3H-2　Ethernet上での全順序放送通信プロトコルの設計と実現

中村　章人、滝沢　誠
(東京電機大学理工学部)

1．はじめに

現在、複数の通信実体の協調動作が分散型データベースシステムで同時実行制御、コミットメント制御、分散型問合せ処理等を実現する上で必要とされており、このためには、複数の実体間での高信頼放送通信が求められている。本論文では、EthernetのMAC層で提供されている低信頼放送サービスを利用して複数の実体に対して全順序放送(TO)サービスを提供するプロトコルについて述べる。TOサービスを利用すると、全実体が同一のPDUを同一の順序で受信する。また、本プロトコル(TOプロトコル)の実現方法についても述べる。

2．信頼性

従来のプロトコルでは、各実体はPDU pが正しく受信されたことを、pが到着し、かつpがあるある条件を満たすこと、例えば通番のチェック等によって決定する。しかし、放送通信サービスに対する正しい受信の概念はこれより複雑である。正しい受信を決定するための方法として、ある1つの主制御実体がこれを行う集中型制御と、各実体が互いに通信し合うことによって自分自信で正しい受信の決定を行う分散型制御がある。TOプロトコルでは、後者を用いている。

[定義] 各実体E_kは、以下の条件を満たすとき、PDU pを正しく受信したという。(1)E_kはpを受信する。(2)E_kは「pの宛先内の全実体がpを受信した」ことを知る。(3)E_kは「pの宛先の各実体は、全宛先実体がpを受信したことを知っている」ことを知る。□

(1)、(2)、(3)をそれぞれpはE_kで受信された、前確認された、確認されたという。複数の実体がPDUを放送する状況で、各実体が同一のPDUを同一の順序で受信するような高信頼放送サービスを定義する。

[定義] 各PDU pとqが共通の宛先のSAPで同一の順序で受信されるとき、かつそのときに限りこの(N-1)高信頼放送サービスを全順序(TO)サービスという。□

従来の2つのSAP間のコネクションの概念をn(≥2)個に拡張したものが、群(cluster)である。

[定義] 1チャネル(1C)サービスとは、PDUの紛失の可能性はあるが、受信されたPDUは全実体で同一の順序である低信頼放送サービスである。□

LANのMAC層と無線網は1Cサービスを提供している。

3．1Cサービス上での全順序(TO)プロトコル

群はn個の実体$E_1,\ldots,E_n$で構成される。各実体E_kが送信したPDUをE_kは受信できるとする。各実体E_kについて、送信ログ$SL_k=(SP_k,<_k)$をE_kが既に送信したPDUの系列、受信ログ$RL_k=(RP_k,<_k)$をE_kが既に受信したPDUの系列とする。ここでSP_kとRP_kは各々、E_kが送信したPDUと受信したPDUの集合である。E_kがq以前にpを受信したときRL_k内で$p <_k q$であり、q以前にpを送信したときSL_k内で$p <_k q$である。$p <_k q$で$p <_k g$かつ$g <_k q$なるgが存在しないとき$p \rightarrow_k q$である。

[定義] RL_k内の全てのpとqに対して、もしpとqがE_jによって放送され、SL_j内で$p \rightarrow_j q$で、RL_k内で$p <_k q$ならば受信ログRL_kは正しいという。□

RL_k^fを、RL_kの先頭からf番目までの部分系列とする。

[定義] 受信ログ$RL_1,\ldots,RL_n$に対して、以下の条件が成り立つとき、かつそのときに限りPDU番号fを障害点FPとする。(1)全てのjとkについて、$RL_j^{f-1} = RL_k^{f-1}$で、これらの受信ログは正しい。(2)あるjについて、RL_j^fは正しくない。□

FPの存在は、ある実体があるPDUを受信できなかったことを意味する。APL_k、PPL_k、RPL_kをそれぞれ確認されたPDU、前確認されたPDU、受信されたPDUからなるS_kの部分ログとする。E_kが放送するPDU pはE_kが既に受信したPDUの確認通知を含んでいる。E_kが放送する各PDU pは以下の情報を持っている。

$p.A_j$ = E_kが次にE_jから受信予定のPDUの通番 (j=1,...,n)。
p.SEQ = pの通番。
p.SRC = pを放送する実体、つまりE_k。
p.BUF = E_k内で利用可能なバッファ数。

各E_kは、変数SEQ_kとPDUの通番をチェックするためのマトリックス$AL_{pq}(p,q=1,\ldots,n)$を持っている。

SEQ_k = E_kが次に放送する予定のPDUの通番。
AL_{jj} = E_kが次にE_jから受信予定のPDUの通番。
AL_{hj} = E_jが次にE_hから受信予定のPDUの通番。を、$AL_{j1},\ldots,AL_{jn}$の中の最小値$minAL_j$は、全実体がg.SEQ < $minAL_j$なるPDU gをE_jから既に受信していることを意味している。群内の各実体は、群開設手続き[TAK87a,b]によって各E_jの初期通番ISS_jと、空バッファ数IBF_jを知っている。最初は$SEQ_k=ISS_k$で$AL_{hj}=ISS_j(h,j=1,\ldots,n)$。各実体は変数$F_1,\ldots,F_n$を持ち、$F_j$は$E_j$内の空バッファ数であり、最初$F_j=IBF_j(j=1,\ldots,n)$である。minFを$F_1,\ldots,F_n$の中の最小値とする。

PDU pの送受信は、以下の条件を満たすとき行われる。

[受信条件] (1)$p.SEQ = AL_{jj}$.(2)$p.A_h < AL_{hh}$. (h=1,...,n).□

[受信動作] (1)$AL_{jj} = AL_{jj} + 1$, $AL_{hj} = p.A_h$ (h=1,...,n).(2)$F_j = p.BUF$.(3)pをRL_kの最後尾、つまりRPL_kの最後尾に追加する。□

[フロー条件] (1)$minAL_k \leq SEQ_k < minAL_k + \min(W, minF/n^2)$.(2)各実体がPDUを受信可能である。□

[送信動作] (1)$p.BUF=F_k$, $p.A_j=AL_{jj}(j=1,\ldots,n)$. (2)$p.SEQ=SEQ_k$, $p.SRC=E_k$, $SEQ_k=SEQ_k+1$.(3)E_kはpを放送する。□

フロー条件が満たされなければ、E_kは十分なバッファが利用できるようになるまで放送を待つ。

分散型制御において各実体E_kがpの正しい受信をど

Totally Ordering Broadcast Protocol on Ethernet
Akihito NAKAMURA, Makoto TAKIZAWA
Tokyo Denki University

のように決定するかが問題となる。ここで、pをE_jが放送したPDUとする。

[前確認(PACK)条件] p.SEQ < $minAL_j$.□

[前確認動作] RPL_k内の先頭からpまでのPDUをRPL_kから取り出し、PPL_kの最後尾に追加する。□

PPL_k RPL_k

< ...a] < b...p q...] ===>

< ...a b...p] < q...]

[定義] 各E_kとRL_k内のPDU pとqに対して、qがE_hでpを前確認し、E_hでpを前確認するg $<_k$ qなるPDU gが存在しないときqはE_hについてpを最初前確認するPDUという。各E_jについて、pを最初前確認するPDUの中で、最後に受信したものをpを最後前確認するPDUという。□

[確認(ACK)条件] PPL_k内のPDU pを最後前確認するPDUが前確認される。□

[確認動作] PS_k内の先頭からpまでのPDUをPPL_kから取り出し、APL_kの最後尾に追加する。□

APL_k PPL_k RPL_k

< ...a] < b...p q...] < .g..] ===>

< ...a b...q] < q...g] < ..]

PDUが紛失した点、つまり障害点FPはE_kがPDU pを受信する度に発見し、リセット手続きが適用される。

[紛失条件] (1)E_jからpを受信したとき、p.SEQ >AL_{jj}ならば、$AL_{jj} \leq$ g.SEQ < p.SEQなるPDU gをE_jから受信していない(j=1,...,n)。(2)E_hからqを受信したとき、あるj(j≠h)について、q.A_j > AL_{jj}ならば、E_kは$AL_{jj} \leq$ g.SEQ < q.A_jなるPDU gをE_jから受信していない(h=1,...,n)。□

[リセット手続き] (1)E_kはPDUの受信を停止し、r.A_h=AL_{hh}(h=1,...,n)なるRST(RESET) PDU rを放送し、SEQ_k=AL_{kk}とする。(2)E_jがE_kからRST rを受信したら、各hについてRL_kがr.$A_h \leq$ p.SEQなるE_hからのPDU pを含むならば、p $<_k$ gである全てのPDU gをRL_kから取り除く。同時に、ALリセット手続きを適用してALを更新する。E_jはPDUの受信を停止し、ra.A_h=AL_{hh}(h=1,...,n)なるRST_ACK PDU raを放送し、SEQ_j=AL_{jj}とする。(3)全実体からRST_ACKを受信し、各hについてr.A_h=AL_{hh}であれば、PPL_kとRPL_k内のPDUをAPL_kに移す。□

[ALリセット手続き] 各E_jとp=last(RL_{kj})に対して、AL_{jh} = p.SEQとする(h=1,...,n)。□

PDUの紛失は、タイムアウト処理によっても発見される。ある実体E_kがPDU pを放送したにも関わらず、E_kを含む全実体がpを受信していないとする。E_kがp→$_k$gなるPDU gを放送しなければ、各実体はpの紛失を発見出来ない。E_kはpを放送した後、タイムアウトが起きたらpの紛失を発見する。この場合、どの実体もpを受信していないので、E_kはpを再放送できる。

4. 評価

本プロトコルの性能を、PDU数と必要時間で評価する。PDU数は、PDU pを確認するために送信されるPDU数とする。ラウンドとは、あるSAPから宛先までの最大遅延時間である。必要時間は、ラウンド数とする。nを実体数としたとき、性能は以下のようになる。

PDU数　最良 1+(n-1)+n = 2n 個

最悪 1+(n-1)+$(n-1)^2$ = n^2-n+1 個

必要時間　最良 3 ラウンド

Ethernet MACサービスの様にPDUを並行放送できない場合　最良 1+(n-1)+n = 2n ラウンド

最悪 1+(n-1)+$(n-1)^2$ = n^2-n+1 ラウンド

5. TOプロトコルの実現

データ転送手続きは、3つのキューPRQUE、ARQUE、PTQUEにより実現されている。各E_kの部分ログPPL_k、RPL_kは各々ARQUE、PRQUEに記憶される。

[送信] フロー条件が満たされるならば、PDU pをPTQUEに記憶し、pを放送する。□

[受信] E_jからpを受信したら、(1)もしpが受信条件を満たせば、pを受け入れPRQUEに記憶する。(2)PRQUEの先頭のPDU qが前確認条件を満たす間、qをPTQUEから取り出しARQUEに記憶する。p.SRC = E_kならば、pをPTQUE、つまりPTQUEの先頭から取り出す。q.RPT=gであれば、ARQUEの先頭からgまでのPDUをARQUEから取り出す。つまり、E_kで確認される。もしPRQUEから最後に取り出したPDUがqであればp.RPT=qとする。(3)もし放送するデータがあれば送信動作を行い、そうでなければ確認通知を含むPDUを放送する。□

[障害] (1)もし紛失条件が成り立てば、リセット手続きを行う。□

6. まとめ

現在、TOプロトコルをEthernet MACサービスを用いて、Unix 4.2BSDとSystem V上でC言語で実現している。プロセス間の通信には、Unixのsocketインターフェイス　を用いている。Ethernet上には、現在m380q、a400 (Facom)、Sun3ワークステーション3台が接続されている。現在、このシステムを使ってTOプロトコルの性能評価を行っている。問題はバッファサイズで、本プロトコルでは、従来の一対一通信に比べて最悪の場合、n^2倍以上のバッファが必要になる。しかし、今日のハードウエア技術により記憶装置のコストは非常に低くなっているので、この問題は数M byteの記憶装置を用いることで解決できると思われる。

参考文献

[BRA] Bracha, G. and Toueg, S., "Asynchronous Consensus and Broadcast Protocols," JACM, Vol.32, No.4, 1985, pp.824-840.

[CHA] Chang, J-.M. and Maxemchuck, M.F., "Reliable Broadcast Protocols," ACM TOCS, Vo;.2, No.3, 1984, pp.251-273.

[NAK] 中村 章人,滝沢 誠,"多チャネル上の全順序放送通信プロトコル",情報処理学会MDP研究会39-1,1988

[SCH] Schneider, F., Gries, D., and Schlichting, R.D., "Fault-Tolerant Broadcasts," Science of Computer Programming, Vol.4, 1984, pp.1-15.

[TAK87a] Takizawa, M., "Design of Highly Reliable Broadcast Communication Protocol," Proc. of IEEE COMPSAC87, 1987, pp.731-740.

[TAK87b] Takizawa, M. "Cluster Control Protocol for Highly Reliable Broadcast Communication," Proc. of the IFIP Conf. on Distributed Prrocessing, Amsterdam, 1987.