１９．情報共有手順の提案と簡単な実装

管制システム部　　金田直樹　塩見　格一

1　はじめに

1.1　情報共有の必要性

近年, 協調的意思決定 (Collaborative Decision Making, CDM) が注目を集めている. CDM においては, 合意を得るために利害関係者間の意識を合わせる必要があり, そのためには無矛盾な情報を共有する必要がある. また, 航空管制に限らず航空に関する業務一般に対して, ひとつの作業を複数人で確認を行いながら行う作業が多い. その意味でも複数の当事者間での情報共有は本質的に重要である. 例えば調整や連絡の不足は情報の共有により改善される可能性が高いと. また, パイロットと管制官の間で同一の情報を共有しないことは危険を招く可能性がある. この問題に対する対策として, TCAS RA ダウンリンクの試みが開始されたり, FMS の持つ情報に基づくコンフリクト警報の改善について研究が行われている. 他にも, 航空に関する業務においては情報共有の失敗であると考えられる見落としや聞き間違いを防ぐために, 復唱が一般的に行われている. これは双方向通信システムにおける誤り訂正である自動再送方式の 1 種であるとモデル化できる. 数人であれば復唱は有効だが, 人数が多くなると全員の復唱を確認する作業だけで非常な手間となってしまう. このように情報の共有は重要な問題と考えられる.

1.2　従来の情報共有方式

管制支援システムは過去にいくつか提唱されてきているが, 本発表では航空管制において重要な意思決定を助けるために, 情報をリアルタイムに矛盾なく共有することが必要であると考え, 情報共有に関する手順を提案する. 現在の情報共有は, クライアント・サーバ方式により, サーバにある唯一のデータベースからクライアントが情報を取得することにより情報共有を実現しているものが殆んどである. しかし, 複数の計算機上にあるデータベースを完全に同期させることは簡単ではない. そのためサーバの冗長構成が取りにくい. またポーリングにより情報を取得するのでリアルタイム性に欠け, またサーバの能力により拡張性が制限されるという問題点があった[1].

サーバを使用しない情報共有方式としては Peer-to-Peer(P2P) システムが代表的なものであり, Gnutella[1] に端を発する, Pure P2P によるファイル共有等が考えられる. Pure P2P システムは完全自立分散システムのため耐障害性が高く, 特定の管理システムを要求しないため, 将来の機能拡張が行いやすいという利点があるが, データ到達時間の偏差が大きく, リアルタイム性に欠けるため, 各ノードが保持するデータが無矛盾であることを保証することが困難であるという問題点がある. Yahoo! Messenger のようにメタ情報をサーバに持たせる Hybrid P2P 方式ならば, この問題点はある程度解決される. しかし, サーバが必要になるため, サーバが使用できないとシステム全体が動作しなくなる, などのクライアント・サーバ方式の問題点を部分的に引き継いでしまう. この問題を解決する方法として, Pure P2P システムにおいて CAN[2] や Chord[3, 4] のような分散ハッシュ表 (Distribution Hush Table, DHT) によるデータの検索方法が知られている. しかし, DHT による分散データベースは検索を効率的に行う仕組みであり, ある特定のデータに対して多くのノードからアクセスがあった場合, 要求されたデータを持つ計算機にアクセスが集中してしまうという問題は依然として残る.

1.3　1 対多通信による情報共有

1 対多の通信を利用した情報共有はリアルタイム性を持ち, 拡張性が高く, 部分的な故障に対して強いという利点がある. しかし, 通常の 1 対多通信は受信者による確認応答 (Ack) が全くないか, すべての受信者が送信者に確認応答を返すかのどちらかである. 確認応答のないものは信頼性に欠け, すべての受信者が送信者に確認応答を返す方法は確認応答が送信者に集中する (Ack Explorsion) という問題がある. そこで本論では, リアルタイム性を持つ情報共有の方法として, 確認応答つき 1 対多通信の改良を提案する. 本提案においては確認応答を分散して行うことにより, 送信者に対する確認応答の集中を避けることができる. これによりリアルタイム性を持ち, データが無矛盾で拡張性が高い情報共有方式が実現可能である. また, 確認応答により, 通信路及び各ノードが正しく動作しているかどうか常時検査を行う. この性質はシステムが使用できないときに警報を発するという完全性の確保にも有効である. この手順は広範囲にわたる応用が可能だが, まずは調整席の管制官が隣接セクタとの情報共有ができることを目標とする. 調整業務の負荷軽減に着目した研究は多くないが, 本提案により調整間に

[1]例えば 1000 台のクライアントが存在する場合, 計算機の能力が十分大きかったとしても, TCP/IP で 1000 セッションを確立することはかなり面倒である.

係る業務負荷の軽減が図られることを希望する.

2　航空管制に係る通信の性質

航空管制に係る通信はいくつかの性質がある. 第一に, 情報が正しく相手に届いたかどうかは重要である. そのため, 必ず確認をとる. 例えば, 管制官はパイロットに指示を復唱させる. 第二に, 管制指示のように重要度が高い情報はリアルタイム性が高く情報量は少ない. 逆に, 気象情報のような重要度が低い情報はリアルタイム性は低く情報量が多い. 第三に, 数万の航空機が同時に飛行している状況は少なくとも我が国では近い将来も含め考えにくい. 第四に, フェールセーフ, またはフェールソフト性が求められる. 例えば 1 つのノードが故障する場合, ある部分がすべて故障する場合, そして自分以外の全部が故障している, すなわちスタンドアロン運転のそれぞれの場合につき, 可能な限り故障していない部分による縮退運転を行うことができることが望ましい. 今回の想定は地対地通信であるので, 通信手段としては, 一般的なディジタルパケット通信を仮定する. 以上のような性質より, 以下のような前提条件の設定を行った.

3　提案手法

3.1　前提条件

リアルタイム性がなく情報量が多い通信は従来手法のカバーする範囲であると考える. そこで本論では重要度の高い, 情報量が少なくリアルタイム性と信頼性を必要とする通信を構成する方法について論じる. 情報量の少ない通信は, パケットに連番を付与することによるパケット損失の検出は難しい. パケットに連番を付与する方法によるパケット損失の検出は, 到着したパケットの連番が非連続であることによりパケットの損失を検出する. このとき, 最後のパケットは後続のパケットがないためパケット損失が検出できない. また, 不完全ながら通信が行えている場合はパケット損失を検出できるが, 全く通信ができない場合にパケット損失を検出できない. そこで本論では逆に, 1 パケットに必要なデータがすべて入るものと仮定する. これはかなり強い仮定である. しかし，前述のような航空管制に係る通信の特徴より, 重要度とリアルタイム性が高い情報の情報量は小さいと仮定することができる. また, 通信相手のアドレスは事前にすべて知っていること, 通信相手のリストはすべての参加者が共有していること, 通信相手に頻繁な増減はないことを仮定する. また, 将来的な動向も考え, 本論では, 通信メディアとして Ethernet[5] と Internet Protocol (IP)[6, 7] を利用して通信すると仮定する.

3.2　従来技術

IP による 1 対多の通信はマルチキャストとブロードキャストの 2 種類がある. ブロードキャストはルータを超えた通信ができないので広範囲の通信には向いていない. 信頼性の高いマルチキャストを実現するための Reliable Multicast という技術が研究されている [8, 9]. Reliable Multicast は Transmission Control Protocol[10] (TCP) と部分的に同様の機能を持つマルチキャストの実装を目指す技術である. 全ての点で TCP と同じ機能を持つ単一のマルチキャストのためのプロトコルを実現することは難しいので, TCP の利点のうち部分的なものを実装する building blocks という形で検討が進められている. この中には誤り訂正符号を付加することによって伝送路の信頼性を得る方法, 確認応答をルータにより集約する Tree Based Ack, 情報が届かなかったことを送信者に通知する NACK (Not Ack) などが提案されている [11].

しかしこれらはマルチキャストの利点のうち, 従来から着目されている利点である, 多くのユーザに対して映像のように大規模なデータを配信することを目的とした技術であり, リアルタイム性を求めているわけではない. また, 誤り訂正符号の付加により伝送路の信頼性を得ても相手に情報が伝達できたかどうかはわからないことから, TCP の完全な代替にはなり得ない.

3.3　情報共有手順

ここでは n 個のノードが情報共有を行うため相互に接続されたネットワークを考える[2]. 各ノードに番号 $0, 1, 2, \ldots, n-1$ を振る. 各ノードは自分の番号を知っているものとする. $A = \{0, 1, \ldots, n-1\}$ とする. 各ノードの送信先アドレスは別の方法を使用し, すべてのノードで共有しているものとする.

以上の状況において, 一般性を失うことなくノード 0 が情報を発信するものとする. 提案する手順は以下の通りである.

Step 1 ノード 0 は全ノード $\forall i \in A$ に対して情報を送信する. これはマルチキャストとして実現される.

Step 2 全ノード $\forall i \in A$ はそれぞれ, 写像 $adj : A \to 2^A$ により, 隣のノードの集合 $B_i = adj(i)$ を選定す

[2] ここではマルチキャストに参加するホストをノードと定義しており，マルチキャストに参加していないクライアントや途中のルータ等は含んでいないのでネットワーク全体をグラフ $G = (V, E)$ としてモデル化したときに $|V| \neq n$ であることに注意せよ.

る[3]. *adj* の詳細については後述する.

Step 3 ノード i はすべての隣のノード $\forall j \in B_i$ に対して 0 からの情報が届いたかどうか, ユニキャストにより問い合わせを行う.

Step 4 ノード i は隣のノード $j \in B_i$ それぞれから 0 からの情報が届いたかどうかユニキャストにより確認応答を受ける. ここで, 自分と相手が同じ情報を持っているかどうか確認を行うために MD5[12] や SHA-1[13] のようなハッシュ関数により計算されたハッシュ値を確認応答に含める.

Step 5 ノード i はすべての隣のノード $\forall j \in B_i$ より確認応答がない場合, ノード 0 に再送要求をユニキャストにより送信する.

Step 6 ノード 0 は再送要求が来るか, 確認応答が来なかった場合に全ノード A に対して再送を行う. タイムアウト時間の決定はいくつかの研究が知られている. ここではパケットにタイムスタンプを付加することによる往復時間 (Round Trip Time, RTT) の計測による方法 [14, 15] を利用することとする. タイムアウト時間を決定するためのアルゴリズムに関する研究に関しては [14, 15] の参考文献等を参照されたい. ここで $0 \in A$ より, ノード 0 も上記 2,3,4 の作業を行うことに注意せよ.

3.4 利点と問題点

上記情報共有手順の利点は, 完全な分散システムであるため, Pure P2P システムと同様, 頑健であり, 耐障害性が高いこと, 一部のノードが故障しても他の部分に影響を与えづらいフェールソフト性を持つこと, そして効率性と拡張性を両立していること, 配送に失敗したときに知らせる, という完全性を持つこと, などが挙げられる. また, この手順はノードが「存在しない」ことを検出できるので, 意図しない disjoin を検出するプロトコルとして応用することも可能である.

問題点としては前述したように, 大容量の情報配送に向かないことが挙げられる. しかし, 本論で述べた手順は, 重要でリアルタイム性が高く, 少ない情報を共有するための手順として設計しているので, この問題点は設計の前提であり, 制約事項ではあるが問題点ではないと考えている.

この手順は写像 *adj* の選び方により性質が異なる. 以下, その点について考察する.

[3] 2^A は A の巾集合を表す. 例えば $A = \{0,1,2\}$ なら, $2^A = \{\{\}, \{0\}, \{1\}, \{2\}, \{0,1\}, \{0,2\}, \{1,2\}, \{0,1,2\}\}$ である.

3.4.1 決定的な場合

$adj(i) = \{i+1 \pmod n\}$ の場合について, 動作, パケット総数の期待値と成功率について考える.

この場合は, ノード i は ノード $i+1 \pmod n$ に情報到達の確認を行う. そのため, ノード $i+1 \pmod n$ に情報が届いておらず, ノード i に情報が届いていれば, ノード i はそのことを検出し, ノード 0 に報告を行う. そのため, 情報が届かないノードの存在をノード 0 は知ることができる. 番号の連続したノード $k, k+1, \ldots, k+l-1$ が故障している場合はノード $k+l$ がノード $k+l-1$ の故障を報告するので, やはり, 故障ノードが存在していることをノード 0 は知ることができる.

この場合の利点は, 第一に *adj* が全射であるので, すべてのノードに対し情報送信のたび必ず確認を行うこと, 第二に動作が単純で判りやすいこと, 第三に効率的に配送確認を行うことができることである. これは以下の理由による. 近年普及しているレイヤ 2 スイッチによるネットワークでは各ノードが行う確認作業である Step 3 及び Step 4 はそれぞれ並列処理が可能である. そのため処理時間は, Step 1, Step 3 及び Step 4 を合わせて最大の RTT の 3/2 倍しか必要としないことが期待される. また, 情報送信が成功した場合のパケット数は Step 1 で 1, Step 3 で n, 及び Step 4 で n であり, 合計 $2n+1$ パケットである. これはノード 0 が TCP によって $n-1$ 個のノードとセッションを確立するために必要なパケット数 $3(n-1)$ よりも少ない.

3.4.2 確率的な場合 1

$|adj(i)| = 1$ であり, その選び方が一様ランダムである場合について, パケット総数の期待値と成功率について考える. この場合は, ノード i は一様ランダムに選ばれたノードに対して情報到達の確認を行う.

パケット数は決定的な場合と同じである. 情報を送信できていないノードが存在するにも関わらず正しく送信が行われたと誤る確率の下界 $P(n)$ は以下のように与えられる.

$$P(n) = \left(1 - \frac{1}{n}\right)^n$$

$P(n)$ は n に関して単調増加であるため $n \geq 6$ ならば

$$\frac{1}{3} < \left(\frac{5}{6}\right)^6 \leq P(n) < \lim_{n\to\infty} P(n) = \frac{1}{e}$$

が成り立つ. また, $n \geq 2$ ならば $P(n) \geq 1/4$ である. この場合, 決定的な場合のように確率 1 の送達確認はできない. しかし, 故障して応答しないノードが存在する場合, 1 度情報を送信するごとに 1/4 より大きな確

率で故障ノードを発見できる. この確率は $n \geq 6$ ならばこの確率は 1/3 に改善され, どんなにノード数が増加しても 1/2 になることはない. 情報の送信を行うごとにこの検査を独立試行として行うので, 期待値 4 回 ($n \geq 6$ なら 3 回) の情報送信により故障ノードを発見できる.

この手法の利点と欠点は, 決定的な場合と異なり, 確率的な確認になるので確率 1 での検出はできない. しかし確率的な検査は想定外の故障に対して頑健であり, 予想されない故障に対処可能な可能性が大きいという重要な利点がある. また, 複数のノードが同時に故障した場合, 故障ノード数が n よりも十分小さければ, 故障ノードを同時に発見できる可能性もある. この利点がありながら, 期待値わずか 3 回で故障ノードを発見できるという効率の低下は問題ではない可能性もある.

3.4.3 確率的な場合 2

$|adj(i)| = 1$ のとき, 高い確率で近くのノードを選び, 低い確率で遠くのノードを選択することとする. このときのパケット数の期待値は決定的な場合と同じであるが, ネットワークをパケットが通過するコストは安くなることが期待される. 具体的な例としては以下のような手順が考えられる.

ノード $i \in A$ と $j \in A$ の間の**距離** $d(i, j)$ を i から j までのホップ数とする[4]. $i, j \in A$ に対し最大の距離 $\max_{i,j \in A} d(i, j) = r$ を**直径**とする.

全てのノード $\forall i$ が
距離 1 のノードの集合 $A^1 = \{j \in A | d(i, j) = 1\}$,
距離 2 のノードの集合 $A^2 = \{j \in A | d(i, j) = 2\}, \cdots$,
距離 r のノードの集合 $A^r = \{j \in A | d(i, j) = r\}$
を保持していると仮定する. このとき i は隣のノード $B_i = adj(i)$ を以下のように選ぶ.

1. $B_i = \emptyset$ とする.
2. 集合 C を以下のように定める. $k = 1, 2, \ldots, r$ とする. 確率 $1/2^k$ で $C := A^k$ と定める. 確率 $1/2^r$ で $C := A^r$ と定める.
3. 集合 C の中から一様ランダムにひとつノードを選び, それを隣のノード B_i のリストに加える.

3.4.4 複数のノードへの確認

$|adj(i)| > 1$ とすると, ノードの検査を何度も行うことと等価であるので故障ノードの発見率を高めることができる. 決定的な場合は複数のノードが同時に故障したときの発見を, 確率的な場合は故障ノードを発見するまでの試行回数の期待値の低下を, それぞれ期待することができる.

[4] i から j までのホップ数はノード i から j までの間に通過するルータの数である. ホップ数はルーティングに依存することに注意せよ.

4 おわりに

航空に関わるシステムは, 安全性を確保するために非常に高い信頼性を要求される. この高い信頼性を実現するために, 多くのシステムにおいては冗長構成による信頼性の向上が図られている. 例えば, 航空機の操舵システムが 3 重冗長であることは良く知られている. また, 航空に関わるシステムにおいてはリアルタイム性は重要である. この意味において, 冗長性を持ち, リアルタイムな情報共有手段である本手法は, CDM のための情報共有ツールとして適しているものと考えられる.

参考文献

[1] NullSoft. http://www.gnutella.com/, March 2000.

[2] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A Scalable Content-Addressable Network. In *Proceedings of the ACM SIGCOMM*, pp. 161–172. ACM Press, August 2001.

[3] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the ACM SIGCOMM*, pp. 149–160. ACM Press, August 2001.

[4] Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *IEEE Transactions on Networking*, Vol. 11, pp. 17 – 32, February 2003.

[5] IEEE LAN / MAN Standards Committee. IEEE 802.3 LAN / MAN CSMA / CD Access Method.

[6] Jon Postel. Internet Protocol. IETF RFC791, September 1981.

[7] Stephen E. Deering and Robert M. Hinden. Internet Protocol, Version 6 (IPv6) Specification. IETF RFC2460, December 1998.

[8] M. Handley, S. Floyd, B. Whetten, R. Kermode, L. Vicisano, and M. Luby. The reliable multicast design space for bulk data transfer. IETF RFC2887, August 2000.

[9] J. C. Lin and S. Paul. Rmtp : A reliable multicast transport protocol. In *Proceedings of IEEE INFOCOM*, pp. 1414–1424, 1996.

[10] Jon Postel. Transmission Control Protocol. IETF RFC793, September 1981.

[11] Allison Mankin, Allyn Romanow, Scott Bradner, and Vern Paxson. Ietf criteria for evaluating reliable multicast transport and application protocols. IETF RFC2357, June 1998.

[12] Ronald L. Rivest. The MD5 Message-Digest Algorithm. IETF RFC1321, April 1992.

[13] Donald E. Eastlake 3rd and Paul E. Jones. US Secure Hash Algorithm 1. IETF RFC3174, September 2001.

[14] Van Jacobson, Bob Braden, and Dave Borman. TCP Extensions for High Performance. IETF RFC1323, May 1992.

[15] W. Richard Stevens. *UNIX Network Programming, Second Edition*, Vol. 1. Prentice Hall PTR, 1998. (篠田陽一訳「UNIX ネットワークプログラミング 第 2 版 Vol.1」ピアソン・エデュケーション (2002)).