ANNEX 1: Ack Vector Explanation

Summary Vectors: They determine the <u>last message received</u> in *principals A, B & C* from other principals…

$Ack_x [Y_A, Y_B, Y_C, …, Y_N]$

Local acknowledgement vector at *principal X*. It tells which operations *X* knows that have been executed on *A,B,C,…,N* <u>from all the other principals</u> with *timestamp* earlier than Y, so *X* can purge all messages from any other *principal* with *timestamp* less than or equal to the **minimal** value for Y in the vector.

Principal A
Summary vector

6	A
10	B
8	C

Principal B

5	A
12	B
8	C

Principal C

5	A
10	B
9	C

$Ack_a [Y_A, Y_B, Y_C]$ $Ack_b [Y_A, Y_B, Y_C]$ $Ack_c [Y_A, Y_B, Y_C]$

Ack vector

| 6 | 4 | 4 | | 5 | 5 | 4 | | 5 | 4 | 5 |

Principal A knows that it already executed **all operations** from A, B & C with <u>*timestamp* less than or equal to 6</u> (since 6 is the **minimal** *timestamp* in *principal A summary vector*).

Principal A knows <u>for sure</u> that *principal B* has already received & executed **all operations** from A, B & C with <u>*timestamp* less than or equal to 4</u>, even though it might have executed operations with *timestamp* later than 4 (in fact, *principal B* has already executed the operations A5, B5 to B12 and C5 to C8 but *principal A* still doesn't know it).

Principal A knows <u>for sure</u> that *principal C* has already received & executed **all operations** from A, B & C with <u>*timestamp* less than or equal to 4</u>, even though it might have executed operations with *timestamp* later than 4 (in fact, *principal C* has already executed the operations A5, B5 to B10 and C5 to C9 but *principal A* still doesn't know it).

Summing-up; *principal A* can purge messages A1, A2, A3, A4, B1, B2, B3, B4, C1, C2, C3, C4 from its *Log* (this is, operations with <u>*timestamp* less than or equal to 4</u>, which is the minimal *timestamp* in the local *Ack$_a$ Vector*), since it's for sure that the other *principals* already received these operations. Thus, after the purge, the <u>*principal A Log*</u> will contain the following messages: <u>A5, A6, B5, B6, B7, B8, B9, B10, C5, C6, C7, C8</u>

Principal B knows <u>for sure</u> that *principal A* has already received & executed **all operations** from A, B & C with <u>*timestamp* less than or equal to 5</u>, even though it might have executed operations with *timestamp* later than 5 (in fact, *principal A* has already executed the operations A6, B5 to B10 and C5 to C8 but *principal B* still doesn't know it).

Principal B knows that it already executed **all operations** from A, B & C with <u>*timestamp* less than or equal to 5</u> (since 5 is the **minimal *timestamp*** in *principal B summary vector*).

Principal B knows <u>for sure</u> that *principal C* has already received & executed **all operations** from A, B & C with <u>*timestamp* less than or equal to 4</u>, even though it might have executed operations with *timestamp* later than 4 (in fact, *principal C* has already executed the operations A5, B5 to B10 and C5 to C9 but *principal B* still doesn't know it).

Summing-up; *principal B* can purge messages A1, A2, A3, A4, B1, B2, B3, B4, C1, C2, C3, C4 from its *Log* (this is, operations with <u>*timestamp* less than or equal to 4</u>, which is the minimal *timestamp* in the local *Ack$_b$ Vector*), since it's for sure that the other *principals* already received these operations. Thus, after the purge, the <u>*principal B Log*</u> will contain the following messages: <u>A5, B5, B6, B7, B8, B9, B10, B11, B12, C5, C6, C7, C8</u>

Principal C knows <u>for sure</u> that *principal A* has already received & executed **all operations** from A, B & C with <u>*timestamp* less than or equal to 5</u>, even though it might have executed operations with *timestamp* later than 5 (in fact, *principal A* has already executed the operations A6, B5 to B10 and C5 to C8 but *principal C* still doesn't know it).

Principal C knows <u>for sure</u> that *principal B* has already received & executed **all operations** from A, B & C with <u>*timestamp* less than or equal to 4</u>, even though it might have executed operations with *timestamp* later than 4 (in fact, *principal B* has already executed the operations A5, B5 to B12 and C5 to C8 but *principal C* still doesn't know it).

Principal C knows that it already executed **all operations** from A, B & C with <u>*timestamp* less than or equal to 5</u> (since 5 is the **minimal *timestamp*** in *principal C summary vector*).

Summing-up; *principal B* can purge messages A1, A2, A3, A4, B1, B2, B3, B4, C1, C2, C3, C4 from its *Log* (this is, operations with <u>*timestamp* less than or equal to 4</u>, which is the minimal *timestamp* in the local *Ack$_c$ Vector*), since it's for sure that the other *principals* already received these operations. Thus, after the purge, the <u>*principal C Log*</u> will contain the following messages: <u>A5, B5, B6, B7, B8, B9, B10, C5, C6, C7, C8, C9</u>