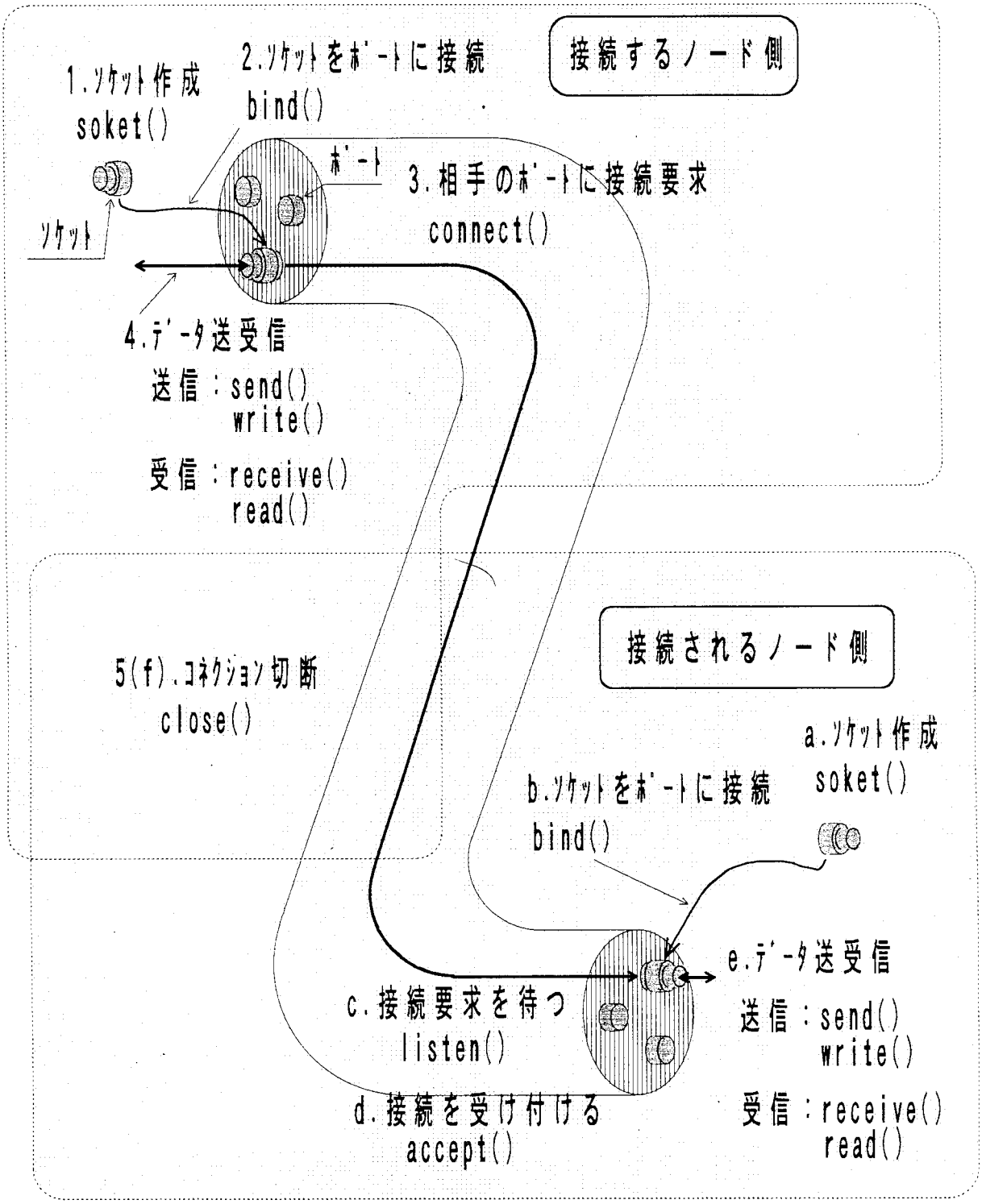
ｱﾌﾟﾘｹｰｼｮﾝ・ﾌﾟﾛｸﾞﾗﾑとﾈｯﾄﾜｰｸ・ﾌﾟﾛﾄｺﾙとのｲﾝﾀﾌｪｰｽを提供する
BSD系UNIXのﾈｯﾄﾜｰｸI/O用ﾌﾟﾛｸﾞﾗﾑ

UNIXのI/Oｵﾍﾟﾚｰｼｮﾝはすべて、open-close-read-writeに統一されているが、ﾈｯﾄﾜｰｸI/Oの場合は複雑であるため、新たなI/Oｲﾝﾀﾌｪｰｽ(ｼｽﾃﾑｺｰﾙ)を追加した。

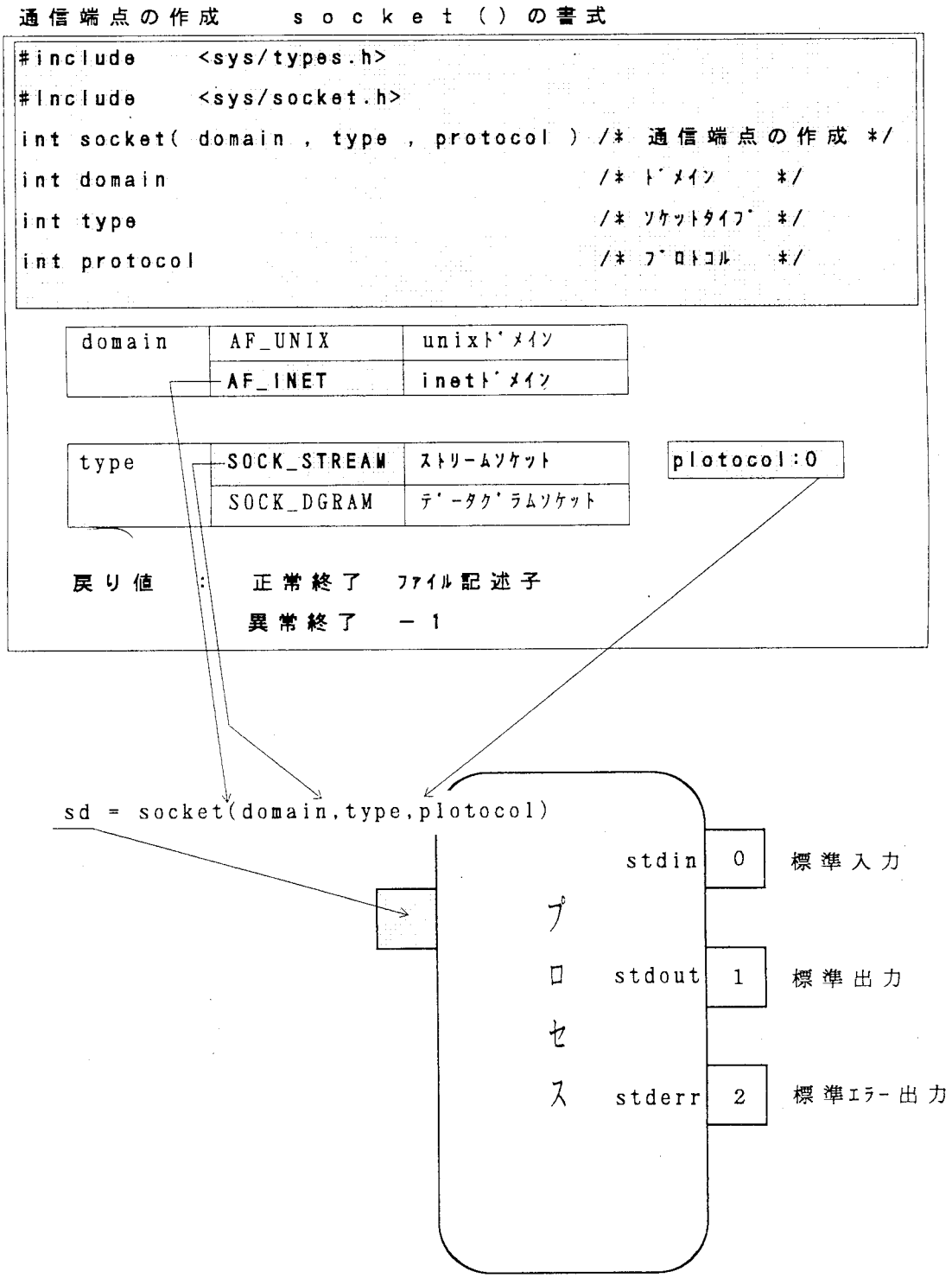
ｻｰﾊﾞ（ｻｰﾋﾞｽを受け付ける側）・ｸﾗｲｱﾝﾄ（ｻｰﾋﾞｽを要求する側）方式でﾉｰﾄﾞ間をｲﾝﾀﾌｪｰｽする。



ソケット・インタフェースを用いたネットワーク・ノード間のインタフェース（ストリームタイプ



（2） ソケットインタフェース関連システムコール



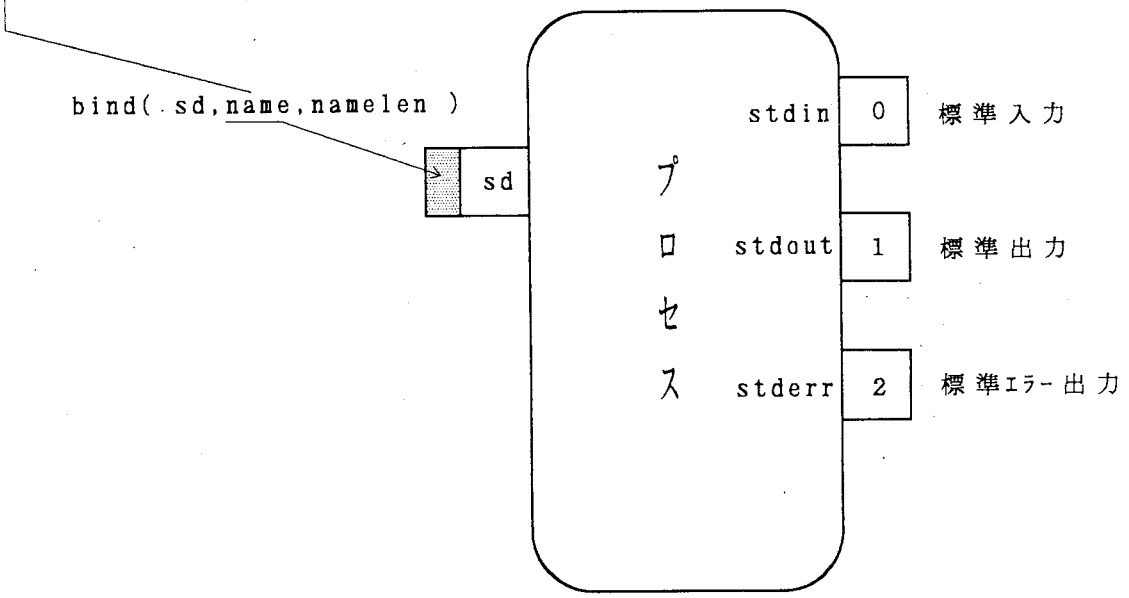
ソケットと名前の結合　　ｂｉｎｄ（）の書式

```
#include     <sys/types.h>
#include     <sys/socket.h>
#include     <netinet/in.h>
int bind( sd,name,namelen ) /* ソケットと名前の結合 */
int sd                      /* ソケットに対するファイル記述子 */
struct sockaddr_in *name    /* ソケットのアドレス情報へのポインタ */
int namelen                 /* ソケットのアドレス情報の長さ */
```

戻り値　：　正常終了　　０
　　異常終了　－１

/usr/include/netinet/in.h

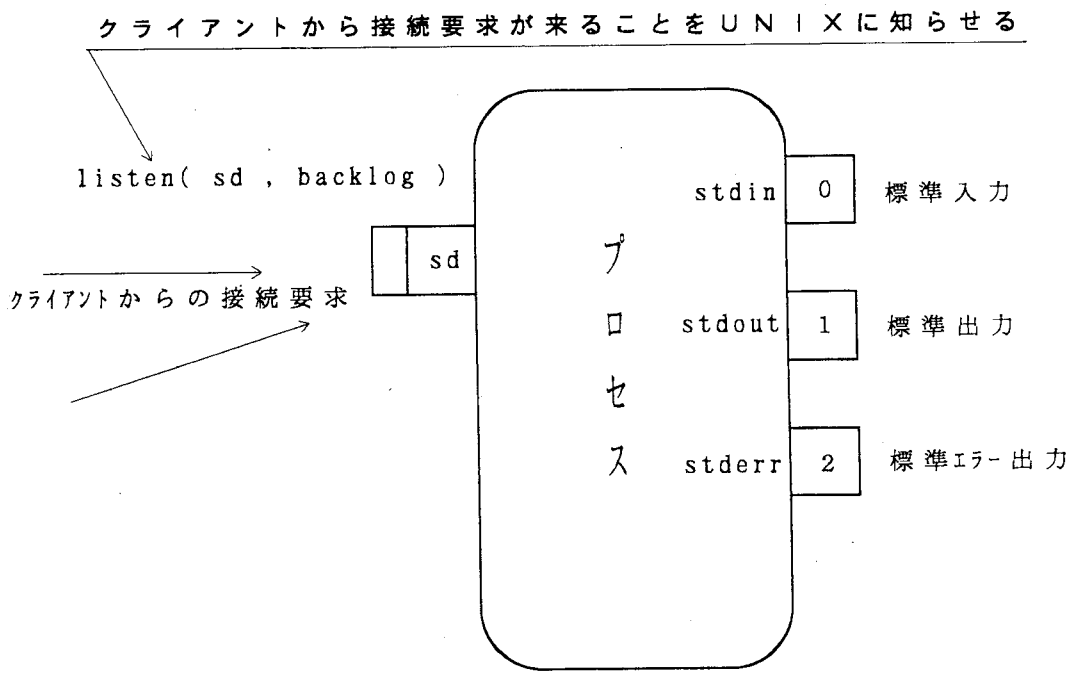
```
struct in_addr {
        u_long s_addr;
};
struct sockaddr_in {
        short   sin_family;--------->AF_INET
        u_short sin_port;----------->ポート番号
        struct  in_addr sin_addr;-->IPアドレス
        char    sin_zero[8];
};
```



ソケットの接続要求の受け入れ　　listen()の書式

```
#include    <sys/types.h>
#include    <sys/socket.h>
int listen( sd ,backlog )    /* ソケットの接続要求の受け入れ */
int sd                       /* ソケットに対するファイル記述子 */
int backlog                  /* 接続要求の待ち行列の最大長 */
```

戻り値　：　正常終了　0
　　異常終了　－1



ソケットの接続許可　　accept()の書式

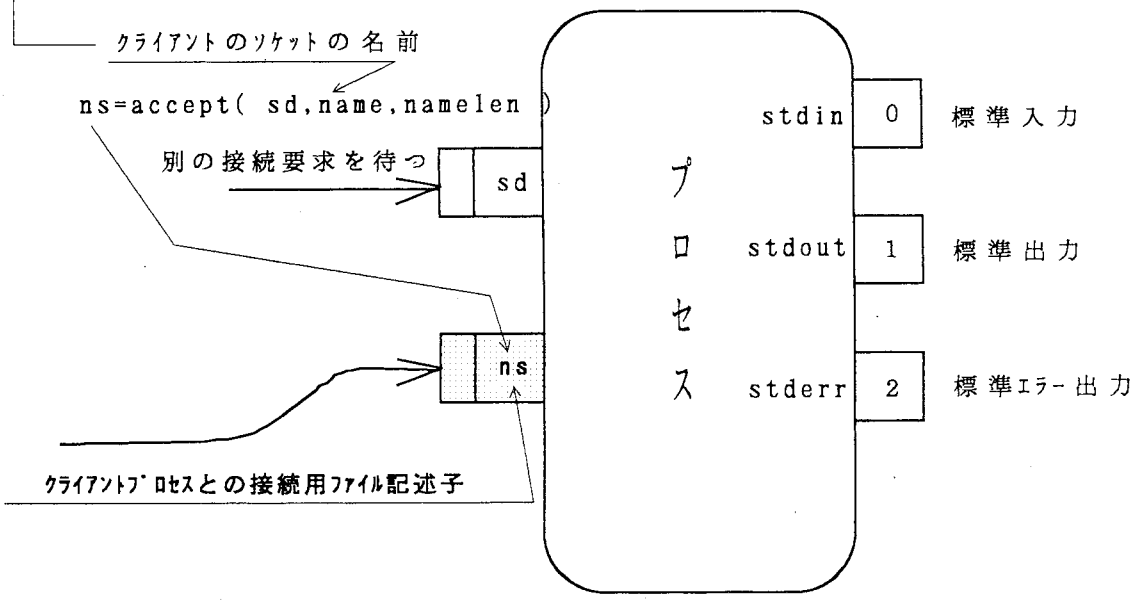
```
#include     <sys/types.h>
#include     <sys/socket.h>
#include     <netinet/in.h>
int accept( sd,name,namelen ) /* ソケットの接続許可 */
int sd                        /* ソケットに対するファイル記述子 */
struct sockaddr_in *name      /* クライアントのソケットアドレス情報へのポインタ */
int namelen                   /* クライアントのソケットアドレス情報の長さ */
```

戻り値　：　正常終了　クライアントと接続するファイル記述子

異常終了　－1

/usr/include/netinet/in.h

```
struct in_addr {
        u_long s_addr;
}
struct sockaddr_in {
        short   sin_family;--------->AF_INET
        u_short sin_port;----------->ポート番号
        struct  in_addr sin_addr;-->IPアドレス
        char    sin_zero[8];
}
```



ソケットの接続　　connect()の書式

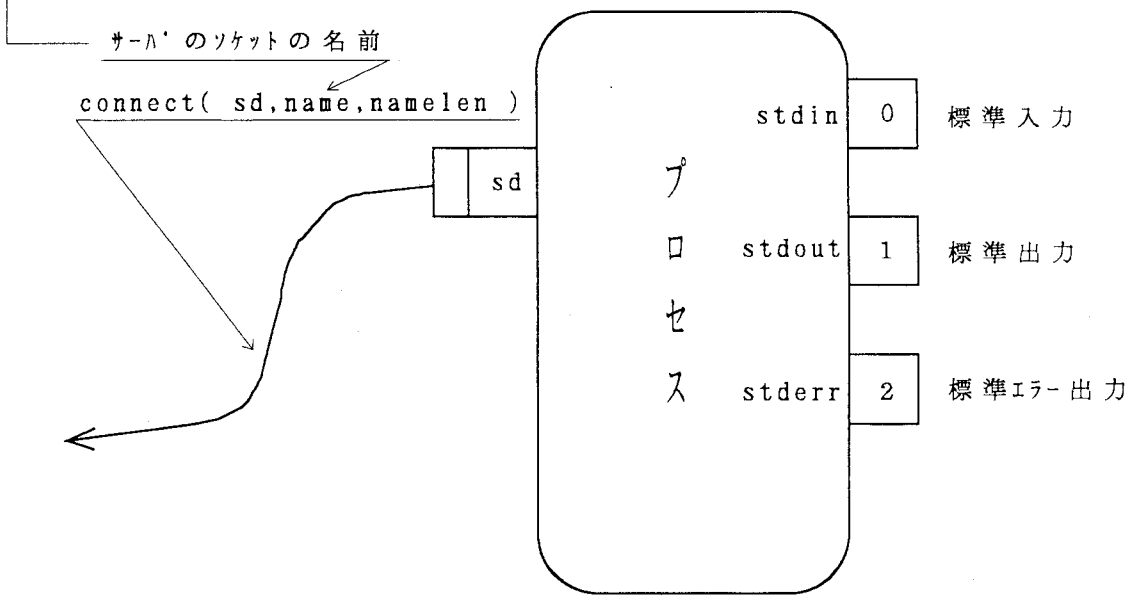
```
#include     <sys/types.h>
#include     <sys/socket.h>
#include     <netinet/in.h>
int connect( sd,name,namelen ) /* ソケットの接続 */
int sd                         /* ソケットに対するファイル記述子 */
struct sockaddr_in *name       /* サーバのソケットアドレス情報へのポインタ */
int namelen                    /* サーバのソケットアドレス情報の長さ */
```

戻り値　：　正常終了　0

異常終了　－1

/usr/include/netinet/in.h

```
struct in_addr {
        u_long s_addr;
}
struct sockaddr_in {
        short   sin_family;--------->AF_INET
        u_short sin_port;----------->ポート番号
        struct  in_addr sin_addr;-->IPアドレス
        char    sin_zero[8];
}
```

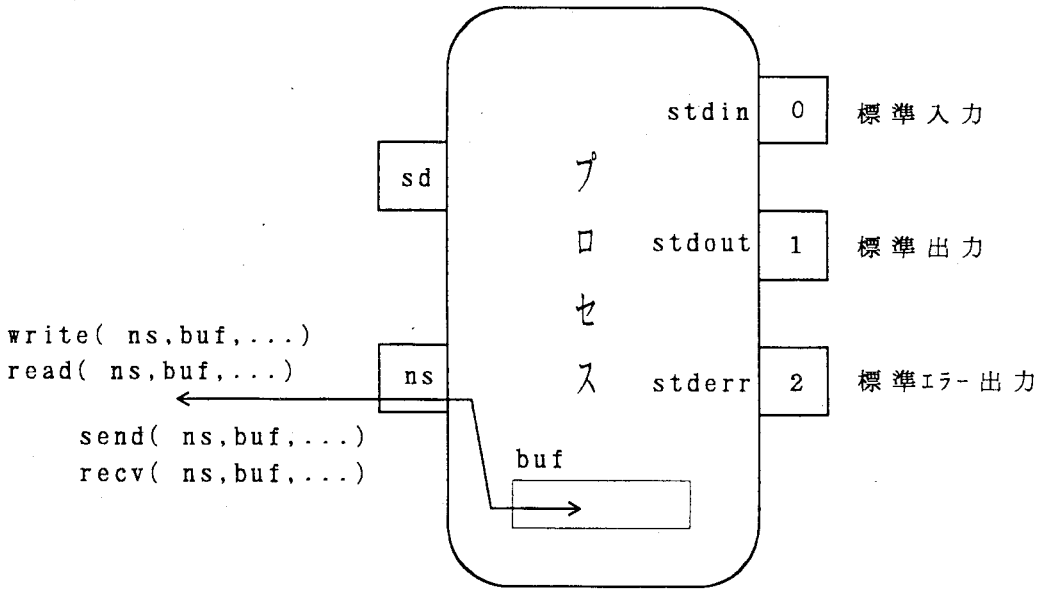


データの送受信　　write, read, send, recvの書式

```
int write( s , buf , npyte )          /* データの送信 */
int read( s , buf , npyte )           /* データの受信 */
int send( s , buf , npyte , flags )   /* データの送信 */
int recv( s , buf , npyte , flags )   /* データの受信 */
int       s                           /* ﾌｧｲﾙ記述子 */
char     *buf                         /* 送受信ﾊﾞｯﾌｧのﾎﾟｲﾝﾀ */
unsigned nbyte                        /* 送受信ﾊﾞｲﾄ数 */
unsigned flags                        /* 送受信ｵﾌﾟｼｮﾝ
                                         ｵﾌﾟｼｮﾝなしの場合0 */
```

戻り値　：　正常終了　送受信したバイト数

異常終了　－１



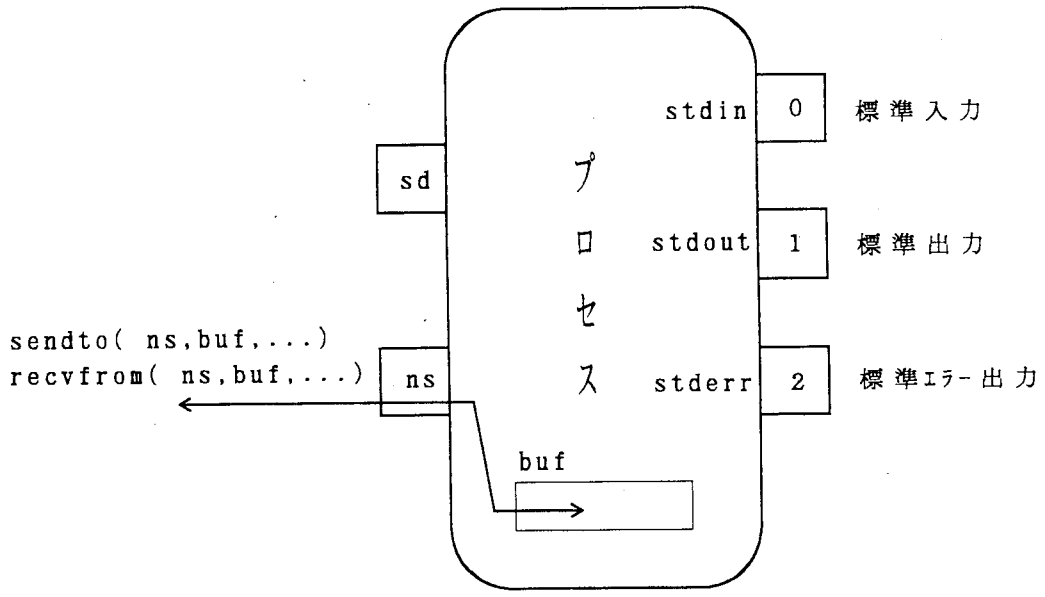
データの送受信　　ｓｅｎｄｔｏ，ｒｅｃｖｆｒｏｍの書式

```
#include <sys/types.h>
#include <sys/socket.h>
int sendto( s , buf , npyte , flags , to , tolen )    /* 送信 */
int recvfrom( s , buf , npyte , flags , to , tolen ) /* 受信 */
int      s                          /* ﾌｧｲﾙ記述子 */
char     *buf                       /* 送受信ﾊﾞｯﾌｧのﾎﾟｲﾝﾀ */
unsigned nbyte                      /* 送受信ﾃﾞｰﾀのﾊﾞｲﾄ数 */
unsigned flags                      /* 送受信ｵﾌﾟｼｮﾝ :なしの場合0 */
struct sockadder *to                /* 相手先ｱﾄﾞﾚｽ構造体のﾎﾟｲﾝﾀ */
int      tolen                      /* ｱﾄﾞﾚｽ構造体のｻｲｽﾞ */
```

戻り値　：　正常終了　送受信したバイト数

　　異常終了　－１

recvfromの場合は tolen のﾎﾟｲﾝﾀｰを指定する。



ホスト情報の取得　　ｇｅｔｈｏｓｔｂｙｎａｍｅの書式

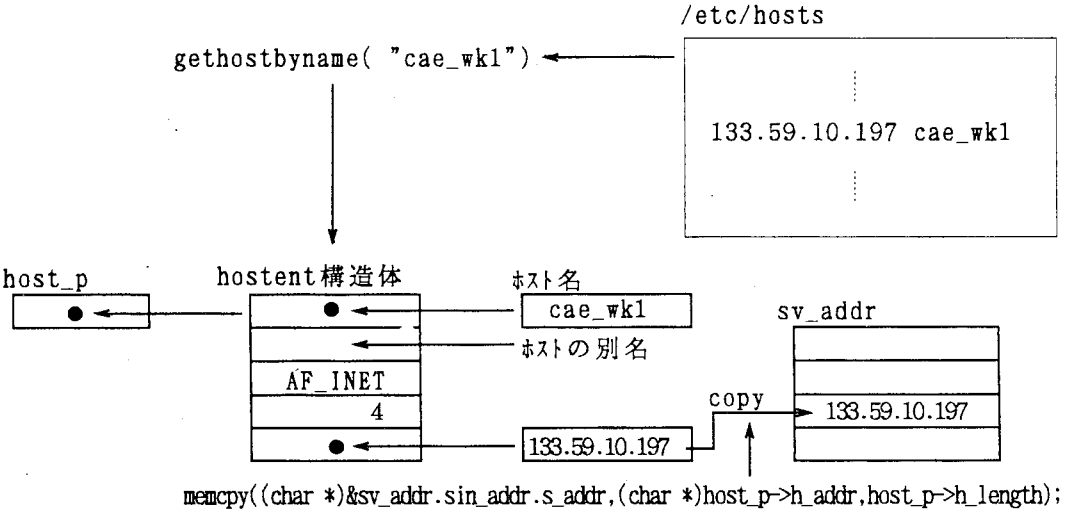
```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
struct hostent *gethostbyname(  name  )
char    *name                          /* ﾎｽﾄ名が格納されている。
                                          文字配列へのﾎﾟｲﾝﾀ */
```

戻り値　：　正常終了　hostent構造体へのポインタ

異常終了　－１

[例]

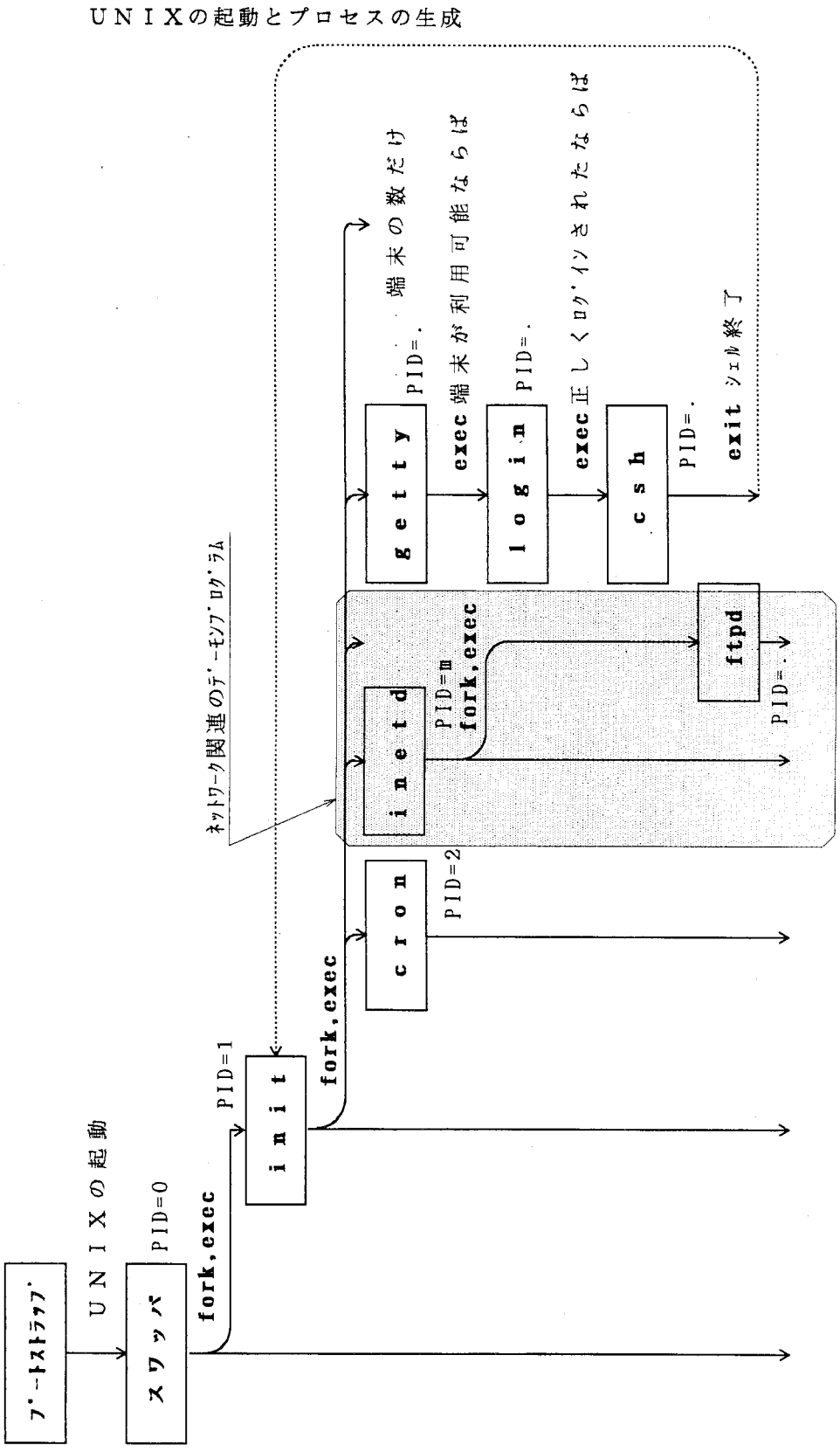
```
struct sockaddr_in  sv_addr ;
struct hostent      *host_p ;
host_p = gethostbyname( "cae_wk1" ) ;
memcpy((char *)&sv_addr.sin_addr.s_addr,(char *)host_p->h_addr,host_p->h_length) ;
```



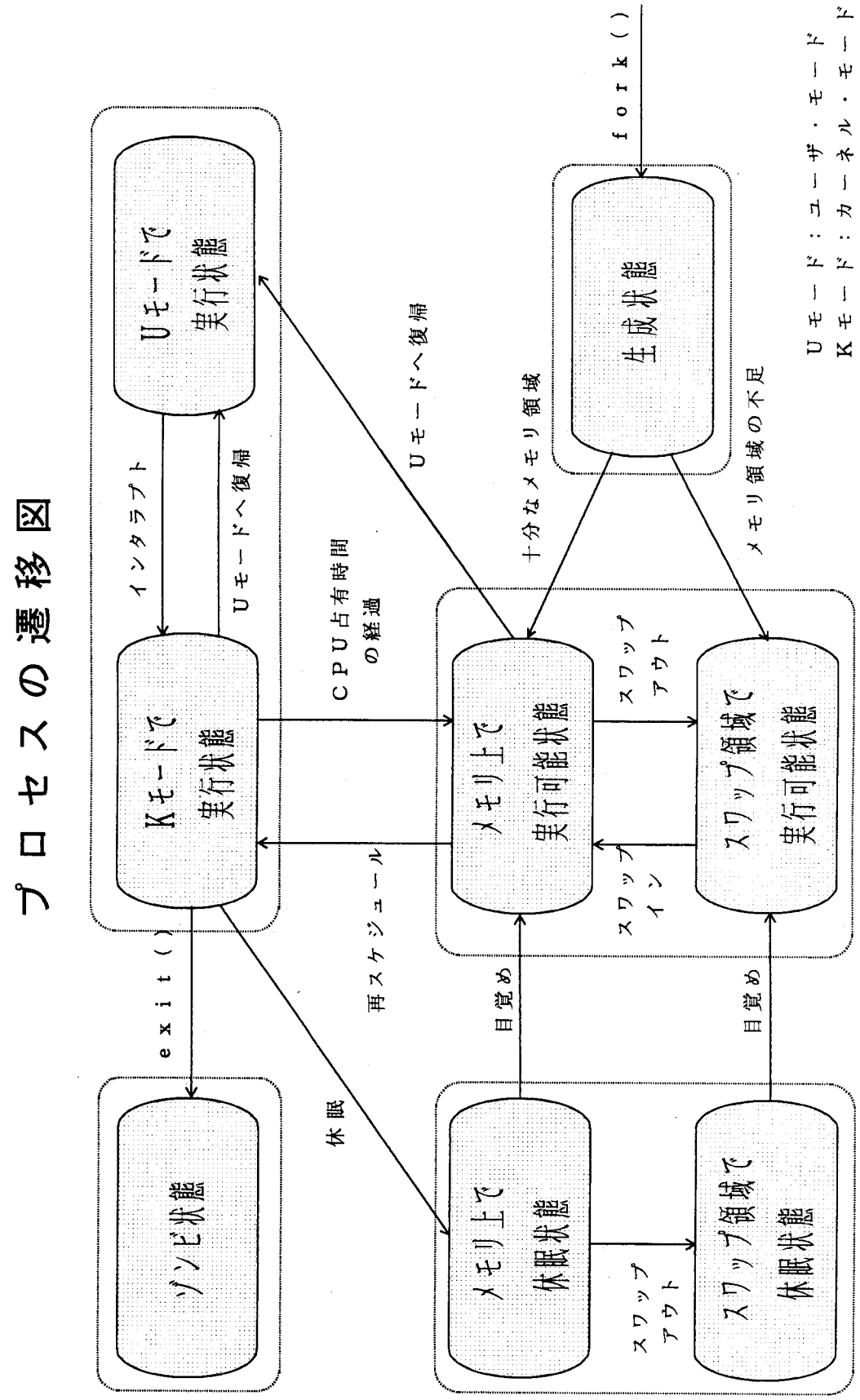
netdb.h

```
struct hostent {
        char  *h_name ;         /* ﾎｽﾄ名 */
        char  **h_aliases;      /* ﾎｽﾄの別名 */
        int    h_addrtype;      /* ｱﾄﾞﾚｽ･ﾌｧﾐﾘ */
        int    h_length         /* ｱﾄﾞﾚｽのｻｲｽﾞ */
        char   *h_addr;         /* ﾎｽﾄｱﾄﾞﾚｽへのﾎﾟｲﾝﾀ */
  };
```

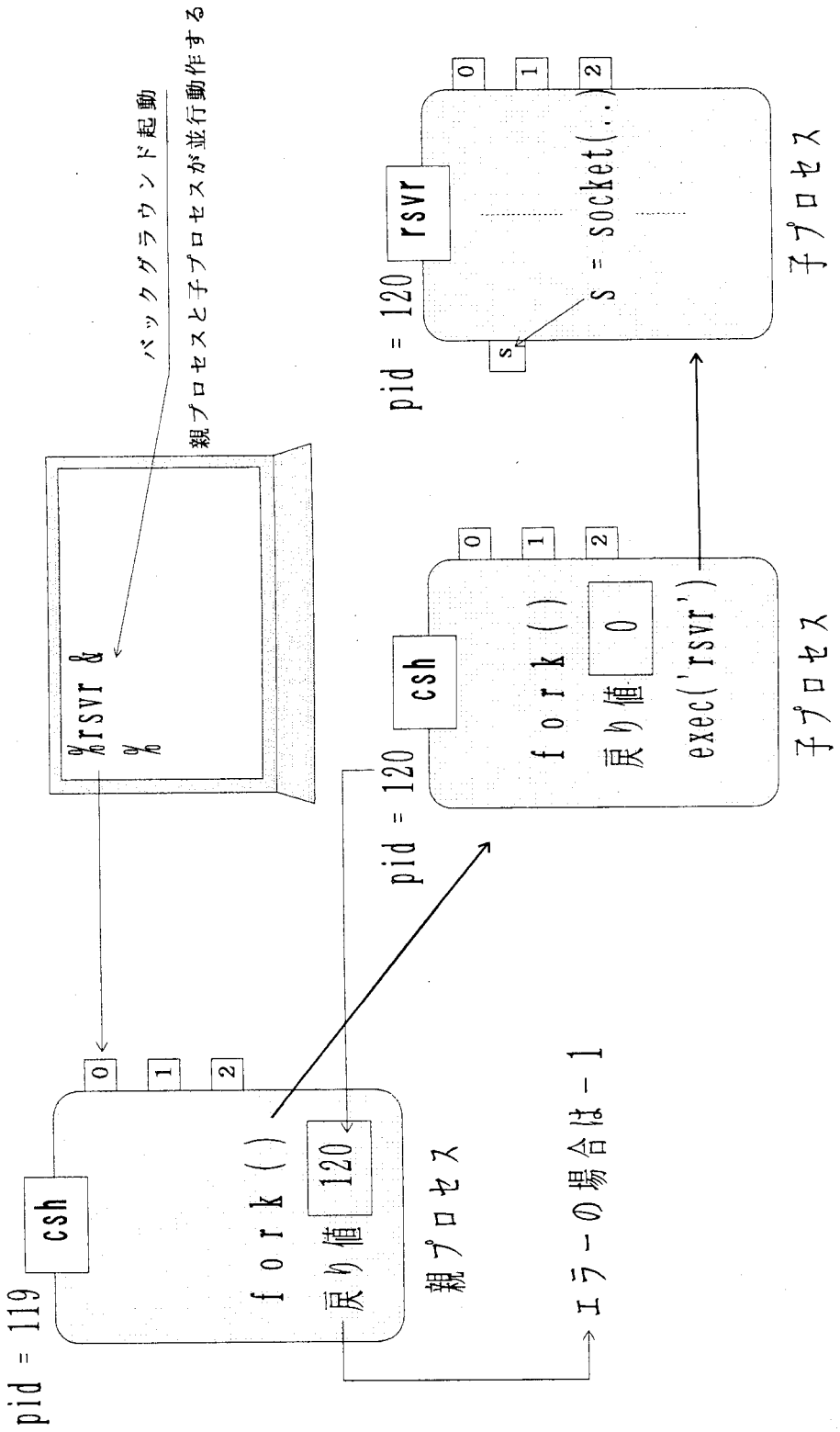
（3） ネットワーク・プログラムの動作環境（UNIX）



② プログラム（プロセス）状態と遷移



③　プロセスの生成例


バックグラウンド起動
親プロセスと子プロセスが並行動作する
%rsvr &
%
pid = 119
csh
f o r k ( )
戻り値 120
親プロセス
0
1
2
pid = 120
csh
f o r k ( )
戻り値 0
exec('rsvr')
子プロセス
pid = 120
rsvr
s = socket(...)
子プロセス
エラーの場合は－1

（4） ネットワーク・プログラミング例

```
/****************** cl_udp.c ***********************
*                                                   *
*     ｿｹｯﾄを用いたﾒｯｾｰｼﾞ交換ﾌﾟﾛｸﾞﾗﾑ例(ｸﾗｲｱﾝﾄ用)    *
*                                                   *
*          ﾎﾟｰﾄ番号 : 9000( UDP )                   *
*                                                   *
*          Useage: cl_udp hostname &                *
*                                                   *
************* UNIX System V ***********************/

#include        <stdio.h>
#include        <sys/types.h>
#include        <sys/socket.h>
#include        <netinet/in.h>
#include        <netdb.h>
#include        <errno.h>
#include        <signal.h>

#define         BUFSIZE     4096
#define         ERR         -1
#define         PORT        9000

char    buf[BUFSIZE];       /*** ﾒｯｾｰｼﾞを格納するﾊﾞｯﾌｧ                    ***/
int     sd ;                /*** ｸﾗｲｱﾝﾄからの接続要求を待つﾌｧｲﾙ記述子 ***/

main( argc , argv  )
 int argc ;
 char *argv[] ;
{
 struct  sockaddr_in server;/*** ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報          ***/
 struct  sockaddr_in client;/*** ｸﾗｲｱﾝﾄﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報          ***/
 int     len ;              /*** ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の長さ    ***/
 int     fromlen ;          /*** ｸﾗｲｱﾝﾄﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の長さ ***/
 int     sd_byte ;          /*** 送信文字数格納ｴﾘｱ                        ***/
 struct  hostent   *host_p ;/*** ｻｰﾊﾞ情報格納用構造体へのﾎﾟｲﾝﾀ          ***/

 int     endproc();         /*** ｼｸﾞﾅﾙに対する処理                        ***/

 if( argc < 2 )
  {
   fprintf( stderr , "Useage : cl_udp hostname¥n" ) ;
   exit( 0 ) ;
  }

 signal( SIGINT , endproc ) ;

 if( ( sd = socket( AF_INET , SOCK_DGRAM , 0 ) ) == ERR )
         {
          perror( "client:socket" ) ;
          exit( 1 ) ;
         }

/*
 * client の設定
 */
 fromlen = sizeof( client ) ;
 memset( (char *)&client , NULL , fromlen ) ;
 client.sin_family = AF_INET ; /* ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の設定 */
 client.sin_addr.s_addr = htonl( INADDR_ANY ) ;
 client.sin_port        = htons( 0 ) ;
 if( bind( sd , &client , fromlen ) == ERR )
         { /* client ﾌﾟﾛｾｽのｿｹｯﾄのﾊﾞｲﾝﾄﾞ */
          perror( "client:bind" ) ;
          exit( 1 ) ;
         }
/*
```

```
 * ｻｰﾊﾞのＩＰｱﾄﾞﾚｽの情報を得る
 */
 host_p = gethostbyname( argv[1] ) ;
 if( host_p == NULL )
        {/* ｻｰﾊﾞｰの情報が得られなかった */
         perror( "client:gethostbyname" );
         close( sd ) ;
         exit( 1 ) ;
        }
/*
 * server の設定
 */
 len = sizeof( server ) ;
 memset( (char *)&server , NULL , len ) ;
 server.sin_family = AF_INET ;
 server.sin_port   = htons( PORT ) ;
 memcpy( (char *)&server.sin_addr.s_addr,
                  (char *)host_p->h_addr,host_p->h_length ) ;

 while( 1 )
  {
   /*
    * 標準入力からデータを読み込む
    */
   fprintf( stdout , "ﾒｯｾｰｼﾞ:" ) ;
   if( fgets( buf , 256 , stdin ) == NULL )
         {
          fprintf( stderr , "End !¥n" );
          close( sd ) ;
          exit( 0 ) ;
         }
   /*
    * server に送信
    */
   sd_byte = strlen( buf ) ;
   if( sendto( sd , buf , sd_byte , 0 , &server , len ) < 0 )
         {/* 送信ｴﾗｰ  */
          perror( "client:sendto" ) ;
          close( sd ) ;
          exit( 1 ) ;
         }
  }
}

int endproc( sig )
int sig ;
{
 signal( sig , SIG_IGN );
 close( sd ) ;
 exit( 0 );
}
```

```
/****************** sv_udp.c ***********************
*                                                   *
*     ｿｹｯﾄを用いたﾒｯｾｰｼﾞ交換ﾌﾟﾛｸﾞﾗﾑ例(ｻｰﾊﾞ用)      *
*                                                   *
*          ﾎﾟｰﾄ番号 : 9000( UDP )                   *
*                                                   *
*          Useage: sv_udp &                         *
*                                                   *
************* UNIX System V ***********************/

#include        <stdio.h>
#include        <sys/types.h>
#include        <sys/socket.h>
#include        <netinet/in.h>
#include        <errno.h>
#include        <signal.h>

#define         BUFSIZE    4096
#define         ERR        -1
#define         PORT       9000

char    buf[BUFSIZE];       /*** ﾒｯｾｰｼﾞを格納するﾊﾞｯﾌｧ                    ***/
int     sd ;                /*** ｸﾗｲｱﾝﾄからの接続要求を待つﾌｧｲﾙ記述子 ***/

main(  )
{
 struct  sockaddr_in server;/*** ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報          ***/
 struct  sockaddr_in client;/*** ｸﾗｲｱﾝﾄﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報          ***/
 int     len ;              /*** ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の長さ    ***/
 int     fromlen ;          /*** ｸﾗｲｱﾝﾄﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の長さ ***/
 int     rec_byte ;         /*** 受信文字数格納ｴﾘｱ                        ***/

 int     endproc();         /*** ｼｸﾞﾅﾙに対する処理                        ***/
 void    ip_disp();         /*** IPｱﾄﾞﾚｽ表示                              ***/

 signal( SIGINT , endproc ) ;

 if( ( sd = socket( AF_INET , SOCK_DGRAM , 0 ) ) == ERR )
         {
          perror( "server:socket" ) ;
          exit( 1 ) ;
         }

 memset( (char *)&server , NULL , sizeof( server ) ) ;
 server.sin_family = AF_INET ; /* ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の設定 */
 server.sin_addr.s_addr = htonl( INADDR_ANY ) ;
 server.sin_port = htons( PORT ) ;
 len = sizeof( server ) ;
 if( bind( sd , &server , len ) == ERR )
         { /* ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのﾊﾞｲﾝﾄﾞ */
          perror( "server:bind" ) ;
          exit( 1 ) ;
         }

 while( 1 )
  {
   fromlen = sizeof( client ) ;
   if( (rec_byte = recvfrom( sd , buf , 256 , 0 , &client , &fromlen )) < 0 )
         {/* 受信ｴﾗｰ */
          perror( "server:recvfrom" ) ;
          close( sd ) ;
          exit( 1 ) ;
         }

   ip_disp( (char *)&client.sin_addr.s_addr ) ;

   buf[rec_byte] = 0x00 ;
```

```
    fprintf( stdout , "%s¥n" , buf ) ;
  }
}

void ip_disp( ip_addr )
 char   *ip_addr ;
{
 unsigned char ip[4] ;
 int           ip_int[4] , i ;

 memcpy( ip , ip_addr , 4 ) ;
 for( i=0 ; i<4 ; i++ ) ip_int[i] = ip[i] ;
 fprintf( stdout , "¥n%d.%d.%d.%d : ",
                       ip_int[0],
                          ip_int[1],
                             ip_int[2],
                                ip_int[3] );

 return ;
}

int endproc( sig )
int sig ;
{
 signal( sig , SIG_IGN );
 close( sd ) ;
 exit( 0 );
}
```

```
/****************** cl_tcp.c ***********************
*                                                   *
*     ｿｹｯﾄを用いたﾒｯｾｰｼﾞ交換ﾌﾟﾛｸﾞﾗﾑ例(ｸﾗｲｱﾝﾄ用)    *
*                                                   *
*          ﾎﾟｰﾄ番号 : 9000( TCP )                   *
*                                                   *
*          Useage: cl_tcp hostname command (&)      *
*              or  cl_tcp hostname (&)              *
*                                                   *
************* UNIX System V ***********************/

#include        <stdio.h>
#include        <sys/types.h>
#include        <sys/socket.h>
#include        <netinet/in.h>
#include        <netdb.h>
#include        <errno.h>
#include        <signal.h>

#define         BUFSIZE     4096
#define         ERR         -1
#define         PORT        9000

char    buf[BUFSIZE];       /*** ﾒｯｾｰｼﾞを格納するﾊﾞｯﾌｧ                    ***/
int     sd ;                /*** ｸﾗｲｱﾝﾄからの接続要求を待つﾌｧｲﾙ記述子 ***/

main( argc , argv  )
 int argc ;
 char *argv[] ;
{
 struct  sockaddr_in server;/*** ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報          ***/
 struct  sockaddr_in client;/*** ｸﾗｲｱﾝﾄﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報        ***/
 int     len ;              /*** ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の長さ   ***/
 int     fromlen ;          /*** ｸﾗｲｱﾝﾄﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の長さ ***/
 struct  hostent   *host_p ;/*** ｻｰﾊﾞ情報格納用構造体へのﾎﾟｲﾝﾀ       ***/

 int     status ;

 int     endproc();         /*** ｼｸﾞﾅﾙに対する処理                    ***/
 int     send_str();        /*** Send strings to the server          ***/
 int     rsh();             /*** Execute command on the remote host ***/

 if( argc < 2 )
  {
   fprintf( stderr , "Useage : cl_tcp hostname command¥n" ) ;
   fprintf( stderr , "     or   cl_tcp hostname¥n" );
   exit( 0 ) ;
  }

 signal( SIGINT , endproc ) ;

 if( ( sd = socket( AF_INET , SOCK_STREAM , 0 ) ) == ERR )
        {
         perror( "client:socket" ) ;
         exit( 1 ) ;
        }

/*
 * client の設定
 */
 fromlen = sizeof( client ) ;
 memset( (char *)&client , NULL , fromlen ) ;
 client.sin_family = AF_INET ; /* ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の設定 */
 client.sin_addr.s_addr = htonl( INADDR_ANY ) ;
 client.sin_port        = htons( 0 ) ;
```

```
 if( bind( sd , &client , fromlen ) == ERR )
         { /* client ﾌﾟﾛｾｽのｿｹｯﾄのﾊﾞｲﾝﾄﾞ */
          perror( "client:bind" ) ;
          close( sd ) ;
          exit( 1 ) ;
         }
/*
 * ｻｰﾊﾞのIPｱﾄﾞﾚｽの情報を得る
 */
 host_p = gethostbyname( argv[1] ) ;
 if( host_p == NULL )
        {/* ｻｰﾊﾞｰの情報が得られなかった */
         perror( "client:gethostbyname" );
         close( sd ) ;
         exit( 1 ) ;
        }
/*
 * server の設定
 */
 len = sizeof( server ) ;
 memset( (char *)&server , NULL , len ) ;
 server.sin_family = AF_INET ;
 server.sin_port   = htons( PORT ) ;
 memcpy( (char *)&server.sin_addr.s_addr,
                  (char *)host_p->h_addr,host_p->h_length ) ;
/*
 * ｻｰﾊﾞﾌﾟﾛｾｽへの接続要求
 */
 if( connect( sd , &server, len ) == ERR )
        {
         perror( "client:connect" );
         close( sd ) ;
         exit( 1 ) ;
        }

 status = 0 ;
 switch( argc )
  {
   case  2 :/** ｻｰﾊﾞをﾒｯｾｰｼﾞ受信ﾓｰﾄﾞにする **/
            if( write( sd , "m" , 1 ) < 0 )
               {
                status = -1 ;
                perror( "client:write" ) ;
                break ;
               }
            if( send_str( ) < 0 ) status = -1 ;
            break ;

   case  3 :/** ｻｰﾊﾞをｺﾏﾝﾄﾞ実行ﾓｰﾄﾞにする **/
            if( write( sd , "c" , 1 ) < 0 )
               {
                status = -1 ;
                perror( "client:write" ) ;
                break ;
               }
            if( rsh( argv[2] ) < 0 ) status = -1 ;
            break ;

   default :/** その他(起動時の引き数が４以上) **/
            break ;
  }
 if( shutdown( sd , 2 ) < 0 ) perror( "client:shutdown" );
 close( sd );
 exit( 0 ) ;
}
```

```
/*
 * ｻｰﾊﾞにﾒｯｾｰｼﾞを送信する
 */
int send_str( )
{
 int sd_byte ;

 while( 1 )
  {
   /*
    * 標準入力からデータを読み込む
    */
   fprintf( stdout , "ﾒｯｾｰｼﾞ:" ) ;
   if( fgets( buf , 256 , stdin ) == NULL )
         {
          fprintf( stderr , "End of file !¥n" );
          break ;
         }
   /*
    * server に送信
    */
   sd_byte = strlen( buf ) ;
   if( write( sd , buf , sd_byte ) < 0 )
         {/* 送信ｴﾗｰ  */
          perror( "send_str:client:write" ) ;
          return( -1 ) ;
         }
   /*
    * 最初の文字が '.' の場合、送信終了
    */
   if( buf[0] == '.' ) break ;
  }
 return( 0 ) ;
}

/*
 * ｻｰﾊﾞにｺﾏﾝﾄﾞを送信する
 */
int rsh( command )
char *command ;
{
 int  rec_byte ;

 if( write( sd , command , strlen( command ) ) < 0 )
       {
        perror( "rsh:client:write" ) ;
        return( -1 ) ;
       }
 while( 1 )
  {
   if( (rec_byte = read( sd , buf , BUFSIZE )) < 0 )
       {
        perror( "rsh:client:read" ) ;
        return( -1 ) ;
       }
   if( rec_byte == 0 ) break ;

   buf[rec_byte] = 0x00 ;
   fprintf( stdout , "%s" , buf ) ;
  }
 return( 0 ) ;
}

/*
 * ｼｸﾞﾅﾙの処理
```

```
 */
int endproc( sig )
int sig ;
{
 signal( sig , SIG_IGN );
 close( sd ) ;
 exit( 0 );
}
```

```
/****************** sv_tcp.c ***********************
*                                                   *
*      ｿｹｯﾄを用いたﾒｯｾｰｼﾞ交換ﾌﾟﾛｸﾞﾗﾑ例(ｻｰﾊﾞ用)      *
*                                                   *
*           ﾎﾟｰﾄ番号 : 9000( TCP )                  *
*                                                   *
*           Useage: sv_tcp (&)                      *
*                                                   *
************* UNIX System V ***********************/

#include        <stdio.h>
#include        <sys/types.h>
#include        <sys/socket.h>
#include        <netinet/in.h>
#include        <errno.h>
/*
#include        <signal.h>
*/

#define         BUFSIZE    4096
#define         ERR        -1
#define         PORT       9000

char    buf[BUFSIZE];       /*** ﾒｯｾｰｼﾞを格納するﾊﾞｯﾌｧ                    ***/
int     sd , ns ;           /*** ｸﾗｲｱﾝﾄからの接続要求を待つﾌｧｲﾙ記述子 ***/

struct  sockaddr_in server;/*** ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報          ***/
struct  sockaddr_in client;/*** ｸﾗｲｱﾝﾄﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報          ***/

void    ip_disp();          /*** IPｱﾄﾞﾚｽ表示                             ***/
void    ch_prog1();         /*** 子ﾌﾟﾛｾｽ1:ｸﾗｲｱﾝﾄからのﾒｯｾｰｼﾞの表示     ***/
void    ch_prog2();         /*** 子ﾌﾟﾛｾｽ2:ｸﾗｲｱﾝﾄからのｺﾏﾝﾄﾞを実行      ***/

main(  )
{
 int     len ;              /*** ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の長さ   ***/
 int     fromlen ;          /*** ｸﾗｲｱﾝﾄﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の長さ ***/
 int     pid     ;          /*** PID(ﾌﾟﾛｾｽID)                           ***/
 int     rec_byte ;

 if( ( sd = socket( AF_INET , SOCK_STREAM , 0 ) ) == ERR )
         {
          perror( "server:socket" ) ;
          exit( 1 ) ;
         }

 len = sizeof( server ) ;
 memset( (char *)&server , NULL , len ) ;
 server.sin_family = AF_INET ; /* ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのｱﾄﾞﾚｽ情報の設定 */
 server.sin_addr.s_addr = htonl( INADDR_ANY ) ;
 server.sin_port = htons( PORT ) ;
 if( bind( sd , &server , len ) == ERR )
         { /* ｻｰﾊﾞﾌﾟﾛｾｽのｿｹｯﾄのﾊﾞｲﾝﾄﾞ */
          perror( "server:bind" ) ;
          exit( 1 ) ;
         }

/*
 * ｸﾗｲｱﾝﾄﾌﾟﾛｾｽからの接続要求の受け入れ
 */
 if( listen( sd , 5 ) == ERR )
         {
          perror( "server:listen" ) ;
          exit( 1 ) ;
         }
```

```
while( 1 )
 {
  /*
   * ｸﾗｲｱﾝﾄﾌﾟﾛｾｽからの接続を待つ
   */
  fromlen = sizeof( client ) ;
  if( ( ns = accept( sd , &client , &fromlen ) ) == ERR )
       {
        perror( "server:accept" ) ;
        close( sd ) ;
        exit( 1 ) ;
       }
  /*
   * 子ﾌﾟﾛｾｽの作成と起動
   */
  if( (rec_byte = read( ns , buf ,BUFSIZE ) ) == ERR )
      {
       perror( "server:read" ) ;
       if( shutdown( ns , 2 ) < 0 ) perror( "server:shutdown" ) ;
       close( sd ) ;
       close( ns ) ;
       exit( 1 ) ;
      }
  pid = -2 ;
  switch( buf[0] )
      {
       case  'm':/** ｸﾗｲｱﾝﾄからのﾒｯｾｰｼﾞを受信する子ﾌﾟﾛｾｽを起動 **/
                 if( ( pid = fork() ) == 0 ) ch_prog1( ) ;
                 break ;

       case  'c':/** ｸﾗｲｱﾝﾄからのｺﾏﾝﾄﾞを実行するﾌﾟﾛｾｽを起動 **/
                 if( ( pid = fork() ) == 0 ) ch_prog2( ) ;
                 break ;
       default  :/** その他 **/
                 break ;
      }
   if( pid > 0 )
      { /* 親ﾌﾟﾛｾｽの処理 */
       close( ns ) ;
       if( pid >= 1 )
         {
          /*
          wait( &status ) ;
          */
          continue ; /* 次のクライアントからの接続要求を待つ */
         }
       else
         {
          if( pid == -1 )
             {/* 子ﾌﾟﾛｾｽの作成・起動に失敗 */
              perror( "server:fork" ) ;
              close( sd ) ;
              exit( 1 ) ;
             }
           else continue ;
         }
      }
 }
}

/*
 * 子ﾌﾟﾛｾｽの処理(ｸﾗｲｱﾝﾄからのﾒｯｾｰｼﾞを受信し表示する)
 */
void ch_prog1( )
{
 int rec_byte ;
 close( sd ) ;
 while( 1 )
```

```
    {
    if( ( rec_byte = read( ns , buf , BUFSIZE ) ) == ERR )
        { /* ｿｹｯﾄを介してｸﾗｲｱﾝﾄﾌﾟﾛｾｽからのﾒｯｾｰｼﾞを受け取る */
         perror( "ch_prog1:server:read" ) ;
         close( ns ) ; /* accept()で得たﾌｧｲﾙ記述子のｸﾛｰｽﾞ */
         exit( 1 );
        }

    ip_disp( (char *)&client.sin_addr.s_addr ) ;

    buf[rec_byte] = 0x00 ;
    fprintf( stdout , "%s¥n" , buf ) ;
    if( buf[0] == '.' )
        {
         if( shutdown( ns , 2 ) == ERR ) perror( "server:shutdown" ) ;
         close( ns ) ; /* accept()で得たﾌｧｲﾙ記述子のｸﾛｰｽﾞ */
         exit( 0 ) ;
        }
   }
}

/*
 * 子ﾌﾟﾛｾｽの処理(ｸﾗｲｱﾝﾄからのｺﾏﾝﾄﾞを実行する)
 */
void ch_prog2( )
{
 int rec_byte ;

 close( sd ) ;
 close( 1 )  ;
 close( 2 )  ;
 dup( ns )   ;   /* 標準出力をｿｹｯﾄへ切り換える */
 dup( ns )   ;   /* 標準ｴﾗｰ出力をｿｹｯﾄへ切り換える */

 if( ( rec_byte = read( ns , buf , BUFSIZE ) ) == ERR )
     { /* ｿｹｯﾄを介してｸﾗｲｱﾝﾄﾌﾟﾛｾｽからのﾒｯｾｰｼﾞを受け取る */
      perror( "ch_prog2:server:read" ) ;
      close( ns ) ;
      exit( 1 );
     }

 buf[rec_byte] = 0x00 ;
 system( buf ) ;
 if( shutdown( ns , 2 ) == ERR ) perror( "server:shutdown" ) ;
 close( ns ) ;
 exit( 0 ) ;
}

/*
 * ｸﾗｲｱﾝﾄのIPｱﾄﾞﾚｽの表示
 */
void ip_disp( ip_addr )
 unsigned char   *ip_addr ;
{
 int           ip_int[4] , i ;

 for( i=0 ; i<4 ; i++ ) ip_int[i] = *( ip_addr + i ) ;
 fprintf( stdout , "¥n%d.%d.%d.%d : ",
                       ip_int[0],
                          ip_int[1],
                             ip_int[2],
                                ip_int[3] );

 return ;
}
```

[参考資料]

異機種接続とLAN絵とき読本
上原　政二　監修　オーム社

TCP／IP通信の基礎と実際
矢吹　道朗　監修　株式会社トリケップス

TCP／IPによるネットワーク構築bit別冊
村井　純・楠本　博之　訳　共立出版