“高い Churn 耐性と検索性能を持つキー順序保存型構造化オーバレイネットワーク Suzaku の提案と評価” (信学技報 Vol. 116, No. 362 (IA2016-65), pp. 11–16) に掲載した Finger Table 更新処理の擬似コード（図 2）にバグがありました．修正した擬似コードを以下に示します（修正は赤字部分）．

```
1  const FFT = 0, BFT = 1;
2  // ft[FFT] is FFT, ft[BFT] is BFT
3  // ft[FFT][0] = successor, ft[BFT][0] = predecessor
4  var ft[2][]: Array of {Array of NodeAndKey};
5  var R: Set of NodeAndKey; // reverse pointers
6  n.updateBoth(dir, i, n1, n2) {
7    tab ← ft[dir];
8    if (i = 0) n1 ← ft[dir][0]; // succ or pred
9    if (n1 = null or n1 = n) {
10     x ← null; goto next;
11   }
12   if (dir = FFT) {y ← n2;} else {y ← ft[FFT][i];}
13   x ← n1.getEnt(dir, i, n, y);
14   if (n1 failed) { we omit the detail of this case }
15   if (i ≠ 0) change(tab, i, n1, true); // active update
16   if ((dir = FFT and x ∈ [n, tab[i]]) or
17     (dir = BFT and x ∈ [tab[i], n])) x ← null; // circulated
18 next:
19   if (x = null and n2 = null) return;
20   if (dir = FFT) { n.updateBoth(BFT, i, n2, x); }
21   else { n.updateBoth(FFT, i + 1, n2, x); }
22 }
23 n.updateFFT(i, n1) {
24   if (i = 0) n1 ← ft[FFT][0]; // successor
25   x ← n1.getEnt(FFT, i, n, null);
26   if (n1 failed) { we omit the detail of this case }
27   if (i ≠ 0) change(ft[FFT], i, n1, true); // active update
28   if (x ∈ [n, ft[FFT][i]]) { // circulated
29     // truncate FFT and BFT to size i
30     for (j←i to ft[FFT].length−1)change(ft[FFT],i,null,false);
31     for (j←i to ft[BFT].length−1)change(ft[BFT],i,null,false);
32     sleep(T_ft);
33     n.updateFFT(0, null);
34   } else {
35     sleep(T_ft);
36     n.updateFFT(i + 1, x);
37   }
38 }
39 n.getEnt(dir, level, p, q) {
40   if (level ≠ 0) {
41     change(ft[1 − dir], level, p, true);// pasv. update 1
42   }
43   if (q ≠ null) {
44     if (dir = BFT) {
45       change(ft[1−dir], level+1, q, false);// pasv. update 2
46     } else R ← R ∪ {q};
47   }
48   return ft[dir][level];
49 }
50 n.change(tab, level, new, addtorev) {
51   old ← tab[level];
52   tab[level] ← new;
53   if (addtorev and new ≠ null) R ← R ∪ {new};
54   if (old ≠ null and (FFT and BFT do not contain old)) {
55     old.removeRev(n);
56   }
57 }
58 n.removeRev(p) {
59   R ← R \ {p};
60 }
```

図 2　Suzaku: finger table 更新アルゴリズム

Fig. 2　Finger table update algorithm of Suzaku.