OUKA
The University of Osaka
Institutional Knowledge Archive

| | |
|---|---|
| Title | 分散処理のための通信アルゴリズムおよび分散プログラムデバッグの研究 |
| Author(s) | 真鍋, 義文 |
| Citation | 大阪大学, 1993, 博士論文 |
| Version Type | VoR |
| URL | https://doi.org/10.11501/3066007 |
| rights | |
| Note | |

*The University of Osaka Institutional Knowledge Archive : OUKA*

https://ir.library.osaka-u.ac.jp/

The University of Osaka

# 分散処理のための通信アルゴリズム および分散プログラムデバッグの研究

1993年1月

真鍋 義文

## 内容梗概

本論文は、分散処理システム構成技術のうち、通信アルゴリズムおよび分散プログラムデバッグに関する研究をまとめたものである。本論文は5章から成り、その内容は以下の通りである。

第1章では、本研究の歴史的背景についてふれ、分散システムにおける通信アルゴリズムおよび分散プログラム開発に関する問題点について述べる。

第2章では、分散システムにおける通信時の経路情報の安全性の問題について考察している。そして、中継者には中継に必要な最小限の情報しか与えず、かつ送り手と受け手の間の匿名性を実現する経路情報の暗号化法を示している。

第3章では、ルーティングの効率と信頼性の問題について述べる。そして、効率を表す尺度である中継処理量、信頼性を表す尺度であるSR-グラフの直径に対して、両者の値が準最適である有向グラフ、およびその上のルーティングの組を与えている。

第4章では、分散プログラム開発支援のためのデバッガに求められる基本機能である、複数のプロセスに関する大域的条件式を用いたブレークポイント設定機能およびトレース機能について述べている。大域的条件式がいかなる形式の時に与えられた条件式を満足する最初の地点をユーザに提示可能であるかを示し、提示可能な場合についてその機能を実現するアルゴリズムを示す。

第5章は結論であり、本研究で得られた結果を総括し、今後に残された課題を指摘する。

# もくじ


**1 序論** 1

1.1 研究の目的 . . . . . 1

1.2 研究の歴史的背景 . . . . . 2

1.2.1 経路情報の安全性 . . . . . 2

1.2.2 グラフとその上のルーティングの組に対する耐故障性と効率 . 3

1.2.3 分散プログラムデバッグの基本機能 . . . . . 5

1.3 本研究の概要 . . . . . 6

1.3.1 経路情報の安全性 . . . . . 6

1.3.2 グラフとその上のルーティングの組に対する耐故障性と効率 . 7

1.3.3 分散プログラムデバッグの基本機能 . . . . . 7

**2 経路情報の安全性** 8

2.1 問題の定式化 . . . . . 8

2.1.1 通信網とルーティング . . . . . 8

2.1.2 経路情報の安全性問題 . . . . . 10

2.1.3 解読者の形態 . . . . . 12

2.2 従来の研究 . . . . . 13

2.3 経路情報の安全性を達成する暗号化法 . . . . . 16

2.3.1 暗号系に対する要求条件 . . . . . 16

2.3.2 共謀が存在しない場合の暗号化法 . . . . . 18

2.3.3 共謀が存在する場合の暗号化法 . . . . . 21

2.4 むすび . . . . . 30

**3 グラフとその上のルーティングの組に対する耐故障性と効率** 31

3.1 諸定義 . . . . . 32

3.1.1 用語の定義 . . . . . 32

3.1.2 中継処理量と SR- グラフの定義 . . . . . 33

3.2 SR- グラフの直径が定数となる有向グラフのルーティング . . . . . 35

3.3 中継処理量が準最適な有向グラフとその上のルーティングの組 . . . . 40

3.4 中継処理量と SR- グラフの直径がともに準最適な有向グラフとその上のルーティングの組 . . . . . 48

3.5 むすび . . . . . 53

**4 分散プログラムデバッグの基本機能** **55**

4.1 従来の研究 . . . . . 55

4.2 モデルおよび定義 . . . . . 59

4.3 ブレークポイント設定機能 . . . . . 62

4.3.1 積条件式に対する停止問題 . . . . . 64

4.3.2 和条件式に対する停止問題 . . . . . 69

4.3.3 一般の大域的条件式に対する停止問題 . . . . . 70

4.4 トレース機能 . . . . . 71

4.5 プロトタイプデバッガの試作 . . . . . 73

4.6 むすび . . . . . 81

**5 結論** **82**

**謝辞** **84**

**参考文献** **85**

**著者の発表論文一覧表** **90**

# 1 序論

## 1.1 研究の目的

近年の情報化社会につれ、通信に対するさまざまな要求が発生し、アプリケーションサービスを含む通信網、いわゆるインテリジェントネットワークが提供されるようになってきている。これらはその初期においては、専用線を用いた第二種事業者による商用サービス、公衆通信網上のサービス、企業内のプライベートネットワークなど独立に形成されていった。また、そのサービスの多くは最初は単一のセンタをベースとした単純なものであった。しかし近年、ネットワークサービスの統合化の要求が生まれ、これらのネットワークの相互接続が生じてきている。しかし、ネットワークの相互接続には整合性にまつわるさまざまな問題が発生する。これは、異なる管理主体を持つさまざまな種類のセンタ・端末が通信網に接続されることに起因するものである。この問題は計算機ネットワークにおける分散処理の問題そのものである。このような問題を解決するための分散処理技術の一層の進展が望まれてきている。

分散処理の目的は、

- 非集中性(負荷が一箇所に集中せず、効率的処理が可能である)
- 耐故障性(一部が故障しても正しく動作する)
- 独立性(各計算機が独立に動作できる)
- 安全性(機密性、不正なアクセスを防止する)
- 部分変更容易性(構成要素の一部分のみの変更が容易である)
- 拡張容易性(システムの拡張が容易である)

などを達成することにある[39][47]。しかしながら、上記の性質を満たす分散処理システムを構成するための要素技術についてはまだ研究途上にある。要素技術の主なものとしては、

- 分散OS
- 通信プロトコル
- 分散データベース
- 分散プログラム用言語

- 分散システム用セキュリティ
- 分散システム開発支援環境

などがあげられ、これらに関する基本問題を解くアルゴリズムが必要である。本論文では、これらの要素技術のうち、通信プロトコルのうちルーティングに関するアルゴリズム、およびソフトウェア開発支援のうちデバッグに関するアルゴリズムについて述べる。

## 1.2 研究の歴史的背景

### 1.2.1 経路情報の安全性

通信の安全性については古くから多くの研究がなされてきている [16]。しかし、それらのほとんどは通信内容の安全性に関する研究であった。通信内容の秘密は保たれたとしても、誰が誰あてに、いつ通信を行なったかを検出する、いわゆるトラフィック解析 (Traffic analysis) の問題は存在する。通信の状況によっては、通信が行なわれたこと自身を秘密にしたい状況も考えられる (例えば、警察への密告がいつ、誰からなされたか、など)。従って、トラフィック解析は安全性に関する一つの大きな問題である。通信路上でのトラフィック解析を防止する方法として、以下のリンク暗号という方式が考えられている [2]。

頂点 $v_0, \ldots, v_n$ はメッセージ送信もしくは中継を担当する装置、辺 $e_0, \ldots, e_{n-1}$ はそれらを結ぶ通信路とする。リンク暗号は、頂点 $u$ から頂点 $v$ までの経路を $v_0(=u), e_0, v_1, \ldots, e_{n-1}, v_n(=v)$、頂点 $v_i$ のみが解くことができる暗号の暗号化関数を $E_i$、復号化関数を $D_i$(すなわち $D_i(E_i(M)) = M$ が任意の $M$ に対して成立する)、送信メッセージを $M$ としたとき、以下のように暗号化および復号化を行なう。

- 頂点 $v_0$ は、メッセージと宛先情報 $\rho(v_0, v_n)$ の組 $(M, \rho(v_0, v_n))$ を暗号化したパケット $E_1(M, \rho(v_0, v_n))$ を辺 $e_0$ に送る (すなわち、$v_1$ に送る)。
- 頂点 $v_i (i = 1, \ldots, n-1)$ は、辺 $e_{i-1}$ から到着した $E_i(M, \rho(v_0, v_n))$ を受けとり、復号化 $D_i(E_i(M, \rho(v_0, v_n)))$ を行なって $(M, \rho(v_0, v_n))$ を得る。宛先情報 $\rho(v_0, v_n)$ から次にこのパケットを送るべき辺 $e_i$ を求め、暗号化 $E_{i+1}(M, \rho(v_0, v_n))$ を行なったパケットを $e_i$ に送る。
- 頂点 $v_n$ は、辺 $e_{n-1}$ から到着した $E_n(M, \rho(v_0, v_n))$ を受けとり、復号化 $D_n(E_n(M, \rho(v_0, v_n)))$ を行なって $(M, \rho(v_0, v_n))$ を得る。宛先情報 $\rho(v_0, v_n)$ から

このパケットの宛先が自分であることを知り、$M$ を受信メッセージとして受理する。

途中の頂点が $M$ の内容を知ることを防止するには、$M$ の代わりに $E_n(M)$ を用いればよい。

すべての辺において、ダミーのメッセージを流して通信量が常に一定になるようにすれば、上記のアルゴリズムにより通信路でのトラフィック解析は不可能となる。これは、同じ宛先情報を持つメッセージが同じ辺を複数個送信されても、暗号化により同一性を判定できないこと、および、同じメッセージが辺 $e_i$ および $e_{i'}$ を通過する時、$e_i$ と $e_{i'}$ を盗聴している者がいたとしても、暗号化 $E_{i+1}$ および $E_{i'+1}$ が異なるため、$E_{i+1}(M, \rho(v_0, v_n))$ と $E_{i'+1}(M, \rho(v_0, v_n))$ の同一性を判定できないからである。

しかし、上記のリンク暗号も、頂点でのトラフィック解析については防止できない。なぜなら、頂点では復号化して宛先情報 $\rho(v_0, v_n)$ を得ることができるからである。$\rho(v_0, v_n)$ の同一性を判定することにより、誰と誰がいつ、どのような頻度で通信を行なっているかの情報を得ることは可能である。さらに、$\rho(v_0, v_n)$ より、送り手と受け手の相互が相手の素性を認識することが可能である。すなわち、匿名性を達成することは不可能である。

計算機ネットワークが単一組織の中で閉じている状況では、頂点でのトラフィック解析や匿名性という問題は起こり得ない。しかし、Internet[43] のようにネットワークが複数の組織を結ぶ形態になると、それら複数の組織の間の方針の違い、争いといった問題は避けられなくなってくるであろう。例えば、A 社と B 社の間の通信状況を、A 社と競合関係にある C 社が知ろうとする状況というものも考えられる。また、ユーザが匿名で通信を行ないたい場合も存在する。

本研究ではこのような、複数の組織によって運営されるネットワークにおいてメッセージを通信する際の、頂点における経路情報の安全性の問題について考察している。

### 1.2.2 グラフとその上のルーティングの組に対する耐故障性と効率

計算機ネットワークおよび通信網の構成問題において、ネットワークを無向グラフあるいは有向グラフでモデル化してグラフ理論的に研究がなされてきた [29][31][45][51]。ここで、頂点は交換機あるいは計算機、辺は通信路を表す。このとき、網の信頼性や効率はそのトポロジー的性質に依存する。網のトポロジーが達成するべき性質として以下の点が考えられる。

- 低コスト (網の設備量が少ない)

- 効率 (通信時の遅れが小さい)
- 信頼性 (故障時の通信可能性が高い)
- ルーティングの単純さ (ルーティング決定のための処理量が少ない)
- 拡張容易性 (既存の網を拡張するときに変更が少ない)

以上の点から最適なネットワークトポロジーのグラフ理論的研究がなされてきた。コストの指標は辺の数、効率は直径 (最短経路の最大中継段数)、信頼性は連結度 (いくつの点を除去しても通信可能であるかを示す) で表し、これらが最適なトポロジーを求める問題が考えられてきた。ルーティングの単純さ、拡張容易性についてはその定式化手法を含めて今後の課題となっている。

しかし、信頼性や効率を考える際、網のトポロジーだけではなく、ルーティングの手法も考慮する必要がある。すなわち、連結度や直径で表される網の潜在的耐故障性や効率が高くとも、ルーティングが悪いと、そのよさを生かしきれていない可能性がある。従って、網の潜在的可能性を本当に実現するルーティングが存在するか否かを判定すること、および、潜在的可能性を生かすルーティングを与えることが実際に網を運用する側面からは重要である。

近年、グラフおよびその上のルーティングの組に対する 2 つの指標が独立に提案された。1 つは効率を計る中継処理量 [9] であり、もう一つは信頼性を計る SR- グラフの直径 [6][20] である。

ルーティング $\rho$ は任意の 2 頂点間に対してある道を規定するものである。この道を経路と呼ぶ。中継処理量 $\xi(G,\rho)$ は各頂点を通る経路数の最大値である。一般に、各頂点で行なわなければならない処理はその頂点におけるメッセージの送受信のみではなく、他の頂点間のメッセージのための処理も存在する。各頂点間の通信量が等しいと仮定すると、中継処理量は中継機能のための頂点の能力低下を計る量である。従って、中継処理量はネットワークの効率を計る一つの指針となり得る。

SR- グラフ $R(G,\rho)/F$ はグラフ $G = (V,E)$、ルーティング $\rho$ および故障集合 $F$ に対して定義される有向グラフである。$R(G,\rho)/F$ の頂点集合は $V - F$ で頂点 $u$ から $v$ の間に辺が存在するのは $G$ における $u$ から $v$ の間の経路が $F$ を通らない場合である。ルーティングテーブルがネットワークに対して 1 度だけ計算されるとすると、すべてのメッセージは故障がある場合にもそれらの経路を用いて通信されなければならない。頂点もしくは辺が故障した場合には、それらを経由していた経路は使用不可能になる。そこで、無故障の経路をいくつか使用することによってメッセージ通信を行なう。この時、メッセージ通信にかかる処理の手間は経路の端点での処理

が支配的になることもある。この仮定の下では、SR- グラフの直径 $D(R(G,\rho)/F)$ はネットワークの信頼性の一つの指標となる。

中継処理量と SR- グラフの直径が小さなルーティングに関しては、以下の 2 つの問題が考えられる。

(1) 任意のグラフに対し、上記評価尺度の最適なルーティングを求める

(2) 上記評価尺度の最適な、グラフとルーティングの組を求める

SR- グラフに関する前者の問題についてはルーティングが最短ルーティング、すなわち $u$ から $v$ の経路として $u$ から $v$ への最短の道のうちから選ぶルーティングである場合には、SR- グラフの直径は故障の数に依存することが知られている [20]。

$k$- 連結無向グラフに対して、最短ではないルーティングで $D(R(G,\rho)/F)$ の値が故障の数に依存しない定数値になるルーティング $\rho$ を求める問題はいくつか考えられている [30][32][42]。

後者の問題については、中継処理量に関して、無向グラフとその上のルーティングの組で、$\xi(G,\rho)$ が下界と同じオーダーのものが求められている [9]。

通信網をグラフでモデル化する場合、無向グラフでモデル化する (すなわち、通信路はすべて双方向と考える) 場合と有向グラフでモデル化する (すなわち、通信路は片方向と考える) 場合が考えられる。故障などを考慮する際には、双方向通信路においても一方向のみの故障を考慮するなど 2 方向の通信路を別々に扱う場合もある。従って、有向グラフでモデル化するのが適当な場合もある。本稿では、有向グラフに関する上記の問題、および 2 つの評価尺度がともに準最適になるルーティングを求める問題について考察する。

### 1.2.3 分散プログラムデバッグの基本機能

分散プログラムの開発は逐次型プログラムの開発に比べると困難である。その大きな原因の一つは、複数のプロセス間の動作の関係がプログラマの予期せぬものになることが多いためである。そのために、分散プログラムデバッグのシステムとして多くの研究がなされてきた [37][41]。それらの研究は以下のように分類される。

- 静的解析によって誤りを発見する
- 動作の概要をユーザにわかりやすく提示する
- 動作させて詳細情報を得る

これらの研究はそれぞれ、有効な場合が異なる。静的解析はプログラムを実際に動作させることなく検出できる点が長所であり、かつ、誤りの可能性をすべて列挙することができるが、実際に誤りではない部分にも警告を出すという false alarm を起こしがちである。ユーザに動作概要を提示する方法は動作のボトルネックになっている箇所を知るなどの大まかな処理の流れを見るのには非常に有効であるが、詳細情報を得るためには他の手法が必要である。動作させて詳細情報を得るのは本質的には万能ではあるが、ツールに的確な指示を行なう能力がユーザに必要となる。

本研究ではこのうちで最も広く用いられている、動作させて詳細情報を得る手法について考察する。分散プログラムのデバッグにおいて重要な問題の一つは、複数のプロセス間の動作の関係で生じる誤りをどのようにして発見し、そのバグを除去するかである。

上記のようなバグを除去するための1つの手段は、複数のプロセスに関係した条件式(大域的条件式)でブレークポイントを設定して、その条件に合致した時に停止することである。また、与えられた条件が成立した時の状態(ソースプログラムの位置および変数の値など)を表示するトレース機能もそのための機能の1つである。本稿では、非決定的な動作の再演を行なうデバッガにおいて、大域的条件式を用いたブレークポイント設定機能およびトレース機能を導入する問題を考察する。

### 1.3 本研究の概要

本研究では、分散システムにおける通信アルゴリズムおよびソフトウェア開発支援に関する上記の問題について考察し、以下の結果を示す。

#### 1.3.1 経路情報の安全性

第2章では、経路情報の安全性の問題に関し、以下の結果を示す。

中継のあるネットワークにおいて、ソースルーティングを行なう場合に、頂点におけるトラフィック解析を防止するため、経路情報を計算するセンタの存在を仮定した場合に

- 中継頂点には中継に必要な情報のみを与え、それ以外の経路情報を与えない
- 送り手頂点に、受け手頂点および経路についての情報を与えない(匿名性)

を満足する、経路情報の暗号化の2つの手法を与える。

一方の手法は頂点の共謀がない場合にのみ有効で、従来の方法と計算量はオーダ的に同じである。もう一方の方法は計算量は大きくなるが、頂点の共謀がある場合

にも有効な手法である。

### 1.3.2 グラフとその上のルーティングの組に対する耐故障性と効率

第3章では、グラフとその上のルーティングの組に対する耐故障性と効率の問題に関し、以下の結果を示す。

1. $k$- 連結な有向グラフが、SR- グラフの直径が故障の数に依存しない定数となるルーティングを持つための十分条件。

2. 任意の頂点数および最大次数に対して、中継処理量が準最適な、有向グラフとその上でのルーティングの組を構成する方法。

3. 任意の頂点数および最大次数に対して、SR- グラフの直径と中継処理量がともに準最適な値となる、有向グラフとその上でのルーティングの組を構成する方法。

### 1.3.3 分散プログラムデバッグの基本機能

第4章では、分散プログラムのデバッガに大域的条件式を用いたブレークポイント機能およびトレース機能を導入する問題に関し、以下の結果を示す。

非決定的な動作の再演を行なう分散プログラム用デバッガに対して、積条件式と和条件式という二つの大域的条件式を考え、条件式を満足する最初の地点で全プロセスを停止するブレークポイント機能の実現性を議論する。そして、積条件式については、条件式を満足する最初の地点で全プロセスを停止することが可能であることを示す。また、和条件式については、最初の地点で全プロセスを停止することは不可能であるが、条件を満足するある地点で停止することは可能であることを示す。さらに、その他の一般的な大域的条件式については、条件式を満足するある地点で停止することも不可能であることを示す。

また、大域的条件式を用いたトレース機能については、積条件式および和条件式のいずれについてもトレース可能であることを示す。

そして、停止およびトレースが可能なおのおのの場合について、その機能を実現するアルゴリズムを示す。

# 2　経路情報の安全性

本章では、中継のあるネットワークにおいて、ソースルーティングを行なう場合に、頂点におけるトラフィック解析を防止するため、経路情報を計算するセンタの存在を仮定した場合に

- 中継頂点には中継に必要な情報のみを与え、それ以外の経路情報を与えない
- 送り手頂点に、受け手頂点および経路についての情報を与えない (匿名性)

を満足する、経路情報の暗号化の 2 つの手法を与える。

一方の手法は頂点の共謀がない場合にのみ有効で、従来の方法と計算量はオーダ的に同じである。もう一方の方法は計算量は大きくなるが、頂点の共謀がある場合にも有効な手法である。

まず、2.1節で用語の定義と問題の定式化を行なう。2.2節で従来の研究について述べる。次に、2.3節で経路情報の暗号化手法を与える。

## 2.1　問題の定式化

本節では、用語の定義と問題の定式化を行なう。

### 2.1.1　通信網とルーティング

ここで考慮する通信網は頂点で中継を行なう網であり、単純な放送型の網 (イーサネットなど) は対象外である。通信網を無向グラフ $G = (V, E)$ で表す。ここで、$V$ は頂点の集合、$E$ は辺の集合であり、$E$ の要素は頂点の非順序対である。ここで、匿名性を考慮すると、頂点は一般に複数の名前を持ち、それらを通信相手に応じて使い分けることになる。それらの名前を仮想頂点と呼ぶ。仮想頂点の集合を $W$ として、仮想頂点に対し、実体としての $G$ の頂点を与える関数を $N : W \to V$ とする。頂点 $v \in V$ に対し、$v$ と隣接している辺の集合を $E_v$ とする。頂点 $v$ と $v'$ の間の道とは、頂点と辺の交互系列 $v_0(= v), e_0, v_1, \ldots, e_{n-1}, v_n(= v')$ で、$e_i = (v_i, v_{i+1}) \in E, v_i \neq v_j (i \neq j)$ を満足するものである。頂点 $v$ と $v'$ の間の距離 $dis(v, v')$ は $v$ と $v'$ の間の道の集合の中での、辺の数の最小値である。$G$ の直径 $D(G)$ は、

$$D(G) = \max_{v,v' \in V, v \neq v'} dis(v, v')$$

で定義される。

頂点 $v$ がある仮想頂点 $w$ にメッセージを送ることができる場合には、$v$ と $N(w)$ の間のある道が決められている。この道を $v$ から $w$ までの経路と呼び、$\rho(v, w)$ と書く。

同一の $(v,w)$ の対に対して耐故障性などの観点から複数の経路が与えられる場合もあるが、議論の単純化のため、経路は1つであるとする。本章の結果は経路が複数与えられる場合にも有効である。また、$N(w')=N(w)$ となる $w$、$w'$ に対して $\rho(v,w)$ と $\rho(v,w')$ が同一となる場合も異なる場合もあるものとする。

ある経路 $\rho=v_0,e_0,v_1,\ldots,e_{n-1},v_n$ に対し、部分系列 $v_k,e_k,v_{k+1},\ldots,e_{k'-1},v_{k'}(0\leq k\leq k'\leq n)$ を $\rho$ の部分経路と呼ぶ。

上記通信網での通信については、パケット通信などに相当する以下の方法を仮定する。

**定義 1 (ルーティング手法)**

1. 各頂点 $v$ は転送用情報 $F_v$ を持つ。

   $F_v$ はルーティングテーブルに相当する情報であり、$E_v$ を含む。

2. 頂点 $v$ が仮想頂点 $w$ にメッセージを送ることが許された時、$v$ はヘッダ情報 $H_{v,w}$ を持つ。

3. 頂点 $v$ が $w$ にメッセージ $M$ を送る時、$v$ は $M$ と $H_{v,w}$ からパケット $P(v,w;M)$ を作り、$H_{v,w}$ から辺 $e\in E_v$ を導出し、$P(v,w;M)$ を辺 $e$ に向かって (すなわち、$e=(v,v')$ とすると $v'$ に) 送る。

   パケット $P$ に対し、$P$ の送り手から受け手までの経路を $\rho(P)$ とする。
   $\rho(P(v,w;M))=\rho(v,w)$ である。

4. 頂点 $v$ がパケット $P$ を受信すると、$v$ は $P$ と $F_v$ からこのパケットの宛先が $v$ であるか否かを判定し、

   (a) もし宛先が $v$ であれば、$v$ は $P$ と $F_v$ からメッセージ $M$ を導出する。

   (b) もし宛先が $v$ でなければ、$v$ は $P$ と $F_v$ から辺 $e'$ を導出する。ここで $e'$ は、$\rho(P)$ における $v$ の次の辺である。そして、$v$ は $P$ もしくは $P$ に対してある変換を行なった $P'$(但し $\rho(P')=\rho(P)$ である) を $e'$ に向かって送る。

□

一般に、パケットのルーティングの方法は多種存在する。そのうち、最も特徴的なものは以下の2つである。

1. ソースルーティング: 送り手において経路は定まっている。すなわち、経路を決定する情報は $H_{v,w}$ のみに含まれている。このとき、$F_v$ は $E_v$ のみになる。

2. ホップ - バイ - ホップルーティング: 中継を行なう頂点に来た時点で初めて次にどこに向かうかを定める。すなわち、経路を決定する情報は $F_v$ にのみ含まれている。このとき、$H_{v,w}$ は $w$ もしくは $N(w)$ のみになる。

実システムのルーティングとしてはこの中間の形態のものも多数存在する。本稿では、ルーティングとしてソースルーティングを仮定する。

### 2.1.2 経路情報の安全性問題

次に、上記のようにして通信を行なう網の中継頂点における解読について定義する。$U \subset V$ を共謀解読者集合とする。各頂点 $u \in U$ は、他の全ての頂点 $u' \in U$ の持つ情報をすべて得ることができると仮定する。ここでは $U$ はただ一つであると仮定するが、$U$ が複数あったとしても、それらの間で情報の共有がなければ独立な解読者集合であるので、以下の議論は各解読者集合に対して成立する。

解読の定義の前に、秘密にすることが不可能な情報を公開情報として、以下のように定義する。

**定義 2 (公開情報)**

頂点 $u \in U$ およびパケット $P$ に対して、$u$ に対する $P$ の公開情報 $R_{u,U}(P)$ を、$\rho(P) = v_0, e_0, v_1, \ldots, e_{n-1}, v_n$ の部分経路 $v_k, e_k, v_{k+1}, \ldots, e_{k'-1}, v_{k'}$ で、

- ある $i(k \leq i \leq k')$ に対して $u = v_i$
- 任意の $j(k < j < k')$ に対して $v_j \in U$

を満足する最長のものとする。

また、頂点 $u \in U$ およびヘッダ情報 $H_{u,w}$ に対して、$u$ に対する $H_{u,w}$ の公開情報 $R_{u,U}(H_{u,w})$ を $\rho(u,w)$ の部分経路 $v_0(=u), e_0, v_1, \ldots, e_{k-1}, v_k$ で、任意の $j(0 \leq j < k)$ に対して $v_j \in U$ を満足する最長のものと定義する。 □

パケット $P$ が $R_{u,U}(P)$ を経由するという情報を $u$ に秘密にすることは不可能である。例えば、図1の例において、$\rho(v,w) = v, e_0, v_1, \ldots, v_5, e_5, v_6$ とし、$U = \{v, v_1, v_3, v_4\}$ とする。このとき、$R_{v_1,U}(P(v,w;M)) = v, e_0, v_1, e_1, v_2$、$R_{v_3,U}(P(v,w;M)) = v_2, e_2, v_3, e_3, v_4, e_4, v_5$ である。また、$R_{v,U}(P(v,w;M)) = R_{v_1,U}(P(v,w;M))$、$R_{v_4,U}(P(v,w;M)) = R_{v_3,U}(P(v,w;M))$ が成立する。図1からは、$U$ は $P(v,w;M)$ が $v$ から $v_5$ までを経由することを検出可能であるように思われるかも知れない。しかし、もし $v_2 \notin U$ が $P(v,w;M)$ に対し

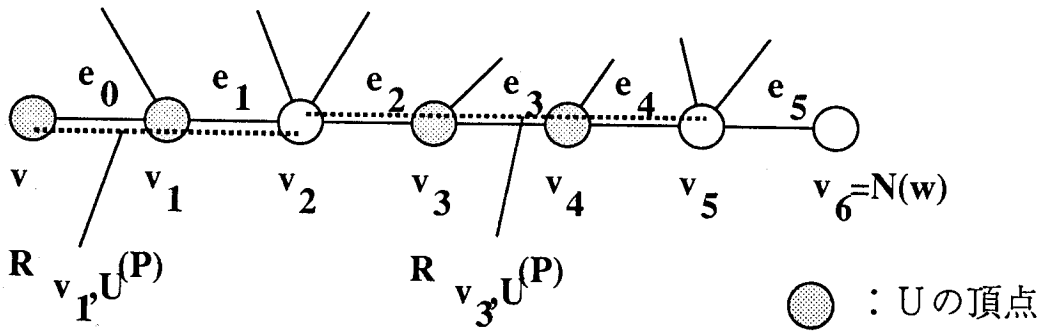


図 1: 公開情報の例

てある変換を施した後に送信すれば、$U$ には $v_2$ に入るパケットと $v_2$ から出て $v_3$ に向かうパケットが同一のものであることを検出不可能かも知れない。従って、これは必ず検出可能な情報であるとは言えない。

また、$\rho(u,w)$ の経路が $R_{u,U}(H_{u,w})$ を含むことを $u$ に秘密にすることは不可能である。例えば、図1の例において、$R_{v,U}(H_{v,w}) = v, e_0, v_1, e_1, v_2$ である。

ここで、経路情報の安全性の問題を以下のように定義する。

**定義 3 (経路情報の安全性問題)**

(a) 共謀集合 $U$ に含まれる頂点 $u$ に対し、$u$ が中継しているパケット $P$ について、$R_{u,U}(P)$ 以外の経路情報は計算量的に安全にする。

(b) 共謀集合 $U$ に含まれる頂点 $u$ に対し、ヘッダ情報 $H_{u,w}$ について、$R_{u,U}(H_{u,w})$ 以外の経路情報は計算量的に安全にする。 □

パケットから経路情報を導出する例としては以下のようなものが考えられる。

- パケット $P$ の送り手または受け手頂点を知る。
- 2つのパケット $P, P'$ の送り手ないし受け手が同一であるか否かを求める。
- パケット $P$ がある頂点 $v$ を通過するか否かを求める。

ヘッダ情報から経路情報を導出する例としては以下のようなものが考えられる。

- $H_{u,w}$ の受け手頂点を知る。
- $H_{u,w}$ と $H_{u,w'}$ が同一であるか否かを求める。
- $H_{u,w}$ が頂点 $v$ を通過するか否かを求める。

ここでは、頂点は通信網 $G$ あるいはルーティング決定方法についての情報を持たないとしている。もしそのような情報があれば、それらを利用して経路情報を得ることが可能である。例えば、$u$ が $G$ を知っていて、$G-u$ が 2 つの連結成分 $G_1$ および $G_2$ に分かれるとする。このとき、$u$ は各パケットの宛先が $G_1$ であるか $G_2$ であるかを検出することが可能である。別の例として、$G$ がリングで頂点集合が $\{v_1, v_2, v_3, v_4, u\}$ であり、$u$ が $G$ およびルーティングが最短経路によるものであることを知っていたとする。このとき、$u$ は $v_4$ から自分に送られてきたパケット $P$ が自分あてでなければ $v_1$ あてであると判定可能である。

### 2.1.3 解読者の形態

次に、解読者の機能について定義を行なう。解読者の機能としては、受動的解読者 (passive adversary) と、悪意のある盗聴者 (malicious adversary) という 2 種類のものが考えられている [25]。前者は要求される動作以外の余計な動作を行なって情報を得ようとするが、要求される動作については正しく行なう解読者である。それに対して後者は、要求される動作を逸脱したいかなる動作も行ない得る盗聴者である。すなわち、経路情報の安全性の問題においては、受動的解読者はパケットの中継、および自分が送り手となる時の送出するパケットの構成などはすべて決められたアルゴリズムに従う。そして、中継・送信・受信を行なう時に得られたパケットを解析して経路情報を引き出そうとする。これに対し、悪意のある解読者は、パケットの中継を正しく行なわない (誤った相手に送る、パケットの内容を変更する)、不正なパケットを送出するなどのことも行なって、通信の妨害をも意図して動作する。本稿では、受動的解読者のみについて考察し、悪意のある解読者は存在しないものとする。すなわち、解読者は何らかの経路情報を得るまでは通信の妨害は行なわないものとする。但し、経路情報を得る目的が通信の妨害を行なうことである場合はあり得るものとする。例えば、解読者 $u \in U$ は中継するパケットが $U$ あてであるか否かを調べ、その情報が得られたら、$U$ あてでないパケットはすべて捨てる、という行為は本稿での解読者が行ない得る。経路情報を得るまでは、解読者はすべてのパケットに対して妨害を行なうか、ランダムに選んだパケットに対して妨害を行なうかしかできない。しかしこのような、経路情報を得ることなく行なう妨害行為はないものとする。特に、$|U| > 1$ の場合には、ランダムに選んだパケットは $U$ の頂点あてであるかも知れない。また、経路情報を得ることなく行なう上記のような妨害行為を防ぐためには、$v$ から $w$ にメッセージ $M$ を送る場合に、その情報 $M$ を分割して複数の経路を用いて通信する方法が考えられている [19]。この手法は本稿の

経路情報暗号化法と併用することができる。

その他、解読者が頂点間を移動するようなモデルも考えられている [25] が、本稿では、解読者は移動を行なわないものとする。

次に、共謀の形態として以下の3つのものを考える。

**定義 4 (共謀形態)**

(1) 共謀なし、すなわち $|U| = 1$。

(2) 共謀あり ($|U| > 1$)、但しパケット $P$ に対してその送り手も受け手も共謀者に含まれない。

(3) 共謀あり ($|U| > 1$)、パケット $P$ に対してその送り手または受け手も共謀者に含まれ得る。 □

共謀形態 (2) と (3) の違いは定義3の要求条件 (a) にのみ関わる。もし形態 (3) の共謀のもとで、パケットの送り手もしくは受け手の情報が得られたとすれば、$U$ の頂点は $U$ あてもしくは $U$ からではないパケットをすべて捨てるということが可能になる。

## 2.2 従来の研究

Chaum [7] は経路情報の安全性についての2つの手法を提案した。この手法の安全性の達成度を表1に示す。追跡不可能なメイルシステム (untraceable mail system) は中継頂点における経路情報の安全性を保証する以下のような暗号化法である。公開鍵暗号方式の存在を仮定し、各頂点 $v_i$ に対する公開鍵による暗号化の関数を $E_i$、秘密鍵による復号化関数を $D_i$ とする。このとき、任意の $X$ に対して $E_i(D_i(X)) = D_i(E_i(X)) = X$ である。$v$ から $w$ への経路 $\rho(v,w) = v_0, e_0, v_1, \ldots, e_{n-1}, v_n$ とする。$v$ が $w$ にメッセージ $M$ を送る場合、$v$ は以下のようなパケット $P$ を作る。

$$P = E_1(R_1, E_2(R_2, \ldots, E_{n-1}(R_{n-1}, E_n(R_n, D_0(R_0, M), A))\ldots))$$

ここで、$R_i (i = 0, 1, \ldots, n)$ は乱数、$A$ は受け手の頂点名を表す。Chaum の方法では、中継を担当するのは MIX という特別な頂点であるとしている。MIX のネットワークは直列接続しか考えられていないが、一般のトポロジーに対処するには、経路を示す情報 $A_i$ が頂点 $v_i$ で得られるように、

$$P = E_1(R_1, A_1, E_2(R_2, A_2, \ldots, E_{n-1}(R_{n-1}, A_{n-1}, E_n(R_n, A_n, D_0(R_0, M)))\ldots))$$

のようなパケットを構成すればよい。このパケット $P$ を受けとったノード $v_1$ は $D_1(P)$ を行なって

$$P' = (R_1, A_1, E_2(R_2, A_2, \ldots, E_{n-1}(R_{n-1}, A_{n-1}, E_n(R_n, A_n, D_0(R_0, M)))\ldots))$$

を得る。$A_1$ より次に送るべき MIX を知り、

$$P_2 = E_2(R_2, A_2, \ldots, E_{n-1}(R_{n-1}, A_{n-1}, E_n(R_n, A_n, D_0(R_0, M)))\ldots)$$

を $v_2$ に送る。$v_2$ など他の頂点についても同様である。最終的に頂点 $v_n$ に送られるパケット $P_n$ は

$$P_n = E_n(R_n, A_n, D_0(R_0, M))$$

である。$v_n$ は $D_n(P_n)$ を行ない、$(A_n, D_0(R_0, M))$ の組を得る。$A_n$ よりこのパケットが自分あてであることを検出し、$E_0(D_0(R_0, M))$ を行なってメッセージ $M$ を得る。この方法の問題点は、送り手頂点 $v_0$ は受け手までの経路を知っている必要がある点である。従って、受け手が送り手から匿名であることは不可能である。また、送り手の頂点はパケットが途中の頂点でどのように変換されるかを知っている。従って、送り手が共謀した場合、中継頂点でどこからどこへのパケットであるかを検出可能である。

Chaum の提案したもう一つの方式、追跡不可能なリターンアドレス (untraceable return address) は $v$ が $v$ 自身を知られることなく、$v$ あてのパケットを構成する方法を $w$ に伝える手法である。$v$ から $w$ への経路を $\rho(v, w) = v_0, e_0, v_1, \ldots, e_{n-1}, v_n$ とすると、$v$ は $w$ に対して、以下のメッセージ $RA$ を伝える。

$$RA = (E_{n-1}(R_{n-1}, A_{n-1}, E_{n-2}(R_{n-2}, A_{n-2}, \ldots, E_1(R_1, A_1, E_0(R_0, A_0))\ldots)), K_0)$$

ここで $K_0$ は $v$ が決めた暗号化鍵である。$R_i (i = 0, \ldots, n-1)$ は乱数、$A_i$ は次に送る MIX を記した情報である。$v$ が $w$ に $RA$ を送るには、前述の追跡不可能なメイルシステムのアルゴリズムを用いればよい。このメッセージを受けとった $w$ は、メッセージ $M$ を $v$ に送る場合に以下のパケットを作る。

$$P = (E_{n-1}(R_{n-1}, A_{n-1}, E_{n-2}(R_{n-2}, A_{n-2}, \ldots, E_1(R_1, A_1, E_0(R_0, A_0))\ldots)), E_{K_0}(M))$$

ここで $E_{K_0}(M)$ は、鍵 $K_0$ を用いた暗号化である。このパケットを受けとった $v_{n-1}$ は、前半部について、$D_{n-1}$ による復号化を行なう。その結果得られた $A_{n-1}$ より次に送るべき MIX を知り、$R_{n-1}$ を鍵としてメッセージを暗号化した

$$P_{n-1} = (E_{n-2}(R_{n-2}, A_{n-2}, \ldots, E_1(R_1, A_1, E_0(R_0, A_0))\ldots)), E_{R_{n-1}}(E_{K_0}(M)))$$

| 要求条件 | 共謀形態 | Chaum 1 | Chaum 2 | 手続き 1 | 手続き 2 |
|---|---|---|---|---|---|
| (a) ($P$ から) | (1) | ○ | △ | ○ | ○ |
| | (2) | ○ (送り手を除く) | △ (受け手を除く) | × | ○ |
| | (3) | × | × | × | ○ |
| (b) ($H_{v,w}$ から) | (1) | × | △ | ○ | ○ |
| | (3) | × | △ | × | ○ |

○ 安全　△ 安全 (1 度のみ使用可)　× 安全でない

Chaum 1: "追跡不可能なメールシステム (Untraceable mail system)"

Chaum 2: "追跡不可能なリターンアドレス (Untraceable return address)"

表 1: 経路情報暗号化法の比較

を $v_{n-2}$ に送る。$v_{n-2}$ など他の MIX について同じ操作を繰り返す。$v$ はパケット

$$P_1 = (E_0(R_0, A_0), E_{R_1}(E_{R_2}(\ldots(E_{R_{n-1}}(E_{K_0}(M)))\ldots)))$$

を受け取る。$v$ は前半部を復号化して $A_0$ よりこのパケットが自分あてであることを知り、後半部を復号化して $M$ を得る。暗号化の鍵として用いた乱数 $R_i (i = 1, \ldots, n-1)$ および $K_0$ はすべて $v$ が知っているので $M$ を復号化することが可能である。また、メッセージの暗号化については慣用暗号系 (秘密鍵による方式) を用いればよい。

上記の方法はメッセージの送り手には経路についての情報を与えない。しかし、$w$ はこの経路情報を 2 度以上使用してはいけない。2 度使用した場合には、パケットの前半部が同一であることから、それらの送り手、受け手の対が同一であることを中継 MIX は検出することが可能である。また、メッセージの受け手はパケットの前半部が中継 MIX でどのような値になるかを知っている。従って、受け手が共謀した場合には経路情報を得ることが可能である。

このように、Chaum の方法は上記の要求条件のすべてを満足することはできない。表 1に安全性を示す。

なお、本稿では中継を行なう通信網を対象としているが、中継のない (放送型の) 通信網では、公開鍵暗号をそのまま用いることにより、匿名性を保ったまま通信を行なうことは容易に実現可能である [36]。

## 2.3 経路情報の安全性を達成する暗号化法

本節では、経路情報を計算するセンタの存在を仮定した場合の、上記の要求条件をすべて満足する暗号化方式を述べる。

### 2.3.1 暗号系に対する要求条件

本稿では、通信網 $G$ の外部に、ヘッダ情報 $H_{v,w}$ を計算するセンタ $C$ が存在すると仮定する。ネットワークによっては通信内容の安全のため、センタを設置して各頂点の公開鍵を管理することも行なわれる。上記の仮定はセンタにある種の付加機能をおくことに相当する。$C$ については以下の仮定をおく。

1. $C$ は $H_{v,w}$ を計算するために必要な情報をすべて持つ。

2. $C$ と各頂点 $v$ との間の通信は安全である。

センタ $C$ が全てのメッセージを中継する方法も考えられるが、その方法は $C$ に非常に大きな処理能力が必要であるので現実的ではない。また、各頂点 $v$ が $w$ に1つメッセージを送るたびに $C$ から新しい $H_{v,w}$ を受け取る ($H_{v,w}$ を使い捨てにする) 方法も考えられるが、この方法もまた、$C$ に非常に大きな処理能力を必要とする。本稿では、$C$ の負担を最低限にするため、$v$ と $w$ の組に対して $C$ は $H_{v,w}$ を1度だけ計算するとする。

また、通信路における解読を防止する方法は使用可能とする。すなわち、各隣接頂点間ではリンク暗号を用い、かつ、ダミーメッセージを流すことによってトラフィック量の変動から通信状況を知られることはないものとする。

まず、共謀のない ($|U| = 1$) 場合について考察する。上記の仮定のもとで、以下の性質を満足する暗号系が存在すれば、経路情報の安全性を達成する方法が存在する。

**定義 5 (暗号系に対する要求条件 [共謀なしの場合])**

送り手 $N_1$、変換者 $N_2$、受け手 $N_3$ の3者が存在するとする。

1. $N_1$ は平文 $M$ を暗号化する。暗号化したメッセージを $E(M)$ とする。

2. $N_2$ は $E(M)$ に乱数 $r$ を加えることができるが、$N_2$ は $E(M)$ を復号化することは不可能である。

   乱数を加えられたメッセージを $R(E(M), r)$ とする。

3. $N_3$ は $R(E(M), r)$ を復号化して $M$ を得ることができる。しかし、$N_3$ は $E(M)$ を求めることは不可能である。 □

上記の条件を満足する暗号系が存在すれば、共謀が存在しない場合の経路情報の安全性は以下のようにして達成される。上記の条件において、$M$、$N_1$、$N_2$、$N_3$ をそれぞれ $\rho(v,w)$、センタ $C$、送り手 $v$、中継ノード $v_i$ とする。送り手 $v$ が $w$ にメッセージを送信することを許され、経路 $\rho(v,w)$ を

$$v_0(=v), e_0, v_1, \ldots, v_{n-1}, e_{n-1}, v_n(=N(w))$$

とする。

このとき $v$ は $C$ から暗号化されたヘッダ情報 $H_{v,w}$ を受け取る。$H_{v,w}$ は系列

$$e_0, E_1(e_1, r_1), E_2(e_2, r_2), \ldots, E_{n-1}(e_{n-1}, r_{n-1}), E_n(\phi_n, r_n)$$

である。ここで、$E_i(e_i, r_i)$ は $e_i$ に乱数 $r_i$ を加えたものを頂点 $v_i$ のみが復号化できるよう、鍵 $k_i$ を用いて暗号化したものである。また、$\phi_n$ は $v_n$ が受け手であることを表す特別な値とする。$v$ はこの暗号を復号化できないので、$H_{v,w}$ から経路に関する情報を得ることができない。例えば、ある $w'$ に対して $\rho(v,w) = \rho(v,w')$ であったとしても、乱数 $r_i$ により $H_{v,w}$ と $H_{v,w'}$ が同じ経路であることを $v$ が検出することは防止されている。

頂点 $v$ はこのヘッダ情報を復号化することなく、さらに乱数を加えることが可能である。よって、$v$ はメッセージ $M$ を $w$ に送る際に、$v$ は乱数を加えたヘッダを $M$ とともに送信する。ヘッダは

$$R(E_1(e_1, r_1), r), R(E_2(e_2, r_2), r), \ldots, R(E_n(\phi_n, r_n), r)$$

という値である。ここで $R$ は乱数を加える関数であり、$r$ が今回使用する値である。$r$ には送信ごとに異なる値を用いる。

このパケットを中継する際、$\rho(v,w)$ 上の頂点 $v_i$ は鍵 $k_i$ を用いて $R(E_i(e_i, r_i), r)$ を復号化して $e_i$ を得る。$v_i$ はそれ以外の情報をパケットから得ることはできない。例えば、$v$ が $w$ に 2 つのメッセージを送ったとしても、$R$ によって乱数 $r$ を加えられているため、それらが同じヘッダ情報から作られたことを検出できない。

また、$v$ は $E_i(e_i, r_i)$ の値を求めることはできないという条件が存在する。$E_i(e_i, r_i)$ の値が求められたとすると、$H_{v,w}$ として使われた $E_i(e_i, r_i)$ と $H_{v,w'}$ として使われた $E_i(e_i, r_i')$ とを $r_i$、$r_i'$ の違いにより区別することができる。従って、乱数の違いから送り手と受け手の対の同一性を検出することを防止するため上記の条件が存在している。

### 2.3.2 共謀が存在しない場合の暗号化法

現在までに提案されている暗号系の多くは定義 5に示す条件、特に暗号を解かずに乱数を加える、という条件を満足していない。エルガマル暗号系 [21] を利用することによって定義 5に示した条件をすべて満足する暗号系を求めることができる。エルガマル暗号の定義を以下に述べる。

**定義 6 (エルガマル暗号系)** [21]

1. (鍵) $p$ : 大きな素数で、$p-1$ が少なくとも 1 つは大きな素数を因数として持つもの。
   $\alpha$ : mod $p$ におけるある原始根[1]。
   $k_B$ : $0 \leq k_B \leq p-1$ を満足するある数。
   $y_B \equiv \alpha^{k_B} \pmod{p}$ とする。
   $B$ に対して送信する場合の公開暗号化鍵: $p$, $\alpha$, $y_B$.
   受け手 $B$ が持つ秘密復号化鍵: $k_B$.

2. (暗号化) $A$ が $B$ にメッセージ $M(0 \leq M \leq p-1)$ を送信する場合には、以下のように行なう。
   $A$ は乱数 $k(0 \leq k \leq p-1)$ を 1 つ選び、
   $c_1 \equiv \alpha^k \pmod{p}$, $c_2 \equiv y_B^k \cdot M \pmod{p}$
   を計算する。暗号文 $(c_1, c_2)$ を $B$ に送る。

3. (復号化) $B$ は $(c_1, c_2)$ を受け取ると、$c_2/c_1^{k_B} \pmod{p}$ を計算して $M$ を得る。

□

この暗号系の安全性は有限体上での対数の計算の複雑さによる、すなわち、$(\mathrm{mod}\ p)$ のもとで、$(x, x^k)$ の対から $k$ を求めるのが困難であることを利用している。

上記暗号系を応用することにより、経路情報の暗号化を以下のようにして行なえばよい。

**手続き 1 (共謀なし ($|U| = 1$) の場合)**

**[準備]**

(1) センタ $C$ は $p-1$ が少なくとも 1 つは大きな素数を因数として持つ素数 $p$ を選ぶ。この値は全頂点に対して共通とする。

[1] $\alpha$ が mod $p$ のもとでの原始根であるとは、$\alpha^0 \alpha^1, \alpha^2, \ldots, \alpha^{p-2}$ が 0 を除く mod $p$ におけるすべての元を網羅していることをいう。

(2) センタ $C$ は各頂点 $v_i$ に対して秘密鍵 $k_i(0 \leq k_i \leq p-1)$ を選ぶ。

(3) 頂点 $v_i$ は $E_{v_i}$ の各辺 $e$ を $0 \leq e \leq p-1$ を満足するようにコード化する。$v_i$ は特別な値 $\phi_i$ を、$0 \leq \phi_i \leq p-1$ を満足し、辺のコードとは異なるように選ぶ。

(4) $v_i$ は $F_{v_i} = \{\phi_i, E_{v_i}$のコード$\}$ を $C$ に送る。

(5) $v_i$ は $C$ から $k_i$ を受け取る。

[仮定]

$G$ の直径を $D$ とする。頂点 $v$ が仮想頂点 $w$ にメッセージ $M(0 \leq M \leq p-1)$ を送るとする。経路 $\rho(v,w)$ を $v_0(=v), e_0, v_1, \ldots, v_{n-1}, e_{n-1}, v_n(=N(w))(n \leq D)$ とする。

$N(w)$ が $C$ に、$v$ から $w$ への経路を教えるよう依頼した時、$C$ は以下の手続き init を実行する。

$v$ は $w$ にメッセージ $M$ を送る際、手続き send を行なう。

頂点 $v_i$ がパケット $P$ を受信した時、手続き receive を行なう。

以下のすべての計算は mod $p$ で行なう。

[手続き init]

$p$ の剰余系での原始根、

$$\alpha_i, \gamma_i (i = 1, \ldots, n, D+1), \epsilon_{i,j} (i = n+1, \ldots, D, \quad j = 1, \ldots, 4)$$

を選んで以下の値を計算する。

$$\begin{aligned} x_i &\equiv e_i \cdot \alpha_i^{k_i} \quad (i = 1, \ldots, n-1), \quad x_n \equiv \phi_n \cdot \alpha_n^{k_n}, \quad x_{D+1} \equiv \alpha_{D+1}^{k_n}, \\ z_i &\equiv \gamma_i^{k_i} \qquad (i = 1, \ldots, n), \qquad\qquad\qquad\quad z_{D+1} \equiv \gamma_{D+1}^{k_n}. \end{aligned}$$

$C$ は $H_{v,w} = \{e_0, (\alpha_1, x_1, \gamma_1, z_1), (\alpha_2, x_2, \gamma_2, z_2), \ldots, (\alpha_n, x_n, \gamma_n, z_n),$
$(\epsilon_{n+1,1}, \epsilon_{n+1,2}, \epsilon_{n+1,3}, \epsilon_{n+1,4}), (\epsilon_{n+2,1}, \epsilon_{n+2,2}, \epsilon_{n+2,3}, \epsilon_{n+2,4}), \ldots, (\epsilon_{D,1}, \epsilon_{D,2}, \epsilon_{D,3}, \epsilon_{D,4}),$
$(\alpha_{D+1}, x_{D+1}, \gamma_{D+1}, z_{D+1})\}$ を $v$ に送る。

[手続き send]

$C$ より受け取った $H_{v,w}$ を $\{e_0, (\alpha_i, x_i, \gamma_i, z_i)(i = 1, 2, \ldots, D+1)\}$ とする。
乱数 $r(0 \leq r \leq p-1)$ を生成して以下の値を計算する。

$$\begin{aligned} \lambda_i &\equiv \alpha_i \cdot \gamma_i^r \quad (i = 1, \ldots, D+1), \\ h_i &\equiv x_i \cdot z_i^r \quad (i = 1, \ldots, D), \qquad h_{D+1} \equiv M \cdot x_{D+1} \cdot z_{D+1}^r. \end{aligned}$$

$v$ はパケット $P(v, w; M) = (\lambda_1, h_1), \ldots, (\lambda_{D+1}, h_{D+1})$ を $e_0$ に送る。

[手続き receive]

受け取ったパケット $P$ を $\{(\lambda_i, h_i)(i = 1, 2, \ldots, D+1)\}$ とする。
$q \equiv h_1/\lambda_1^{k_i}$ を計算する。
**If** $q \equiv \phi_i$ **then** $h_{D+1}/\lambda_{D+1}^{k_i}$ を計算して $M$ を得る。
**else begin** $p$ の剰余系における原始根 $\epsilon, \epsilon'$ を選び、パケット

$$P' = (\lambda_2, h_2), \ldots, (\lambda_D, h_D), (\epsilon, \epsilon'), (\lambda_{D+1}, h_{D+1})$$ を $q$ に送る。

**end** □

ここで、$\gamma_i$, $z_i$ は乱数を加えるための項である。任意の $r$ に対して
$z_i^r \equiv \gamma_i^{r \cdot k_i} \pmod{p}$ が成立するため、$r$ をどのように選んでも経路上の頂点で求められる $q$ の値に変化はない。従って、送り手は暗号化されたヘッダ情報を復号化することなく、乱数を加えることが可能である。

経路の長さ $n$ についての情報を隠すため、$C$ はランダムな値 $\epsilon_{i,j}(i = n+1, \ldots, D, j = 1, \ldots, 4)$ を選んで長さ $D - n$ のダミーを付け加える。従ってすべての $H_{v,w}$ は同じ長さを持つ。パケットを受け取った $v_i$ もダミー $(\epsilon, \epsilon')$ を加えてパケット長を一定に保つ。もし $v_i$ が $D+1$ より短いパケットを受け取った場合には、$v_i$ は隣の頂点が不正な行為を行なっていると判定できる。

この暗号系の安全性は以下の定理で示される。

**定理 1** 手続き 1 の経路情報暗号が $|U| = 1$ の場合に解読可能であれば、エルガマル暗号系も解読可能である。 □

**(証明)**

まず、ヘッダ情報 $H$ から情報をとり出すことを考える。送り手 $v$ の受け取る情報は $(\alpha_i, x_i, \gamma_i, z_i)(i = 1, \ldots, D+1)$ である。$x_i$ と $z_i$ の定義より、各 4 つ組はエルガマル暗号における平文 $e_i$(または $\phi_n$) と "1" の暗号文である。もしも $e_i$(または $\phi_n$) の値がこれから得られた場合には、エルガマル暗号系もこのような選択平文攻撃が可能である。

次に、経路 $\rho(v, w)$ 上の頂点 $v'$ がパケット $P$ を受け取った場合を考える。ある対 $(\lambda, h) \equiv (\alpha_i \cdot \gamma_i^r, e_i \cdot \alpha_i^{k_i} \cdot \gamma_i^{r \cdot k_i})$ について考察する。ここで、$v' \neq v_i$ とする。$\gamma_i$ が原始根であるので、$\alpha_i$ はある整数 $x$ を用いて $\gamma_i^x$ と書くことができる。従って、$(\lambda, h) \equiv (\gamma_i^{x+r}, e_i \cdot (\gamma_i^{k_i})^{x+r})$ と表すことができる。よって、この対はエルガマル暗号系の暗号

文で、使用した乱数 $k$ が $x+r$ である場合であるとみなすことができる。$r$ は乱数であるので、もしこの対から $e_i$ を求めることができたとしたら、エルガマル暗号も多くの $k$ に対して解くことが可能である。

その他の経路情報、例えば2つのパケットが同じ $e_i$ を持つか否かを検出するなどの問題についても、同じことをエルガマル暗号に対して行なうことができることを同様の方法で証明できる。従って、この暗号系を解くことができればエルガマル暗号を解くことも可能となる。 □

逆に、もしもエルガマル暗号が解読可能ならば、この暗号系も解読可能なのは明らかである。従って、この暗号系とエルガマル暗号系の安全性は同等である。

この暗号系は $|U|=1$、すなわち共謀が存在しない場合には安全であるが、$|U|>1$、すなわち共謀が存在する場合には安全ではない。2つの頂点 $v_A$ と $v_B$ が共謀すると仮定する。このとき以下のような解読が可能である。$v_A$ がパケット $P$ を受け取った場合に、$v_A$ はこのパケットが $v_B$ を経由するか否かを、パケットの各部分を $v_B$ の秘密鍵で解くことによって検出できる。ある枝 $e(e \in E_{v_B})$ のコードが得られればこのパケットは $v_B$ を経由するものとみなすことができる。ヘッダ情報 $H$ から情報を得る場合にも同様の共謀が可能である。

### 2.3.3 共謀が存在する場合の暗号化法

本節では、共謀が存在し得る場合 $(|U| \geq 1)$ のアルゴリズムを示す。

**手続き 2 ($|U| \geq 1$ の場合)**

**[準備]**：(2)(5) 以外は手続き 1 と同じである。

(2') $C$ は各頂点 $v_i$ に対して $2D+1$ 個の秘密鍵 $k_{i,j}(j=1,\ldots,2D+1)$ を、$0 \leq k_{i,j} \leq p-1$ を満足するように選ぶ。

(5') $v_i$ は $C$ から $k_{i,j}(j=1,\ldots,2D+1)$ を受け取る。

**[仮定]**：手続き 1 と同じ。

**[手続き init]**

$p$ における剰余系の原始根

$$\alpha_i, \gamma_i(i=1,\ldots,n,D+1), \epsilon_{i,j}(i=n+1,\ldots,D,\ j=1,\ldots,4)$$

を選び、以下の値を計算する。

$$\begin{aligned}
\beta_i &\equiv \alpha_i \cdot \textstyle\prod_{j=1}^{i-1} \alpha_j^{k_{j,2(i-j)}} \quad (i=1,\ldots,n), & \beta_{D+1} &\equiv \alpha_{D+1} \cdot \textstyle\prod_{j=1}^{n-1} \alpha_j^{k_{j,2D}},\\
x_i &\equiv e_i \cdot \textstyle\prod_{j=1}^{i} \alpha_j^{k_{j,2(i-j)+1}} (i=1,\ldots,n-1), & x_n &\equiv \phi_n \cdot \textstyle\prod_{j=1}^{n} \alpha_j^{k_{j,2(n-j)+1}},\\
x_{D+1} &\equiv \alpha_{D+1}^{k_{n,2D+1}} \cdot \textstyle\prod_{j=1}^{n-1} \alpha_j^{k_{j,2D+1}},\\
\delta_i &\equiv \gamma_i \cdot \textstyle\prod_{j=1}^{i-1} \gamma_j^{k_{j,2(i-j)}} \quad (i=1,\ldots,n), & \delta_{D+1} &\equiv \gamma_{D+1} \cdot \textstyle\prod_{j=1}^{n-1} \gamma_j^{k_{j,2D}},\\
z_i &\equiv \textstyle\prod_{j=1}^{i} \gamma_j^{k_{j,2(i-j)+1}} \quad (i=1,\ldots,n), & z_{D+1} &\equiv \gamma_{D+1}^{k_{n,2D+1}} \textstyle\prod_{j=1}^{n-1} \gamma_j^{k_{j,2D+1}}.
\end{aligned}$$

$C$ は $H_{v,w} = \{e_0, (\beta_1, x_1, \delta_1, z_1), (\beta_2, x_2, \delta_2, z_2), \ldots, (\beta_n, x_n, \delta_n, z_n),$
$(\epsilon_{n+1,1}, \epsilon_{n+1,2}, \epsilon_{n+1,3}, \epsilon_{n+1,4}), (\epsilon_{n+2,1}, \epsilon_{n+2,2}, \epsilon_{n+2,3}, \epsilon_{n+2,4}), \ldots, (\epsilon_{D,1}, \epsilon_{D,2}, \epsilon_{D,3}, \epsilon_{D,4}),$
$(\beta_{D+1}, x_{D+1}, \delta_{D+1}, z_{D+1})\}$ を $v$ に送る。

[手続き send]

$C$ より受け取った $H_{v,w}$ を $\{e_0, (\beta_i, x_i, \delta_i, z_i)(i=1,2,\ldots,D+1)\}$ とする。
乱数 $r_{0,1}, r_{0,2} (0 \le r_{0,1}, r_{0,2} \le p-1)$ を生成し、以下の値を計算する。

$$\begin{aligned}
\lambda_i &\equiv \beta_i \cdot \delta_i^{r_{0,1}} \quad (i=1,\ldots,D+1),\\
h_i &\equiv x_i \cdot z_i^{r_{0,1}} \quad (i=1,\ldots,D), & h_{D+1} &\equiv M \cdot x_{D+1} \cdot z_{D+1}^{r_{0,1}},\\
\kappa_i &\equiv \delta_i^{r_{0,2}} \quad (i=1,\ldots,D+1), & g_i &\equiv z_i^{r_{0,2}} \quad (i=2,\ldots,D+1).
\end{aligned}$$

$v$ はパケット $P(v,w;M) = (\lambda_1, h_1, \kappa_1), (\lambda_2, h_2, \kappa_2, g_2), (\lambda_3, h_3, \kappa_3, g_3), \ldots,$
$(\lambda_{D+1}, h_{D+1}, \kappa_{D+1}, g_{D+1})$ を $e_0$ に送る。

[手続き receive]

受け取ったパケット $P$ を $\{(\lambda_1, h_1, \kappa_1), (\lambda_i, h_i, \kappa_i, g_i)(i=2,3,\ldots,D+1)\}$ とする。
$q \equiv h_1/\lambda_1^{k_{i,1}}$ を計算する。

**If** $q \equiv \phi_i$ **then** $h_{D+1}/\lambda_{D+1}^{k_{i,2D+1}}$ を計算して $M$ を得る。

**else begin** 乱数 $r_{i,1}, r_{i,2} (0 \le r_{i,1}, r_{i,2} \le p-1)$ を生成し、以下の値を計算する。

$$\begin{aligned}
\lambda'_j &\equiv \lambda_j/\lambda_1^{k_{i,2j-2}} \quad (j=2,\ldots,D+1), & h'_j &\equiv h_j/\lambda_1^{k_{i,2j-1}} \quad (j=2,\ldots,D+1),\\
\kappa'_j &\equiv \kappa_j/\kappa_1^{k_{i,2j-2}} \quad (j=2,\ldots,D+1), & g'_j &\equiv g_j/\kappa_1^{k_{i,2j-1}} \quad (j=2,\ldots,D+1),\\
\lambda''_j &\equiv \lambda'_j \cdot \kappa_j^{\prime r_{i,1}} \quad (j=2,\ldots,D+1), & h''_j &\equiv h'_j \cdot g_j^{\prime r_{i,1}} \quad (j=2,\ldots,D+1),\\
\kappa''_j &\equiv \kappa_j^{\prime r_{i,2}} \quad (j=2,\ldots,D+1), & g''_j &\equiv g_j^{\prime r_{i,2}} \quad (j=3,\ldots,D+1).
\end{aligned}$$

$p$ における剰余系の原始根 $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ を選び、パケット
$P' = (\lambda''_2, h''_2, \kappa''_2), (\lambda''_3, h''_3, \kappa''_3, g''_3), \ldots, (\lambda''_D, h''_D, \kappa''_D, g''_D), (\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4),$
$(\lambda''_{D+1}, h''_{D+1}, \kappa''_{D+1}, g''_{D+1})$ を $q$ に送る。

end □

手続き 1 と同じように、$\epsilon$ はダミー、$\delta_i$ と $z_i$ は乱数を加えるための項であり、経路上の頂点で計算される $q$ の値には影響しない。この手続きでは、パケットの各部分は経路上のすべての頂点の鍵を用いて暗号化される。これにより一部の頂点が共謀した場合の解読を防止している。また、経路上のすべての頂点でさらに乱数を加えることにより、ある頂点の入りパケットと出パケットの間の対応関係を導出することを防止している。

(例)

$D = 3$、$\rho(v,w) = v, e_0, v_1, e_1, v_2$ とする。$C$ は init を行なって以下の値を計算する。

$$\begin{array}{lll}
\beta_1 \equiv \alpha_1, & \beta_2 \equiv \alpha_1^{k_{1,2}} \cdot \alpha_2, & \beta_4 \equiv \alpha_1^{k_{1,6}} \cdot \alpha_4, \\
x_1 \equiv e_1 \cdot \alpha_1^{k_{1,1}}, & x_2 \equiv \phi_2 \cdot \alpha_1^{k_{1,3}} \cdot \alpha_2^{k_{2,1}}, & x_4 \equiv \alpha_1^{k_{1,7}} \cdot \alpha_4^{k_{2,7}}, \\
\delta_1 \equiv \gamma_1, & \delta_2 \equiv \gamma_1^{k_{1,2}} \cdot \gamma_2, & \delta_4 \equiv \gamma_1^{k_{1,6}} \cdot \gamma_4, \\
z_1 \equiv \gamma_1^{k_{1,1}}, & z_2 \equiv \gamma_1^{k_{1,3}} \cdot \gamma_2^{k_{2,1}}, & z_4 \equiv \gamma_1^{k_{1,7}} \cdot \gamma_4^{k_{2,7}}.
\end{array}$$

$C$ は $H_{v,w} = \{e_0, (\beta_1, x_1, \delta_1, z_1), (\beta_2, x_2, \delta_2, z_2), (\epsilon_{3,1}, \epsilon_{3,2}, \epsilon_{3,3}, \epsilon_{3,4}), (\beta_4, x_4, \delta_4, z_4)\}$ を $v$ に送る。これを受け取った $v$ は send を実行して以下の値を計算する。

$$\begin{array}{lll}
\lambda_1 \equiv \alpha_1 \cdot \gamma_1^{r_{0,1}}, & \lambda_2 \equiv \alpha_1^{k_{1,2}} \cdot \alpha_2 \cdot \gamma_1^{r_{0,1} \cdot k_{1,2}} \cdot \gamma_2^{r_{0,1}}, \\
\lambda_3 \equiv \epsilon_{3,1} \cdot \epsilon_{3,3}^{r_{0,1}}, & \lambda_4 \equiv \alpha_1^{k_{1,6}} \cdot \alpha_4 \cdot \gamma_1^{r_{0,1} \cdot k_{1,6}} \cdot \gamma_4^{r_{0,1}}, \\
h_1 \equiv e_1 \cdot \alpha_1^{k_{1,1}} \cdot \gamma_1^{r_{0,1} \cdot k_{1,1}}, & h_2 \equiv \phi_2 \cdot \alpha_1^{k_{1,3}} \cdot \alpha_2^{k_{2,1}} \cdot \gamma_1^{r_{0,1} \cdot k_{1,3}} \cdot \gamma_2^{r_{0,1} \cdot k_{2,1}}, \\
h_3 \equiv \epsilon_{3,2} \cdot \epsilon_{3,4}^{r_{0,1}}, & h_4 \equiv M \cdot \alpha_1^{k_{1,7}} \cdot \alpha_4^{k_{2,7}} \cdot \gamma_1^{r_{0,1} \cdot k_{1,7}} \cdot \gamma_4^{r_{0,1} \cdot k_{2,7}}, \\
\kappa_1 \equiv \gamma_1^{r_{0,2}}, & \kappa_2 \equiv \gamma_1^{r_{0,2} \cdot k_{1,2}} \cdot \gamma_2^{r_{0,2}}, \\
\kappa_3 \equiv \epsilon_{3,3}^{r_{0,2}}, & \kappa_4 \equiv \gamma_1^{r_{0,2} \cdot k_{1,6}} \cdot \gamma_4^{r_{0,2}}, \\
g_2 \equiv \gamma_1^{r_{0,2} \cdot k_{1,3}} \cdot \gamma_2^{r_{0,2} \cdot k_{2,1}}, & g_3 \equiv \epsilon_{3,4}^{r_{0,2}}, \\
g_4 \equiv \gamma_1^{r_{0,2} \cdot k_{1,7}} \cdot \gamma_4^{r_{0,2} \cdot k_{2,7}}. &
\end{array}$$

$v$ はパケット $P = (\lambda_1, h_1, \kappa_1), (\lambda_2, h_2, \kappa_2, g_2), (\lambda_3, h_3, \kappa_3, g_3), (\lambda_4, h_4, \kappa_4, g_4)$ を $e_0$ に送る。頂点 $v_1$ がこのパケットを受信し、$q \equiv h_1/\lambda_1^{k_{1,1}} \equiv e_1$ であるので、このパケッ

トは $e_1$ に送られるべきであることを知る。従って $v_1$ は以下の値をさらに計算する。

$$\begin{aligned}
\lambda_2' &\equiv \alpha_2 \cdot \gamma_2^{r_{0,1}}, & \lambda_4' &\equiv \alpha_4 \cdot \gamma_4^{r_{0,1}},\\
h_2' &\equiv \phi_2 \cdot \alpha_2^{k_{2,1}} \cdot \gamma_2^{r_{0,1}\cdot k_{2,1}}, & h_4' &\equiv M \cdot \alpha_4^{k_{2,7}} \cdot \gamma_4^{r_{0,1}\cdot k_{2,7}},\\
\kappa_2' &\equiv \gamma_2^{r_{0,2}}, & \kappa_4' &\equiv \gamma_4^{r_{0,2}},\\
g_2' &\equiv \gamma_2^{r_{0,2}\cdot k_{2,1}}, & g_4' &\equiv \gamma_4^{r_{0,2}\cdot k_{2,7}},\\
\lambda_2'' &\equiv \alpha_2 \cdot \gamma_2^{r_{0,1}+r_{0,2}\cdot r_{1,1}}, & \lambda_4'' &\equiv \alpha_4 \cdot \gamma_4^{r_{0,1}+r_{0,2}\cdot r_{1,1}},\\
h_2'' &\equiv \phi_2 \cdot \alpha_2^{k_{2,1}} \cdot \gamma_2^{(r_{0,1}+r_{0,2}\cdot r_{1,1})\cdot k_{2,1}}, & h_4'' &\equiv M \cdot \alpha_4^{k_{2,7}} \cdot \gamma_4^{(r_{0,1}+r_{0,2}\cdot r_{1,1})\cdot k_{2,7}},\\
\kappa_2'' &\equiv \gamma_2^{r_{0,2}\cdot r_{1,2}}, & \kappa_4'' &\equiv \gamma_4^{r_{0,2}\cdot r_{1,2}},\\
g_4'' &\equiv \gamma_4^{r_{0,2}\cdot r_{1,2}\cdot k_{2,7}}.
\end{aligned}$$

なお、$\lambda_3', h_3', \kappa_3', g_3', \lambda_3'', h_3'', \kappa_3''$ および $g_3''$ はダミー値に対する計算なので、その結果は意味のない値である。

$v_1$ はパケット $P' = (\lambda_2'', h_2'', \kappa_2''), (\lambda_3'', h_3'', \kappa_3'', g_3''), (\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4), (\lambda_4'', h_4'', \kappa_4'', g_4'')$ を $e_1$ に送る。$v_2$ がこのパケットを受け取り、$q \equiv h_2''/\lambda_2''^{k_{2,1}} \equiv \phi_2$ を計算してこのパケットが自分あてであることを知る。従って $v_2$ は $h_4''/\lambda_4''^{k_{2,7}} \equiv M$ を計算して $M$ を得る。

□

この暗号系の安全性は以下の定理で示される。

**定理 2** 手続き 2 の経路情報暗号が $|U| \geq 1$ の場合に解読可能であれば、エルガマル暗号系も解読可能である。 □

**(証明)**

まず、ヘッダ情報から経路情報を求める場合について考える。送り手の頂点 $v$ はヘッダ情報 $(\beta_i, x_i, \delta_i, z_i)(i = 1, \ldots, D+1)$ を持つ。$v \in U$ が、この経路 $\rho(v, w)$ が $v, e_0, \ldots, e_{n-1}, v_n$ であるか否かを検出しようとする場合、特に $\rho(v, w)$ が頂点 $v_i$ の次に $e_i$ を経由するか否かを $v$ が検出しようとする場合を考える。ここで、2 通りの状況を考える。

1. $v_i \notin U$
2. $v_i \in U$ で、ある $j < i$ について $v_j \notin U$ が成立

   (すべての $j \leq i$ に対して $v_j \in U$ であるなら、$e_i$ は公開情報となる)

まず、前者の場合について考察する。最悪の場合、すなわちすべての $v_j (j < i)$ が $U$ に含まれるとする。この場合、$k_{j,l} (j < i, l = 1, 2, \ldots, 2D+1)$ の値はすべて知っ

ているので、$v$ は 4 つ組

$$(\alpha_i, e_i \cdot \alpha_i^{k_{i,1}}, \gamma_i, \gamma_i^{k_{i,1}})$$

を $(\beta_i, x_i, \delta_i, z_i)$ から得ることができる。これはエルガマル暗号で平文が $e_i$ と "1" である場合とみなすことができる。従って、もしもこれから $e_i$ を得ることができれば、エルガマル暗号も解読可能である。ヘッダ $H$ と $H'$ の比較などについても同様である。

次に、後者の場合を考える。最悪の場合として、すべての $v_l(l \neq j)$ が $U$ に含まれている場合を考える。ここで $j = i-1$ とする。それ以外の場合についても同様にして証明できる。この場合、$v$ は 4 つ組

$$(\alpha_{i-1}, e_{i-1} \cdot \alpha_{i-1}^{k_{i-1,1}}, \gamma_{i-1}, \gamma_{i-1}^{k_{i-1,1}})$$

を $(\beta_{i-1}, x_{i-1}, \delta_{i-1}, z_{i-1})$ から、4 つ組

$$(\alpha_i \cdot \alpha_{i-1}^{k_{i-1,2}}, e_i \cdot \alpha_{i-1}^{k_{i-1,3}} \cdot \alpha_i^{k_{i,1}}, \gamma_i \cdot \gamma_{i-1}^{k_{i-1,2}}, \gamma_{i-1}^{k_{i-1,3}} \cdot \gamma_i^{k_{i,1}})$$

を $(\beta_i, x_i, \delta_i, z_i)$ から得ることができる。もし $v_i$ が $\alpha_i$ と $\gamma_i$ の値を推測することができれば、$v$ は

$$(\alpha_{i-1}^{k_{i-1,2}}, \alpha_{i-1}^{k_{i-1,3}}, \gamma_{i-1}^{k_{i-1,2}}, \gamma_{i-1}^{k_{i-1,3}})$$

を後者の 4 つ組から得ることができる。従って、$\rho(v,w)$ が $e_i$ を通るか否かを求めるためには、$v$ は少なくとも、$(\alpha_{i-1}, \gamma_{i-1})$ の値を知っているもとで、後者の 4 つ組のうちの少なくとも 1 つが真に $\alpha_{i-1}^{k_{i-1,2}}$ (あるいは $\alpha_{i-1}^{k_{i-1,3}}$, $\gamma_{i-1}^{k_{i-1,2}}$, $\gamma_{i-1}^{k_{i-1,3}}$) であるか否かを検出できなければならない。$k_{i-1,2}$ および $k_{i-1,3}$ の値を $v$ は知らないので、これは選択平文攻撃で平文が "1" である場合に等しい。その他の情報を得る場合についても同様にして証明される。従って、ヘッダ情報から経路についての情報を得られればエルガマル暗号に対する同様な攻撃が可能である。

次に、経路 $\rho(v,w)$ 上の頂点 $v'$ が $v$ から $w$ へのパケット $P$ を受け取った時に、$P$ から経路情報を得る場合を考える。ここでは特に $e_i$ を含む部分から情報を得ることを考える。

ここで 3 種類の解読について考える。

(1) 頂点 $v'$ で得られた 4 つ組から $e_i$ を導出

(2) 頂点 $v'$ を通る 2 つのパケットの今後の経路の比較をする (経路の同一性の判定)

(3) 頂点 $v'$ を通るあるパケット $P$ と頂点 $v''$ を通るあるパケット $P'$ について、$P = P'$ か否かの判定 (パケットの同一性の判定)

証明の前に、頂点 $v_j$ で、$v$ から $w$ へのパケットの $e_i$ を含む 4 つ組、すなわち、パケットの $(i-j)$ 番目の部分 $(\lambda_{i-j}, h_{i-j}, \kappa_{i-j}, g_{i-j})$ ($v$ では $(\lambda_i, h_i, \kappa_i, g_i)$ であった部分) がどのような値になって $v_{j+1}(j < i < n)$ に送られるかを以下に示す。

$$
\begin{aligned}
\lambda_{i-j} \equiv\ & \alpha_i \cdot \gamma_i^{r_{0,1}+\sum_{m=1}^{j} r_{m,1}\cdot\prod_{l=0}^{m-1} r_{l,2}} \\
& \cdot \prod_{m=j+1}^{i-1}(\alpha_m^{k_{m,2(i-m)}} \cdot \gamma_m^{(r_{0,1}+\sum_{q=1}^{j} r_{q,1}\cdot\prod_{l=0}^{q-1} r_{l,2})\cdot k_{m,2(i-m)}}), \\
h_{i-j} \equiv\ & e_i \cdot \alpha_i^{k_{i,1}} \cdot \gamma_i^{(r_{0,1}+\sum_{m=1}^{j} r_{m,1}\cdot\prod_{l=0}^{m-1} r_{l,2})\cdot k_{i,1}} \\
& \cdot \prod_{m=j+1}^{i-1}(\alpha_m^{k_{m,2(i-m)+1}} \cdot \gamma_m^{(r_{0,1}+\sum_{q=1}^{j} r_{q,1}\cdot\prod_{l=0}^{q-1} r_{l,2})\cdot k_{m,2(i-m)+1}}), \\
\kappa_{i-j} \equiv\ & \gamma_i^{\prod_{m=0}^{j} r_{m,2}} \cdot \prod_{m=j+1}^{i-1} \gamma_m^{(\prod_{l=0}^{j} r_{l,2})\cdot k_{m,2(i-m)}}, \\
g_{i-j} \equiv\ & \gamma_i^{(\prod_{m=0}^{j} r_{m,2})\cdot k_{i,1}} \cdot \prod_{m=j+1}^{i-1} \gamma_m^{(\prod_{l=0}^{j} r_{l,2})\cdot k_{m,2(i-m)+1}}.
\end{aligned}
$$

まず、(1) の場合を考える。ここで、共謀の形態によってさらに 2 つの場合に分ける。

(1-1) $v_i \notin U$

(1-2) $v_i \in U$

まず、場合 (1-1) を考える。最悪の場合、すなわちすべての $v_j(j < i)$ が $U$ に含まれる場合を考える。このとき、すべての $r_{j,1}, r_{j,2}(j = 0, \ldots, i-1)$ を $U$ は知っている。$v_{i-1}$ において、$e_i$ の含まれる 4 つ組は

$$(\alpha_i \cdot \gamma_i^{x_1}, e_i \cdot \alpha_i^{k_{i,1}} \cdot \gamma_i^{x_1 \cdot k_{i,1}}, \gamma_i^{x_2}, \gamma_i^{x_2 \cdot k_{i,1}})$$

と書くことができる。ここで、$x_1, x_2$ は $r_{j,1}, r_{j,2}(j = 0, \ldots, i-1)$ から成る式で、その値を $U$ の頂点は知っている。$U$ はヘッダ情報より $(\gamma_i, \gamma_i^{k_{i,1}})$ を得ることができる。従って、$U$ は上記の 4 つ組から $(\alpha_i, e_i \cdot \alpha_i^{k_{i,1}})$ を得ることができる。これは平文が $e_i$ であるエルガマル暗号の暗号文である。従って、$e_i$ をこの 4 つ組から得られればエルガマル暗号を解くことができる。

次に場合 (1-2) を考える。解読者の頂点を $v_a(a < i)$ とする。このとき、$a < j < i$ を満足するある $j$ について $v_j \notin U$ が成立すると仮定する。そうでなければ、$e_i$ は公開情報となる。ここで、最悪の場合、すなわちすべての $v_l(l < a)$ が $U$ に含まれる場合を考える。また、$a = i-2$ および $j = i-1$ とする。その他の場合も同様にして証明される。$v_{i-2}$ において、$e_i$ を含む 4 つ組は

$$(\alpha_i \cdot \gamma_i^{x_1} \cdot \alpha_{i-1}^{k_{i-1,2}} \cdot \gamma_{i-1}^{x_1 \cdot k_{i-1,2}}, e_i \cdot \alpha_i^{k_{i,1}} \cdot \gamma_i^{x_1 \cdot k_{i-1,2}} \cdot \alpha_{i-1}^{k_{i-1,3}} \cdot \gamma_{i-1}^{x_1 \cdot k_{i-1,3}}, \gamma_i^{x_2} \cdot \gamma_{i-1}^{x_2 \cdot k_{i-1,2}}, \gamma_i^{x_2 \cdot k_{i,1}} \cdot \gamma_{i-1}^{x_2 \cdot k_{i-1,3}})$$

と書くことができる。ここで、$x_m(m=1,\ldots,4)$ は $U$ が知っている値である。また、$v_{i-2}$ は $e_{i-1}$ を含む 4 つ組から

$$(\alpha_{i-1}\cdot\gamma_{i-1}^{x_1},\gamma_{i-1}^{x_2})$$

を得ることができる。もしも $v_i$ が $\alpha_i$ と $\gamma_i$ の値を推測することができたと仮定すると、$e_i$ を含む 4 つ組から、$v_{i-2}$ は

$$(\alpha_{i-1}^{k_{i-1,2}}\cdot\gamma_{i-1}^{x_1\cdot k_{i-1,2}},\alpha_{i-1}^{k_{i-1,3}}\cdot\gamma_{i-1}^{x_1\cdot k_{i-1,3}},\gamma_{i-1}^{x_2\cdot k_{i-1,2}},\gamma_{i-1}^{x_2\cdot k_{i-1,3}})$$

を得ることができる。従って、$v_{i-2}$ がこのパケットが $e_i$ を通るか否かを検出するには、$v_{i-2}$ は少なくとも上記 4 つ組のうちの少なくとも 1 つが、既知の $(\alpha_{i-1}\cdot\gamma_{i-1}^{x_1},\gamma_{i-1}^{x_2})$ という値に対して、$\alpha_{i-1}^{k_{i-1,2}}\cdot\gamma_{i-1}^{x_1\cdot k_{i-1,2}}$ (または $\alpha_{i-1}^{k_{i-1,3}}\cdot\gamma_{i-1}^{x_1\cdot k_{i-1,3}},\gamma_{i-1}^{x_2\cdot k_{i-1,2}},\gamma_{i-1}^{x_2\cdot k_{i-1,3}}$) であるか否かを検出できる必要がある。これは、平文が "1" であるエルガマル暗号である。従って、パケットから $e_i$ の値を導出することができればエルガマル暗号系で選択暗号攻撃ができることになる。

次に (2) の解読について考える。解読者を $v_a$ とする。ここで、経路 $\rho(v,w)$ 上で、$v_j\notin U(j<a)$ となる頂点が存在するとする。そうでなければ、$v$ から $v_a$ までの部分経路は $R_{v,U}(H_{v,w})$ に含まれるので公開情報となる。ここで、最悪の場合すなわち、$v_i,v_j$ 以外のすべての頂点が $U$ に含まれているとする。さらに 2 つの場合に分ける。

(2-1) $a<i$

(2-2) $a=i$

まず場合 (2-1) を考える。ここで、$a=i-1$ とする。他の場合も同様の方法で証明される。$v_j\notin U$ であるため、$r_{j,1},r_{j,2}$ は $U$ が知らない値で、他の $r_{m,l}(m\neq j,l=1,2)$ は $U$ が知っている値である。さらに、$\gamma_i$ は原始根であるので、$\alpha_i$ はある $x_0$ に対して $\gamma_i^{x_0}$ と書くことができる。従って $v_{i-1}$ において、$e_i$ を含む 4 つ組は

$$(\gamma_i^{x_0+r_{j,1}\cdot x_1+r_{j,2}\cdot x_2+x_3},e_i\cdot\gamma_i^{(x_0+r_{j,1}\cdot x_1+r_{j,2}\cdot x_2+x_3)\cdot k_{i,1}},\gamma_i^{r_{j,2}\cdot x_4},\gamma_i^{(r_{j,2}\cdot x_4)\cdot k_{i,1}})$$

と書くことができる。ここで、$x_m(m=1,\ldots,4)$ は $r_{m,l}(m\neq j,l=1,2)$ に関する式であり、その値を $U$ が知っているものである。この 4 つ組は 2 つのエルガマル暗号であるとみることができる。前 2 つは平文が $e_i$ で乱数 $k$ が $x_0+r_{j,1}\cdot x_1+r_{j,2}\cdot x_2+x_3$ である場合と見なせる。後 2 つは平文が "1" で乱数 $k$ が $r_{j,2}\cdot x_4$ である場合と見なせる。$r_{j,1}$ および $r_{j,2}$ はパケットによって異なるので、$x_1\equiv x_2\equiv 0(\mathrm{mod}\ \ p-1)$ である場合を除けば、2 つの異なるパケット $P$ および $P'$ が同じ平文 $e_i$ に対する暗号文

であるか否かを検出するのは不可能である。そうでなければ、同じ平文 $M$ と異なった乱数 $k$ の多くのエルガマル暗号文を解くことが可能である。

例外である $x_1 \equiv x_2 \equiv 0(\mathrm{mod}\ \ p-1)$ の場合には、$v_j$ において $r_{j,1}, r_{j,2}$ によって乱数を加えても暗号文は変化しない。その場合には、$x_1 \equiv \prod_{l=0}^{j-1} r_{l,2}$ であり、上記4つ組の後半 $(\kappa_i, g_i)$ はある $X_1, X_2$ に対して

$$(\gamma_i^{(\prod_{m=0}^{j-1} r_{m,2})\cdot X_1}, \gamma_i^{(\prod_{m=0}^{j-1} r_{m,2})\cdot k_{i,1}\cdot X_2})$$

となる。従って、$(\kappa_i, g_i) \equiv (1,1)$ であり、$v_j$ はそのパケットが不適当であることを検出することができる。

次に場合 (2-2) を考える。$a = i$ とする。このとき、$v_a$ は4つ組

$$(\gamma_i^{x_0+r_{j,1}\cdot x_1+r_{j,2}\cdot x_2+x_3}, e_i\cdot\gamma_i^{(x_0+r_{j,1}\cdot x_1+r_{j,2}\cdot x_2+x_3)\cdot k_{i,1}}, \gamma_i^{r_{j,2}\cdot x_4}, \gamma_i^{(r_{j,2}\cdot x_4)\cdot k_{i,1}})$$

から $e_i$ を導出することができる。しかし、$e_i$ は $R_{v_a,U}(P)$ に含まれる。$v_a$ において $e_i$ の含まれる4つ組から2つのパケットが同じ経路を含むことを知ることができるためには、異なる $r_{j,1}$ および $r_{j,2}$ で乱数化されたパケットが同一の $\gamma_i$ もしくは $x_0$(すなわち、$\alpha_i$) を用いていることを検出する必要がある。しかし、任意の原始根 $\gamma'$ に対して、$\gamma_i$ はある自然数 $X$ を用いて $\gamma'^X$ と書くことができる。従って、$e_i$ を含む4つ組は

$$(\gamma'^{(x_0+r_{j,1}\cdot x_1+r_{j,2}\cdot x_2+x_3)\cdot X}, e_i\cdot\gamma'^{(x_0+r_{j,1}\cdot x_1+r_{j,2}\cdot x_2+x_3)\cdot X\cdot k_{i,1}}, \gamma'^{r_{j,2}\cdot x_4\cdot X}, \gamma'^{(r_{j,2}\cdot x_4)\cdot X\cdot k_{i,1}})$$

と書くことができる。これは、もしセンタ $C$ と $v_j$ が他の ($e_i$ を通る) 送り手と受け手の対に対して $(\alpha_i, \gamma_i, r_{j,1}, r_{j,2})$ の値として、任意の $r$ について

$$(\gamma'^{x_0\cdot X+x_3\cdot(X-1)-r\cdot x_1\cdot X}, \gamma', r_{j,1}\cdot X + r, r_{j,2}\cdot X)$$

を用いた場合にも同じ4つ組が現れる。従って、$v_a$ が送り手と受け手の対を求めることは不可能である。

最後に、場合 (3) を考える。ここで、$v$ から $w$ までの経路の $v'$ と $v''$ の間に $v_j \notin U$ となる頂点 $v_j$ が存在すると仮定する。そうでなければ、$v'$ と $v''$ は $v''$ で受け取ったあるパケットが $v'$ で受け取ったパケットと同じかどうかを検出できる、なぜならば $v'$ から $v''$ への部分経路は $R_{v',U}(P(v,w;M))$ に含まれるからである。一般性を失うことなく、2つの共謀解読者を $v_{j-1}$ および $v_{j+1}$ とする。また、$j+1 = i$ とする。その他の場合も同様にして証明される。

$e_i$ を含む4つ組の $v_{j-1}(= v_{i-2})$ での内容は以下のように表すことができる。

$$(\alpha_i\cdot\gamma_i^{x_1}\cdot\alpha_{i-1}^{k_{i-1,2}}\cdot\gamma_{i-1}^{x_1\cdot k_{i-1,2}}, e_i\cdot\alpha_i^{k_{i,1}}\cdot\gamma_i^{x_1\cdot k_{i-1,2}}\cdot\alpha_{i-1}^{k_{i-1,3}}\cdot\gamma_{i-1}^{x_1\cdot k_{i-1,3}}, \gamma_i^{x_2}\cdot\gamma_{i-1}^{x_2\cdot k_{i-1,2}}, \gamma_i^{x_2\cdot k_{i,1}}\cdot\gamma_{i-1}^{x_2\cdot k_{i-1,3}})$$

ここで $x_m(m=1,\ldots,4)$ は $U$ が知っている値である。$v_{j-1}$ はさらに、$e_{i-1}$ を含む4つ組から $(\alpha_{i-1}\cdot\gamma_{i-1}^{x_1},\gamma_{i-1}^{x_2})$ を得ることができる。

$v_j(=v_{i-1})$ において、この4つ組は

$$(\alpha_i\cdot\gamma_i^{x_1+x_2\cdot r_{i-1,1}},e_i\cdot\alpha_i^{k_{i,1}}\cdot\gamma_i^{(x_1+x_2\cdot r_{i-1,1})\cdot k_{i,1}},\gamma_i^{x_2\cdot r_{i-1,2}},\gamma_i^{x_2\cdot r_{i-1,2}\cdot k_{i,1}})$$

のように変換されて $v_{j+1}$ に送られる。まず、$v_{j-1}$ において、4つ組のすべての部分は $v_j$ の鍵を用いて暗号化されている。この4つ組から情報を得ることの不可能性は(1)の場合と同様である。従って、$v_{j-1}$ において、この4つ組から得られる値は $x_1$ と $x_2$ のみである。$v_{j+1}$ において、4つ組の各部分は $r_{i-1,1}$ と $r_{i-1,2}$ によって乱数を加えられる。従って、$v_{j+1}$ で受け取ったパケットが $v_{j-1}$ で受け取ったあるパケットと同じものか否かを検出するのは受け取った4つ組が $x_1,x_2,\alpha_i,\gamma_i$ を使っているか否かを検出することに相当する。(2-2)の場合と同様に、$\alpha_i$ は $\gamma_i^{x_0}$ と書け、$\gamma_i$ は任意の原始根 $\gamma'$ について $\gamma'^X$ と書くことができるので、この4つ組は

$$(\gamma'^{(x_0+x_1+x_2\cdot r_{i-1,1})\cdot X},e_i\cdot\gamma'^{(x_0+x_1+x_2\cdot r_{i-1,1})\cdot X\cdot k_{i,1}},\gamma'^{x_2\cdot r_{i-1,2}\cdot X},\gamma'^{x_2\cdot r_{i-1,2}\cdot X\cdot k_{i,1}})$$

と書くことができる。従って、もしセンタ $C$ と頂点 $v_j$ が別の送り手、受け手の対に関して $(\alpha_i,\gamma_i,r_{j,1},r_{j,2})$ の値として、任意の $r$ について

$$(\gamma'^{x_0\cdot X+x_1\cdot(X-1)-r\cdot x_2\cdot X},\gamma',r_{i-1,1}\cdot X+r,r_{i-2,2}\cdot X)$$

を使用した場合にも同じ4つ組を得ることができる。従って、$v_{j-1}$ におけるあるパケットと $v_{j+1}$ における別のパケットが同じであることを検出することは不可能である。但し、例外として $x_2\equiv 0(\mathrm{mod}\ \ p-1)$ であれば、$r_{i-1,1}$ と $r_{i-1,2}$ によって乱数を加えても $v_{j+1}$ で得られる値に変化がない。しかし、その場合には $v_j$ にはこの4つ組が不適当であることを、4つ組の後半が $(\gamma_i^{x_2},\gamma_i^{x_2\cdot k_{i,1}})\equiv(1,1)$ となることにより検出できる。 □

この方法はいかなる共謀形態においても両方の要求条件を満足する。各手法の比較を表1に示す。

この手続きの計算量は以下の通りである。手続き1において、initで累乗を $2(n+1)$ 回実行する。同様に、sendでは $2(D+1)$ 回の累乗が必要である。また、receiveでは高々2回の累乗が必要である。ここで、$n$ は経路の長さである。Chaum [7] の手法をRSA暗号系 [44] で実現した場合には、送信時に $n$ 回の累乗、受信時に1回の累乗が必要である。従って、もし $D\approx n$ ならば、initは $C$ で各送り手・受け手の対に対して1度しか計算しないので、本暗号系の計算量はChaumの手法の約2倍となる。

手続き2では、initが$2n^2+5n-2$回の累乗計算を必要とし、sendは$2D+1$回の累乗計算を必要とする。receiveは高々$4D$回の累乗計算を行なう。もしも$D \approx n$であれば、送り手頂点に対する計算量はChaumの手法の約2倍となる。受信時の計算量は送り手の計算量の約2倍となる。

## 2.4 むすび

本章では、ネットワークにおける経路情報の安全性について考察し、2種類の経路情報の暗号化法を示した。

第一の方法は共謀がない場合に用いられ、計算量は既存の方法と比べて定数倍程度となる。第二の方法は既存の方法と比べると計算量は大きいが、既存の手法では対処不可能であった任意の共謀形態に関して安全である。

経路情報の暗号化に関しては、以下の問題が今後の課題として考えられる。

- すべての経路を1箇所で求めるのは信頼性、安全性、および効率の面で問題となる場合もある。センタ$C$をおかない場合の手法、すなわち、$G$の頂点が協力してヘッダ情報$H_{v,w}$を生成する際に、いくつかの頂点が共謀して解読しようとしても不可能な方法も必要である。

- 本稿ではルーティングとしてソースルーティングを仮定したが、トラフィックの変動に対応して経路を動的に変化させるなどの目的でホップ-バイ-ホップルーティングを行なう場合もある。その場合に有効な手法、すなわち、経路情報が$F_v$にのみある場合の暗号化法も必要である。

# 3　グラフとその上のルーティングの組に対する耐故障性と効率

本章では、グラフ $G$ とその上のルーティング $\rho$ の組に対する耐故障性と効率の問題について議論する。

中継処理量 (forwarding index) $\xi(G,\rho)$ は、各頂点を経由する経路の最大値であり、ネットワークにおけるルーティングの効率をはかる一つの指標である。SR- グラフ (surviving route graph) の直径 $D(R(G,\rho)/F)$ は故障 $F$ が起きた場合に通信を行なうときに必要な経路の数の最大数であり、ネットワークにおけるルーティングの信頼性をはかる一つの指標である。$\xi(G,\rho)$ および $D(R(G,\rho)/F)$ を小さくするルーティングを求めることが重要である。

本章では有向グラフに対して上記の問題を考察し、以下の結果を示す。

1. $k$- 連結な有向グラフ $G(k \neq 2,4)$ が $|F| < k$ を満たす任意の $F$ に対して $D(R(G,\rho)/F) \leq 6$ を満足するルーティング $\rho$ を持つための十分条件。

2. $gcd(n,d) = 1, d \geq 2$ を満たす任意の $n$ と $d$ に対して、$\xi(G,\rho_1) < n\log_d n$ を満足する、頂点数 $n$、最大次数 $d$ の有向グラフ $G$ とその上でのルーティング $\rho_1$ の組を構成する方法。

3. $d \geq 2$ を満たす任意の $n$ と $d$ に対して、$\xi(G,\rho_2) < 2n\lceil\log_d n\rceil$ を満足する、頂点数 $n$、最大次数 $d$ の有向グラフ $G$ とその上でのルーティング $\rho_2$ の組を構成する方法。

4. $d \geq 3$、$n > d^4$ および $gcd(n,d) = 1$ を満たす任意の $n$ と $d$ に対して $\xi(G,\rho_1') < 2n\log_d n$ を満足し、かつ、$|F| < d-1$ を満たす任意の $F$ に対して $D(R(G,\rho_1')/F) \leq 3$ を満足する、頂点数 $n$、最大次数 $d$ の有向グラフ $G$ とその上のルーティング $\rho_1'$ の組を構成する方法。

5. $d \geq 3$ および $n > d^4$ を満たす任意の $n$ と $d$ に対して、$\xi(G,\rho_2') < 3n\log_d n$ を満足し、かつ、$|F| < d-1$ を満たす任意の $F$ に対して $D(R(G,\rho_2')/F) \leq 3$ を満足する、頂点数 $n$、最大次数 $d$ の有向グラフ $G$ とその上のルーティング $\rho_2'$ の組を構成する方法。

まず、3.1節で、本章で用いる用語の定義を行なう。3.2節で、$k$- 連結有向グラフに対し、SR- グラフの直径が定数となるルーティングが存在する十分条件を示す。3.3節で、中継処理量が準最適な値となる有向グラフとその上のルーティングの組を示す。3.4節で、SR- グラフの直径と中継処理量がともに準最適な値となる有向グラフとその上のルーティングの組を示す。

## 3.1 諸定義

### 3.1.1 用語の定義

本節では本章で用いられる表記の定義を行なう。

$gcd(n,d)$ を $n$ と $d$ の最大公約数とする。ある関数 $f(n)$ が $o(1)$ であるとは、$\lim_{n-\infty} f(n) = 0$ となることをいう。有向グラフ $G = (V, E)$ は頂点集合 $V$ と有向辺の集合 $E$ の組である。頂点 $v$ の入次数および出次数は $v$ に入る、または $v$ から出る有向辺の数である。$G$ の最大次数 $\Delta(G)$ は $v \in V(G)$ の中での入次数および出次数の最大値である。

ある頂点集合 $U \subset V(G)$ に対し、$U$ の誘導部分グラフは頂点集合が $U$ となる最大の $G$ の部分グラフである。

$(u, v) \in E$ であるとき、$u$ は $v$ の先行点、$v$ は $u$ の後続点と呼ぶ。$G$ における頂点 $v_0$ から $v_k$ への遊歩道とは、頂点と辺の交互系列 $v_0, e_0, v_1, \ldots, v_{i-1}, e_i, v_i, \ldots, e_k, v_k$ で、$e_i = (v_{i-1}, v_i) \in E$ を満足するものである。含まれる頂点がすべて異なる遊歩道を道と呼ぶ。道の長さとは、その道に含まれる辺の数である。頂点 $u$ から $v$ までの距離とは、$u$ から $v$ への最短の道の長さである。$G$ の直径とは任意の 2 頂点間の距離の最大値であり、$D(G)$ と書く。

有向グラフ $G$ が強連結であるとは、任意の 2 頂点間に道が存在することをいう。$G$ が $k$- 連結であるとは、任意の $k-1$ 個の頂点を取り除いても強連結であることをいう。$G$ の連結度 $\kappa(G)$ を、$G$ から取り除いたときにトリビアルもしくは連結でなくなる頂点の数の最小値と定義する。$\Delta(G) = d$ であるグラフ $G$ に対し、$\kappa(G) \leq d$ が成立する。

頂点 $v \in V$ および頂点集合 $U \subset V$-$\{v\}$ に対し、$v$-$U$ ファンとは $v$ から $U$ のすべての頂点に対する $|U|$ 個の、頂点を共有しない道の集合をいう。同様に、$U$-$v$ ファンとは $U$ のすべての頂点から $v$ までの $|U|$ 個の頂点を共有しない道の集合である。$k$-連結な有向グラフに対して、以下の性質が成立する。

**性質 1** [5]

$G = (V, E)$ を $k$- 連結な有向グラフとする。$U$ を、$|U| \leq k$ を満足する任意の $V$ の部分集合、$v$ を $V - U$ の任意の頂点とする。

このとき、$v$-$U$ ファンと $U$-$v$ ファンが存在する。 □

その他のグラフに関する表記については、文献 [28] を参照のこと。

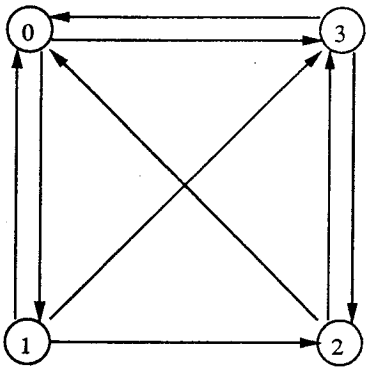


図 2: 有向グラフ $G$ の例

### 3.1.2 中継処理量と SR- グラフの定義

有向グラフ $G = (V, E)$ に対し、ルーティング $\rho$ は相異なる頂点対 $(x, y) \in V \times V$ に対して $x$ から $y$ への道を与える関数である。この道を $\rho$ による経路と呼ぶ。

頂点 $v \in V(G)$ とルーティング $\rho$ に対して、$\xi_v(G, \rho)$ を、$v$ を通る $\rho$ の経路の数と定義する。中継処理量 $\xi(G, \rho)$ は $\xi(G, \rho) = \max_{v \in V(G)} \xi_v(G, \rho)$ として定義される。図 2に示す有向グラフ $G$ において、$\rho$ が以下のように定義されていたとする。

$$\rho(u,v) = \begin{cases} u, (u,v), v & \text{if} (u,v) \in E \\ 0, (0,1), 1, (1,2), 2 & u = 0, v = 2 \\ 2, (2,3), 3, (3,0), 0, (0,1), 1 & u = 2, v = 1 \\ 3, (3,0), 0, (0,1), 1 & u = 3, v = 1 \end{cases} \quad (1)$$

このルーティング $\rho$ に対して、$\xi_0(G, \rho) = 2$、$\xi_1(G, \rho) = 1$、$\xi_2(G, \rho) = 0$、$\xi_3(G, \rho) = 1$ が成立するので、$\xi(G, \rho) = 2$ となる。

グラフ $G$ に対して実現可能な最小の中継処理量を $\xi(G) = \min_\rho \xi(G, \rho)$ と定義する。有向グラフ $G$ に対する $\xi(G)$ の下界は無向グラフに対する下界 [9] と同様にして以下の式で与えられる。

**性質 2** $|V(G)| = n$ および $\Delta(G) = d (d \geq 2)$ を満足する任意の有向グラフ $G(n, d)$ に対し、$\xi(G(n, d)) \geq (1 + o(1)) n \log_d n$ が成立する。 □

**(証明)**

$G$ を、$\Delta(G) = d$ を満足する任意の有向グラフとする。$v$ を任意の頂点とする。$v$ から距離 $i$ の頂点は高々 $d^i$ 個である。従って、$v$ から距離 $i$ 以下の頂点の数 $n_i$ は $n_i \leq d + d^2 + \ldots + d^i = d \cdot (d^i - 1)/(d - 1)$ となる。よって、$v$ からの距離が $i$ より大き

い頂点の数 $n_i'$ は $n_i' \geq n-1-d\cdot(d^i-1)/(d-1)$ となる。よって、$n_i' \geq 0$ となるのは、$n-1 \geq d\cdot(d^i-1)/(d-1)$、すなわち、$1 \leq i \leq \log_d n-1$ の時である。

ここで、$v$ から $n_i'$ 個の経路は途中 $i$ 個の頂点を経由することになる。従って、$v$ からの経路が他の頂点を経由する数の総和 $\zeta_v$ は

$$\zeta_v \geq \sum_{i=1}^{\log_d n-1} (n-1-d\cdot(d^i-1)/(d-1)) = \log_d n\cdot(n-1/(d-1))+(n-d^2-1)/(d-1)$$

$$= (1+o(1))n\log_d n$$

である。これをすべての $v$ について和を取ると他の頂点を経由する回数の総数 $\zeta$ が求められる。

$$\zeta \geq (1+o(1))n^2\log_d n$$

従って、中継処理量の最小値は

$$\xi(G(n,d)) \geq \zeta/n \geq (1+o(1))n\log_d n$$

となる。この下界値を満足するのは直径が $O(\log_d n)$ である場合となることは明らかである。 □

次に、SR- グラフの定義を行なう。$F \subset V \cup E$ を故障集合とする。故障集合は頂点故障 $F_v$ と辺故障 $F_e$ から成る。

SR- グラフ $R(G,\rho)/F = (V',E')$ は以下のように定義される有向グラフである。

$V' = V - F_v$, $E' = \{(x,y) \mid V(\rho(x,y)) \cap F_v = \phi \text{ and } E(\rho(x,y)) \cap F_e = \phi\}$

ここで、$V(\rho(x,y))$ および $E(\rho(x,y))$ は $\rho(x,y)$ に含まれる頂点と辺の集合である。

SR- グラフの直径を $D(R(G,\rho)/F)$ と書く。図2のグラフ $G$ および式 (1) のルーティング $\rho$ および $F = \{3,(1,0)\}$ に対する SR- グラフ $R(G,\rho)/F$ を図3に示す。この直径 $D(R(G,\rho)/F) = 2$ である。SR- グラフの直径の下界については以下の性質が成立する。

**性質 3** 任意の有向グラフ $G$ およびその上のルーティング $\rho$ に対して、

$$\max_{|F|<\kappa(G)} D(R(G,\rho)/F) \geq 2$$

が成立する。 □

もし $|F| \geq \kappa(G)$ であれば、$G/F$ が非連結になることもあり、$\max D(R(G,\rho)/F)$ は無限大となる。従って、SR- グラフの直径が最適になるグラフは連結度最大のグラフである。

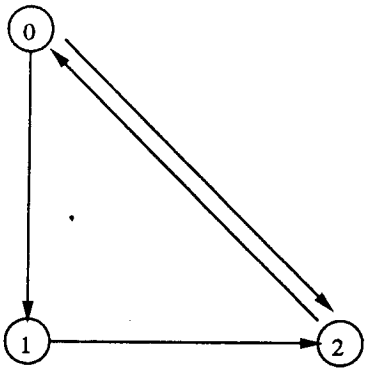


図 3: SR- グラフの例

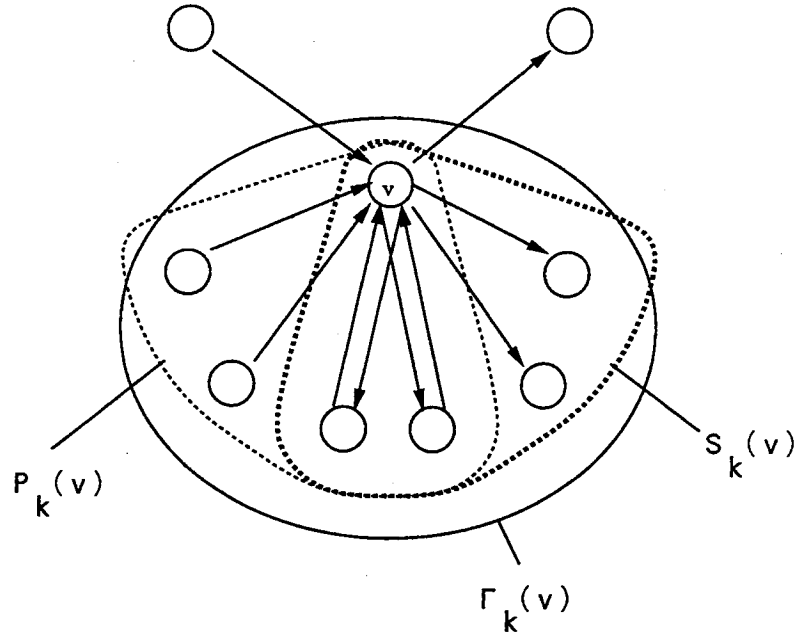


図 4: $P_k(v)$、$S_k(v)$ および $\Gamma_k(v)(k=5)$

### 3.2 SR- グラフの直径が定数となる有向グラフのルーティング

本節では $k$- 連結 $(k \neq 2,4)$ な有向グラフが、SR- グラフの直径が小さい定数であるようなルーティング $\rho$ を持つための十分条件を示す。

まず、条件 $DC_k(m)$ を示す。この条件は無向グラフについての文献 [32][42] に示されているものと同種の性質である。$v \in V$ に対して、$P_k(v)$ $(S_k(v))$ を $v$ と $v$ の $k-1$ 個の先行点 (後続点) から成る頂点集合とする。$P_k(v)$ と $S_k(v)$ の例を図 4に示す。$\Gamma_k(v)$ を $P_k(v) \cup S_k(v)$ と定義する。

**定義 7** 条件 $DC_k(m)$

$G=(V,E)$ が以下の条件を満足する $m$ 個の頂点 $v_0, v_1, \ldots, v_{m-1}$ を持つ時に $DC_k(m)$

を満足するという。

各頂点 $v_i$ に対して $P_k(v_i)$ および $S_k(v_i)(i=0,\ldots,m-1)$ が存在して、

$$\Gamma_k(v_i)\cap\Gamma_k(v_j)=\phi(\forall i\neq j)$$

を満足する。 □

**定理 3** $k$- 連結 $(k\neq 2,4)$ 有向グラフ $G$ が $DC_k(k)$ を満足すれば、$G$ 上のルーティング $\rho$ で $\max_{|F|<k} D(R(G,\rho)/F)\leq 6$ を満足するものが存在する。 □

文献 [42] と同様な議論により、$DC_k(k)$ を満足するあるグラフのクラスが以下のように示される。

**性質 4** $0<\epsilon<4^{-1/3}$ を満足する任意の $\epsilon$ に対し、ある $n_0>0$ が存在して、頂点数 $n\geq n_0$ および最大次数 $\Delta(G)\leq\epsilon\cdot n^{1/3}$ を満足するすべての $k$- 連結有向グラフは条件 $DC_k(k)$ を満足する。 □

(証明)

$k$- 連結有向グラフ $G$ の頂点数を $n$、最大次数を $d$ とする。まず、$G$ が $DC_k(\lceil n/(4d^2+1)\rceil)$ を満たすことを示す。条件 $DC_k(\lceil n/(4d^2+1)\rceil)$ を満足する頂点集合 $M$ は以下のアルゴリズムで得ることができる。初期状態で $M$ は空とし、候補頂点集合 $C$ を $V$ にする。任意の頂点 $x$ を $C$ から選んで $M$ に入れ、$G$ の辺を無向辺でおきかえたグラフで $x$ からの距離が 2 以下の頂点 ($x$ を含む) を $C$ から取り除く。これを $C$ が空になるまで繰り返す。各ステップにおいて、$C$ の頂点は高々 $1+2d+2d(2d-1)=4d^2+1$ 個減少する。従って、$M$ の要素数は少なくとも $|M|\geq\lceil n/(4d^2+1)\rceil$ となる。$M$ の頂点を $DC_k(\lceil n/(4d^2+1)\rceil)$ の条件を満足する頂点とすることができるのは明らかである。

ここで、$0<\epsilon<4^{-1/3}$ を満足するある $\epsilon$ が与えられたとする。$\epsilon'=1/\epsilon^3-4>0$ とする。$d_0$ を、$\epsilon'\cdot d^3\geq d$ を満足する最小の $d$ とし、$n_0$ を $n_0=4d_0^3+d_0$ とおく。$G$ を、頂点数 $n\geq n_0$ および最大次数 $d\leq\epsilon n^{1/3}$ を満足する有向グラフとする。$G$ は $DC_k(\lceil n/(4d^2+1)\rceil)$ を満足するので、$\lceil n/(4d^2+1)\rceil\geq k$ を示せばよい。$k\leq d$ であるので、$d\leq n/(4d^2+1)$、すなわち $n\geq 4d^3+d$ を示せばよい。$n_0$ の選び方から、$d\leq d_0$ ならば $n\geq 4d^3+d$ は成立する。$d>d_0$ の時、$d_0$ の選び方より、$d$ は $d\leq\epsilon'\cdot d^3$ を満足する。すなわち、$4d^3+d\leq(1/\epsilon^3)d^3$ が成立する。$(1/\epsilon^3)d^3\leq n$ より、$n\geq 4d^3+d$ が成立する。 □

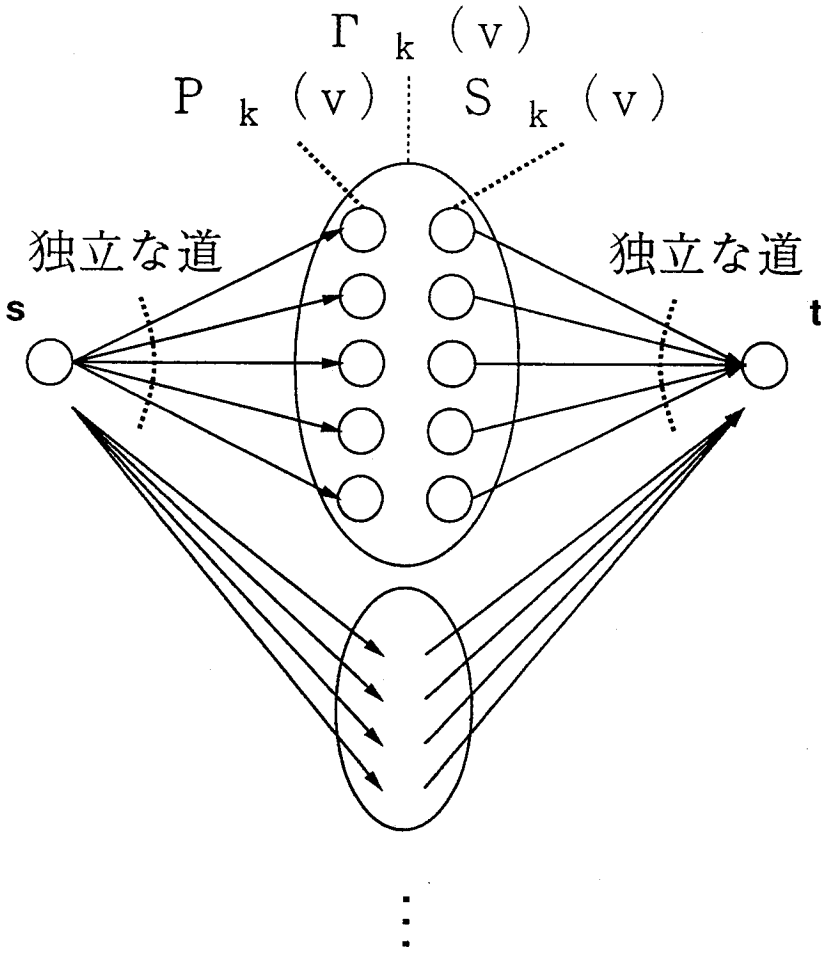


図 5: 経路の選び方

上記定理のルーティング $\rho$ は以下のように定義する。$v_i(i = 0, 1, \ldots, k-1)$ を、条件 $DC_k(k)$ を満足する頂点とする。$\Gamma = \cup_{0 \le i < k} \Gamma_k(v_i)$ とする。$G(\Gamma_k(v))$ を、$\Gamma_k(v)$ によって誘導される部分グラフとする。

$|P_k(v_i)| = k$ ($|S_k(v_i)| = k$) と性質 1 より、$P_k(v_i)$ ($S_k(v_i)$) および $u \notin P_k(v_i)$ ($u \notin S_k(v_i)$)$(i = 0, 1, \ldots, k-1)$ に対して、$u - P_k(v_i)$ ファン ($S_k(v_i) - u$ ファン) が存在する (図 5)。

$u - P_k(v_i)$ ファンに含まれる $u \notin P_k(v_i)$ から $v \in P_k(v_i)$ への道を $\psi_{fan}(u, v; P_k(v_i))$ と書く。$S_k(v_i) - v$ ファンに含まれる $u \in S_k(v_i)$ から $v \notin S_k(v_i)$ への道を $\psi_{fan}(u; S_k(v_i), v)$ と書く。任意の $u \in P_k(v_i)$ と $v \in S_k(v_i)$ に対して、$G(\Gamma_k(v_i))$ において $u$ から $v$ への道が存在する。この道を $\psi_c(u, v; \Gamma_k(v_i))$ と書く。

ここで、$u_i$ および $u_j$ $(i \neq j)$ をそれぞれ $S_k(v_i)$ および $P_k(v_j)$ の頂点とする。$u_i - P_k(v_j)$ ファンの道を $u_i$ から $P_k(v_j)$ への経路として選ぶと、$S_k(v_i) - u_j$ ファンの道を $S_k(v_i)$ から $u_j$ への道として選ぶことができないかもしれない。それは、$u_i - P_k(v_j)$

ファンに含まれる $u_i$ から $u_j$ への道は $S_k(v_i) - u_j$ ファンにものとは異なるかもしれないからである。$u_i - P_k(v_j)$ ファンと $S_k(v_i) - u_j$ ファンのどちらを経路として選ぶかを決めるため、有向グラフ $T(k) = (V_T, E_T)$ を考える。ここで、$T(k)$ の頂点 $i \in V_T (i = 0, 1, \ldots, k-1)$ は各 $\Gamma_k(v_i)$ に対応し、$(i_1, i_2) \in E_T$ であれば、$(i_2, i_1) \notin E_T$ を満足するとする。

$u$ と $v$ を $V$ の任意の相異なる頂点とする。グラフ $T(k)$ が与えられた時、ルーティング $\rho$ を以下のように定める。$T(k)$ の求め方については後述する。

$$\rho(u,v) = \begin{cases} \psi_{fan}(u, v; P_k(v_i)) & \text{if } u \in V - \Gamma \text{ and } v \in P_k(v_i) \quad (2) \\ \psi_{fan}(u; S_k(v_i), v) & \text{if } u \in S_k(v_i) \text{ and } v \in V - \Gamma \quad (3) \\ \psi_{fan}(u, v; P_k(v_j)) & \text{if } u \in \Gamma_k(v_i) \text{ and } v \in P_k(v_j) \text{ such that } (i,j) \in E_T \quad (4) \\ \psi_{fan}(u; S_k(v_j), v) & \text{if } u \in S_k(v_j) \text{ and } v \in \Gamma_k(v_i) \text{ such that } (i,j) \in E_T \quad (5) \\ \psi_{fan}(u; S_k(v_i), v) & \text{if } u \in S_k(v_i) \text{ and } v \in \Gamma_k(v_i) - S_k(v_i) \quad (6) \\ \psi_c(u, v; \Gamma_k(v_i)) & \text{if } u \in P_k(v_i) \text{ and } v \in S_k(v_i) \quad (7) \\ \text{don't care} & \text{otherwise} \end{cases}$$

$\rho$ の定義では任意の頂点対間に経路が高々 1 つしか定義されないことは明らかである。$\rho$ は以下の性質を持つ。

**補題 1** $|F| < k$ を満足する任意の $F$ に対して、$D(R(G,\rho)/F) \leq 2D(T(k)) + 2$ が成立する。 □

**(証明方針)** $|F| < k$ かつ $\Gamma_k(v_i)$ の数が $k$ であるので、$G(\Gamma_k(v_I))$ に $F$ の要素が存在しない $\Gamma_k(v_I)$ が少なくとも 1 つは存在する。$u$ と $v$ を $V - F$ に含まれる任意の相異なる頂点とする。$R(G,\rho)/F$ における $u$ から $v$ への距離を $dis_R(u,v)$ と書くことにする。

$dis_R(u,v) \leq 2D(T(k)) + 2$ を証明するため、以下の補題を証明すればよい。

**補題 2** 任意の $u \in P_k(v_I)$ および $v \in S_k(v_I)$ に対し、$dis_R(u,v) \leq 1$ である。 □

**補題 3** 任意の $u \in V - \Gamma$ に対し、$dis_R(u,v) = 1$ を満足する頂点 $v \in P_k(v_I)$ が存在する。 □

**補題 4** 任意の頂点 $u \in \Gamma_k(v_i)$ $(i \neq I)$ に対し、$dis_R(u,v) \leq D(T(k))$ を満足する $v \in P_k(v_I)$ が存在する。 □

**補題 5** 任意の頂点 $u \in S_k(v_I)$ に対し、$dis_R(u,v) \leq D(T(k)) + 1$ を満足する $v \in P_k(v_I)$ が存在する。 □

**補題 6** 任意の頂点 $v \in V - \Gamma$ に対し、$dis_R(u,v) = 1$ を満足する $u \in S_k(v_I)$ が存在する。 □

**補題 7** 任意の頂点 $v \in \Gamma_k(v_i)$ $(i \neq I)$ に対し、$dis_R(u,v) \leq D(T(k))$ を満足する $u \in S_k(v_I)$ が存在する。 □

**補題 8** 任意の頂点 $v \in P_k(v_I)$ に対し、$dis_R(u,v) \leq 1$ を満足する $u \in S_k(v_I)$ が存在する。 □

(**補題 1の証明**)

以上の補題より、任意の $u$、$v$ の対に対して、$dis_R(u,v) \leq 2D(T(k))+2$ であることを以下のようにして証明できる。

(Case 1) $u \in P_k(v_I)$ かつ $v \in S_k(v_I)$ の場合。この場合は補題 2より明らかである。

(Case 2) $u \notin P_k(v_I)$ かつ $v \in S_k(v_I)$ の場合。補題 3または 4または 5より、$dis_R(u,u') \leq D(T(k))+1$ を満足する $u' \in P_k(v_I)$ が存在する。補題 2より、$dis_R(u', v) \leq 1$ である。従って、$dis_R(u,v) \leq D(T(k))+2$ となる。

(Case 3) $u \in P_k(v_I)$ かつ $v \notin S_k(v_I)$ の場合。補題 6または 7または 8より、$dis_R(v',v) \leq D(T(k))$ を満足する $v' \in S_k(v_I)$ が存在する。補題 2より、$dis_R(u,v') \leq 1$ である。従って、$dis_R(u,v) \leq D(T(k))+1$ が成立する。

(Case 4) $u \notin P_k(v_I)$ かつ $v \notin S_k(v_I)$ の場合。補題 3または 4または 5より、$dis_R(u,u') \leq D(T(k))+1$ を満足する $u' \in P_k(v_I)$ が存在する。補題 6または 7または 8より、$dis_R(v',v) \leq D(T(k))$ を満足する $v' \in S_k(v_I)$ が存在する。さらに、補題 2より、$dis_R(u',v') \leq 1$ が成立する。従って、$dis_R(u,v) \leq 2D(T(k))+2$ が成り立つ。

上記より、任意の $u$ と $v$ の組に対して、$dis_R(u,v) \leq 2D(T(k))+2$ が成立する。

□

次に、補題 2-8を証明する。

(**補題 2の証明**) $G(\Gamma_k(v_I))$ が故障を含んでいないため、式 (7) より明らか。

(**補題 3の証明**) 式 (2) より、$u$ から $P_k(v_I)$ への、$k$ 個の独立な道が存在する。$|F| < k$ であるので、少なくとも 1 つは無故障である。従って、$dis_R(u,v) = 1$ を満足する $v \in P_k(v_I)$ が存在する。

(**補題 4の証明**) $i_0(=i), i_1, i_2, \ldots, i_{d-1}, i_d(=I)$ を $T(k)$ における $i$ から $I$ への長さ $d \leq D(T(k))$ の道とする。式 (4) より、$u$ から $P_k(v_{i_1})$ までの $k$ 個の道のうち少なくとも 1 つは無故障である。$u_{i_1}$ を、この道の $P_k(v_{i_1})$ における端点とすると、$dis_R(u,u_{i_1}) = 1$ が成立する。これを繰り返すことにより、$j = 0,1,\ldots,d-1$ に

対して $dis_R(u_{i_j}, u_{i_{j+1}}) = 1$ を満足する頂点 $u_{i_{j+1}} \in P_k(v_{i_{j+1}})$ が存在する。従って、$dis_R(u, v) \leq d \leq D(T(k))$ を満足する $v = u_{i_d} \in P_k(v_I)$ が存在する。

(**補題 5の証明**) $T(k)$ において、$I$ から $I$ への、長さ $D(T(k)) + 1$ 以下の遊歩道が存在する。なぜならば、$(I, i) \in E_T$ を満足する頂点 $i \in V_T$ と、$i$ から $I$ への長さ $D(T(k))$ 以下の道が存在するからである。あとは補題 4の証明と同様にして、$dis_R(u, v) \leq D(T(k)) + 1$ を満足する頂点 $v \in P_k(v_I)$ が存在することが証明される。

(**補題 6の証明**) 補題 3の証明で、式 (2) の代わりに式 (3) を用いればよい。

(**補題 7の証明**) $T(k)$ において、$I$ から $i$ への、長さ $D(T(k))$ 以下の道が存在する。あとは補題 4の証明と同様である。

(**補題 8の証明**) $v \in S_k(v_I) \cap P_k(v_I)$ である場合は、$u = v$ とすれば、$dis_R(u, v) = 0$ となる。$v \in P_k(v_I) - S_k(v_I)$ である場合は、式 (6) より、$dis_R(u, v) = 1$ を満足する頂点 $u \in S_k(v_I)$ が存在する。 □

文献 [30] には、以下の有向グラフ $T_1(k)$ および $T_2(k)$ の直径が 2 以下であることが示されている。$m$ を $\lfloor k/2 \rfloor$ とする。

$k$ が奇数の場合には $T_1(k) = (V_{T_1}, E_{T_1})$ は以下のように定義される。

$$V_{T_1} = \{0, 1, \ldots, 2m\}, E_{T_1} = \bigcup_{\alpha=1}^{m} E_{T_1,\alpha}$$

$$E_{T_1,\alpha} = \{\, (i, j) \mid j \equiv i + \alpha \pmod{2m+1} \,\}$$

$k$ が偶数で $k \neq 2, 4$ の場合には、$T_2(k) = (V_{T_2}, E_{T_2})$ は以下のように定義される。

$$V_{T_2} = \{0, 1, \ldots, 2m-1\}, E_{T_2} = \bigcup_{\alpha=1}^{m} E_{T_2,\alpha}$$

$$E_{T_2,\alpha} = \begin{cases} \{(i,j) | j \equiv i + \alpha \pmod{2m}\} & \text{if } \alpha = 1, 3, 4, \ldots, m-1 \\ \{(i,j) | i \text{ is odd and } j \equiv i + 2 \pmod{2m}\} & \\ \quad \cup \{(i,j) | i \text{ is even and } j \equiv i - 2 \pmod{2m}\} & \text{if } \alpha = 2 \\ \{(i,j) \mid 0 \leq i < m \text{ and } j \equiv i + m \pmod{2m}\} & \text{if } \alpha = m \end{cases}$$

$D(T_i(k)) \leq 2 (i = 1, 2)$ が成立することと性質 1より、定理 3が証明される。

### 3.3 中継処理量が準最適な有向グラフとその上のルーティングの組

本節では中継処理量が準最適なルーティング $\rho$ を持つ有向グラフ $G$ の構成法を与える。3.1.2節で述べたように、中継処理量が最適なルーティングを持つグラフは直径が $O(\log_d n)$ でなければならない。一般化 de Bruijn グラフ [29] $G_B(n, d)$ は $O(\log_d n)$ の直径を持つ。$G_B(n, d) = (V, E)$ は以下のように定義されている。

$$V = \{0, 1, \ldots, n-1\}, E = \{(u, v) | v \equiv d \cdot u + p \pmod{n},\ p = 0, 1, \ldots, d-1\}$$

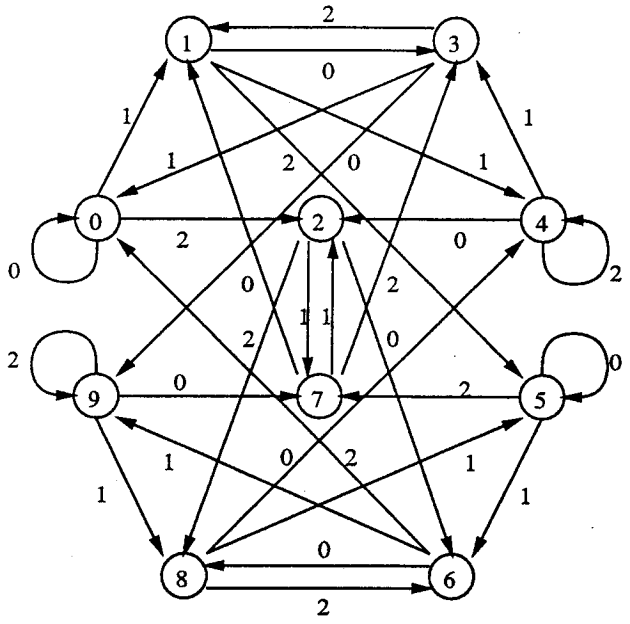


図 6: $G_B(10,3)$ (辺 $(u,v)$ のラベルは $v \equiv 3u+p \pmod{10}$ の $p$ の値)

$G_B(10,3)$ を図 6に示す。

$\Delta(G_B(n,d)) = d$、$\kappa(G_B(n,d)) = d-1$、$D(G_B(n,d)) = \lceil \log_d n \rceil$ である [31]。$D(G_B(n,d))$ を $D$ と書く。このとき、$d^{D-1} < n \leq d^D$ である。

準備として、$G_B(n,d)$ の遊歩道の表記法を定義する。

頂点 $u$ から $v$ への長さ $m$ の遊歩道が存在するのは、

$$v \equiv u \cdot d^m + p_1 \cdot d^{m-1} + p_2 \cdot d^{m-2} + \cdots + p_{m-1} \cdot d + p_m \pmod{n}$$

を満足する整数 $p_1, p_2, \ldots, p_m (0 \leq p_i < d (i = 1,2,\ldots,m))$ が存在する時、かつその時に限られる。この遊歩道を $u$ と $m$ けたの $d$ 進数 $(p_1, p_2, \ldots, p_m)$ とで表現することにする。例えば、

$$1 \equiv 0 \cdot 3^3 + 1 \cdot 3^2 + 0 \cdot 3 + 2 \pmod{10}$$

であるので、$G_B(10,3)$ には長さ 3 の 0 から 1 への遊歩道 $0,(0,1),1,(1,3),3,(3,1),1$ が存在する。この遊歩道は 0 と $(1,0,2)$ で表現する。従って、もし、

$$v \equiv u \cdot d^m + x \pmod{n} \tag{8}$$

を満足する整数 $x (0 \leq x < d^m)$ が存在すれば、$u$ から $v$ への長さ $m$ の遊歩道が存在し、それは $u$ および、$x$ を表す $m$ けたの $d$ 進数で表現することができる。この方法では一般に道ではなく遊歩道が得られる。しかし、得られた遊歩道が同じ頂点を

2度以上通る場合に、そこを短絡すれば道が得られるので、以降は経路を遊歩道として定義する。

直径の定義より、$m < D$ のとき、$u$ から $v$ への長さ $m$ の遊歩道は存在しないこともある。与えられた $u$,$v$ の対に対し、式 (8) を満足する最小の非負整数 $x$ を $P_m(u,v)$ とする。$P_m(u,v) < d^m$ であるとき、$P_m(u,v)$ によって与えられる $u$ から $v$ への遊歩道を $W_m(u,v)$ と書き、

$$W_m(u) = \{W_m(u,v) | v \text{ は } P_m(u,v) < d^m \text{ を満足する}\}$$

とする。

この方法では、同一の $u$ と $v$ に対して長さ $D$ の遊歩道が複数得られる場合もある。なぜならば、ある $j > 0$ に対し、$P_D(u,v) + jn < d^D$ となることがあるからである。$W_{D,j}(u,v)$ を、$P_D(u,v) + jn$ によって得られた遊歩道とする。また、

$$W_{D,j}(u) = \{W_{D,j}(u,v) | v \text{ は } P_D(u,v) + jn < d^D \text{ を満足する}\}$$

とする。ここで、$W_{D,0}(u,v) = W_D(u,v)$ とする。$G_B(n,d)$ 上のルーティング $\rho_1$ を以上の記法を用いて以下のように定義する。

**定義 8** $\rho_1(u,v) = W_{D,0}(u,v)$ □

$gcd(n,d) = 1$ のとき、$\rho_1$ は $n \to \infty$ のときに最適な中継処理量を達成する。以下の定理でそれを示す。

**定理 4** $gcd(n,d) = 1$ のとき、$\xi(G_B(n,d), \rho_1) < n\log_d n$ である。 □

この定理を証明するため、下記の性質を用いる。

**性質 5** [48]

$$p \cdot x \equiv q \pmod{n}$$

の解 $x$ の個数は、

$$\begin{cases} gcd(p,n) & q \text{ が } gcd(p,n) \text{ の倍数である} \\ 0 & \text{otherwise} \end{cases}$$

□

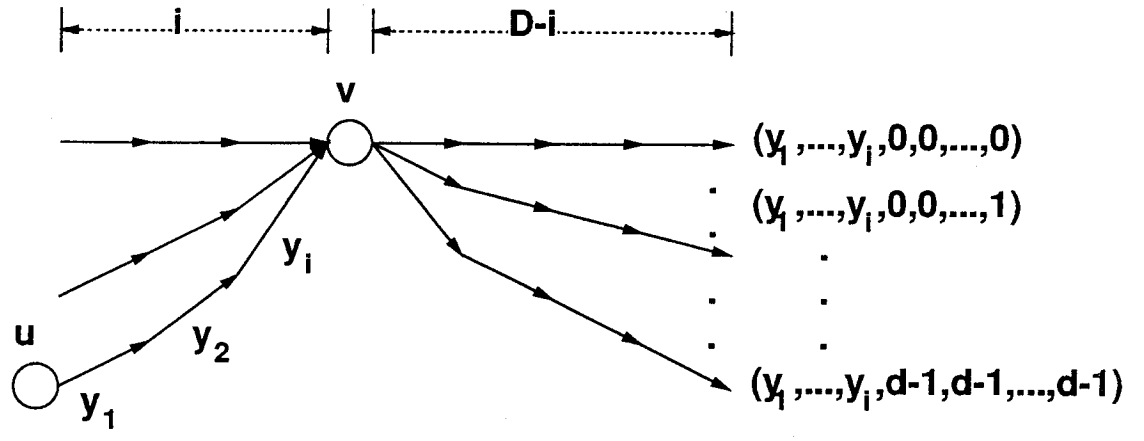


図 7: $v$ を通る長さ $D$ の遊歩道

(定理 4の証明)

$n$ の $D$ けた $d$ 進数による表現を $(r_1, r_2, \ldots, r_D)$ とする、すなわち

$$n = \sum_{j=1}^{D} r_j \cdot d^{D-j}$$

とする。任意の頂点 $v$ に対し、$v$ を通る経路の数を数える。$v$ と異なる任意の頂点 $u$ に対して、$\rho_1$ によって定義される $u$ から $V$ の任意の頂点への経路は $n$ 未満の $D$ けた $d$ 進数で表現される、すなわち、

$$(0,0,\ldots,0),(0,0,\ldots,1),\ldots,(r_1,r_2,\ldots,r_D-1)$$

である。ここで、以下の式を考える。

$$v \equiv d^i \cdot u + y \pmod{n} \tag{9}$$

$u$ と $v$ がある $y(0 \leq y < d^i)$ に対して式 (9) を満足するとき、$y$ の $i$ けた $d$ 進数表現を $(y_1, y_2, \ldots, y_i)$ とすると、$u$ から $v$ への、$(y_1, y_2, \ldots, y_i)$ で表現される遊歩道が存在し、かつ、$u$ から $V$ のある頂点へ $v$ を経由する、

$$(y_1, y_2, \ldots, y_i, q_1, q_2, \ldots, q_{D-i}) \ (0 \leq q_j < d)$$

で表現される $d^{D-i}$ 個の長さ $D$ の遊歩道が存在する (図 7)。この $d$ 進数が $n$ より小さければ、この遊歩道は $\rho_1$ の経路である。

従って、$\rho_1$ で定義される $u$ から $V$ への経路のうち、$i$ 番目に通る頂点が $v$ になるものの個数は、

$$\begin{cases} d^{D-i} & \text{if } 0 \leq y < \sum_{j=1}^{i} r_j \cdot d^{i-j} \\ \sum_{j=i+1}^{D} r_j \cdot d^{D-j} & \text{if } y = \sum_{j=1}^{i} r_j \cdot d^{i-j} \\ 0 & \text{otherwise} \end{cases}$$

である。ここで、(9)式において$y$を与えられた値とし、$u$を未知数とみなす。性質5および$gcd(n,d)=1$より、式(9)は各整数$y$ $(0 \leq y < n)$に対して解$u$を1つだけ持つ。従って、$v$を$i$番目に通る経路の数は、

$$d^{D-i} \cdot \sum_{j=1}^{i} r_j \cdot d^{i-j} + \sum_{j=i+1}^{D} r_j \cdot d^{D-j} = n$$

となる。$i$は$1,2,\ldots,D-1$の値を取るので、$v$を経由する経路の数は$(D-1)n$である。よって、

$$\xi_v(G_B(n,d),\rho_1) \leq (D-1)n < n\log_d n$$

であるので、

$$\xi(G_B(n,d),\rho_1) < n\log_d n$$

が成立する。 □

次に、$gcd(n,d) \neq 1$となる場合を考える。$t = \lfloor d^D/n \rfloor$とする。このとき、$tn \leq d^D < (t+1)n$が成立し、$1 \leq t < d$である。この場合のルーティング$\rho_2$を以下のように定義する。

**定義 9**

$$\rho_2(u,v) = W_{D,j}(u,v)$$

ここで、$j$は$P_D(u,v) \equiv j \pmod t (0 \leq j < t)$とする。 □

**定理 5** $\xi(G_B(n,d),\rho_2) < 2n\lceil \log_d n \rceil$が成立する。 □

(**証明**)

$n$の$D$けた$d$進数表現を$(r_1,r_2,\ldots,r_D)$とし、$v$を任意の頂点とする。先の定理の証明と同じく、$v$を通る経路数を数える。$v$と異なる任意の頂点$u$に対し、$u$から$V$への$d^D$個の長さ$D$の経路のうち、

$$(0,0,\ldots,0),(0,0,\ldots,1),\ldots,(r_1,r_2,\ldots,r_D-1)$$

で表現される$n$個の経路が$W_{D,0}(u)$であり、

$$(r_1,r_2,\ldots,r_D),(r_1,r_2,\ldots,r_D+1),\ldots,(2n-1\text{の}D\text{けた}d\text{進数表現})$$

で表現される$n$個の経路が$W_{D,1}(u)$である。$W_{D,2}(u)$なども同様である。

$\rho_2$の定義より、$u$から$w$への経路として$W_{D,j}(u)$の遊歩道が用いられたとすると、$u$から$w+1$,$w+2$, ..., $w+t-1$への経路として$W_{D,j}(u)$の遊歩道は用いられ

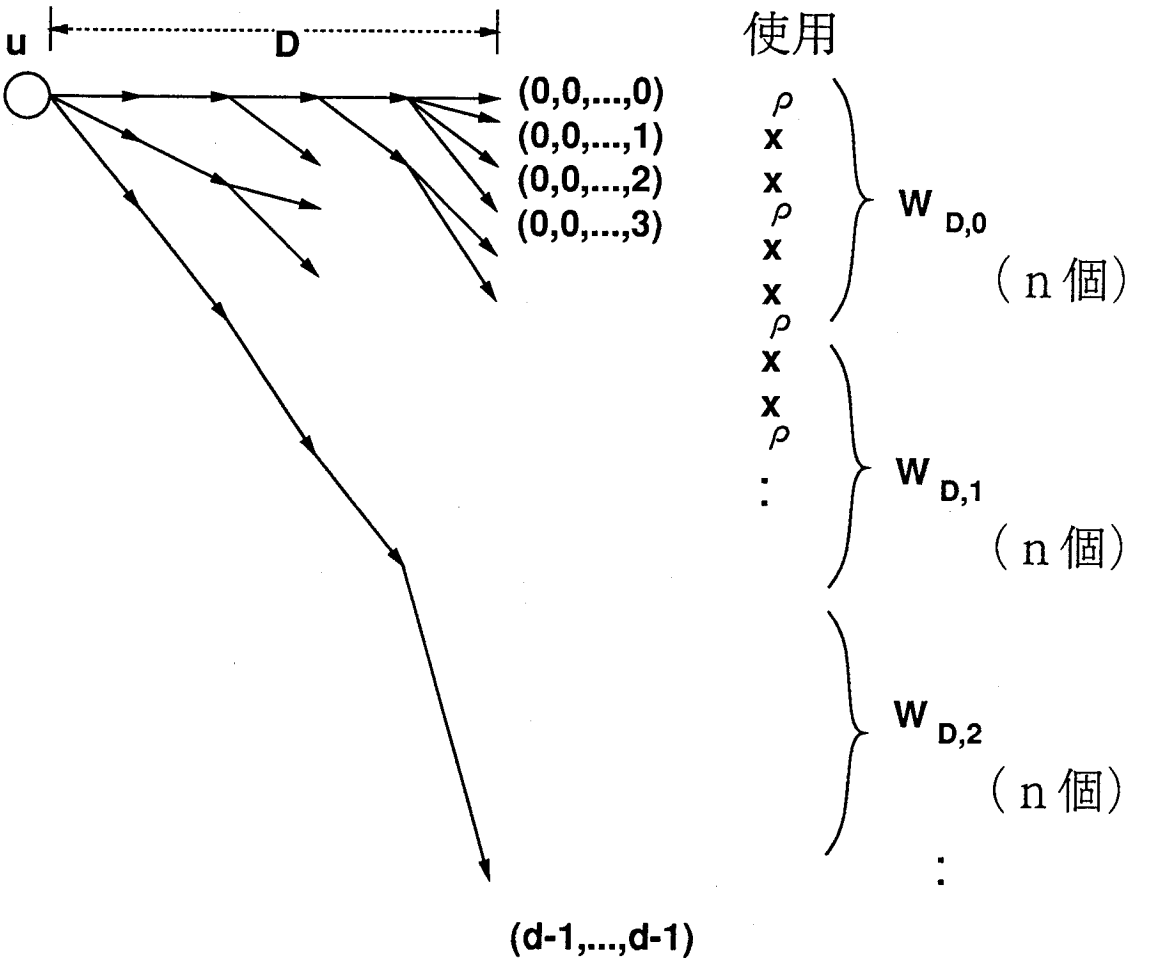


図 8: $u$ からの経路の使用状況

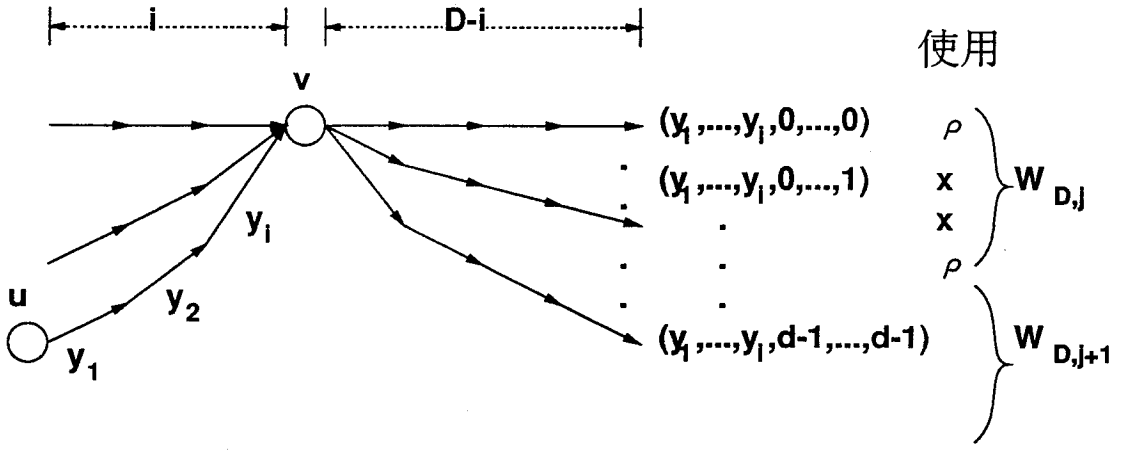


図 9: $v$ を通る $\rho_2$ の経路

ず、次に $W_{D,j}(u)$ の遊歩道が用いられるのは、$u$ から $w+t$ への場合である (但し、$W_{D,j}(u,w+t)$ が定義されている場合のみ)。すなわち、$W_{D,j}(u)$ の遊歩道は $t$ 個おきに $\rho_2$ の経路として用いられる (図 8)。

ここで、式 (9) を考える。$0 \leq y < d^i$ である場合には、$y$ の $i$ けた $d$ 進数表現を $(y_1, y_2, \ldots, y_i)$ とおくと、

$$(y_1, y_2, \ldots, y_i, q_1, q_2, \ldots, q_{D-i}) \ (0 \leq q_j < d)$$

で表現される $d^{D-i}$ 個の長さ $D$ の $u$ から $V$ への遊歩道が $v$ を経由する。$j_0$ を、これら $d^{D-i}$ 個の遊歩道の中に、少なくとも 1 つは $W_{D,j}(u)$ の要素が存在するような最小の $j$ とする。このとき、$j' > j_0 + 1$ なる $j'$ に対しては、これら $d^{D-i}$ 個の遊歩道の中に $W_{D,j'}(u)$ の要素は存在しない。なぜならば $|W_{D,j}(u)| = n > d^{D-i}$ であるからである。この $d^{D-i}$ 個のうちで $\alpha$ 個が $W_{D,j}(u)$ の要素で、残りの $d^{D-i} - \alpha$ が $W_{D,j+1}(u)$ と仮定する。このとき、高々 $\left\lceil \frac{\alpha}{t} \right\rceil + \left\lceil \frac{d^{D-i}-\alpha}{t} \right\rceil$ 本の遊歩道が $u$ からの経路として $\rho_2$ で用いられる (図 9)。

これを用いて、$\rho_2$ で定義される $u$ から $V$ への経路のうち、$i$ 番目に通る頂点が $v$ となるものの数を評価する。性質 5 より、式 (9) の解 $u$ の数は $v - y$ が $gcd(d^i, n)$ の倍数である場合に $gcd(d^i, n)$ となる。相異なる $y (0 \leq y < d^i)$ に対する $d^i$ 個の式に対する解 $u$ の個数の和は $d^i$ である、なぜならばこれら $d^i$ 個の式のうち、$gcd(d^i, n)$ 個に 1 つの式が $gcd(d^i, n)$ 個の解を持つからである。

従って、

$$\begin{aligned} \xi_v(G_B(n,d), \rho_2) &\leq \sum_{i=1}^{D-1} d^i \left( \left\lceil \frac{\alpha}{t} \right\rceil + \left\lceil \frac{d^{D-i}-\alpha}{t} \right\rceil \right) \leq \sum_{i=1}^{D-1} d^i \left( \frac{d^{D-i}}{t} + \frac{2(t-1)}{t} \right) \\ &= \frac{(D-1)d^D}{t} + \frac{2(t-1)(d^D-d)}{(d-1)t} \end{aligned}$$

となる。なぜなら

$$\left\lceil \frac{b}{a} \right\rceil \leq \frac{b}{a} + \frac{a-1}{a}$$

が成立するからである。ここで、

$$X = 2nD - \xi_v(G_B(n,d), \rho_2)$$

とおくと、$d^D < (t+1)n$ より、

$$X > 2nD - \frac{t+1}{t}(D-1)n - \frac{2(t-1)\{(t+1)n-d\}}{t(d-1)}$$

となり、さらに $t \leq d-1$ より

$$\begin{aligned} X &> 2nD - \tfrac{t+1}{t}(D-1)n - \tfrac{2(t-1)\{(t+1)n-t-1\}}{t^2} \\ &> \tfrac{n\{(D-1)(t^2-t)+2\}}{t^2} + \tfrac{2(t^2-1)}{t^2} \end{aligned}$$

となる。ここで $t \geq 1$、$D \geq 2$ より、$X > 0$ となる。従って、

$$\xi_v(G_B(n,d), \rho_2) < 2n\lceil \log_d n \rceil$$

であるので

$$\xi(G_B(n,d), \rho_2) < 2n\lceil \log_d n \rceil$$

が成立する。 □

定理5からはまた、準最適な中継処理量を持つ無向グラフとルーティングの組を得ることができる。$G'_B(n,d)$ を、$G_B(n,d)$ の有向辺を無向辺で置き換えた無向グラフとする。このとき $\Delta(G'_B(n,d)) = 2d$ である。従って、任意の $n$ および $d$ に対し、頂点数 $n$、最大次数 $\Delta = d$ の無向グラフ $G_u(n,d)$ を $G'_B(n, \lfloor d/2 \rfloor)$ から得る ($d$ が奇数の場合は、適当に枝を加える) ことができる。$\rho_2$ をルーティングとして用いれば、

$$\xi(G_u(n,d), \rho_2) = \xi(G_B(n, \lfloor d/2 \rfloor), \rho_2) < 2n\lceil \log_{\lfloor d/2 \rfloor} n \rceil$$

が得られる。現在までに知られている、中継処理量の小さい無向グラフの構成法としては、

$$\xi(G(n,d), \rho) < 3n\log_{\lfloor d/3 \rfloor} n$$

を満足するものが示されている [9]。$n \geq \lfloor d/2 \rfloor \cdot \lfloor d/3 \rfloor$ の場合には

$$2n\lceil \log_{\lfloor d/2 \rfloor} n \rceil < 3n\log_{\lfloor d/3 \rfloor} n$$

が成立するので、この範囲については既知の結果よりも中継処理量の小さいグラフが得られる。

### 3.4 中継処理量とSR-グラフの直径がともに準最適な有向グラフとその上のルーティングの組

前節までは、SR-グラフの直径および中継処理量単独についての議論を行なってきた。本節では、中継処理量が準最適で、かつSR-グラフの直径も準最適な有向グラフとその上のルーティングの組を示す。グラフは前節の一般化de Bruijnグラフ$G_B(n,d)$を用いる。$gcd(n,d) = 1$の場合、$G_B(n,d)$上のルーティング$\rho_1'$を、以下のように定義する。

**定義 10**

$$\rho_1'(u,v) = \begin{cases} W_{D-1}(u,v) & \text{if } P_{D-1}(u,v) < d^{D-1} \\ W_{D,0}(u,v) & \text{otherwise} \end{cases}$$

□

**定理 6** $gcd(n,d) = 1$かつ$n > d^4$のとき、

$$\xi(G_B(n,d),\rho_1') < 2n\log_d n$$

かつ

$$\max_{|F|<d-1} D(R(G_B(n,d),\rho_1')/F) \leq 3$$

が成立する。 □

(証明)

(中継処理量)

任意の頂点$v$に対し、$W_{D-1}$の遊歩道で$v$を経由するものを考える。$P_{D-1}(u,w) < d^{D-1}$を満足する$u$と$w$の組に対し、$W_{D-1}(u)$で定義される$u$から$w$への道は$(D-1)$けた$d$進数$(q_1,q_2,\ldots,q_{D-1})$$(0 \leq q_i < d)$で表現される。定理4の証明と同様に、式(9)を考える。$v$が$i$番目の頂点として現れる$W_{D-1}(u)$の遊歩道の数は、

$$\begin{cases} d^{D-i-1} & \text{if } 0 \leq y < d^i \\ 0 & \text{otherwise} \end{cases}$$

となる。$gcd(n,d) = 1$であるので、式(9)の解$u$の数は、任意の$y$に対して1個である。従って、$v$を$i$番目の頂点として通る$W_{D-1}$の遊歩道の数は$d^{D-i-1} \cdot d^i = d^{D-1}$となる。$i$は$1,2,\ldots,D-2$の値を取るので、$v$を経由する$W_{D-1}$の経路の数は高々$(D-2)\cdot d^{D-1}$となる。

従って、

$$\xi_v(G_B(n,d),\rho_1') \leq \xi_v(G_B(n,d),\rho_1) + (D-2)\cdot d^{D-1}$$

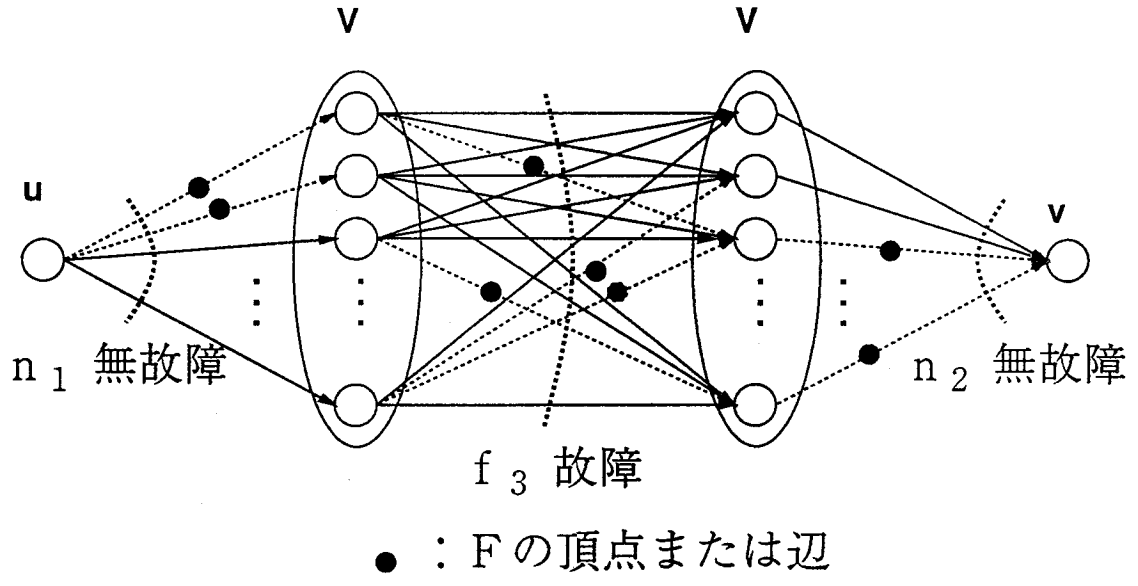


図 10: SR- グラフの直径の評価

$$= (D-1)\cdot d^D + (D-2)\cdot d^{D-1} < 2n\log_d n$$

であるので、

$$\xi(G_B(n,d),\rho_1') < 2n\log_d n$$

が成立する。

**(SR- グラフの直径)**

$V-F$ の相異なる頂点を $u$ および $v$ とする。$u$ から $V$ までの $n$ 個の経路のうち、$n_1$ 個が無故障で、かつ $V$ から $v$ への $n$ 個の経路のうち、$n_2$ 個が無故障で、$V$ から $V$ への $n^2$ 個の経路のうち $f_3$ 個が故障しているとする。もし、$n_1 > 0$、$n_2 > 0$ および $n_1 \cdot n_2 > f_3$ が成立すれば、ある 2 頂点 $u'$ および $v'$ が存在して $u$ から $u'$ への経路、$u'$ から $v'$ への経路、$v'$ から $v$ への経路すべてが無故障である。従って、

$$D(R(G_B(n,d),\rho_1')/F) \leq 3$$

が成立する (図 10)。

以降では、頂点の故障のみを考慮する。辺に故障が存在する場合は、その辺に隣接した頂点のうちの 1 つが故障しているとみなすことができる。

まず $u$ から $V$ への経路を考え、$n_1$ を評価する。$W_{D-1}(u)$ の $d^{D-1}$ 個の経路のうちで、無故障なものの数を求める。$f$ をある故障頂点とする。定理 4の証明と同じく、

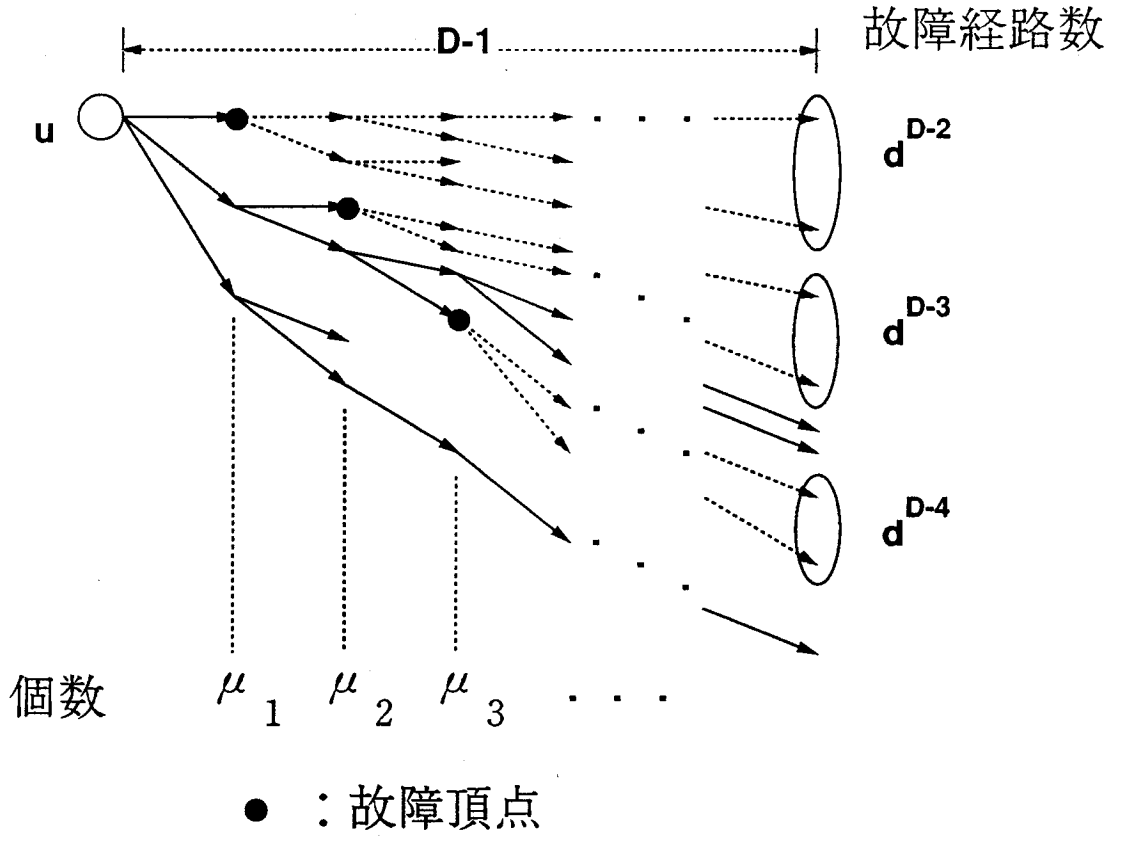


図 11: $u$ からの無故障な経路数

以下の式を考える。

$$f \equiv u \cdot d^i + y \pmod{n} \qquad (10)$$

$u$ と $f$ がある $i(1 \leq i \leq D-1)$ およびある $y(0 \leq y < d^i)$ に対して上式を満足すると仮定する。$y$ の $i$ けた $d$ 進数表現を $(y_1, \ldots, y_i)$ とすると、

$$(y_1, \ldots, y_i, q_1, \ldots, q_{D-i-1})(0 \leq q_j \leq d-1)$$

で表現される $d^{D-i-1}$ 個の経路が $f$ を経由するか $f$ を端点とするため、故障している (図 11)。$\mu_i(1 \leq i \leq D-1)$ を、$i$ に対して式 (10) を満足する $y(0 \leq y < d^i)$ を持つ故障頂点 $f$ の個数とする。このとき、$u$ から $V$ への経路のうちで無故障なものの数は

$$n_1 \geq d^{D-1} - \sum_{i=1}^{D-1} \mu_i \cdot d^{D-i-1}$$

となる。ここで、$|F| < d-1$ より、$\mu_i \leq d-2$ であり、$n_1 > 0$ が成立する。

次に、$n_2$ に関して同様の議論を行なう。以下の式を考える。

$$v \equiv f \cdot d^i + y' \pmod{n} \qquad (11)$$

$v$ と $f$ がある $i(1 \leq i \leq D-1)$ およびある $y'(0 \leq y' < d^i)$ に対して上式を満足すると仮定する。このとき同様にして $d^{D-i-1}$ 個の経路は $f$ を経由するか $f$ を端点とするため故障する。$\nu_i(1 \leq i \leq D-1)$ を、$i$ に対して式 (11) を満足する故障頂点 $f$ の数とする。このとき、$V$ から $v$ への経路のうちで無故障なものの数 $n_2$ は

$$n_2 \geq d^{D-1} - \sum_{i=1}^{D-1} \nu_i \cdot d^{D-i-1} > 0$$

となる。

ここで、$i_1 + i_2 \leq D-1$ と仮定すると $\mu_{i_1} + \nu_{i_2} \leq |F| + 1$ となることを示す。$\mu_{i_1} + \nu_{i_2} > |F| + 1$ とする。このとき、$i = i_1$ に対して式 (10) を満足し、かつ $i = i_2$ に対して式 (11) を満足する 2 つの相異なる頂点 $f_1, f_2 \in F$ が存在する。すなわち、以下の式が成立する。

$$f_k \equiv u \cdot d^{i_1} + y_k \pmod{n}, \quad v \equiv f_k \cdot d^{i_2} + y'_k \pmod{n}$$

$$0 \leq y_k < d^{i_1}, 0 \leq y'_k < d^{i_2} \ (k = 1, 2)$$

上式より

$$f_1 - f_2 \equiv y_1 - y_2$$

および

$$(f_1 - f_2)d^{i_2} + (y'_1 - y'_2) \equiv 0$$

が成立し、

$$(y_1 - y_2)d^{i_2} + (y'_1 - y'_2) \equiv 0$$

となる。$f_1 \neq f_2$ より $y_1 - y_2 \neq 0$ であり、$|(y_1 - y_2)d^{i_2}| \geq d^{i_2}$ となる。$|y'_1 - y'_2| < d^{i_2}$ より、

$$(y_1 - y_2)d^{i_2} + (y'_1 - y'_2) = jn$$

を満足する整数 $j(\neq 0)$ が存在する。しかし、$i_1 + i_2 \leq D-1$ であるので、$0 \leq y_k < d^{i_1}, 0 \leq y'_k < d^{i_2}$ より

$$|(y_1 - y_2)d^{i_2} + (y'_1 - y'_2)| \leq d^{D-1} - 1 < n$$

が成立する。これは上式に矛盾する。従って、$i_1 + i_2 \leq D-1$ であれば、$f_1 = f_2$、すなわち $\mu_{i_1} + \nu_{i_2} \leq |F| + 1$ となる。

$i = 3, \ldots, D-1$ の場合には $\mu_i, \nu_i \leq |F| \leq d-2$ が成立するので、

$$\begin{aligned} n_1 \cdot n_2 \geq \ & (d^{D-1} - \mu_1 d^{D-2} - \mu_2 d^{D-3} - \textstyle\sum_{i=3}^{D-1}(d-2)d^{D-i-1}) \\ & \cdot (d^{D-1} - \nu_1 d^{D-2} - \nu_2 d^{D-3} - \textstyle\sum_{i=3}^{D-1}(d-2)d^{D-i-1}) \end{aligned} \qquad (12)$$

が成立する。$n > d^4$、すなわち、$D \geq 5$ の場合には、$i_1 + i_2 \leq 4$ に対して $\mu_{i_1} + \nu_{i_2} \leq d-1$ である。この式と $0 \leq \mu_i, \nu_i \leq d-2$ $(i = 1, 2)$ より、式 (12) の右辺は $\mu_1 = 1$、$\nu_1 = d-2$、$\mu_2 = 1$ かつ $\nu_2 = d-2$ (あるいは $\mu_1 = d-2$、$\nu_1 = 1$、$\mu_2 = d-2$ かつ $\nu_2 = 1$) が成立する時に最小になる。従って、

$$n_1 \cdot n_2 > (d-2)(d+1)(d^{2D-5} + d^{2D-6})$$

である。

最後に、故障を端点以外の頂点に持つ経路の数 $f_3$ を評価する。ある故障頂点を $f$ とする。ある頂点 $w$ に対し、以下の式を考える。

$$f \equiv w \cdot d^i + y \pmod{n} \qquad (13)$$

すべての $y(0 \leq y < d^i)$ に対するこの式の解 $w$ の数の総和は、定理4の証明と同じようにして、$d^i$ であることが求められる。上式の解である各頂点 $w$ に対し、$f$ を経由する経路の数は高々 $d^{D-i} + d^{D-i-1}$ である。ここで、$d^{D-i}$ は $W_D(w)$ の経路、$d^{D-i-1}$ は $W_{D-1}(w)$ の経路の分である。従って、

$$f_3 \leq (d-2) \cdot (\sum_{i=1}^{D-1} d^i \cdot d^{D-i} + \sum_{i=1}^{D-2} d^i \cdot d^{D-i-1}) = (d-2) \cdot \{(D-1) \cdot d^D + (D-2) \cdot d^{D-1}\}$$

である。$d \geq 3$ より、

$$F(d, D) = (n_1 \cdot n_2 - f_3)/(d-2)d^{D-1}$$

とおくと、$F(d, D) = (d+1) \cdot (d^{D-4} + d^{D-5}) - \{d(D-1) + D - 2\}$ となる。$D$ について $F(d, D)$ が最小になるのは $d \geq 3$, $D \geq 5$ より $D = 5$ の時である。$F(d, 5) = d^2 - 2d - 2$ は $d \geq 3$ の時、$F(d, 5) > 0$ を満足する。よって、$n_1 \cdot n_2 - f_3 > 0$ が成立し、

$$\max_{|F| < d-1} D(R(G_B(n, d), \rho_1')/F) \leq 3$$

である。 □

$gcd(n, d) \neq 1$ となる場合については、$\rho_2$ に以下のように変更を加える。

**定義 11**

$$\rho_2'(u, v) = \begin{cases} W_{D-1}(u, v) & \text{if } P_{D-1}(u, v) < d^{D-1} \\ W_{D,j}(u, v) & \text{otherwise} \end{cases}$$

ここで、$j$ は $P_D(u, v) \equiv j \pmod{t} (0 \leq j < t)$ とする。 □

**定理 7** $n > d^4$ および $d \geq 3$ であれば、

$$\xi(G_B(n,d),\rho_2') < 3n\log_d n$$

および

$$\max_{|F|<d-1} D(R(G_B(n,d),\rho_2')/F) \leq 3$$

が成立する。 □

(**証明**)

(**中継処理量**)

頂点 $v$ を経由する $W_{D-1}$ の経路の数を評価する。定理5の証明と同様の議論により、$v-y$ が $gcd(d^i,n)$ の倍数である場合、式 (9) は $gcd(d^i,n)$ 個の解を持つ。従って、解の数の総和は $d^i$ となる。よって、定理5の証明と同様にして、$v$ を経由する $W_{D-1}$ の経路の数は高々 $(D-2)\cdot d^{D-1}$ である。従って、

$$\begin{aligned}\xi_v(G_B(n,d),\rho_2') &\leq \xi_v(G_B(n,d),\rho_2) + (D-2)\cdot d^{D-1}\\ &\leq \tfrac{(D-1)d^D}{t} + \tfrac{2(d^D-d)(t-1)}{(d-1)t} + (D-2)d^{D-1}\\ &< \tfrac{(D-1)(t+1)n}{t} + \tfrac{2(t+1)n-2d}{d-1} + \tfrac{(D-2)(t+1)n}{d}\\ &< 3n(D-1) < 3n\log_d n\end{aligned}$$

となる。ここで、$d \geq 3$ であるので、$1 \leq t < d$ に対して $d^D < (t+1)n$ が成立し、$n > d^4$ であるので $D \geq 5$ が成立することによる。

SR- グラフの直径に関する証明については、定理6の証明と同様である。 □

### 3.5 むすび

本章では、以下の結果を示した。

- $k$- 連結な有向グラフが SR- グラフの直径が 6 以下となるルーティングを持つための十分条件
- 任意の頂点数および最大実数に対し、中継処理量が準最適になる有向グラフとルーティングの組を構成する方法
- 任意の頂点数および最大次数に対し、SR- グラフの直径と中継処理量がともに準最適である有向グラフとルーティングの組を構成する方法

以下の問題が今後の課題として残されている。

- $k$- 連結な有向グラフがSR- グラフの直径が最適値2となるルーティングを持つための条件を求める問題

- SR- グラフの直径と中継処理量の両方が最適値となる有向グラフとルーティングの組を求める問題

などが考えられる。なお、このうちの最初の問題に関しては、本結果ののち、$k$- 連結な有向グラフがSR- グラフの直径が3および最適値2となるルーティングを持つための十分条件が求められている[49][50]。

また、本稿の議論では故障の発生時にルーティングを変更しないことを仮定した。ルーティングを動的に変更することを評価に含めた場合の耐故障性・効率をいかに定式化し、どのようなルーティングが最適であるかを求める問題も重要である。

# 4　分散プログラムデバッグの基本機能

本章では、非決定的な動作の再演を行なう分散プログラム用デバッガにブレークポイント設定機能およびトレース機能を導入する問題に関し、以下の結果を示す。

積条件式と和条件式という二種類の大域的条件式に対して、条件式を満足する地点で停止するブレークポイント設定機能の実現性を議論する。そして、積条件式については、条件式を満足する最初の地点で全プロセスを停止することが可能であることを示す。また、和条件式については、最初の地点で全プロセスを停止することは不可能であるが、条件を満足するある地点で停止することは可能であることを示す。さらに、その他の一般的な大域的条件式については、条件式を満足するある地点で停止することも不可能であることを示す。

また、大域的条件式を用いたトレース機能 (条件式成立時点の状態を出力する機能) については、積条件式および和条件式のいずれについてもトレースが可能であることを示す。

そして、停止およびトレースが可能なおのおのの場合について、その機能を実現するアルゴリズムを示す。

4.1節では大域的条件式検出に関する従来の研究を示す。4.2節では分散システムおよびデバッガのモデルを示す。4.3節ではブレークポイント設定機能の実現性を示す。4.4節ではトレース機能の実現性について示す。4.5節では、4.3節および4.4節で示したブレークポイント設定およびトレース機能の有効性を確認するために行なったプロトタイプデバッガの作成およびその評価について述べる。

## 4.1　従来の研究

分散プログラムのデバッグで重要な問題のうちの1つは、複数のプロセスが関係しているバグをいかにして検出するかである [37][41]。

上記の目的を達成するための1つの手段は、複数のプロセスに関係した条件式 (大域的条件式) でブレークポイントを設定して、その条件に合致した時に停止することである。また、与えられた条件が成立した時の状態 (ソースプログラムの位置および変数の値など) を表示するトレース機能もそのための機能の1つである。

例えば、プロセス $p_i(i = 1,2)$ がおのおの変数 $x_i$ を持ち、$p_i$ は $x_i = 1$ が成立するとき、ある共有データに対してアクセス権を持つことを表すものとする (図12)。このとき、同時に $x_1 = 1$ と $x_2 = 1$ が成立する場合が存在するプログラムにはバグがあり、そのためどこかの地点でエラーが顕在化する。この種のバグを検出するためには、“$x_1 = 1 \bigcap x_2 = 1$” という式が成立するか否かを検出する機構が必要である。

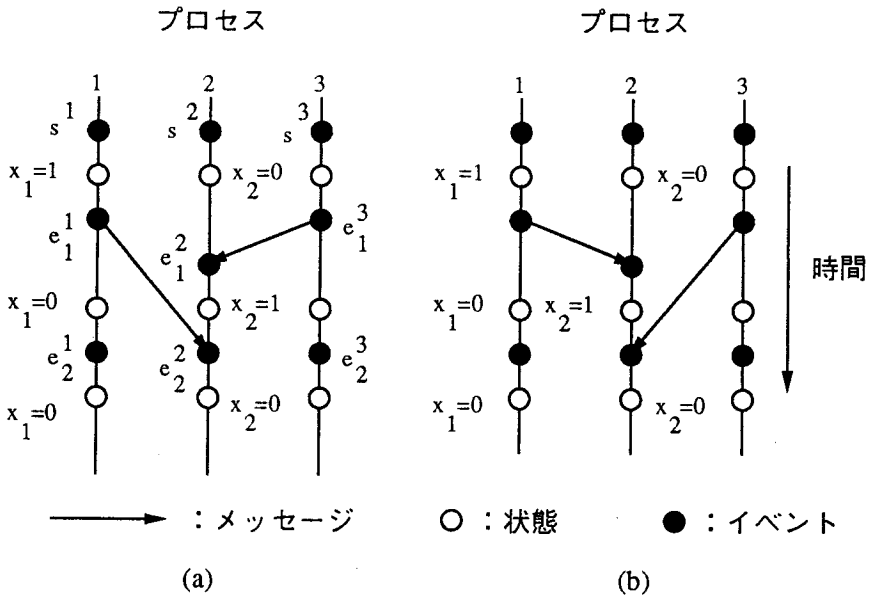


図 12: 分散プログラムの動作例。 (a): バグ顕在化、 (b): バグ非顕在化

このような、複数のプロセスに関する論理式を大域的条件式 (global predicate) と呼ぶことにする。

文献 [26] では、2 種類の大域的条件式について考察している。1 つは和条件式 (Disjunctive Predicate) DP であり、これは単純条件式を "$\cup$" でつないだものである。ここで、単純条件式とは一つのプロセスにおいて、任意の時点で真偽が判定可能な条件式である。もう一つは積条件式 (Conjunctive Predicate) CP であり、単純条件式を "$\cap$" でつないだものである。本稿ではこの 2 種類の条件式について考察する。

大域的条件式の成立を検出する問題には、3 つの大きな要求条件がある。まず第一に、プローブイフェクト (probe effect) がなるべく小さいものであるべきであるという点である。分散プログラムは非同期的に動作するため、そのタイミングによって動作の状況が異なるという非決定性が存在する。よって、もしも大域的条件式成立検出のための動作が実行中に行なわれると、その動作は検出を行なわない場合のものと異なる場合がある。従って、検出のための動作はなるべく小さいものであることが望ましい。

第二に、デバッガはユーザに、条件式成立直後の状態をなるべく見せることが望ましい。ユーザがセットした条件式はバグの状態を表すものであることが多いので、その条件式成立以降の動作はユーザには意味のないものである場合が多い。さらに、余分な動作を行なうことによってユーザがバグの原因を発見することの妨げになる可能性もある。

例えば、図 12の例で、大域的条件式を "$x_1 = 1 \cap x_2 = 1$" とする。この条件が成立

した場合には、プロセス1および2はそれぞれ、$x_1 = 1$および$x_2 = 1$が成立した直後で停止させるのが望ましい。しかし、この式にはプロセス3に関する条件式は存在しないため、プロセス3の停止地点は明確ではない。このような場合にプロセス3を停止させないという方法が考えられている。具体的には、そのプロセスが、すでに停止したプロセスからのメッセージを待つ状態になってブロックさせられるまで動作させるというものである[34]。しかし、この条件式で表現されるバグの原因はプロセス1や2以外のプロセスに存在することもある。この例では、プロセス3からのメッセージの内容の誤りによって、条件式が成立したという可能性もある。そのような場合に、プロセス3を動作させるとプロセス3がなぜそのメッセージを送信したかがわかりにくくなる。例えば、プロセス3がメッセージ送信を行なったサブルーチンから脱出して、メッセージ送信を決定した変数をすべてクリアしてしまうなどの場合が考えられる。従って、与えられた条件式に対して、それを成立させる最小限の動作を行なった地点で全プロセスを成立させるのが最適な停止地点であると考えられる。

第三に、デバッガは同じ動作を何度も繰り返してテストできることが望ましい。サイクリックデバッグ法は逐次型プログラムのデバッグにおいて一般的に用いられている方法である[22]。サイクリックデバッグでは、ユーザはまずブレークポイントをセットしてプログラムを動作させる。ブレークポイントの条件式が成立して停止したら、ユーザはその時の変数の状態などを見て誤りを発見できた場合には、さらにその原因を求めるために別のブレークポイントをセットしてプログラムを再度実行する。その条件式が成立した場合にはさらにその原因を探るブレークポイントをセットし再実行、という手順を繰り返して行く方法である。

分散プログラムに対してサイクリックデバッグを行なうためには、2度目の実行においても同じ状況が再現されなければならない。例えば、図12の実行例に対して、1回目の実行が図12(a)の通りであり、誤りが検出されたとする。しかし、2回目の実行が、メッセージ通信の遅れが2回目で異なることにより、図12(b)のような動作になることもある。この場合には、条件式が成立しないため、バグは顕在化しない。このように、分散プログラムにおいては、2回目の実行において1回目の実行と異なる動作を行なうことがあるため、サイクリックデバッグのためには、同一の動作を繰り返してテストできることが望ましい。

大域的条件式成立検出の手法は3つに分類される。

- 実行中検出
- 実行後検出

- 再演時検出

実行中検出はプログラムの実行時に条件式検出の手続きを実行するものである[15][26][38]。この方法は再演の手続きも不要であり、プログラムを事前に1度実行させる必要もない。Millerと Choi[38] は linked predicate と呼ぶ大域的条件式を実行中に検出するアルゴリズムを提案している。Linked predicate は "$SP_1 \rightarrow SP_2 \rightarrow \ldots SP_n$" という形式である、ここで $SP_i$ は単純条件式である。ここで、"$\rightarrow$" は後述する Lamport の因果関係である [33]。従って、"$SP_1 \rightarrow SP_2 \rightarrow \ldots SP_n$" は "$SP_1$ が成立し、そののち $SP_2$ が成立し、(中略)、そののち $SP_n$ が成立" を意味する。この条件式は成立の検出が比較的容易である。その理由は、この式自身をプロセス間のメッセージにのせて検出していくことが可能だからである。しかし、実行中に検出を行なうためプローブイフェクトは一般に大きくなる。さらに、式が成立した直後の状態をユーザに見せることは不可能である。その理由はこのアルゴリズムは与えられた "$SP_1 \rightarrow SP_2$" という式が与えられた場合に $SP_1$ が成立しても停止させないため、$SP_1$ を持つプロセスがそれ以上の実行を行なうことを許しているためである。

Haban と Weigel[26] はプログラム実行時に積条件式と和条件式の成立を検出するアルゴリズムを示している。しかし、このアルゴリズムも条件式が成立した直後の状態をユーザに見せることは不可能である。プローブイフェクトの問題もやはり存在する。

Cooper と Marzullo[15] はある条件式 $P$ が成立した直後の状態を "Currently $P$" と呼び、その状態で停止する問題を考えている。しかし、このアルゴリズムはいくつかのプロセスの動作を強制的にブロックするため、プローブイフェクトは非常に大きくなる。また、条件式 $P$ によっては、"Currently $P$" において停止することは不可能である。しかし、論文中ではそのことに言及されていない。一般的な大域的条件式に対して、条件式成立時点で停止することが不可能なことは本稿で後述する。

実行後検出は実行の際に各プロセスごとに独立にイベントの発生、変数の値の変化などの情報を収集し、そのログを実行後に解析するという方法である [12][17][23]。実行中のログ収集アルゴリズムは単純であり、実行後にいかなる複雑な解析も可能である。しかし、実行前にログとして収集するべきデータを規定しておく必要があるため、ログのサイズは大きくなりがちであり、プローブイフェクトは大きくなる可能性が高い。また、収集していなかったデータが必要であることが解析中に明らかになったとしても、そのためにもう一度ログを取るのは難しい。なぜならば、2度目の実行はタイミングの違いからふるまいが異なることがあり、バグが顕在化しな

い可能性もあるからである。

再演時検出は以下のような手法である。最初の実行の際、再演のために必要な最小限の情報を収集し、そのログをもとに、実行状況を再演させながら実行する際に大域的条件検出を行なう手法である。各プロセスのプログラムにおいて非同期的な割り込みやタイムアウトなどの時間に依存したイベントが存在しない場合には、実行を再演するためのログは非常に小さくなることが知られている [34]。その手法による分散プログラム再演手法がいくつか提案させている [10][34][46]。しかし、これらの研究は再演のみを議論し、大域的条件式検出のようなデバッグ機能については議論していない。

### 4.2 モデルおよび定義

まず、分散システムおよびデバッガについての仮定をおく。本稿では各プロセスのプログラムでメッセージとして送信される値は各プロセスの初期値およびそのプロセスが受信する値およびその順序のみに依存するものと仮定する。すなわち、各プロセスの動作は決定的とする。従って、非同期的な割り込みやタイムアウトなどのイベントは存在しないものとする。

メッセージパッシングのみで通信を行なう分散システムのモデルを以下のように定義する [33]。

- 有限数のプロセスと有限数のチャネルから成る
- チャネルは無限長、エラーなしの FIFO バッファである
- メッセージの遅れは有限ではあるが上限がない

メッセージの送受信は特殊なイベント送信イベントおよび受信イベントと呼ぶ。その他のイベントは内部イベントと呼ぶ。各プロセスの初期状態も特殊なイベントとする。この分散システムの動作において、因果関係 “$<$” を以下のように定義する。

**定義 12** [33]

各プロセスのイベントに対する因果関係 “$<$” は以下の 3 つの条件を満足する最小限の関係である。

1. $a$ と $b$ が同一プロセスに対するイベントで $a$ の方が $b$ より以前であれば、$a < b$ である。

2. $a$ がメッセージの送信イベントで $b$ がそのメッセージに対する受信イベントであれば、$a < b$ である。

3. もし $a < b$ かつ $b < c$ が成立すれば $a < c$ である。

イベント $a$ と $b$ に対し、$a < b$ または $a = b$ が成立する時、$a \leq b$ と書く。 □

本稿ではメッセージパッシングで通信を行なう場合のみを対象としているが、分散システムによっては、共有メモリで通信を行なうものもある [34]。そのようなシステムをメッセージパッシングで以下のようにシミュレートすることが可能である。

- 各共有メモリを仮想的プロセスとみなす。
- 共有メモリへの書き込みは書き手プロセスからメモリプロセスへのメッセージ送信とみなす。
- 共有メモリからの読み出しは読み手プロセスからメモリプロセスへのメッセージとそれに対するメモリプロセスから読み手プロセスへの返答メッセージとみなす。

以下ではメッセージパッシングの場合のみを対象とするが、その結果は共有メモリ通信の場合にも適用可能である。

次に、後述の補題の証明のため、分散システムにおける、Chandy と Lamport の“意味のある全域状態 (meaningful global state)”[11] をフォーマルに定義する。

**定義 13** $E_i$ をプロセス $i$ のイベントの集合とする。プロセスのイベントの $N$ 個組 $s = (e_1, e_2, \ldots, e_N)$ $(e_i \in E_i)$ が全域状態であるとは、以下の条件が成立する時およびその時に限る。

任意の $e_i' \in E_i$ に対し、$e_i' > e_i$ ならば任意の $j (1 \leq j \leq N)$ に対して $e_i' \not< e_j$ が成立する。

$U$ をすべての全域状態の集合とする。 □

意味のある全域状態を以下では単に全域状態と呼ぶ。Lamport の定義 [11] では、全域状態はイベント系列の組に対して定義されている。ここでは簡単化のため、イベント系列ではなく、最終のイベントのみで表現している。

全域状態 $s = (e_1, e_2, \ldots, e_N)$ はプロセス $i$ がイベント $e_i$ の実行を終えて、次のイベントの実行を行なう前の状態を表す。そして、全域状態は、各プロセスの動作のタイミングによっては同時に起こり得る状態の組を表している。

次に、この全域状態に対する因果関係を以下のように定義する。

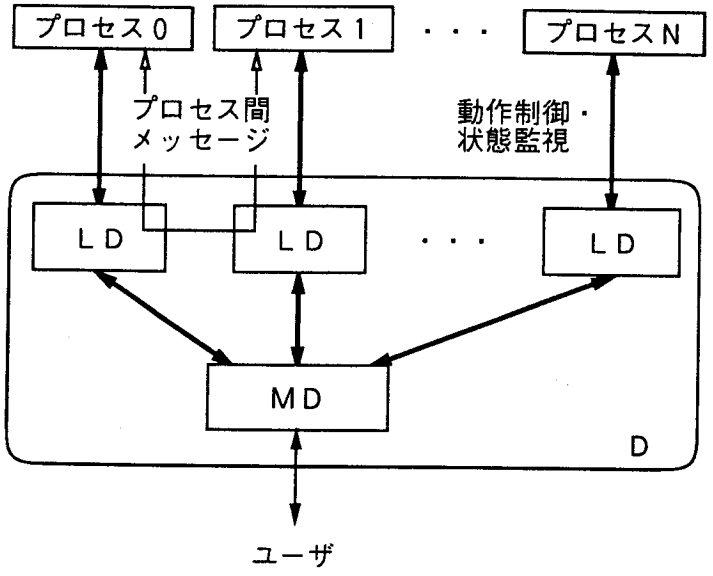


図 13: 分散プログラム用デバッガのモデル

**定義 14** 2つの全域状態 $s = (e_1, e_2, \ldots, e_N)$ および $s' = (e'_1, e'_2, \ldots, e'_N)$ に対し、すべての $i(1 \leq i \leq N)$ に対して $e_i \leq e'_i$ が成立する時、かつその時に限り $s \leq s'$ とする。また、$s \leq s'$ かつ $s \neq s'$ が成立する時、かつその時に限り $s < s'$ である。 □

次に、再演をベースとする分散プログラム用デバッガのモデルを述べる。再演をベースとするデバッガ **D** は1つのメインデバッガ MD と各プロセスに対して存在するローカルデバッガ $\mathrm{LD}_i$ から成る (図13)。

MD と各 $\mathrm{LD}_i$ の間には双方向の1対1の通信路が存在する。$\mathrm{LD}_i$ はプロセス $i$ の動作の停止および再開を行ない、プロセス $i$ が停止している時にその変数の値、現在の地点などを調べることができる。MD は $\mathrm{LD}_i$ にコマンドを送り、その結果を受信し、その結果をユーザに出力する。**D** はプログラムを複数回実行する。1回目の実行をモニタ実行、2回目以降の実行を再演実行と呼ぶ。

モニタ実行時には、各 $\mathrm{LD}_i$ は独立にプロセス $i$ の通信履歴を保存する。ここで保存する通信履歴は各受信イベントに対する番号の系列である。各受信イベントにおける番号は、その受信イベントで実際にメッセージを受信した場合にはその送信を行なったプロセスの番号であり、メッセージ受信に失敗した場合には $Null$ という特別な値である。すなわち、プロセス $i$ において $\alpha$ 回目の受信イベントでプロセス $j$ の送信したメッセージを受信した場合、$i$ の通信履歴の $\alpha$ 番目は $j$ となる。実際のプログラムにおける送信および受信の手続きの形式はプロトタイプデバッガの実現の項で後述する。

モニタ実行のあと、ユーザはデバッガに対してデバッガコマンド、例えば "stop if $P$"($P$ が成立した時に停止せよというブレークポイント設定コマンド) を MD に与える。

再演実行時には、プロセス $i$ へのメッセージ到着は $LD_i$ によって制御され、メッセージの到着順序はモニタ時と同一になるようにされる。これにより、再演時の実行はモニタ時と同一になる [34]。

再演実行時には、$LD_i$ はプロセス $i$ の動作を以下のように制御する。但し、ここであげた機能は逐次型プログラムに対するデバッガが一般に持っている機能である。

**定義 15** $LD_i$ は再演時にプロセス $i$ に対して以下の操作が可能である。

1. 次に実行するイベントが送信イベント、受信イベント、内部イベントのいずれであるかを知る。
2. $i$ に対して送られたメッセージの送り手プロセスを知り、メッセージをバッファする。
3. 次のイベントを実行してプロセスを停止する。但し、受信イベントの場合にはその時受信するメッセージを指定して実行させる。
4. プロセス $i$ の現在の状態を調べる。 □

$LD_i$ で使用可能な関数および手続きを図 14に示す。

### 4.3 ブレークポイント設定機能

本節では、再演実行時にユーザが設定したブレークポイントに対し、その成立を検出して停止するアルゴリズムについて議論する。まず、大域的条件式を定義する。

単純条件式は逐次型プログラムのデバッガでブレークポイントの条件式として使われている式であり、プログラム中の現在の行の番号や変数の値などに関する式である。単純条件式はプログラムの実行中の任意の地点においてその真偽がその時点で判定可能なものとする。すなわち、その真偽がプログラムの実行の履歴に依存するものではないとする。プロセス $i$ に対するある単純条件式を $SP_i$ と書くことにする。プロセス $i$ でイベント $e$ を実行した直後の $SP_i$ の値を $SP_i(e)$ と書く。

次に、2 つの大域的条件式、積条件式と和条件式を定義する。和条件式 (Disjunctive Predicate) DP は単純条件式を "$\cup$" でつないだものである。

DP $P = SP_1 \cup SP_2 \cup \ldots, \cup SP_n$ とすると、$P$ が真になるのは $SP_i(i = 1, \ldots, n)$ のうちの少なくとも 1 つが真である時、かつその時に限る。積条件式 (Conjunctive

```
const  I = ...; /* プロセス番号 */
  N = ...; /* プロセスの総数 */
  SPExist = ...; /* このプロセスに単純条件式が存在すれば真 */
  Null = 0;
  Infinity = ...; /* ∞ */
  Forever =false;
type simple_predicate = ...;
  process = 0 .. N;
  status = (passive, active);
  message_no = 0 .. Infinity /* メッセージ番号 */
  event_type = ( send, receive, internal ); /* 送信 / 受信 / 内部イベントの別 */
  message = record  dest, soc : process;  data :  ... end; /* プロセス間メッセージ */
  list_of_var = ...; /* トレースで出力する変数のリスト */
  list_of_value = ...; /* list_of_var の値のリスト */
function NextEvent : event_type; begin /* 次のイベントのタイプを返す */ end;
procedure LastEvent(var e : event_type, var p : process);
  begin /* 最後に実行したイベントのタイプを返す。
    e =send(receive) のとき、 p は受け手 (送り手) のプロセス番号
    e =internal のとき、p = Null */ end;
procedure ExecNext(rm : message);
  begin /* 次のイベントの実行。
      受信イベントであれば、rm で与えられたメッセージを受信。 */ end;
function JudgePred(SP : simple_predicate) : boolean;
    begin /* 条件式の真偽を返す。 */ end;
function ReadPM : message; begin /* メッセージを読む。 */ end;
procedure InsertQueue(m : message);
  begin /* メッセージを内部キューに挿入する。 */ end;
function ReadQueue(i : process) : message;
  begin /* i から最初に受信したメッセージを返す。なければ Null を返す。 */ end;
function ReadHistory : process;
  begin /* 通信履歴の次の要素を読んで、ポインタを次に進める。 */ end;
function GetV(list : list_of_var) : list_of_value;
  begin /* list で与えられた変数の現在の値を返す。 */ end;
```

図 14: $\mathrm{LD}_I$ の関数

Predicate) CP は単純条件式を "$\bigcap$" でつないだものである。
CP $P = SP_1 \bigcap SP_2 \bigcap \ldots \bigcap SP_n$ とすると、$P$ が真になるのは $SP_i(i = 1, \ldots, n)$ のすべてが真である時、かつその時に限る。大域的条件式は $U$ から $\{true, false\}$ への関数となる。ここで、

$$G(P) = \{s \in U | P(s) = true\}$$

とする。

次に、大域的条件式 $P$ に対し、$P$ が真になる最初の全域状態、$Inf(P)$ を以下のように定義する。

**定義 16** 大域的条件式 $P$ に対し、
$Inf(P) = \{s \in G(P) | s' \notin G(P) \text{ for any } s' \in U \text{ such that } s' < s\}$ □

図 12(a) の動作例に対して、CP $P_1 = (x_1 = 1 \bigcap x_2 = 1)$ とすると、$Inf(P_1) = \{(e_1^1, e_1^2, e_1^3)\}$ である。DP $P_2 = (x_1 = 1 \bigcup x_2 = 1)$ に対して
$Inf(P_2) = \{(e_1^1, s^2, s^3), (s^1, e_1^2, e_1^3)\}$ である。一般に $Inf(P)$ は複数の全域的状態となる。

### 4.3.1 積条件式に対する停止問題

本節では、積条件式 $P$ が与えられた場合に、$Inf(P)$ で停止するアルゴリズムについて議論する。

アルゴリズムの概要は以下の通りである。まず、各プロセスに対し、状態 active および passive を定義する。あるプロセスが実行中の時、active と呼び、そうでない時に passive と呼ぶ。状態が passive であるプロセスは他のプロセスから動作せよというコマンドを受けとった場合にのみ active になる。すべてのプロセスが passive である時にシステムが停止していると呼ぶ。

積条件式 $P$ が与えられた時、$P$ に自プロセスの単純条件式が存在しないプロセスは常に真である単純条件式を持つと仮想的に考える。初期状態において、自プロセスの単純条件式が偽のプロセスは active でそれ以外は passive とする。そして、active なプロセスは自分の単純条件式が真になるまで動作を行なおうとする。active なプロセスがメッセージ $M$ を受信する時、メッセージ $M$ の送り手プロセス $S$ が passive であるために動作が進まない場合がある。従って、受信するべきメッセージが届いていない場合には、その送り手のプロセス $S$ に対して $M$ 送信まで動作せよというコントロールメッセージを送る。ここで、次に受信するメッセージ $M$ の送り手プロ

セスがどのプロセスであるかは、モニタ実行時に収集した通信履歴に書かれているので求めることができる。

このコントロールメッセージを受信した$S$は、もし$M$の送信がまだであればactiveになり、$M$の送信後、自プロセスの単純条件式が真になるまで動作する。すなわち、あるプロセスがactiveになるのは自プロセスの単純条件式が偽であるかメッセージを送信しなければならない時である。

コントロールメッセージでは、どのメッセージを$S$が送らなければならないかを特定できなければならない。なぜならコントロールメッセージを受信した$S$では、$S$がすでに送ったメッセージがまだ(通信の遅れで)相手$R$に届いていないために発送されたコントロールメッセージなのか、自分が本当にまだ送信していないメッセージを$R$が要求しているのかが不明であるからである。従って、コントロールメッセージはメッセージの識別子を持つ。ここで識別子は$S$から$R$へのメッセージにふったシーケンス番号である。そのため、$\mathrm{LD}_i$は各チャネルごとに送受信したメッセージの数を数えておく。このようにすると送信するメッセージ自身にはシーケンス番号をつける必要がない。

最後に、全プロセスがpassiveになった時にそのことを検出できなければならない。この問題は停止検出問題[18]と呼ばれていて、システムに対するさまざまな仮定の元での多くのアルゴリズムが提案されている。停止を検出する部分を停止検出部と呼ぶ。状態がpassiveからactiveへ、またその逆に変化した場合に停止検出部に送られる。コントロールメッセージが停止検出部を必ず経由するようにすると停止検出は簡単になる。停止検出部は分散アルゴリズムとして構成する、すなわち$\mathrm{LD}_i$に実現することも、集中型のアルゴリズムとして実現する、すなわちMDに実現することも可能である。文献[13]に示されているアルゴリズムは停止検出の分散アルゴリズムとして使用可能である。

積条件式$P$に対して$Inf(P)$で停止するアルゴリズムの概要を図15に示す。

次に、このアルゴリズムが$Inf(P)$で停止することの証明を以下に示す。まず、準備として以下の補題を証明する。

**定義 17** 同一のプロセスの2つのイベント$e$および$e'$に対して、$min(e,e')$を、以下のように定義する。

$$min(e,e') = \begin{cases} e & \text{if } e \leq e' \\ e' & \text{otherwise} \end{cases}$$

```
program HaltAtBreakpoint /* LD_I のプログラム */
type external_event_type = ( MessageArrival, Control-MessageArrival, ExecFinish,
    TeminationDetected ); /* ExecFinish: 被デバッグプログラム 1 ステップ終了 */
  control_message = record dest, soc : process; mno : message_no end;
    /* メッセージ送信の要求 */
var s, /* i に送信したメッセージ数 */
  r, /* i から受信したメッセージ数 */
  ms: /* i が受信した送信要求。 */ array [process] of message_no;
  waitp : process; /* プロセス waitp からのメッセージ待ち */
  bstat, cstat : status; /* 直前および現在の status */
  extevent : external_event_type; /* 発生した外部イベント */
  e : event_type ; /* 被デバッグプログラムのイベントのタイプ */
  SP : simple_predicate; /* 自プロセスの単純条件式 */
  m : message; /* 被デバッグプログラムのメッセージ */
  cm : control_message; /* コントロールメッセージ (送信要求) */
  i : process; /* プロセス番号 */
procedure Wait(var event : external_event_type); begin /* 外部イベント待ち */ end;
procedure SendCM(req : control_message); begin /* コントロールメッセージ送信 */ end;
function ReadCM : control_message; begin /* コントロールメッセージ受信 */ end;
procedure SendST(stat : status); begin /* 停止検出部に status を送信 */ end;
function TermTest(s, ms : array [process] of message_no, SP : simple_predicate)
    : status; begin /* ローカルな status のテスト */ end;
procedure TryExec(r : array [process] of message_no, var waitp : process);
  begin /* 被デバッグプログラムの次のイベントを実行しようとする。
    次が受信でメッセージが到着していなければ送り手プロセス i に
      cm.mno = r[i] + 1 としたコントロールメッセージ cm を送信 */ end;
```

図 15: $LD_I$ の停止アルゴリズム

```
begin /* MAIN */
  for i := 1 to N do begin s[i] := 0; r[i] := 0; ms[i] := 0 end;
  waitp := Null;
  cstat := TermTest(s, ms, SP); /* status チェック */
  if cstat =active then TryExec(r, waitp);
  repeat Wait(extevent); /* 外部イベント待ち */
    case extevent of /* 外部イベントの種類により */
      ControlMessageArrival: /* コントロールメッセージ到着 */
        begin
          cm := ReadCM; /* コントロールメッセージ読み込み */
          ms[cm.soc] := cm.mno; bstat := cstat;
          cstat := TermTest(s, ms, SP); SendST(cstat); /* status チェック */
          if bstat =passive and cstat =active then TryExec(r, waitp)
        end; /* End of ControlMessageArrival. */
      ExecFinish: /* 被デバッグプログラム 1 ステップ終了 */
        begin
          LastEvent(e, i);
          if e = send then s[i] := s[i] + 1; /* 送信メッセージ数更新 */
          if e = receive then r[i] := r[i] + 1; /* 受信メッセージ数更新 */
          cstat := TermTest(s, ms, SP); /* status のチェック */
          if cstat =active then TryExec(r, waitp) else SendST(cstat)
        end; /* End of ExecFinish. */
      MessageArrival: /* 被デバッグプログラムメッセージ到着 */
        begin
          m := ReadPM; /* メッセージ読み込み */
          if waitp ≠ m.soc then InsertQueue(m) /* 受信待ちのものでない */
            else begin  waitp := Null; ExecNext(m) end
        end; /* End of MessageArrival. */
      TeminationDetected: ..... /* Inf(P) で停止 */;
    end /* End of case */
  unitl Forever
end.
```

図 16: $\mathrm{LD}_I$ の停止アルゴリズム (続き)

2つの全域状態 $s = (e_1, e_2, \ldots, e_N)$ および $s' = (e'_1, e'_2, \ldots, e'_N)$ に対し、$min(s, s')$ を

$$min(s, s') = (min(e_1, e'_1), min(e_2, e'_2), \ldots, min(e_N, e'_N))$$

と定義する。 □

**補題 9** $s, s' \in U$ の時、$min(s, s') \in U$ である。 □

**(証明)**

$min(s, s') = (e''_1, e''_2, \ldots, e''_N) \notin U$ とする。$U$ の定義より、ある $i$ と $j$ に対して $\exists e'''_i \in E_i, e'''_i > e''_i$ かつ $e'''_i < e''_j$ が成立する。$e''_i = min(e_i, e'_i)$ より、$e''_i = e_i$ または $e''_i = e'_i$ が成立する。一般性を失うことなく、$e''_i = e_i$ とする。このとき、

$$e'''_i > e''_i = e_i \text{かつ} e'''_i < e''_j = min(e_j, e'_j) \leq e_j,$$

が成立するので $s \in U$ に反する。 □

**補題 10** $P$ が CP であれば、$|Inf(P)| \leq 1$ が成立する。 □

**(証明)**

$P$ を $SP_{i_1} \bigcap SPi_2 \bigcap \ldots \bigcap SP_{i_k}$ とする。

2つの相異なる全域的状態 $s$ と $s'$ が $Inf(P)$ に属すとする。もし $s < s'$ ならば $Inf(P)$ の定義から $s' \notin Inf(P)$ となるので、$s \not< s'$ が成立する。従って、$min(s, s') < s$ が成立する。このとき、

$P(min(s, s')) = SP_{i_1}(min(e_{i_1}, e'_{i_1})) \bigcap SP_{i_2}(min(e_{i_2}, e'_{i_2})) \bigcap \ldots \bigcap SP_{i_k}(min(e_{i_k}, e'_{i_k})) = true$

また、$min(s, s') \in U$ が成立する。よって、$min(s, s') \in G(P)$ となる。

$min(s, s') < s$ および $min(s, s') \in G(P)$ より $s \in Inf(P)$ に反する。従って、$|Inf(P)| \leq 1$ である。 □

**定理 8** CP $P$ に対して、図15のアルゴリズムは $Inf(P)$ で停止する。 □

**(証明)**

$|Inf(P)| = 1$、すなわち $P$ が成立する全域状態が存在する時、その全域状態を $inf = (t_1, t_2, \ldots, t_n)$ とする。上記アルゴリズムによって全プロセスが $inf$ で停止することを示すためには、以下の2つを示す必要がある。

- $s < inf, s \in U$ を満足する全域状態で全プロセスが停止することがない。
- 各プロセス $i$ が $t_i$ より先まで動作することがない。

まず、前者を示す。全プロセスがある全域状態 $s \in U$ で停止したと仮定する。このとき、各プロセスが active になる条件より、$s$ で各 $SP_i$ は真である。従って、$s$ は $G(P)$ の要素となる。もしも $s < inf$ であれば、これは $Inf(P)$ の定義に反する。従って前者は成立する。

次に後者を示す。プロセス $i$ が次のイベントを実行するのは、以下の条件のどちらかが成立する時のみである。

(1) $i$ に対する単純条件式 $SP_i$ が偽である。

(2) $i$ がまだ送信していないメッセージを要求するコントロールメッセージを受信した。

あるプロセス $i$ が $t_i$ を越えて実行したとする。この実行中に途中で現れる全域的状態 $s'$ を、$s' = (s_1, s_2, \ldots, s_N) \leq inf$ および $s_i = t_i$ を満足する状態とする。この全域的状態において $s_i = t_i$ を満足するプロセスの集合を $S_{inf}$ とする。$S_{inf}$ に含まれる任意のプロセス $i$ は $s_i = t_i$ を満足するので、上記の動作ルールの (1) によっては動作しない。従って、$S_{inf}$ に含まれるプロセスが動作するのは、プロセス $j \notin S_{inf}$ が $t_j$ を実行するよりも前に、$S_{inf}$ に含まれるあるプロセス $k$ に向けて、まだ送っていないメッセージを要求するコントロールメッセージを送る場合に限られる。このコントロールメッセージに対応するメッセージ送信イベントを $e_k$、受信イベントを $e_j$ とする。この時、$e_k > t_k$、$e_k < e_j$ および $e_j < t_j$ が成立する。後ろの 2 式より $e_k < t_j$ が成立する。$e_k > t_k$ および $e_k < t_j$ が成立する $e_k$ が存在することは $inf \in U$ に反する。従って $t_i$ を越えて動作するプロセスは存在しない。 □

### 4.3.2 和条件式に対する停止問題

与えられた条件式が DP である場合には、1 度の再演では $s \in Inf(P)$ となる状態 $s$ で停止することは不可能である。以下にその証明を示す。

**定理 9** 条件式 $P$ が DP である場合には、1 度の再演で $s \in Inf(P)$ で停止するデバッガ **D** は存在しない。 □

(証明)

以下の例を考える。プロセス数 $n$ が 2 で $P = SP_1 \bigcup SP_2$ とする。プロセス 1 の初期状態 $s^1$ で $SP_1(s^1) = false$、プロセス 2 の初期状態 $s^2$ で $SP_2(s^2) = false$ とする。プロセス 1 と 2 は通信を行なわないとする。プロセス 1 の次のイベントを $e^1$、プロセス 2 の次のイベントを $e^2$ とする。ここで **D** が $e^1$ を動作させたとする。この場合にはプロセス 1 でそのあとずっと $SP_1 = false$ であり、プロセス 2 では $SP_2(e^2) = true$ となる場合が存在する。このとき $Inf(P) = \{(s^1, e^2)\}$ となるので $Inf(P)$ で停止不可能である。

また、**D** が $e^2$ を動作させたとする。この場合にはプロセス 2 でそのあとずっと $SP_2 = false$ であり、プロセス 1 では $SP_1(e^1) = true$ となる場合が存在する。このとき $Inf(P) = \{(e^1, s^2)\}$ となるので $Inf(P)$ で停止不可能である。従って、1 度の再演で $Inf(P)$ で停止できるデバッガは存在しない。 □

$P$ が DP の場合には、ある $s \in G(P)$ で停止することは可能である。停止アルゴリズムは実行中検出のアルゴリズム [38] を用いればよい。

### 4.3.3 一般の大域的条件式に対する停止問題

CP、DP 以外の一般の大域的条件式については、1 度の再演では、ある $s \in G(P)$ で停止することも不可能である。それを以下に示す。

**定理 10** 任意の大域的条件式 $P$ に対し、1 度の再演で $s \in G(P)$ で停止するデバッガ **D** は存在しない。 □

**(証明)**

以下の例を考える。プロセス数 $n$ が 2 で $P = (SP_1^1 \bigcap SP_1^2) \bigcup (SP_2^1 \bigcap SP_2^2)$ とする。プロセス 1 の初期状態 $s^1$ で $SP_1^1(s^1) = false$ かつ $SP_2^1(s^1) = true$、プロセス 2 の初期状態 $s^2$ で $SP_1^2(s^2) = true$ かつ $SP_2^2(s^2) = false$ とする。プロセス 1 と 2 は通信を行なわないとする。プロセス 1 の次のイベントを $e^1$、プロセス 2 の次のイベントを $e^2$ とする。ここで **D** が $e^1$ を動作させたとする。この場合にはプロセス 1 で $e^1$ 以降ずっと $SP_1^1 = SP_2^1 = false$ であり、プロセス 2 では $SP_2^2(e^2) = true$ となる場合が存在する。このとき $G(P)$ には $(s^1, e^2)$ が含まれ、かつ、プロセス 1 の $e^1$ 以降の状態を含む全域状態は $G(P)$ に含まれないので $G(P)$ で停止不可能である。

また、**D** が $e^2$ を動作させたとする。この場合にはプロセス 2 で $e^2$ 以降ずっと $SP_2^1 = SP_2^2 = false$ であり、プロセス 1 では $SP_1^2(e^1) = true$ となる場合が存在する。このとき $G(P)$ には $(e^1, s^2)$ が含まれ、かつ、プロセス 2 の $e^2$ 以降の状態を含む

全域状態は$G(P)$に含まれないので$G(P)$で停止不可能である。従って、1度の再演で$G(P)$で停止できるデバッガは存在しない。 □

また、再演を2回行なうことを許せば、任意の大域的条件式について、$Inf(P)$で停止することは可能である。これは、1回目の実行において、各プロセスのイベントの因果関係と、各状態での単純条件式の真偽の値をファイルなどに保存し、それを解析するという実行後検出の方法で$Inf(P)$を求めた後、2回目の実行でその地点で停止すればよい。

### 4.4 トレース機能

多くの逐次型プログラムのデバッガは以下のような形式のトレース機能を持つ。

trace [condition] print [expression]

このコマンドの意味は、[condition]に書かれた条件が成立した時に[expression]で書かれた式の値を出力し、動作を続行するというものである。ここで[expression]にはプログラムの変数、プログラムカウンタなどから成る式が書かれる。分散プログラムのデバッガにおいては、この[condition]に大域的条件の記述を許し、[expression]には複数のプロセスのプログラム変数、プログラムカウンタなどから成る式を許すように拡張することが考えられる。そして、その意味は大域的条件$P$が与えられた時、$Inf(P)$の全域状態における値を出力することとする。

条件式$P$がCPである場合には、トレースは停止アルゴリズムを使うことで実現可能である。プロセスが$Inf(P)$で停止すると、デバッガは[expression]に登場する変数などの現在の値を読み出して[expression]の値を計算し、その結果を出力した後動作を続行すればよい。

条件式$P$がDPである場合には、前節で述べたように、$Inf(P)$で停止することは不可能である。しかし、トレースは式の値を出力するだけであるので、実行中に、式に登場する変数の値を保存しておくことにより、$G(P)$のある状態において、$Inf(P)$の時の式の値を出力することが可能である。

そして、大域的条件成立の実行後検出の場合にはログは実行終了まで保存する必要があるが、それではログのサイズが大きくなるので、保存した値は不要になった時点で破棄することが望ましい場合も存在する。本節では、条件式$P$がDPであるトレースコマンドの処理を述べる。その特徴は以下の通りである。

- $G(P)$を満足する状態において$Inf(P)$の時の式の値を出力する。
- トレースのために保存した値は使われることがないと判明した最初の時点で破棄する 。

$LD_i$ におけるトレース処理のアルゴリズムの概要を以下に述べる。$P$ を $SP_{j_1} \cup SP_{j_2} \cup \ldots \cup SP_{j_k}$ とする。$e_i$ をプロセス $i$ のあるイベントとする。$i$ が $e_i$ を実行するために各プロセスが最小限の実行を行なった時点の全域状態を $Inf[e_i]$ と書く。すなわち、$Inf[e_i]$ は条件式 $SP_i$ が $(s_i = e_i)$ である場合の $Inf(SP_i)$ となる全域状態である。

$LD_i$ は $N$ 個の仮想的プログラムカウンタ $inf_i[1..N]$ を持つ。$i$ が最後に実行したイベントを $e_i$ とすると、$inf_i[1..N]$ は常に、$(inf_i[1], inf_i[2], \ldots, inf_i[N]) = Inf[e_i]$ を満足するようにする。ここで、仮想的プログラムカウンタは実際のプログラムカウンタのシミュレートではなく、実行したイベントの数を数えておくことで実現できる。仮想的プログラムカウンタの管理を行なうアルゴリズムは後述する。このとき、プロセス $i$ で $SP_i$ が真になった時、$(inf_i[1], inf_i[2], \ldots, inf_i[N]) = Inf(SP_i)$ となる。

トレースコマンドの [expression] に登場する $i$ の変数を $x^i_1, x^i_2, \ldots, x^i_{m_i}$ とする。プロセス $i$ が初期状態もしくは送信イベントを実行した直後の状態で $LD_i$ はその時の $x^i_1, x^i_2, \ldots, x^i_{m_i}$ の値を $inf_i[i]$ の値とともに内部に保存する。受信イベントもしくは内部イベントの直後の値は必要ではない。それは、受信イベントもしくは内部イベント $e_i$ は $Inf(SP_j)$ $(j \neq i)$ に含まれることはないからである。$SP_i$ が真になると、$LD_i$ は MD に $SP_i$ が真になったことを、現在の $inf_i[1..N]$ の値および $x^i_1, x^i_2, \ldots, x^i_{m_i}$ の値とともに送る。MD がこのメッセージを受信すると各 $LD_j$ に対し、$inf_i[j]$ の時点の $x^j_1, x^j_2, \ldots, x^j_{m_j}$ の値を送るように要求する。その返答を受け取った後、MD は $Inf(SP_i)$ における [expression] の値を計算して出力する。それが終了すると MD は全プロセスに動作を継続するよう指令を出す。

$inf_i[1..N]$ の値を正しいものに保つために $LD_i$ は送信メッセージに現在の $inf_i[1..N]$ をのせて送る。受信イベントを実行する際には、受け手プロセス ($j$ とする) はメッセージから $inf_i[1..N]$ を取り出して $inf_j[k] = max(inf_j[k], inf_i[k])(k = 1, 2, \ldots, N)$ を計算する。このようにすると常に $(inf_i[1], inf_i[2], \ldots, inf_i[N]) = Inf[e_i]$ となるのは明らかである。

次に、不要になった値の破棄について考える。現在の $inf_i[j]$ の値が $a$ であるとすると、今後 $SP_i$ が真になる時の $inf_i[j]$ の値は $a$ 以上である。従って、$inf_j[j] < a$ を満足する時点の $x^j_1, x^j_2, \ldots, x^j_{m_j}$ の値は今後 $SP_i$ に関しては使われることがない。従って、ある $a$ に対して、全ての $i$ について $inf_i[j] \geq a$ が成立すれば、$inf_j[j] < a$ を満足する時点の $x^j_1, x^j_2, \ldots, x^j_{m_j}$ の値をプロセス $j$ は破棄することができる。逆に、ある $i$ に対して $inf_i[j] < a$ であれば、$SP_i$ が次に真になる時点の $inf_i[j]$ の値は $a$ 未満である可能性があり、$a$ 未満の時点の値を使う可能性があるためにプロセス $j$ はこ

の値を破棄できない。

従って、古い値の破棄のため、$inf_i[j]$ の値が変化すると、$LD_i$ はその新しい値を $LD_j$ に送る。その値をもとに、各 $LD_i$ は保存している値が必要がどうかを判断し、破棄を行なう。アルゴリズムは図17に示す。

### 4.5 プロトタイプデバッガの試作

本アルゴリズムの実際のデバッグでの有効性を確認するため、プロトタイプデバッガ ddbx-p の試作を行なった。

試作上の制約は以下の通りである。

- 対象言語は C、OS は UNIX 4.2BSD 以上。
- プロセスの動的な生成 / 消滅は行なわない。
- 通信路の動的な生成 / 解放は行なわない。

各プロセスは異なる計算機上に存在してもよい。再演のため、システムが提供する通信プリミティブのみを使用してプログラムを作成する必要がある。ddbx-p では、通信相手の指定方法として、プロセス指定とポート指定の 2 種類を用意している。プロセス指定は通信相手のプロセスに与えられた番号を陽に指定するものである。また、ポート指定は分散アルゴリズムの理論モデル (例えば [27])、すなわち、各プロセスからは通信ポートのみが見え、全プロセス間のトポロジーはプロセスに情報として与えられない限り不明というモデルに従った通信方法である。

システムが提供するプリミティブは以下の通りである。

- `int msinit(&argc, argv)` : 初期化処理
- `int mssnd(process/port, message)` : 指定したプロセス / ポートへの送信
- `int msrcv(mode, process/port, message)` : 指定したプロセス (ポート) から受信
- `int msrcva(mode, message)` : 任意のプロセス (ポート) から受信
- `int msrcvs(mode, process/port_set, message)` : 指定したプロセス (ポート) 集合の中から受信
- `int msterm()` : 終了時処理

```
program SelectiveTracing; /* LD_I のプログラム:HaltAtBreakPoint と同じ変数 / 手続きは省略 */
const  Varlist = ...; /* 出力する式の中の I の変数のリスト */
type  external_event_type = ( MessageArrival. Control-MessageArrival,
    CounterMessageArrival, ExecFinish, OutputMessageArrival, RemoveMessageArrival );
  remove_message = record dest, soc : process; pc : message_no end;
    /* 保存した値の破棄要求 */
  counter_message = record  dest, soc : process;
    inf : array [process] of message_no  end; /* 他プロセスの inf */
var  inf : array [process] of message_no ;
  waitcp : process; /* waitcp からの CounterMessage 待ち */
  opc : message_no; /* 出力命令メッセージ */
  rem : remove_message; /* 破棄指令メッセージ */
  ctm : counter_message; /* カウンターメッセージ */
procedure StoreV(vlist : list_of_value, pc : message_no);
  begin /* vlist と pc をローカルに保存 */ end;
procedure RemRec(rem : remove_message);
  begin /* プログラムカウンタが rem.pc よりも小さい値は rem.soc には
  '不要' の印をつける。全ての predicate について '不要' の印がついた値は破棄 */ end;
function ReadV(counter : message_no) : list_of_value;
  begin /* counter の時点の変数の値をローカルから読み出す */ end;
procedure SendV(vlist : list_of_value); begin /* 変数の値を MD に返答。 */ end;
procedure InitOut(vlist : list_of_value, inf : array [process]  of message_no);
    begin /* MD に SP が真になったことを vlist と inf とともに報告 */ end;
function ReadOM : message_no; begin /* 変数の値を出力せよという命令を受信 */ end;
procedure SendRM(rem : remove_message); begin /* 破棄指令メッセージを送信 */ end;
function ReadRM : remove_message; begin /* 破棄指令メッセージを受信 */ end;
procedure InsertQueueCTM(ctm : counter_message);
  begin /* カウンタメッセージ (inf) をキューに保存 */ end;
function ReadQueueCTM(i : process) : counter_message;
  begin /* i からの最初のカウンターメッセージを読む。存在しなければ Null を返す */ end;
```

図 17: $LD_I$ のトレースアルゴリズム

```
procedure Maxinf(rinf : array [process] of message_no,
    var inf : array [process] of message_no); /* inf の最大値を求める */
var  i : process; rem : remove_message;
begin for i := 1 to N do /* inf の値の更新を行なう */
  if rinf[i] > inf[i] then begin inf[i] := rinf[i];
    if (SPExist) then /* inf の値が変化したので破棄指令を送る */
      begin rem.soc := I; rem.dest := i; rem.pc := inf[i]; SendRM(rem) end
    end
end;
begin /* MAIN */
  for i := 1 to N do begin s[i] := 0; r[i] := 0; ms[i] := 0; inf[i] := 0 end;
  waitp := Null ; waitcp := Null ;
  StoreV(GetV(Varlist), inf[I]); /* 初期状態の変数の値を保存 */
  cstat := TermTest(s, ms, SP);
  if cstat =passive and (SPExist) then InitOut(GetV(Varlist), inf);
  if cstat =active or (SPExist) then TryExec(r, waitp);
  repeat Wait(extevent); /* 外部イベント待ち */
    case extevent of
      ExecFinish: /* 被デバッグプログラム 1 ステップ終了 */
        begin  LastEvent(e, i);
          if e = receive then begin r[i] := r[i] + 1;
            ctm := ReadQueueCTM(i); /* inf の読み込み */
            if ctm = Null then waitcp := i else Maxinf(ctm.inf, inf)
            end
          if e = send then begin s[i] := s[i] + 1;
            inf[I] := inf[I] + 1; ctm.soc := I; ctm.dest := i; ctm.inf := inf;
            SendCTM(ctm); /* inf の値を送信 */
            StoreV(GetV(Varlist), inf[I]); end; /* 送信直後の変数値保存 */
          if waitcp = Null then begincstat := TermTest(s, ms, SP);
            if cstat =passive and (SPExist) then InitOut(GetV(Varlist), inf);
            if cstat =active or (SPExist) then TryExec(r, waitp)
            end
        end; /* End of ExecFinish. */
```

図 18: $LD_I$ のトレースアルゴリズム (続き)

```
    ControlMessageArrival: /* コントロールメッセージ到着 */
      begin cm := ReadCM; /* 読み込み */
        if not (SPExist) then begin
          ms[cm.soc] := cm.mno; /* 単純条件式を持たないプロセスのみ送信命令有効 */
          if waitcp = Null then begin
            bstat := cstat; cstat := TermTest(s, ms, SP);
            if bstat =passive and cstat =active then TryExec(r, waitp)
            end
          end
      end; /* End of ControlMessageArrival. */
    MessageArrival: /* 被デバッグプログラムメッセージ到着 */
      begin  m := ReadPM; /* メッセージ読み込み */
        if waitp ≠ m.soc then InsertQueue(m) /* 受信待ちのものでない */
          else begin  waitp := Null; ExecNext(m) end
        end; /* End of MessageArrival. */
    CounterMessageArrival: /* inf メッセージの到着 */
      begin  ctm := ReadCTM; /* メッセージ読み込み */
        if waitcp ≠ ctm.soc then InsertQueueCTM(ctm) /* 受信待ちのものでない */
          else begin waitcp := Null;
          Maxinf(ctm.inf, inf); /* max を取るとともに、破棄要求送信 */
          cstat := TermTest(s, ms, SP);
          if cstat =passive and (SPExist) then InitOut(GetV(Varlist), inf);
          if cstat =active or (SPExist) then TryExec(r, waitp)
          end
      end; /End of CounterMessageArrival. */
    OutputReqArrival: /* 出力要求到着 */
      begin opc := ReadOM; SendV(ReadV(opc)) end ;
    RemoveReqArrival: /* 破棄要求到着 */
      begin rem := ReadRM; RemRec(rem) end;
  end  /* End of case. */
 unitl Forever
end.
```

図 19: $\mathrm{LD}_I$ のトレースアルゴリズム (続き)

ここで、 mode はWAITMODE あるいはNOWAITMODE という値をとり、メッセージが受信プロセスにまだ届いていなかった場合、WAITMODE の時にはメッセージが届くまでブロックされ、NOWAITMODE の場合はすぐに error を返す。

msinit, msterm はそれぞれ、通信の初期化処理と終了時処理を行なうルーチンであり、それぞれプログラムの最初と最後におく必要がある。また、プログラム中では、整数型変数 prno(自分のプロセス番号)、 tnpr(全プロセス数: プロセス指定の場合)、 degree(接合しているポートの数: ポート指定の場合) が使用可能である。

コンパイル時のオプションを変更することにより、通常実行、モニタ実行、再演実行を行なう各実行プログラムを得ることができる。また、実行プログラムを起動するには init と呼ぶプログラムを使用することが必要である。 init は、複数計算機上で動作させる場合と単一計算機上で動作させる場合で異なる。ここでは、単一計算機上で動作させる場合のみを記す。この時は、

```
init [programfile] [networkfile]
```

を実行する。第一引数は実行プログラム名を記述したファイルの名前である。そのファイルの行数は起動するプロセスの数、各行の内容は各プロセスで実行させるプログラムを表す。[programfile] のファイルの内容が以下のようであったとすると、

```
a.out0 arg01 arg02
a.out1 arg11 arg12
a.out2 arg21 arg22
a.out3 arg31 arg32
```

プロセスを 4 つ生成し、それらに 0、 1、 2、 3 というプロセス番号を与え、プロセス 0 はa.out0 arg01 arg02 (arg01、 arg02 は a.out0 の引数である) を実行する。プロセス 1、 2、 3 はそれぞれ 2、 3、 4 行目に書かれたプログラムを実行する。ここで、モニタ実行、再演実行を行なうプログラム名を記述しておくと、それぞれモニタ実行、再演実行を行なうことができる。このときのプロセス番号はプロセス間通信の際に用いる番号であり、実際の UNIX の pid とは異なる。第二引数はポート指定の通信を行なう場合にのみ用いられる、隣接行列を表したファイルの名前である。 [networkfile] のファイルの内容が以下のようであったとする。

```
0 1 0 1
1 0 1 0
0 1 0 0
1 0 0 0
```

1行目はプロセス0がどのプロセスとの間に通信路が存在するかを表す。値が1の場合に通信路は存在する。この例では、プロセス1と3との間に通信路が存在する。従って、プロセス0はdegreeが2であり、2つのポートを持つ(但し、各ポートがどのプロセスと接続されているかは各プロセスには情報として与えられない)。2、3、4行目は同様に、プロセス1、2、3がどのプロセスと接続しているかを表す。例えばプロセス2はdegreeが1であり、プロセス1とのみ接続されている。

モニタ実行を行なった後、再演デバッガを起動する。その場合のコマンドは以下の通りである。

```
ddbx [programfile] [networkfile]
```

引数の意味はinitの場合と同じである。プロセス起動終了後、プロンプトを出力してデバッガのコマンド待ちの状態になる。

ddbx-pの主要なコマンドを図20に示す。前節までに述べたように、[global predicate]がDPである場合には、$Inf(P)$で停止させることは不可能である。従って、一般にstopの条件式に一般のand/orからなる条件式が与えられたとき、および、複数のstop命令が与えられたとき、そのうちのただ一つのCPに対してのみ、$Inf(P)$で停止させることが可能であり、それ以外の条件式については、条件式を成立したことを(あとから)検出して停止させることしかできない。この、ただ一つのCPをprimaryと呼ぶ。primaryを変更する命令が、switchである。stop, traceコマンドを入力した時に、番号(commandno)が割り振られる。switchは指定した[commandno]のコマンドの[predicateno]番目のCPをprimaryとする。primary predicateにはstatusコマンドを実行した時に*印がつけられる。デフォルトでは最初に設定したコマンドの最初のCPがprimaryとなる。

この試作したddbx-pの評価について述べる。まず、履歴保存のための手間の増加について文献[46]と同様の測定を行なった。測定を行なった計算機はSUN4-260(OSはSUNOS-4.0.3、メモリ64MB)である。通常実行において`NOWAITMODE`のmsrcvを行なった場合、1回あたりのCPU timeは約153$\mu$sとなった。それに対し、履歴保存実行を行なった場合、1回あたりのCPU timeは約196$\mu$sであった。この差約43$\mu$sが履歴保存のための実行時間の増加分である。msrcva、msrcvsの場合も、保存するデータ量および保存手続きは同一であるので、実行時間の増加は受信命令でほぼ同一であった。従って、メッセージ通信の頻度が4～5ミリ秒に1回であれば、履歴保存のために実行時間の増加割合が1%程度とあまり大きくない。

また、1回の受信命令ごとに保存される履歴情報は数バイト(大抵の場合、2～3バイト)である。従って、受信命令ばかりを繰り返すプログラムであっても、1秒あ

- run　：再演の開始
- cont　：再演の続行
- rerun：再演をもう一度最初から開始
- stop if [global condition]：ブレークポイント設定

  [global condition]::= [conjunctive condition] |
  　　[conjunctive condition] or [global condition]
  [conjunctive condition] ::= [simple condition] |
  　　[simple condition] and [conjunctive condition]
  [simple condition] ::= [processno] : [local condition]
  [local condition]　::= [boolean expression] | at [lineno]

  ここで、[processno] はプロセス番号、
  [boolean expression] はプログラムの変数などに関する論理式、
  [lineno] はプログラムの行番号である。
  (例) stop if 2:i>j and 5:x=0 or 2: at 110
- trace [process expression] if [global condition]：トレースの設定

  [process expression] ::= [processno] : [local expression] |
  　　[processno] : [local expression] , [process expression]
  [local expression] ::= [expression] | source

  source は、その地点のソースプログラムを意味する。
  (例) trace 4: source, 2:k if 4:x=0 and 2:y=0
- status　：設定された stop, trace コマンド表示
- delete [commandno]：stop, trace コマンド取り消し
- switch [commandno], [predicateno]　：primary の変更
- print [process expression]：式の値の出力
- list [process lineno]：ソースプログラム表示

  [process lineno] ::= [processno] : [lineno] , [lineno]
- quit　：デバッガの終了

図 20: ddbx-p の主要なコマンド

たり 10 数 K バイトの履歴情報を保存するだけで済む。従って、数十分程度の履歴を保存することも可能である。

また、ddbx-p をグループ内での分散プログラム作成において実際に試用し、評価を受けた。主な意見を以下に記す。

- 一般に、プロセス間通信としてソケットなどを用いる場合、プロセス間通信のために多くの手続きを記述する必要があるが、ここで提供している通信プリミティブは簡単に使用できる。その代わりに、デバッグをしない通常実行のみである場合にも init を用いてプロセス起動を行なわなければならないという欠点を持つ。

- 最小限実行を行なった地点で全プロセスを停止できる機能は、あるメッセージの送受信のあたりに誤りがあり、送信側と受信側のどちらにバグがあるかわかっていない場合に特に有効である。このとき、受信時点にブレークポイントを設定すると、送信側プロセスは対応する送信直後の状態で停止するので、送信側と受信側を対比して見ることができる。

- 本デバッガはプロトタイプ版であるため、プロセスの制御には UNIX 上のデバッガである dbx を使用している。従って dbx と対話的処理を行なうために実行速度は遅い。例えば、ブレークポイントを 1 つ設定して動作させた場合でも、通常実行の場合と比べて実行速度は 10 倍程度遅くなる。ブレークポイントの数が増える、あるいは式が複雑になると速度はさらに遅くなる。

- 全プロセスをデバッガが管理する必要があるため、デバッグ対象プロセスと非対象プロセスとが混在した分散プログラム (例えば、デバッグ済みのサーバを使用するクライアントプログラム) のデバッグが不可能である。

また、分散プログラムのデバッグの効率向上のためには、動作状況のユーザへの提示方法も重要である。メッセージの流れに沿ったデバッグ手法、すなわち、ユーザの着目プロセスをメッセージ通信に応じて送り手 / 受け手プロセス間で移動する手法により、分散プログラム動作状況の理解が容易になり、デバッグの効率が向上すると考えられる [1]。

プロセスが動的に生成 / 消滅する、および通信路を動的に生成 / 解放する分散プログラムに対しては、通信プリミティブおよび、再演 / 停止アルゴリズムに対して少しの変更を加えることにより同様の手法が実現可能である [35]。

## 4.6 むすび

本章では、分散プログラムのデバッガに対する基本機能のうち、大域的条件を用いたブレークポイント設定およびトレース機能について検討した。主な結果は以下の通りである。

1. 大域的条件 $P$ が積条件式である場合には、$P$ が成立する最初の全域状態 $Inf(P)$ で停止することが可能であり、$Inf(P)$ で停止するアルゴリズムを示した。

2. 大域的条件 $P$ が和条件式である場合には、$P$ が成立する最初の全域状態 $Inf(P)$ で停止することが不可能であり、$P$ が成立するある全域状態で停止することのみが可能である。また、一般の大域的条件式 $P$ に対しては、$P$ が成立している状態で停止することも不可能であることを示した。

3. 大域的条件 $P$ が積条件式・和条件式のいずれの場合にも、$P$ が成立しする時点で $Inf(P)$ の全域状態における各プロセスの変数などからなる式の値を出力するトレース機能が可能であり、それを実現するアルゴリズムを与えた。

今後検討するべき課題としては、時制論理などを用いたさらに複雑な大域的条件式の有効性とその実現可能性について考察することが考えられる。また、本稿では非決定的な動作を行なう分散プログラムの一つの動作をテストする問題を考察したが、ある一つのタイミングの動作で誤りが検出されなかったとしても、非決定性のため、別のタイミングの動作には誤りが存在する可能性がある。従って、誤りがないことを保証するには、さまざまなタイミングの動作に対して同一のテストを行なう必要がある。そこで、異なるタイミングの動作を自動的に生成し、かつその網羅性を保証するツールが存在すれば、そのツールが生成する各動作に対して本デバッガなどを用いたテストを行なうことにより、誤りのないことを保証できる。誤りのないことを保証するためには、上記のタイミング自動生成ツールも必要であると考えられる。

# 5 結論

本研究では、分散システム構築のための要素技術のうち、通信プロトコルのうちルーティングに関するアルゴリズム、および分散プログラム開発支援のうちデバッグに関するアルゴリズムについて考察した。本論文で示した結果を要約すると次のようになる。

1. 第2章では、経路情報の安全性の問題に関し、以下の結果を示した。

   中継のあるネットワークにおいて、ソースルーティングを行なう場合に、頂点におけるトラフィック解析を防止するため、経路情報を計算するセンタの存在を仮定した場合に

   - 中継頂点には中継に必要な情報のみを与え、それ以外の経路情報を与えない
   - 送り手頂点に、受け手頂点および経路についての情報を与えない (匿名性)

   を満足する、経路情報の暗号化の2つの手法を示した。

   一方の手法は頂点の共謀がない場合にのみ有効で、従来の方法と計算量はオーダ的に同じである。もう一方の方法は計算量は大きくなるが、頂点の共謀がある場合にも有効な手法である。

2. 第3章では、耐故障性および効率が準最適なルーティングを求める問題に関し、以下の結果を示した。

   (a) $k$- 連結な有向グラフ $G(k \neq 2,4)$ が $|F| < k$ を満たす任意の $F$ に対して $D(R(G,\rho)/F) \leq 6$ を満足するルーティング $\rho$ を持つための十分条件。

   (b) $gcd(n,d) = 1, d \geq 2$ を満たす任意の $n$ と $d$ に対して、$\xi(G,\rho_1) < n\log_d n$ を満足する、頂点数 $n$、最大次数 $d$ の有向グラフ $G$ とその上でのルーティング $\rho_1$ の組を構成する方法。

   (c) $d \geq 2$ を満たす任意の $n$ と $d$ に対して、$\xi(G,\rho_2) < 2n\lceil\log_d n\rceil$ を満足する、頂点数 $n$、最大次数 $d$ の有向グラフ $G$ とその上でのルーティング $\rho_2$ の組を構成する方法。

   (d) $d \geq 3$、$n > d^4$ および $gcd(n,d) = 1$ を満たす任意の $n$ と $d$ に対して $\xi(G,\rho_1') < 2n\log_d n$ を満足し、かつ、$|F| < d-1$ を満たす任意の $F$ に対して $D(R(G,\rho_1')/F) \leq 3$ を満足する、頂点数 $n$、最大次数 $d$ の有向グラフ $G$ とその上のルーティング $\rho_1'$ の組を構成する方法。

(e) $d \geq 3$ および $n > d^4$ を満たす任意の $n$ と $d$ に対して、$\xi(G, \rho_2') < 3n\log_d n$ を満足し、かつ、$|F| < d-1$ を満たす任意の $F$ に対して $D(R(G, \rho_2')/F) \leq 3$ を満足する、頂点数 $n$、最大次数 $d$ の有向グラフ $G$ とその上のルーティング $\rho_2'$ の組を構成する方法。

3. 第4章では、非決定的な動作の再演を行なう分散プログラム用デバッガにブレークポイント設定機能およびトレース機能を導入する問題に関し、以下の結果を示した。

まず、大域的条件式を満足する地点で停止するブレークポイント設定機能については、

(a) 積条件式については、条件式を満足する最初の地点で全プロセスを停止することが可能である。

(b) 和条件式については、最初の地点で全プロセスを停止することは不可能であるが、条件を満足するある地点で停止することは可能である。

(c) 一般的な大域的条件式については、条件式を満足するある地点で停止することも不可能である。

また、大域的条件式を用いたトレース機能については、積条件式および和条件式のいずれについてもトレースが可能であることを示した。

そして、停止およびトレースが可能なおのおのの場合について、その機能を実現するアルゴリズムを示した。

今後の課題として残された問題は数多くあるが、通信プロトコルについては、センタでルーティングを決定するのではなく、ネットワーク上で協調してルーティングを求める場合において、経路情報の安全性、高い耐故障性、高い効率を達成する方法が必要であると思われる。また、分散プログラムのデバッグについては、開発者が想定していなかったタイミングの実行を自動的に生成することによって、誤りを顕在化させる手法が重要であると思われる。

本論文の結果は、分散処理システムを構築するにあたっての基礎となるものである。実際の分散処理システム構築にあたっては、そのアプリケーションに依存した各種の解決すべき問題があると考えられるが、本論文の成果が役立つものと確信している。

謝辞

本研究をまとめるに際し、懇切なる御指導並びに御助言を頂いた大阪大学基礎工学部都倉信樹教授に心から感謝の意を表します。また、御助言と励ましを頂いた大阪大学基礎工学部橋本昭洋教授、宮原秀夫教授、菊野亨教授に厚く感謝致します。

本研究は、日本電信電話株式会社基礎研究所における業務の一環として担当したものであり、この間御指導、御鞭撻頂いた国際電気通信基礎技術研究所岡田桂治博士(もと情報通信基礎研究部第五研究室長)、ソフトウェア研究所市川晴久グループリーダ、ソフトウェア研究所武末勝主幹研究員、基礎研究所情報科学研究部尾内理紀夫グループリーダに心から御礼申し上げます。

また、本研究の過程で、貴重な御意見を頂いた大阪大学基礎工学部萩原兼一助教授、増澤利光助教授および名古屋工業大学和田幸一助教授に感謝致します。

さらには、本研究を進めるに当たり、有益な討論、協力を頂いた日本電信電話株式会社ソフトウェア研究所ソフトウェア基礎技術研究部今瀬真グループリーダ、総合企画本部曽根岡昭直担当課長(もとソフトウェア開発技術研究部主任研究員)、基礎研究所情報科学研究部青柳滋己社員に心から感謝致します。

本研究の遂行、本論文の執筆は以上の方々をはじめとする関係各位の多くの方々により支えられてきました。ここに心から感謝の意を表します。

## 参考文献


[1] 青柳、真鍋:"分散プログラミングのためのデバッグ手法の一提案", 情報科学若手の会シンポジウム (1991-07).

[2] Baran, P., "On Distributed Communications: IX. Security, Secrecy, and Tamper-Free Considerations," RM-3765-PR, The Rand Corp., Santa Monica, Calif. (Aug. 1964).

[3] Beaver, D., Micali, S., and Rogaway, P.: "The Round Complexity of Secure Protocols," Proc. of 22nd Symp. on Theory of Computing, pp. 503-513(May 1990).

[4] Ben-Or, M., Goldwasser, S., and Wigderson, A.: "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation," Proc. of 20th Symp. on Theory of Computing, pp.1-10(May 1988).

[5] Bollobas, B.: "Extremal Graph Theory," Academic Press, London (1978).

[6] Broder, A., Dolev, D., Fischer, M., and Simons, B.: "Efficient fault tolerant routing in networks," Proc. of 16th Symp. on Theory of Computing, pp. 536-541 (May 1984).

[7] Chaum, D. L.: "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," Communications of ACM, Vol. 24, No. 2, pp. 84-88 (Feb. 1981).

[8] Chaum, D., Crepeau, C., and Damgard, I.: "Multiparty Unconditionally Secure Protocols," Proc. of 20th Symp. on Theory of Computing, pp.11-19(May 1988).

[9] Chung, F.R.K., Coffman, E.D., Reiman, M.I., and Simon, B.: "The Forwarding Index of Communication Networks," IEEE Trans. on Information Theory, Vol. IT-33, No. 2, pp. 224-232 (Mar. 1987).

[10] Carver, R. H., and Tai, K.: "Reproducible Testing of Concurrent Programs Based on Shared Variable", Proc. 6th Int. Conf. on Distributed Computing Systems, pp. 428-433(May 1986).

[11] Chandy, K. M., and Lamport, L.: "Distributed Snapshots: Determining Global States of Distributed Systems," ACM Trans. on Computer Systems, Vol. 3. No. 1, pp. 63–75(Feb. 1985).

[12] Choi, J.-D., Miller, B. P., and Netzer, R.: "Techniques for Debugging Parallel Programs with Flowback Analysis," Technical Report 786, Univ. of Wisconsin-Madison, Computer Science Department(Aug. 1988).

[13] Cohen, S., and Lehmann, D.: "Dynamic Systems and Their Distributed Termination," Proc. of 2nd Symp. on Principles of Distributed Computing, pp. 29–33(1982).

[14] Cooper, R.: "Pilgrim: A Debugger for Distributed Systems," Proc. of 7th Int. Conf. on Distributed Computing Systems, pp. 458–465(Sep. 1987).

[15] Cooper, R. and Marzullo, K.: "Consistent Detection of Global Predicates," Proc. of ACM Workshop on Parallel and Distributed Debugging, pp.167–174(May 1991).

[16] Denning, D. E. R.: "Cryptography and Data Security", Addison-Wesley Publishing (1982).

[17] Denning, A. and Schonberg, E.: "The task recycling technique for detecting access anomalies on-the-fly," Technical Report RC 15385 (68543), IBM T.J. Watson Research Center (Jan. 1990).

[18] Dijkstra, E. W., and Scholten, C. S.: "Termination Detection for Diffusing Computations," Information Processing Letteers Vol. 11, No.1, pp. 1–4(1980).

[19] Dolev, D., Dwork, C., Waarts, O., and Yung, M.: "Perfectly Secure Message Transmission," Proc. of 31st Symp. on Foundations of Computer Science, pp. 36–45(Oct. 1990).

[20] Dolev, D., Halpern, J.Y., Simons, B., and Strong, H.R.: "A new look at fault tolerant routing," Proc. of 16th Symp. on Theory of Computing, pp. 526-535 (May 1984).

[21] ElGamal, T.: "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Trans. on Information Theory, Vol. IT–31, No. 4, pp. 469–472 (July 1985).

[22] Fairley, R. E.: "Software Engineering Concepts," McGraw-Hill, pp. 288–289.

[23] Garcia-Molina, H., Germano, F. Jr., and Kohler, W.H.: "Debugging a Distributed Computing System" IEEE Trans. on Software Engineering, Vol. SE–10. No. 29, pp.210–219(Mar. 1984).

[24] Goldreich, O.: "Towards a Theory of Software Protection and Simulation by Oblivious RAMs," Proc. of 19th Symp. on Theory of Computing, pp.182–194(May 1987).

[25] Goldreich, O., Micali, S., and Wigderson, A.: "How to Play Any Mental Game," Proc. of 19th Symp. on Theory of Computing, pp.218–229(May 1987).

[26] Haban, D. and Weigel, W.: "Global Events and Global Breakpoints in Distributed Systems," Proc. of 21st Hawaii International Conference on System Sciences, pp. 166–175(Jan. 1988).

[27] 萩原兼一: "分散環境における問題解法のメッセージ複雑度について", Proc. of 8th Mathematical Programming Symposium, Japan, pp.41–50 (1987–11).

[28] Harary, F.: "Graph Theory," Addison-Wesley, Reading, MA, (1969).

[29] Imase, M. and Itoh, M.: "Design to Minimize Diameter on Building-Block Network," IEEE Trans. on Computers, Vol. C–30, No. 6, pp. 439-442(June 1981).

[30] Imase, M. and Manabe, Y.: "Fault Tolerant Routings in a $\kappa$-connected Network," Information Processing Letters, Vol. 28, pp.171-175 (July 1988).

[31] Imase, M., Soneoka, T., and Okada, K.: "Connectivity of Regular Directed Graphs with Small Diameter," IEEE Trans. on Computers, Vol. C–34, No. 3, pp. 267–273(Mar. 1985).

[32] Kawaguchi, K. and Wada, K.: "Optimal Fault- Tolerant Network Routings for (k+1)-connected Graphs," Univ. of Wisconsin-Milwaukee Computer Science Tech. Rep. UWM EE&CS 88-2 (Mar. 1988).

[33] Lamport, L.: "Time, Clocks, and the Ordering of Events in a Distributed System," Communications of ACM, Vol. 21, No. 7, pp. 558–565(July 1978).

[34] LeBlanc, T. J., and Mellor-Crummey, J. M.: "Debugging Parallel Programs with Instant Replay," IEEE Trans. on Computers, Vol. C-36, No. 4, pp. 471-480(April 1987).

[35] 真鍋、青柳:"再演をベースとした分散プログラムのデバッグ手法について"、情処研報 92-DPS-56-13 (1992-07).

[36] 満保、木下、辻井: "送受信者追跡の不可能な通信プロトコルに関する研究", 信学論 D-I, Vol.J74-D-I, No. 7, pp.429-434(1991-07).

[37] McDowell, C.E. and Helmbold, D.P.: "Debugging Concurrent Programs," ACM Computing Surveys, Vol. 21, No. 4, pp.593-622(Dec. 1989).

[38] Miller, B. P., and Choi, J.-D.: "Breakpoints and Halting in Distributed Programs," Proc. of 8th Int. Conf. on Distributed Computing Systems, pp. 316-323(June 1988).

[39] Mullender, S.: "Distributed Systems", Addison-Wesley (1989).

[40] Ostrovsky, R.: "Efficient Computation on Oblivious RAMs," Proc. of 22nd Symp. on Theory of Computing, pp. 514-523(May 1990).

[41] Pancake, C.M. and Utter, S.: "A Bibliography of Parallel Debuggers, 1990 Edition," ACM SIGPLAN Notices, Vol. 26, No. 1, pp.21-37(Jan. 1991).

[42] Peleg, D. and Simons, B.: "On Fault Tolerant Routing in General Graphs," Proc. of 5th Symp. on Principles of Distributed Computing, pp. 98-107 (Aug. 1986).

[43] Quaterman, J. S. and Hoskins, J. C.: "Notable Computer Networks," Communications of ACM, Vol. 29, No. 10, pp. 932-971(Oct. 1986).

[44] Ribest, R., Shamir, A., and Adleman, L.: "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," Communicaitions of ACM, Vol. 21, No. 2, pp.120-126(Feb. 1978).

[45] Soneoka, T., Nakada, H., and Imase, M.: "A Design of $d$-Connected Digraph with Minimum Number of Edges and Quasiminimal Diameter I," Discrete Applied Mathematics, Vol.27 pp.255-265(1990).

[46] 高橋直久: "データ共有型並列プログラムの部分再演法について", 信学技報 COMP89-98 (1989-12).

[47] 塚本、八田他: "分散処理技術," 情報処理 Vol.28, No.4 (1987-04).

[48] van der Waerden, B.L.: "Algebra," Springer-Verlag, Berlin, pp.63-65 (1964).

[49] Wada, K. and Kawaguchi, K.: "Efficient fault-tolerant fixed routings on (k+1)-connected digraphs," Discrete Applied Mathematics Vol. 37/38, pp. 539-552(1991).

[50] 和田、川口: "K+1 連結有向グラフに対する最適な耐故障性路線割当," 信学技報 COMP89-10(1989-05).

[51] Wilcov, R.S.: "Analysis and Design of Reliable Computer Networks", IEEE Trans. on Communication, Vol. COM-20, No. 6, pp. 660-678(June 1972).

## 著者の発表論文一覧表

* 印をつけたものは本論文に直接関連する発表論文である。

### 論文


1 真鍋、萩原、都倉: "一層一行配線問題の片側トラック数の最小性", 信学論 Vol. J67-A, No.5, pp.503-510 (1984-05).

2 真鍋、萩原、都倉: "超立方体環グラフと超立方体グラフの最小2分割幅", 信学論 Vol. J67-D, No. 6, pp.647-654 (1984-06).

3 真鍋、萩原、都倉: "樹枝状単調減少論理回路を実現するCMOS回路に対する最小分離度配置問題", 信学論 Vol. J68-D, No. 9, pp. 1571-1579 (1985-09).

4 曽根岡、真鍋、今瀬、井上:"高信頼度二重化ネットワークの最適構成", 信学論 Vol. J70-A, No. 2, pp.252-260 (1987-02).

*5 Y. Manabe, M. Imase, and T. Soneoka: "Reliable and Efficient Fixed Routings on Digraphs", Trans. of IEICE, Vol. E-71, No. 12, pp. 1212-1220 (Dec. 1988).

6 森保、曽根岡、真鍋:"分散システムにおける全域状態監視アルゴリズム", 信学論 Vol. J74-D1, No.4 pp.273-282 (1991-04).

*7 Y. Manabe, M. Imase: "Global Conditions in Debugging Distributed Programs", Journal of Parallel and Distributed Computing, Vol.15, No.1, pp.62-69(May 1992).

*8 Y. Manabe, M. Imase, and T. Soneoka: "Untraceable Route Information for Networks", Submitted to IEEE Trans. Communication.

9 T. Soneoka, M. Imase, and Y. Manabe:"Design of a d-connected digraph with a minimum number of edges and quasiminimal diameter II", to appear in Discrete Applied Mathematics.


### レター


1 M. Imase and Y. Manabe: "Fault Torelant Routings in a k-connected Networks", Information Processing Letters Vol. 28, pp.171-175 (July 1988).

国際会議

1 M. Imase, T. Soneoka, and Y. Manabe:"A Fault Tolerant Routing in Networks", ICIAM '87 (1987).

2 T. Soneoka, Y. Manabe, and M. Imase:"A Design of Reliable Networks Using Node Redundancy", 19th Southeastern Conference on Combinatorics, Graph Theory, and Computing(Feb. 1988), also in Congressus Numerantium Vol. 67 (Dec. 1988).

*3 Y. Manabe and S. Aoyagi:"Debugging Dynamic Distributed Prorams Using Global Predicates", 4th IEEE Symp. on Parallel and Distributed Processing, pp.402-407 (Dec. 1992).

全国大会

*1 真鍋、今瀬、曽根岡:"固定ルーチングにおける効率と信頼性", 昭63信学会春季全国大会 SD5-9 (1988-03).

*2 真鍋、今瀬、曽根岡:"ネットワークにおけるルート情報暗号化の一手法", 昭63信学会秋季全国大会 A-93 (1988-10).

3 青柳、真鍋:"分散プログラムデバッガのためのチェックポイントアルゴリズム", 情報処理学会第44回全国大会 2G-6(1992-03).

研究会

1 真鍋、萩原、都倉:"一層一行配線問題の片側トラック数の最小性について", 信学技報 CAS83-05 (1983-05).

2 真鍋、萩原、都倉:"超立方体環と超立方体グラフの最小2分割幅について", 信学技報 AL83-41 (1983-11).

3 真鍋、萩原、都倉:"単調減少論理関数を実現するCMOS回路に対する配置問題", 信学技報 CAS84-103 (1984-10).

4 曽根岡、井上、真鍋、今瀬:"二重化構造を持つトポロジーの信頼性", 信学技報 CAS85-92 (1985-10).

5 真鍋、能條:“通信網における故障形態と社会的影響との関係のモデル化”, 信学技報 IN86-16 (1986-06).

6 今瀬、真鍋:“ネットワークにおける障害耐力のある固定ルーチング方式について”, 信学技報 COMP86-70 (1987-01).

7 真鍋、今瀬、曽根岡:“分散アルゴリズムの通信計算量に対するトポロジー制限の影響”, 信学技報 COMP87-13 (1987-06).

*8 真鍋、今瀬、曽根岡:“グラフ上の固定ルーチングについて”, 信学会回路とシステム軽井沢ワークショップ pp.111-116 (1988-05).

9 森保、曽根岡、真鍋:“分散システムにおける同期誤り検出法”, 信学技報 IN88-152 (1989-03).

*10 真鍋、今瀬:“分散プログラムのデバッグにおける大域的条件について”, 信学技報 COMP89-99 (1989-12).

11 青柳、真鍋:“分散プログラミングのためのデバッグ手法の一提案”, 情報科学若手の会シンポジウム (1991-07).

*12 真鍋、青柳:“分散プログラム用デバッガ ddbx-p の試作開発”, 信学技報 IN91-105 (1991-09).

13 青柳、真鍋:“分散デバッガのためのチェックポイント・ロールバックアルゴリズム”, 日本ソフトウェア科学会第 10 回ソフトウェア研究会 (1992-04).

*14 真鍋、青柳:“再演をベースとした分散プログラムのデバッグ手法について”, 情処研報 92-DPS-56-13 (1992-07).

特許

*1 真鍋、今瀬:“分散プログラムデバッグ方式”, 特願平 2-45231 (1990-12).

2 青柳、真鍋:“疑似時計装置”, 特願平 4-29212 (1992-02).

*3 真鍋:“分散プログラムデバッグ方式”, 特願平 4-139559 (1992-06).