広く分散した動的な情報のための情報共有手法の最適化

八木 哲

NTT 未来ねっと研究所

本稿では，広く分散した多数の情報源において頻繁に変化する情報を観測し，その最新の情報を広く分散した多数の利用者に提供するための情報共有手法における，最適化について述べる．まず，情報共有手法の課題は次のとおりである．(a) 広く分散した多数の情報源に起因する，多量の更新処理．(b) 広く分散した多数の利用者に起因する，多量の参照処理．(c) 頻繁に変化する情報に起因する，高い頻度の更新処理．(d) 最新情報の共有．これらの課題に対処するために，我々は次の特徴を持つ情報共有手法を提案している．「利用者は情報の詳細を段階的に参照する」という参照パターンを前提条件として，(1) 自律型モジュールを分散配置し，配置した地域で観測された情報の管理を担当させる．(2) 各自律型モジュールは，観測された情報から詳しさの異る複数の要約を動的に生成し，利用頻度の高い要約だけを相互に複製して，観測された情報への索引として用いる．これらにより，参照処理と更新処理を各自律型モジュールに分散・局所化する．自律型モジュールを跨る更新処理の頻度を削減する．換言すれば，自律型モジュール間の通信頻度を削減する．しかし，我々の情報共有手法では自律型モジュールの間で要約が冗長に転送されることがある．この場合，自律型モジュール間の通信頻度を削減できないことがある．そこで，本稿では要約の冗長な転送を排除する方法を示す．

1 はじめに

広く分散した多数の情報源において頻繁に変化する情報を観測し，その最新の情報を広く分散した多数の利用者に提供できる仕組みがあれば，より利便性の高い広域情報案内サービスを構築できる．たとえば，道路交通情報 [1]・地域情報・気象環境情報 [2][3][4][5] を提供できる仕組みがあれば，デジタルシティ [6]・カーテレマティクス・歩行者 ITS[7] を融合させたような，計画の立案やドア・ツー・ドアの行動を支援するサービスを構築できる．ネットワーク情報 [8]・計算機資源情報を提供できる仕組みがあれば，広域ネットワーク監視サービス [9] やグローバルコンピューティング向けの計算機資源管理サービス [10] を構築できる．このような仕組みを，情報共有基盤と呼ぶことにする．情報共有基盤を実現するには，次の課題に対処できる情報共有手法が求められる．

(a) 広く分散した多数の情報源に起因する，多量の更新処理．

(b) 広く分散した多数の利用者に起因する，多量の参照処理．

(c) 頻繁に変化する情報に起因する，高い頻度の更新処理．

(d) 最新情報の共有．

従来，次のような情報共有手法が用いられている [10][11][12][13][14]．課題 (a) のために，自律型モジュールを分散配置し，配置した地域で観測された情報の管理を担当させる．これにより，観測された情報に対する更新処理を分散・局所化する．課題 (b) のために，各自律型モジュールは，観測された情報を互いに複製あるいはキャッシュする．これにより，観測された情報に対する参照処理を分散・局所化する．このときの課題 (c) は「複製やキャッシュに対する高い頻度の更新処理」である．課題 (c) のために，複製やキャッシュに対する更新処理の頻度を制限する．しかし複製やキャッシュが最新情報を保持し難くなるために，課題 (d) には対処し難い．情報に時刻印や有効期限を添付しておけば無効な情報を棄却できるが，課題 (d) に対処できるわけではない．

一方，我々は次の特徴を持つ情報共有手法 [15][16] を提案している．「利用者は情報の詳細を段階的に参照する」という参照パターンを前提条件として，(1) 自律型モジュールを分散配置し，配置した地域で観測された情報の管理を担当させる．(2) 各自律型モジュールは，観測された情報から詳しさの異る複数の要約を動的に生成し，利用頻度の高い要約だけを相互に複製

して，観測された情報への索引として用いる．これらにより，参照処理と更新処理を各自律型モジュールに分散・局所化する．自律型モジュールを跨る更新処理の頻度を削減する．

両手法の方針は，複数の自律型モジュールの使用と，自律型モジュール間の依存関係の削減である．換言すれば，自律型モジュール間の通信頻度の削減である．そこで「自律型モジュール間の通信頻度」を尺度として両手法を比較すれば，次のようである．我々の手法では，自律型モジュールの間で要約が冗長に転送されることがある．この場合，従来手法よりも「自律型モジュール間の通信頻度」を削減できないことがある．そこで，本稿では要約の冗長な転送を排除する方法を示す．まず，我々の情報共有手法の枠組 (2 章) と実現法の概要 (3 章) を示す．次に，要約の冗長な転送を排除するという問題をモデル化し (4 章)，解法を示す (5 章)．また，解法の効果を考察する (6 章)．最後に，本稿の内容をまとめ，今後の課題を示す (7 章)．

2　情報共有手法の枠組

前提条件とする「利用者は情報の詳細を段階的に参照する」という参照パターンの詳細を次に示す．

- 各利用者は，第 1 段階では大多数の情報の概観を参照する．第 2 段階では興味を持った情報の詳細を参照する．
- 各情報は，第 1 段階では大多数の利用者から概観を参照される．第 2 段階では興味を持った利用者から詳細を参照される．

これに基づいて，情報共有手法の枠組を示す．参照パターンの第 1 段階では，各情報の概観が参照の対象である．利用者が網羅的に参照するために，各情報の参照頻度は比較的高い．そこで，情報源において観測した頻繁に変化する情報から，次の特徴を持つ情報を動的に生成し，これを利用者に提供する．

(1) 観測した情報の特徴だけを反映する．
(2) 観測した情報と比べて時間的な変化が少ない．

この情報を ‘抽象化した情報’ と呼ぶことにする．‘抽象化した情報’ の複製を利用者のいる地域に配置することで，‘抽象化した情報’ に対する参照処理を分散・局所化する．‘抽象化した情報’ の前記 (1)(2) の特徴により，‘抽象化した情報’ の複製に対する更新処理の頻度を削減する．参照パターンの第 2 段階では，各情報の詳細が参照の対象である．利用者が選択的に参照するために，各情報の参照頻度は比較的低い．そこで，情報源において観測した頻繁に変化する情報を，そのまま利用者に提供する．この情報を ‘素な情報’ と呼ぶことにする．‘素な情報’ を情報源のある地域ごとに個別管理することで，‘素な情報’ に対する更新処理を分散・局所化する．‘抽象化した情報’ を索引として，参照する ‘素な情報’ を利用者に選択させることにより，‘素な情報’ に対する参照処理を分散させる．

このような枠組を持つ情報共有手法を課題 (a)(b)(c)(d) に照らせば，次のようである．課題 (a) のために，自律型モジュールを分散配置し，配置した地域の ‘素な情報’ の管理を担当させる．これにより，‘素な情報’ に対する更新処理を分散・局所化する．課題 (b) のために，各自律型モジュールは，‘素な情報’ から ‘抽象化した情報’ を生成し，これを相互に複製して，‘素な情報’ への索引として用いる．これにより，‘抽象化した情報’ に対する参照処理を分散・局所化する．‘素な情報’ に対する参照処理を分散させる．課題 (c)(d) のために，前記 (1)(2) の特徴を持つ ‘抽象化した情報’ を複製の対象とする．これにより，複製に対する更新処理の頻度を削減した上で，複製が最新情報を保持できるようにする．

3　情報共有手法の実現法の概要

3.1　構成

2 章で示した情報共有手法の実現には，‘抽象化した情報’ を生成し，これを自律型モジュールの間で複製するための，体系的な仕組みが必要である．ここでは，2 章で示した情報共有手法に基づいた情報共有基盤である WISE (Wide-area Information Sharing Engine) [16] の概要を示す．まず，WISE の構成を図 1に示す．地域ごとにセンサと RM (Resource Manager)

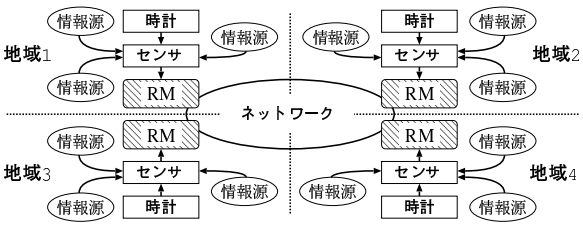

図 1: WISE の構成

がある．センサは，情報源において情報を観測し，観測した情報に時刻印を押して， RM に渡す． RM は，2 章で示した自律型モジュールに相当する．利用者は，近傍の RM を介して情報を参照する． RM の中核は，次に示す 3 つの機能要素である．データベースは，‘素な情報’ と ‘抽象化した情報’ を格納する．抽象化機能は，‘抽象化した情報’ を生成し，これを他の RM に複製する．このとき， RM が相互に複製する ‘抽象化した情報’ が持つ情報量と，前提条件とする参照パターンの第 1 段階に必要な情報量とが，一致しない場合がある．前者の方が少ない場合，利用者は近傍の RM が保有する ‘抽象化した情報’ に加えて他の RM が保有する ‘素な情報’ を参照する．この結果，参照処理を各 RM に分散・局所化できない．前者の方が多い場合， RM は利用されない ‘抽象化した情報’ を他の RM に複製する．この結果，複製に対する更新処理の頻度を削減できない．この問題に対応するために，学習機能は他の RM に複製すべき ‘抽象化した情報’ を参照頻度に基づいて選定する．次節以降では， RM の中核であるデータベース・抽象化機能・学習機能の概要を示す．

3.2 ‘素な情報’ と ‘抽象化した情報’ を格納するデータベース

まず，‘素な情報’ と ‘抽象化した情報’ を，次に示す単位データ d の集合として表す．

- $d = (g, t, a, s, v)$
  - g：地域． d が生成された地域を示す．識別名 [12] を用いる．
  - t：時刻． d の生成時刻を示す．単位時間ごとに単調増加するシーケンス番号を用いる．
  - a：抽象度．値が大きいほど抽象化されており，よりいっそうに，観測した情報の特徴だけを反映し，観測した情報と比べて時間的な変化が少ないことを示す．‘素な情報’ は 0．‘抽象化した情報’ は正の整数．
  - s：情報源． d を生成した情報源を示す． g に対する相対識別名 [12] を用いる．
  - v：値．

前提条件とする参照パターンに則して利用者が単位データを参照するように，次に示す WISE 情報モデル (IMW : Information Model for WISE) と呼ぶグラフ構造を d の集合に与える．

- $IMW = (N, AA, AG)$
  - N はノードの集合． $N = \{n_f \mid n_f = \{d_f \mid d_f$ は，地域が g_f，抽象度が a_f である単位データ$\}\}$．
  - AA は，ノードの抽象度の高低関係を表すアークの集合． $AA = \{(n_{ah}, n_{al}) \mid n_{ah}$ は，抽象度 ah の単位データを要素に持つノード． n_{al} は，抽象度 al の単位データを要素に持つノード． $ah > al$． $\}$．
  - AG は，ノードの地域の隣接関係を表すアークの集合． $AG = \{\ (n_{a1}, n_{a2}) \mid n_{a1}$ は，抽象度 $a1$ の単位データを要素に持つノード． n_{a2} は，抽象度 $a2$ の単位データを要素に持つノード． $a1 = a2$． $\}$．

グラフ構造を与えられた d の集合を， IMW のインスタンスと呼ぶことにする．図 2に例示する．この例では，抽象度 0・抽象度 1・抽象度 2 の単位データを要素に持つノードがある．抽象度の大きい単位データを要素に持つノードほど，大まかな地域情報を持つ．抽象化機能が，抽象度 0 の単位データから抽象度 1 以上の単位データを生成する (3.3 節参照)．利用者は，次のようにして単位データを参照する．

(1) 全地域を一瞥する．すなわち，ノードの地域の隣接関係を示すアークをたどりながら，抽象度 2 の単位データを参照する．

(2) 興味のある地域を選択する．すなわち，抽象度 2 の単位データを要素に持つノードのなかから，興味のある地域のノードを選択する．

(3) 選択した地域の概要を知る．すなわち，選択したノードから，ノードの抽象度の高低関係を示すアークをたどり，抽象度 1 の単位データを参照する．

(4) 選択した地域の詳細を知る．すなわち，さらにノードの抽象度の高低関係を示すアークをたどり，抽象度 0 の単位データを参照する．

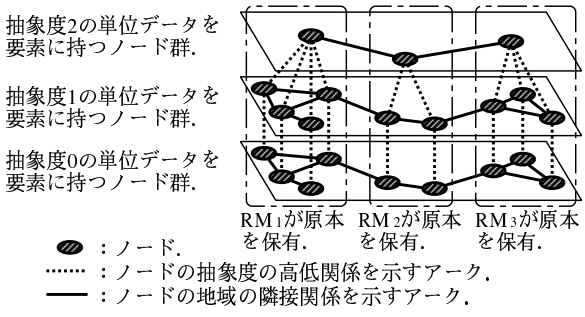

図 2: IMW のインスタンス

またこの例では，RM_1・RM_2・RM_3 が単位データを共有している．各 RM は，IMW のインスタンスのトポロジ情報を保有する．さらに，配置された地域のセンサから受け取った抽象度 0 の単位データと，その抽象度 0 の単位データから生成した抽象度 1 以上の単位データを，単位データの原本として保有する．他の RM が保有する単位データの原本に関しては，その RM への参照リンクと部分的に複製を保有する．たとえば RM_1 は，図 2左の 9 つのノードの要素である単位データの原本を保有する．図 2中央の 5 つのノードの要素である単位データに関しては，RM_2 への参照リンクと部分的に複製を保有する．図 2 右の 7 つのノードの要素である単位データに関しても同様である．RM_2・RM_3 の学習機能が，RM_2・RM_3 が保有する単位データの原本のなかから，RM_1 に複製すべき単位データを選定する (3.4 節参照)．

3.3 ‘抽象化した情報’ を生成する抽象化機能

抽象化機能の動作を説明する．基本となる動作は，ノード n_{al} の要素である抽象度 al の単位データから，ノード n_{ah} の要素である抽象度 $ah(> al)$ の単位データを生成する動作である．この動作は次の特徴を持つ．(1)n_{al} の要素である単位データが表す情報の特徴だけを，n_{ah} の要素である単位データが表す情報に反映させる．(2)n_{al} の要素である単位データの値の変化が，n_{ah} の要素である単位データの値に現れ難いようにする．以上の動作を，抽象化動作と呼ぶことにする．具体的には，次のような動作である．

- 精度削減型の動作：単位データの値の有効桁数を削減することで，単位データを生成する．あるいは，単位データの値を予め定めた種別に分類し，種別名を値とすることで，単位データを生成する．
- 要素選択型の動作：注目度の高い情報・一次判断に必要な情報・情報の概説・情報の定義情報など，特徴的な情報を表す単位データを選択することで，単位データを生成する．
- 要素集約型の動作：単位データの値の集合に対して統計関数・恣意的な関数を適用することで，単位データを生成する．

抽象化機能は，IMW のインスタンスに対して抽象化動作を行う．すなわち，抽象度 0 の単位データの更新を契機として，更新された抽象度 0 の単位データから，ノードの抽象度の高低関係を示すアークにより対応付けられた，抽象度 1 の単位データを生成する．生成した抽象度 1 の単位データから，ノードの抽象度の高低関係を示すアークにより対応付けられた，抽象度 2 の単位データを生成する．これを繰り返し，連鎖反応的に単位データを生成する．また，生成した単位データを他の RM に複製する．

3.4 複製すべき ‘抽象化した情報’ を選定する学習機能

学習機能の動作を説明する．学習機能は「自律型モジュール間の通信頻度」を尺度として，RM が保有する単位データの原本のなかから，他の RM に複製すべき単位データを選定する．RM_1 の学習機能は，単位データ d を RM_1 から RM_2 に複製するか否かを，次のようにして判断する．

- r は，RM_2 の近傍の利用者による d （RM_1 が保有する原本と RM_2 が保有する複製) の参照頻度．
- u は，RM_1 による d の更新頻度．
- r と u を用いて次のように判断する．厳密には通信プロトコルに依存した重み付けが r と u に必要であるが，ここでは単純化のために省略した．
  - $r > u$ の場合．同じ d が繰り返し転送されることを防ぐために，d を RM_1 から RM_2 に複製する．RM_2 の近傍の利用者は，RM_2 が保有する d の複製を参照する．
  - $r < u$ の場合．複製しても参照されない d の転送を防ぐために，d を RM_1 から RM_2 に複製しない．RM_2 の近傍の利用者は，RM_2 を介して RM_1 が保有する d の原本を参照する．

4 問題のモデル化

学習機能は，単位データごと，あるいはノードごとに，複製するか否かを判断する．従って，同じ抽象度 0 の単位データから生成された抽象度の異る複数の単位データが，同一の RM に複製されたり同一の RM を介して参照されることがある．このとき，抽象度の高い単位データは抽象度の低い単位データから生成可能であるという意味において，単位データは冗長に転送されている．しかし，特定の抽象度の単位データだけを複製し，より抽象度の高い単位データを複製先の

RM において生成すれば，単位データの冗長な転送を排除できる．以上の問題をモデル化する．

(1) IMW のインスタンスから，ノードとノードの抽象度の高低関係を示すアークから構成される木を抽出する．図 3に例示する．ここでは，RM_1 から RM_2 への単位データの冗長な転送を排除する場合を取り上げる．

(2) RM_1 において，RM_1 が保有する単位データの原本を要素に持つノードに，属性として ‘参照頻度’・‘更新頻度’・‘転送状態’・‘通信頻度’ を設ける．

(3) ‘参照頻度’ は，RM_2 の近傍の利用者による，ノードの要素である単位データ（RM_1 が保有する原本と RM_2 が保有する複製）の参照頻度．‘更新頻度’ は，RM_1 による，ノードの要素である単位データの更新頻度．‘転送状態’ は，‘参照’ か ‘複製’ か ‘生成’．‘通信頻度’ は，ノードの要素である単位データを RM_1 から RM_2 へ転送する頻度．

(4) ‘転送状態’ が ‘参照’ のとき，ノードの要素である単位データを RM_1 から RM_2 に複製しない．RM_2 の近傍の利用者は，RM_2 を介して RM_1 が保有する単位データの原本を参照する．‘通信頻度’ は ‘参照頻度’ に等しい．

(5) ‘転送状態’ が ‘複製’ のとき，ノードの要素である単位データを RM_1 から RM_2 に複製する．RM_2 の近傍の利用者は，RM_2 が保有する単位データの複製を参照する．‘通信頻度’ は ‘更新頻度’ に等しい．

(6) ‘転送状態’ が ‘生成’ のとき，このノードの子であるノードの ‘転送状態’ は，‘複製’ か ‘生成’ である．このノードの要素である単位データは，RM_2 において生成される．RM_2 の近傍の利用者は，RM_2 において生成された単位データを参照する．‘通信頻度’ は 0 である．

(7) ‘通信頻度’ の合計値を最小にする，各ノードの ‘転送状態’ を求める．

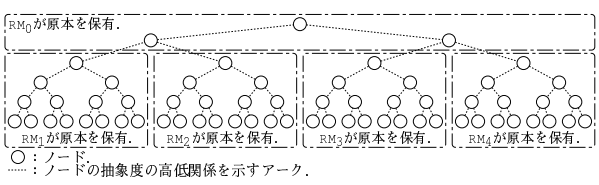

図 3: IMW のインスタンスから抽出した木

5 解法

5.1 概要

解法の概要を，単純な例題を用いて示す．図 4に例題を示す．‘参照頻度’ は，根から葉へ均一に減少している．すなわち，常に一定数の利用者が，根から葉へ均一に分散しながら単位データを参照している．‘更新頻度’ は，葉から根へ均一に減少している．すなわち，抽象化機能が，葉から根へ単位データの時間的な変化が均一に減少するように，単位データを生成している．このとき，‘通信頻度’ の合計値を最小にする，各ノードの ‘転送状態’ を求める．この例題では，抽象度が等しいノードの ‘参照頻度’ と ‘更新頻度’ が等しいために，抽象度が等しいノードを一つの仮想的なノード (単純仮想ノードと呼ぶことにする) と見なすことで，問題を単純化できる．そこで，次のパラメタを用いて解法の概要を示す．

- r_a $(a = 0, 1, 2, 3)$ は，抽象度が a である単純仮想ノードの ‘参照頻度’．単純仮想ノードを構成するノードの ‘参照頻度’ の合計値である．
- u_a $(a = 0, 1, 2, 3)$ は，抽象度が a である単純仮想ノードの ‘更新頻度’．単純仮想ノードを構成するノードの ‘更新頻度’ の合計値である．

まず，抽象度が ac である単純仮想ノードを複製することにすれば，この単純仮想ノードの ‘転送状態’ は ‘複製’ である．抽象度が ac より高い単純仮想ノードの ‘転送状態’ は ‘生成’ になる．換言すれば，抽象度が ac より低い単純仮想ノードの ‘転送状態’ は ‘参照’ になる．このときの ‘通信頻度’ の合計値 T_{ac} を次に示す．

$$T_{ac} = \begin{cases} u_0 & ac = 0 \\ \sum_{i=0}^{ac-1} r_i + u_{ac} & ac = 1, 2, 3 \end{cases} \quad (1)$$

T_{ac} の最小値は，T_0 から T_3 へ昇べきの順に探索することで求めることができる．こうして求めた T_{ac} の最小値と，全てのノードの ‘転送状態’ を ‘参照’ にし

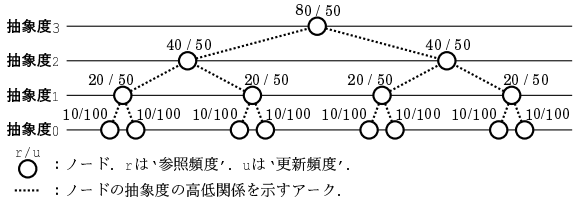

図 4: 問題を表す木

た場合の‘転送頻度’の合計値とで，値が小さい方における‘転送状態’が解である．

5.2 アルゴリズム

一般には，5.1節の例題のように‘参照頻度’と‘更新頻度’が均一に変化するとは限らない．そこで，木を帰りがけ順になぞりながら，段階的に解を求める．まず，葉を含む高さ2の部分木 T_{1i} $(i = 0, 1, 2, ..., n)$ における解を求める．この場合，次に示す T_{1i} $(i = 0, 1, 2, ..., n)$ の状態 (1)(2)(3) から，‘通信頻度’の合計値が最小になる状態を選択することで，5.1節の例題の場合と同様に解を求めることができる．

- **状態** (1)：子であるノードを複製することにした状態．親であるノードの‘転送状態’は‘生成’．子であるノードの‘転送状態’は‘複製’．
- **状態** (2)：親であるノードを複製することにした状態．親であるノードの‘転送状態’は‘複製’．子であるノードの‘転送状態’は，‘通信頻度’が小さくなるように‘複製’・‘参照’から個別に選択．
- **状態** (3)：ノードごとに複製するか否かを選択することにした状態．各ノードの‘転送状態’は，‘通信頻度’が小さくなるように‘複製’・‘参照’から個別に選択．

次に，T_{1i} $(i = 0, 1, 2, ..., n)$ を仮想的なノード (仮想ノードと呼ぶことにする) と見なし，仮想ノードとその親であるノードから構成される高さ2の部分木 T_{2j} $(j = 0, 1, 2, ..., m)$ における解を求める．この場合，次に示す仮想ノードの状態 (a)(b) を求めれば，T_{2j} $(j = 0, 1, 2, ..., m)$ における状態 (1)(2)(3) に相当する状態を求めることができる．T_{2j} $(j = 0, 1, 2, ..., m)$ の状態 (1)(2)(3) に相当する状態から，‘通信頻度’の合計値が最小になる状態を選択することで，T_{1i} $(i = 0, 1, 2, ..., n)$ の場合と同様に解を求めることができる．

- **状態** (a)：‘転送状態’が‘複製’であるノードに相当する状態．この状態は，仮想ノードの親であるノードの要素である単位データが，複製先のRMにおいて生成可能な状態である．仮想ノードとみなした部分木が状態 (1)(2) であるとき，部分木の根であるノードの要素である単位データは複製先のRMにあり，条件を充たす．従って，部分木の状態 (1)(2) から‘通信頻度’の合計値の小さくなる方の状態を選択すればよい．このときの部分木を構成する各ノードの‘転送状態’を‘複製転送状態’，‘通信頻度’の合計値を‘複製通信頻度’と呼ぶことにする．
- **状態** (b)：‘転送状態’を‘通信頻度’が小さくなるように‘複製’・‘参照’から選択したノードに相当する状態．この状態は，仮想ノードとみなした部分木の‘通信頻度’の合計値が最小になる状態である．従って，部分木の状態 (1)(2)(3) から‘通信頻度’の合計値が最小になる状態を選択すればよい．このときの部分木を構成する各ノードの‘転送状態’を‘最善転送状態’，‘通信頻度’の合計値を‘最善通信頻度’と呼ぶことにする．‘最善転送状態’は仮想ノードとみなした部分木における解であり，‘最善通信頻度’はそのときの‘通信頻度’の合計値である．

同様にして，T_{2j} $(j = 0, 1, 2, ..., m)$ を仮想ノードとみなし，仮想ノードとその親であるノードから構成される高さ2の部分木 T_{3k} $(k = 0, 1, 2, ..., l)$ における解を求める．これを繰り返し，木全体を含む仮想ノードの状態 (a)(b) を求める．このような操作を行う疑似コードを，付録に示す．

5.3 アルゴリズムの適用例

課題 (a)(b)(c)(d) を含む分りやすい例題として，次のような広域ネットワーク監視サービスを取り上げる．監視対象は4つの組織のネットワークである．各ネットワークには8つのサブネットがある．このとき，各サブネットにおけるネットワーク・ルータ・サーバの負荷情報を監視し，大まかな負荷情報を‘サービスのユーザビリティを示す情報’として，詳細な負荷情報を‘性能管理や障害管理のための情報’として，各組織の利用者と管理者に提供する．利用者と管理者は，‘サービスのユーザビリティを示す情報’を常時参照する．管理者は，‘性能管理や障害管理のための情報’を，‘サービスのユーザビリティを示す情報’が「高負荷」を示したときに参照する．

我々の情報共有手法を用いる場合は，負荷情報を定期的に収集するセンサをサブネットごとに置き，収集した負荷情報を管理するRMをネットワークごとに置く．図1と同様の構成である．各RMは，図3と同様のIMWのインスタンスを用いて，抽象度の高い単位データを‘サービスのユーザビリティを示す情報’として提供する．抽象度の低い単位データを‘性能管理や障害管理のための情報’として提供する．

このときの，付録に示した疑似コードの実行過程

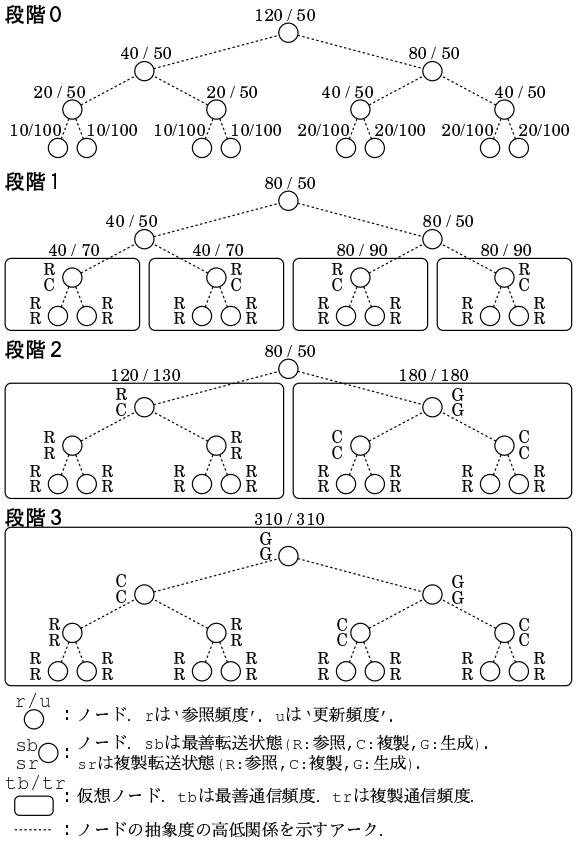

図 5: 実行過程

を図5に示す．段階 0 では，ある RM からある RM への単位データの冗長な転送を排除する問題を木で表している．‘サービスのユーザビリティを示す情報’が「高負荷」を示したために，‘性能管理や障害管理のための情報’も参照されているという，参照頻度が比較的高い状況である．段階 1 では，葉を含む高さ 2 の部分木を仮想ノードとして，5.2 節で示した状態 (a)(b) を求めている．状態 (a) として状態 (2) を選択している．‘複製転送状態’は，親であるノードは‘複製’，子であるノードは‘参照’である．‘複製通信頻度’は，親であるノードの‘更新頻度’と子であるノードの‘参照頻度’の和である．状態 (b) として状態 (3) を選択している．‘最善転送状態’は全てのノードで‘参照’である．‘最善通信頻度’は全てのノードの‘参照頻度’の和である．段階 2 では，高さ 3 の部分木を仮想ノードとして状態 (a)(b) を求めている．段階 3 では，木全体を仮想ノードとして状態 (a)(b) を求めている．段階 3 の‘最善転送状態’が解であり，‘最善通信頻度’が‘通信頻度’の合計値 (310) である．なお，疑似コード適用前の‘通信頻度’の合計値は 380 である．

一方，従来の情報共有手法を用いる場合は，負荷情報を定期的に収集する SNMP エージェントをサブネットごとに置き，収集した負荷情報を管理する SNMP マネージャをネットワークごとに置く．SNMP マネージャは，収集した負荷情報をそのまま相互に複製し，‘サービスのユーザビリティを示す情報’あるいは‘性能管理や障害管理のための情報’として提供する．このときの‘通信頻度’の合計値は，我々の情報共有手法において RM が全ての抽象度 0 の単位データを相互に複製する場合と同じであり，800 である．

6 考察

前提条件とする「利用者は情報の詳細を段階的に参照する」という参照パターンに基づいて「自律型モジュール間の通信頻度」を削減するには，利用者が必要とする詳しさに応じた情報だけを自律型モジュールの間で通信すれば良い．従来の情報共有手法では，観測した情報をそのまま相互に複製するために，前記のような通信はできない．我々の情報共有手法では，抽象度の異る単位データを動的に生成し，利用頻度の高い単位データだけを相互に複製するために，前記のような通信ができる．しかし，利用者が詳しい情報を頻繁に必要とする場合には，本稿で示したアルゴリズムを適用しても抽象度の低い単位データが相互に複製されるために，従来手法との差は減少する．この場合，共有する情報の詳しさを制限する方法が考えられる．たとえば「自律型モジュール間の通信頻度」に対する RM の処理能力やネットワークの帯域の余力が閾値を下回ったこと契機として，抽象度の低い単位データから順に，RM に閉じた局所的な利用に切り替える．

7 おわりに

本稿では，我々が提案している情報共有手法において，RM 間の単位データの冗長な転送を排除するという最適化の方法を示した．また，その効果を考察した．今後の課題として，RM 間の単位データの交換に用いる，即時性・信頼性・転送効率などの要件を満たす通信プロトコルの確立がある．

参考文献

[1] (財) 道路交通情報通信システムセンタ:VICS の挑戦, (財) 道路交通情報通信システムセンタ (1996).

[2] (財) 日本気象協会 tenki.jp. http://tenki.jp/

[3] 国土交通省 川の防災情報. http://www.river.go.jp/

[4] 国土交通省 地震計ネットワーク情報. http://www.nilim.go.jp/japanese/database/nwdb/

[5] 環境省 大気汚染物質広域監視システム. http://w-soramame.nies.go.jp/

[6] 石田亨：ディジタルシティの現状, 情処会誌, Vol.41, No.2, pp.163-168 (2000).

[7] 国土交通省道路局 地域の ITS・歩行者の ITS. http://www.its.go.jp/ITS/j-html/index.html

[8] Miller, M.A. : *Managing Internetworks with SNMP - Second Edition*, M & T Books (1997). トップスタジオ (訳) : SNMP インターネットワーク管理, 翔泳社 (1998).

[9] 荒野, 西郷 : 大規模商用ネットワークの運用事例, 情処会誌, Vol.39, No.10, pp.969-975 (1998).

[10] Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C. : Grid Information Services for Distributed Resource Shareing, *Proc. 10th IEEE International Symposium on Hight-Performance Distributed Computing* (2001).

[11] Mockapetris, P.,V. and Dunlap, K.,J., : Development of the Domain Name System, *SIGCOMM'88*, pp.123-133, ACM (1988).

[12] 大山, 千田, 戸部, 窪田, 田中, 空 : X.500 ディレクトリ入門, 東京電気大学出版局 (1997).

[13] 馬場, 山口 : DNS を用いた広域負荷分散の実装, 情処学会研究会報告, DSM-9-7, pp.37-42 (1998).

[14] 服部, 呉, 安田, 横井 : ディレクトリサービスを利用した都市情報の分散型データベース構築に関する検討, 情処学会論文誌, Vol.41 No.12, pp.3307-3313 (2000).

[15] 八木, 高橋 : 広域分散した動的な情報のための共有システムの構想, 2001-DPS-103-1,pp.519-520 (2001).

[16] 八木, 高橋 : 広域分散した動的な情報の共有システム WISE のデータベース, 2002-DBS-126-3,pp.17-23 (2002).

付録　疑似コード

「'」で囲った文字列は各ノードに付属する変数である．「"」で囲った文字列は定数である．

```
単位データの冗長な転送を排除する ()
{
  状態 (a)(b) を求める (根であるノード);
}
状態 (a)(b) を求める (ノード)
{
  //' 通信頻度' が小さくなるように "参照 " と
  //"複製 " から ' 転送状態' を選択.
  if( ' 参照頻度' < ' 更新頻度' ){
    ' 通信頻度' = ' 参照頻度';
    ' 転送状態' = "参照 ";
  } else {
    ' 通信頻度' = ' 更新頻度';
    ' 転送状態' = "複製 ";
  }
  if( ノードは葉である ){
  // このノードだけで仮想ノードを作る場合.
    // 状態 (a)(b) における各値を求める.
    ' 最善通信頻度' = ' 通信頻度';
    ' 最善転送状態' = ' 転送状態';
    ' 複製通信頻度' = ' 更新頻度';
    ' 複製転送状態' = "複製 ";
  } else {
  // 真の子孫とともに仮想ノードを作る場合.
    // 子であるノードに移動.
    子であるノードに対して順に「状態 (a)(b) を
    求める ()」を適用;

    // 状態 (a)(b) における各値を求める.
    ' 最善通信頻度の和' =
        子であるノードの' 最善通信頻度' の和;
    ' 複製通信頻度の和' =
        子であるノードの' 複製通信頻度' の和;
    if( ' 複製転送頻度の和' ≦
        ' 最善転送頻度の和' + ' 通信頻度' )
    {
    // 状態 (b) として状態 (1) を選択する場合.
      ' 最善通信頻度' = ' 複製転送頻度の和';
      ' 最善転送状態' = "生成 ";

      // 状態 (a) として状態 (1) を選択.
      ' 複製通信頻度' = ' 複製転送頻度の和';
      ' 複製転送状態' = "生成 ";

      真の子孫であるノードにおいて' 複製転送
      状態' を' 最善転送状態' に代入する;
    } else {
    // 状態 (b) として状態 (2) か状態 (3) を
    // 選択する場合.
      ' 最善通信頻度' =
          ' 最善転送頻度の和' + ' 通信頻度';
      ' 最善転送状態' = ' 転送状態';

      if( ' 複製転送頻度の和' ≦
          ' 最善転送頻度の和' + ' 更新頻度' )
      {
      // 状態 (a) として状態 (1) を選択する場合.
        ' 複製通信頻度' = ' 複製転送頻度の和';
        ' 複製転送状態' = "生成 ";
      } else {
      // 状態 (a) として状態 (2) を選択する場合.
        ' 複製通信頻度' =
            ' 最善転送頻度の和' + ' 更新頻度';
        ' 複製転送状態' = "複製 ";

        真の子孫であるノードにおいて' 最善転送
        状態' を' 複製転送状態' に代入する;
      }
    }
  }
}
```