社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
DC2011-25 (2011-10)

ネットワークコーディングに基づくゴシップの提案

徳山 瞬† 土屋 達弘† 菊野 亨†

† 大阪大学
〒 565-0871 大阪府吹田市山田丘 1-5
E-mail: {s-tokuym, t-tutiya, kikuno}@ist.osaka-u.ac.jp

あらまし ゴシップは，分散システムにおけるブロードキャスト手法の一つである．ゴシップでは，ブロードキャストを行うノードが，メッセージをランダムに選択されたいくつかのノードに対して送信し，受信したノードも同様にメッセージの転送を行う．この過程を繰り返すことで全ノードに対するメッセージの伝搬が高い確率で実現される．しかし，この手法では多くのノードが同一メッセージを複数回受信するなど，冗長なメッセージが多いという問題がある．そこで，本研究では，ネットワークコーディングを利用したゴシッププロトコルの最適化について提案する．提案手法では，各ノードは受信したメッセージからランダムに新たなメッセージを作り出し，それを送信する．これらのメッセージは全て，オリジナルのブロードキャストメッセージの断片のランダムな線形結合となっている．これにより，各ノードは完全に同一なメッセージを受信することがほとんどなくなり，通常のゴシッププロトコルより，低いメッセージコストで高い信頼性を実現することができる．

キーワード ゴシップ，ブロードキャスト，ネットワークコーディング，信頼性

Network Coding-Based Gossip

Shun TOKUYAMA†, Tatsuhiro TSUCHIYA†, and Tohru KIKUNO†

† Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
E-mail: {s-tokuym, t-tutiya, kikuno}@ist.osaka-u.ac.jp

Abstract Gossip is a broadcasting method for distributed systems. In gossip, the node that initiates a broadcast sends the broadcast message to some randomly selected nodes. Upon receiving a message, a node then forwards the message to randomly selected nodes. As a result of repeating this process, the message is eventually propagated through the whole network. An obvious problem with gossip is that it incurs significant message overhead: many nodes receive the identical message multiple times, waisting network resources. To address this problem we propose a network coding-based gossip protocol. In the proposed protocol, a broadcast message is not diffused as it is. Rather, it is divided into some fragments and nodes encode and forward random linear combinations of these fragments. This prohibits nodes from receiving an identical message multiple times and, as a result, increases reliability with less message overhead than the ordinary gossip.

Key words Gossip, broadcast, network coding, reliability

1. はじめに

ゴシップは，分散システムにおけるブロードキャスト手法の一つである [2], [6], [12]．ゴシップではメッセージを送信する際に，隣接ノード全てにメッセージを送信するのではなく，ランダムに選んだいくつかのノードにのみメッセージを送信する．これを繰り返すことでメッセージはシステム全体に広まっていく．ブロードキャストのためのスパニングツリー等を管理する必要がなく，また，ランダム性によりメッセージを複数の経路から受信できることから，高い信頼性を実現することができる．ゴシップは，複製データベース [1], [5], 故障検知 [8], [14], 分散情報管理 [10], [13], ライブストリーミング [3] 等の応用で利用されている．

一方，高い信頼性の代償として，ゴシップはメッセージオーバーヘッドが大きいという問題がある．具体的には，多くのノードが同一メッセージを複数回受信するなど，冗長なメッセージ

Initiation of broadcast of M:
  Send M to f randomly chosen nodes;

When a node receives a message M:
  If (M is received for the first time)
    Send M to f uniformly randomly chosen nodes;

図 1　従来のゴシッププロトコル

の送受信が避けられない.

そこで，本論文では，ネットワークコーディング [9] を利用した新たなゴシッププロトコルを提案する．提案手法は，従来のゴシッププロトコルと比較して，少ないトラフィックで高いメッセージ到達率 (信頼性) を実現する．ネットワークコーディングとは，送信ノードや受信ノードだけでなく，中継ノードでもメッセージの符号化，復号化を行うような通信手法を意味する．提案手法ではブロードキャストメッセージはそのまま転送されるのではなく，幾つかの固まりに分割され，そのランダム結合がメッセージとして送信される．各ノードは複数のメッセージを受信し，それらを復号することでオリジナルのメッセージを得ることができる.

ネットワークコーディングを利用したゴシッププロトコルの既存研究として，[4] や [11] がある．これらの先行研究では，メッセージの伝搬速度に関心が置かれており，ノードや通信の故障や信頼性については考慮されていない．また，ゴシップは同期式ラウンドに従って進行することを仮定し，各ラウンドにおいてノードが通信できるのは，別の一つのノードのみとしている．一方，本研究では，故障ノードを仮定した上で信頼性を評価している．また，ノードは複数のノードに対し同時に通信が可能という，より現実に即した設定を想定している.

2. 従来手法とその問題点

本節では従来のゴシッププロトコルについて説明する．図 1 は従来のゴシッププロトコルのアルゴリズムを示している．ノードがメッセージの送信を開始する際には，ランダムに f 個の隣接ノードを選び送信する．メッセージを受け取った際，それが初めてであれば開始時と同様にランダムに f 個の隣接ノードを選びそのメッセージを転送する．ここで，f はファンアウトと呼ばれる.

この手法ではランダムにメッセージを送信するノードを選ぶため，多くのノードが同じメッセージを受け取る一方，メッセージが一度も届かないノードも存在する．表 1 はノード数 1000，故障率 10% の環境下での単純なシミュレーションの結果である．この表は f の値を変化させた時に一つのノードが何度同じメッセージを受け取ったかを示している．各々の値はシミュレーションを 100 回行った平均値である.

この表から $f = 4$ で到達率が 97% の時に 30% 近いノードが 5 回以上同じメッセージを受け取っていることが分かる．また $f = 7$ で到達率が 99. 7% の時には 75% ものノードが 5 回以上同じメッセージを受け取っている.

表 1　正常なノードが同じメッセージを受け取った回数 (1000 ノード，故障率：10%)

	0	1	2	3	4	$\geqq 5$
$f = 4$	2.9%	10.9%	18.5%	21.6 %	18.8 %	27.3%
$f = 5$	1.2%	5.0%	11.7%	17.0%	19.2%	45.6%
$f = 6$	0.6%	2.5%	6.6%	11.8%	16.2%	62.3%
$f = 7$	0.3%	1.1%	3.6%	7.6%	12.1%	75.1%

3. 提案手法

本節ではネットワークコーディングを利用したゴシッププロトコルについて説明する．これによって，同じブロードキャストメッセージを何度も受け取ることを減らすことが出来る．図 2 にアルゴリズムを示す.

3.1 ブロードキャスト開始時

ブロードキャストを開始するノードは，ブロードキャストメッセージを k 個のブロックに分割する．ここで $F_1, F_2, \cdots, F_k$ が分割されたブロックであるとする．元のメッセージを l ビットであるとし，$b = \lceil l/k \rceil$ とする．分割されたブロックは全て，大きさ q のガロア体 $GF(q)$ 上の $\lceil b/\log_2(q) \rceil$ 次のベクトルである.

次に，開始ノードはそれらのブロックからランダム線形コーディングを使って送信するメッセージを作り出す．各々のブロックに対し，$GF(q)$ 上からランダムにある数を選んで係数とし，それらの線形結合を作る．それぞれのブロックの係数の集合とこの線形結合を併せて，送信メッセージとする．ここで係数は 0 を含まないものとする．なお，全てのブロックは $GF(q)$ 上のベクトルであり，加算や乗算は $GF(q)$ 上の演算とする.

メッセージを作成する度，開始ノードはそれを f_{init} 個の隣接ノードに対して送信する．初期ファンアウト f_{init} は通常のファンアウト f よりも大きな値とすべきである．これは開始ノードとその他のノードの負荷を等しくするためである．その他のノードは多くて $k * f$ のメッセージを送信するので，f_{init} を $k * f$ に設定する.

また，それぞれのメッセージはヘッダに ID を持っているとする．ID は元のブロードキャストメッセージ毎に定められる値で，これによりノードが複数のメッセージのブロードキャストを処理することができるようにする．以降ではアルゴリズムがどのように動作するかを単一のブロードキャストメッセージに対して説明していく.

3.2 メッセージの送信

従来の手法と違い，本手法では受信ノードは受け取ったメッセージそのまま送信するのでなく，既に受信しているメッセージに符号化を施し新たなメッセージを作り出してそれを送信する.

ブロードキャスト毎に，ノードはバッファを準備し，受信したメッセージを格納するものとする．図 2 では，*Received* がこのバッファを表している．ノードがメッセージ m を受信したときの処理は，以下の 3 ステップからなる.

```
Initiation of broadcast of M:
  Divide M into k fragments F_1, ..., F_k;
  Choose a set RN of f_init random nodes;
  For p ∈ RN
    Create a message msg from F_1, ..., F_k
      with random linear encoding;
    Send msg to node p;

When a node receives a message m:
  { Step 1 }
  If (msg is informative)
    Add msg to Received; { Received is initially empty. }
    { Step 2 }
    Choose a set RN of f random nodes;
    For p ∈ RN
      Create a message msg from the messages in Received
        with random linear encoding;
      Send msg to node p;
    { Step 3 }
      If (|Received| = k)
        Decode the broadcast message from Received;
```

図 2　ネットワークコーディングを利用したゴシッププロトコル

- **Step 1:** m が意味のあるメッセージであるならばバッファに格納し，*Step* 2 に移動する．そうでなければ m を破棄する．
- **Step 2:** バッファから f 個の新たなメッセージを作り出し，ランダムに選んだノードに送信する．*Step* 3 に移動．
- **Step 3:** バッファ内のメッセージ数が k に達したら，オリジナルのブロードキャストメッセージを復号する．

a）Step 1

メッセージ m を受信したときバッファに既に $m_1, \cdots, m_{s-1}$ が格納されているとする．$m_1, \cdots, m_{s-1}$ と m は全て二つの部分から成る．一つは $F_1, \cdots, F_k$ の線形結合から成るペイロード，もう一つは係数ベクトルである．$m_1, \cdots, m_{s-1}$ の係数ベクトルは線形独立である．

メッセージ m の係数ベクトルが $m_1, \cdots, m_{s-1}$ の全ての係数ベクトルと線形独立である時，m を意味のあるメッセージという．*Step* 3 で述べるように，オリジナルのメッセージは互いに線形独立な係数ベクトルを持つ k 個の任意なメッセージからのみ復号できるからである．

b）Step 2

生成されるメッセージのペイロード部分は，バッファに蓄えられている $m_1, \cdots, m_s$ のランダム線形結合となる．ここで，m_s は *Step* 1 で受信した新規メッセージとする．具体的には，ペイロード部分は以下の式で与えられる．

$$\sum_{i=1}^{s} a_i m_i$$

ここで a_i は $GF(q)$ 上からランダムに選ばれた係数である．

係数ベクトルは，以下の式で与えられる．

$$\sum_{i=1}^{s} a_i e_i$$

ここで e_i は m_i の係数ベクトルを示す．

c）Step 3

バッファ内に格納されているメッセージの数が k に達したら，オリジナルのブロードキャストメッセージの復号が可能になる．この時 k 個のメッセージの係数ベクトルは全て線形独立となるので，k 元連立方程式を解くことによって復号できる．

4. 最　適　化

本節では前節の手法の改良を提案する．これは，予備実験により，提案手法の性能を調べた結果得られるパフォーマンスの向上がわずかであることが判明したためである．この原因として，二つの問題が基本提案手法にあることが分かった．これらを解決するため，最適化を施した新たなアルゴリズムを図 3 に示す．

4.1　無意味なメッセージの拡散防止

一つ目の問題は，極初期の段階ではメッセージの拡散が非効率的になる，ということである．あるノードが受信したメッセージが一つだけである時，新たな送信メッセージを作成しようとするのなら，全てのメッセージが線形従属となってしまう．このため無意味なメッセージが多くなり，オリジナルのメッセージを復号するためにより多くのメッセージを受け取らなくてはならないことになる．

これを防ぐため以下の対策をとる．

（1）　初めてメッセージを送るノードに対しては二つのメッセージを送信する．

（2）　二つ以上メッセージを受信している場合のみ，新たなメッセージを作り，別のノードに送信する．

1 の対策は，メッセージを交換したことのあるノードを記憶しておくことで実現できる．*Contacts* というバッファがこのために使われる．これは最初は空であり，ノード p からメッセージを受信したり (Step 1)，ノード p にメッセージを送信することによって (Step 2)，対象ノード名が格納されていく．もしノード p が *Contacts* の中にあれば，そのノードは既にメッセージを複数個受け取っていることが保証される．なければ p にメッセージを送信する際には二つのメッセージを作り出して送信する．

2 の対策は，受信したメッセージ数が 1 の時には送信を行わないようにすることで容易に実現できる．

4.2　動的ファンアウト

もう一つの問題は，メッセージが十分に広まった段階では新たなメッセージの送信が無駄になることが多いことである．ノードが k 番目のメッセージを受け取った時，多くの他のノードは k やそれに近い個数のメッセージを受け取っている．これは，メッセージの伝搬が全てのノードでほぼ等しく広まっていくためである．このような場合，新たなメッセージを送信することは無駄なトラフィックを増やす要因となる．

```
Initiation of broadcast of M:
  Divide M into k fragments F_1, ..., F_k;
  Choose a set RN of f_init random nodes;
  For p ∈ RN
    Create two messages msg, msg' from F_1, ..., F_k
      with random linear encoding;
    Send msg, msg' to node p;

When a node receives a message m:
  { Step 1 }
  If (msg is informative)
    Add msg to Received; { Received is initially empty. }
    Add p to Contacts; { Contacts is initially empty. }
    { Step 2 }
    If (|Received| ≧ 2)
      Choose a set RN of f(|Received|) random nodes;
      For p ∈ RN
        Create a message msg and send it to p;
        If (p ∉ Contacts)
          Create another message msg' and send it to p;
          Add p to Contacts;
    { Step 3 }
    If (|Received| = k)
      Decode the broadcast message from Received;
```

図 3　最適化された提案手法アルゴリズム

これを解決するため，動的ファンアウトを導入する．これは，メッセージを受信した数に伴いファンアウトを動的に減らすというものである．ファンアウトの値は関数 $f(|Received|)$ で与えられるものとする．ここで $Received$ は受信した意味あるメッセージを格納しているバッファを示しており，$|Received|$ はそのメッセージ数である．

ネットワークの構造や k の値等の要素によって効率的な関数 $f(|Received|)$ は決定される．本研究の現段階では事前にシミュレーションを行い $f(|Received|)$ を決定している．

5. シミュレーション結果

本節では，シミュレーションの結果を示す．各々のノードはシステム中から，等しくランダムにノードを選択できるものとする．これは *peer sampling service* [7] を使うことで実現できる．ノード数は 100 の場合と 500 の場合を想定する．

演算は $GF(2^8)$ 上で行うものとする．分割数 k は 4, 6, 8 と変化させる．動的ファンアウトは表 2 の通りとする．f_d はデフォルトのファンアウトの値とする．この値は 4 から 7 とする．

表 2　ファンアウト $f(|Received|)$

$\|Received\|$	1	2	3	4	5	6	7	8
$k=4$	–	f_d	0	0	–	–	–	–
$k=6$	–	f_d	2	0	0	0	–	–
$k=8$	–	f_d	f_d	1	0	0	0	0

ブロードキャスト開始ノードのファンアウト f_{init} は $k * f_d$ と定める．

故障ノードの割合は 0%から 30%まで 10%ずつ変化させる．故障ノードに送られたメッセージはすぐに破棄されるとする．ノードから送られるメッセージは指数分布に従う遅延時間後に受信ノードに到達する．ノードでの符号化によるオーバーヘッドは無視する．その代わり，メッセージの遅延時間に含まれるものと仮定する．また，メッセージの係数ベクトルはそのペイロードと比べて無視できる程度に小さいものとする．これは，ブロードキャストメッセージを十分に大きくすることで実際の環境でも成り立つ．

シミュレーションの結果を図 4 と図 5 に示す．これらの図は信頼性と通信コストの関係を示している．横軸は通信コストを，縦軸はブロードキャストメッセージを復号できなかった正常なノードの割合を表している．通信コストは，ブロードキャストメッセージをそのまま送信するときはメッセージ数，符号化したメッセージを送信するときはメッセージ数の $1/k$ 倍とする．

図中の各点は f_d を $4, \cdots, 7$ まで一つずつ変化させた場合を示している．それぞれの点の値は，1000 回シミュレーションを行った平均値である．4 つの線は，従来のゴシッププロトコルと提案手法の $k=4, k=6, k=8$ に対応している．

図から，提案手法の方がより少ない通信コストで高い信頼性を達成していることが分かる．例えば，ノード数が 500，故障ノード割合 10%の場合（図 5(b)），$k=8$，$f_d=4$ の時に 0.3%の未到着のノードが存在するが，通信コストは 1500 以下である．従来のゴシッププロトコルではこの信頼性を達成するにはその 2 倍の通信コストが必要である．

他に分かることとして，k の値が大きくなるほど，ブロードキャストメッセージが届かないノードの割合が少なくなっていくことがある．これは実験を行った全ての場合で見られる．その理由として以下の二つが挙げられる．一つは k を大きくするほど係数ベクトルの長さが大きくなり，線形独立な係数ベクトルを作り出す可能性が減るからである．もう一つの理由は，メッセージが小さいほど伝搬するメッセージ量をコントロールしやすいためである．例えば k が大きいほど，より効率的な動的ファンアウトの変化が可能となる．

6. ま と め

本研究では，ネットワークコーディングを利用したゴシッププロトコルを提案した．提案手法では元のブロードキャストメッセージに符号化を施し，サイズの小さなメッセージを作り出してそれを転送する．各ノードは元のメッセージの分割数 k に等しい，線形独立な係数ベクトルを持つメッセージを受信することで復号が可能となる．これにより，従来のゴシッププロトコルと異なり，メッセージを何回も受け取った場合でも，それぞれのメッセージから異なった情報を得ることができる．

シミュレーションにより，提案手法が信頼性と通信コストのトレードオフを改善できることを示した．また，k の値が大きくなるほどより効果的となることが分かった．

しかし，提案手法があらゆる環境で優れた結果を出すことが

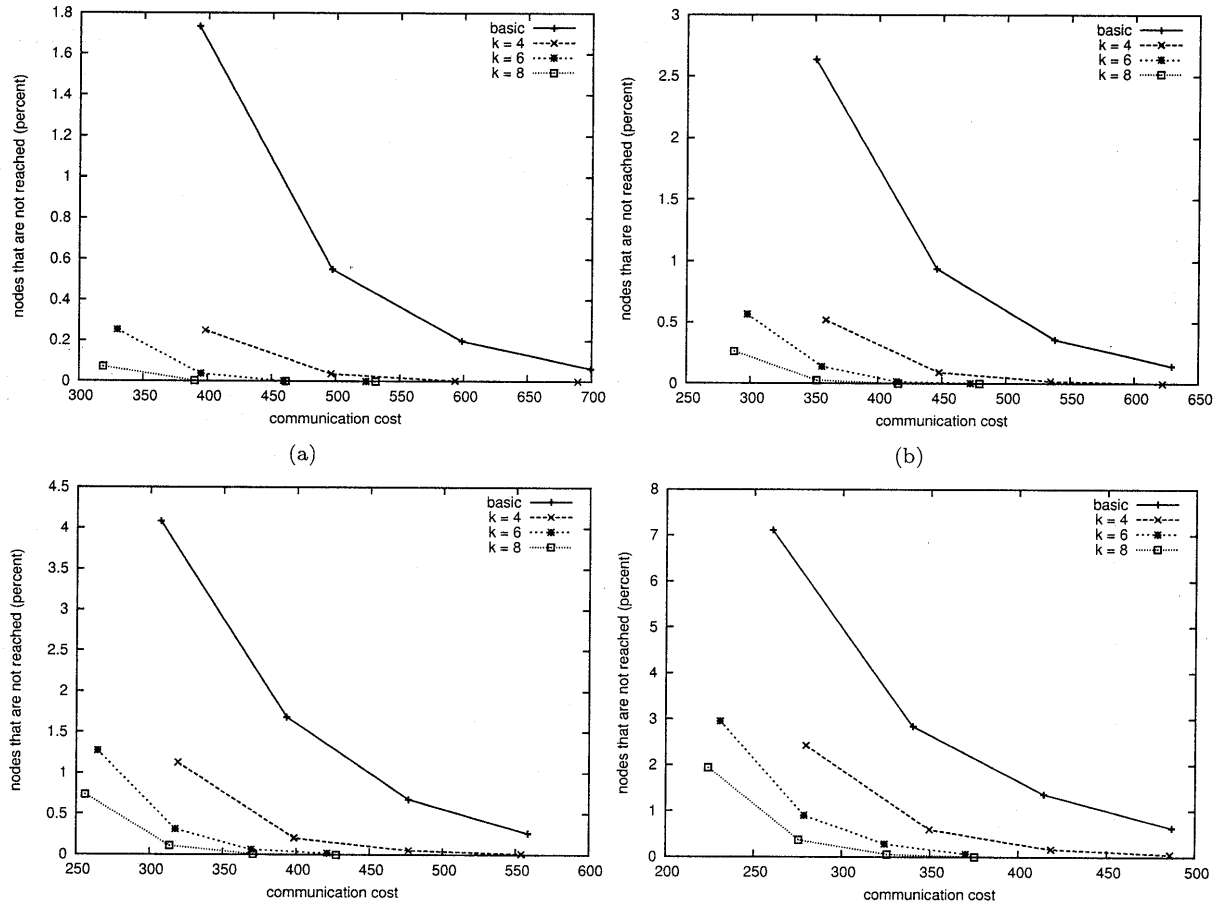

図 4 コストとメッセージ未着ノード．100 ノード，故障ノード (a) 0%, (b) 10%, (c) 20%, (d) 30%

できると断言することはできない．これは，シミュレーションで理想的な設定を仮定しているためである．例えば，符号化や復号化にかかるオーバーヘッドは今回考慮していない．将来的にはより現実に即した設定で実験を行う予定である．また，動的ファンアウトにおける関数 $f(|Received|)$ の定め方も問題として残っている．本論文では予備実験で効果的であった値を使用したが，より具体的な定め方が課題として挙げられる．

謝　辞

本研究の一部は，文部科学省グローバル COE プログラム「アンビエント情報社会基盤創成拠点」の支援を受けて行った．

文　献

[1] D. Agrawal, A. El Abbadi, and R. Steinke. Epidemic algorithms in replicated databases. In *Proceedings of the Sixteenth ACM Symposium on Principles of Database Systems*, pages 161–172, 1997.

[2] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.

[3] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: optimal performance trade-offs. In *Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '08, pages 325–336, New York, NY, USA, 2008. ACM.

[4] S. Deb, M. Médard, and C. Choute. Algebraic gossip: A network coding approach to optimal multiple rumor mongering. *IEEE Transactions on Information Theory*, 52(6):2486–2507, June 2006.

[5] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Ann. ACM Symp. Principles of Distributed Computing (PODC)*, pages 1–12, Aug. 1987.

[6] P. T. Eugster, R. Guerraoui, S. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. In *Proceedings of the 2001 International Conference on Dependable Systems and Networks (DSN '01)*, pages 443–452, July 2001.

[7] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems*, 25, August 2007.

[8] A. Lakshman and P. Malik. Cassandra - a decentralized structured storage system. In *3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware (LADIS 09)*, Oct. 2009.

[9] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, Feb. 2003.

[10] A. Montresor, M. Jelasity, and O. Babaoglu. Robust aggregation protocols for large-scale overlay networks. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, pages 19–28. IEEE Computer Society, 2004.

[11] D. Mosk-Aoyama and D. Shah. Information dissemination via network coding. In *Proc. ISIT*, pages 1748–1752, July 2006.

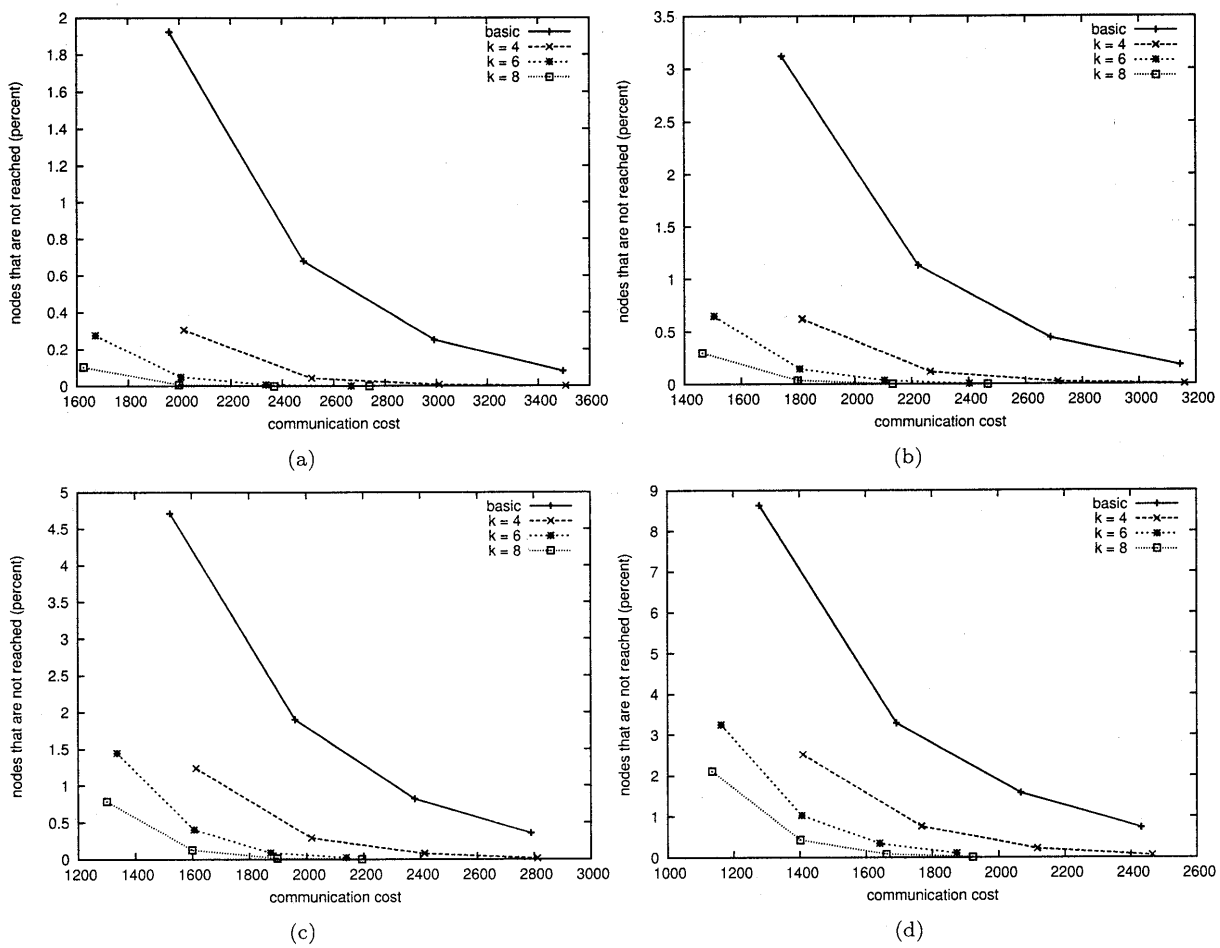

図 5　コストとメッセージ未着ノード. 500 ノード, 故障ノード (a) 0%, (b) 10%, (c) 20%, (d) 30%

[12] Q. Sun and D. Sturman. A gossip-based reliable multicast for large-scale high-throughput applications. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2000)*, pages 347–358, June 2000.

[13] R. van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems (TOCS)*, 21(2):164–206, 2003.

[14] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98)*, pages 55–70, Sept. 1998.