Multi-version Timestamp example

Given the following schedule:

T1	T2	T3
R(q)		
W(q)		
	R(q)	
	W(q)	
R(q)		
W(q)		
	W(q)	
		R(q)
		W(q)
		Commit

Below we trace through the actions.

1) Assume an initial state of q.0 where its Read Time Stamp (RTS) = 0 and its Write Time Stamp (WTS) = 0.
2) T1 performs a read of q. It's going to read the initial state and set the RTS of q.0 to 1.
3) T1 performs a write of q. This creates a new version q.1 and sets the RTS and WTS of q.1 to 1.
4) T2 performs a read of q. It reads version q.1 and sets the RTS of q.1 to 2.
5) T2 performs a write of q. This creates a new version q.2 and sets the RTS and WTS of q.2 to 2.
6) T1 performs a read of q. In normal timestamp ordering, this would abort T1 (and subsequently T2) because T2 had already overwritten q. Since we have kept previous versions, we read q.1 instead of q.2 and the read is successful.
7) T1 performs a write of q. Unfortunately, the RTS of q.1 is set to 2 – meaning T2 has already read a version of q that T1 had yet to produce. This aborts T1.
8) Since T2 had read q.1, when T1 aborted it cascaded into T2 and T2 aborts.
9) T3 performs a read of q. It reads q.0 and sets the RTS of q.0 to 3.
10) T3 performs a write of q. This creates a new version q.3 and sets the RTS and WTS of q.3 to 3.
11) T3 commits. q.0 is no longer needed and is released.

Let's look at another example:

T1	T2	T3	T4
R(q)			
W(q)			
		R(q)	
		W(q)	
	R(q)		
	W(q)		
		W(q)	
			R(q)
			W(q)
Commits			
			Commit
		Commit	

Below we trace through the actions:

1) Assume an initial state of q.0 where it's Read Time Stamp (RTS) = 0 and its Write Time Stamp (WTS) = 0.
2) T1 performs a read of q. It's going to read the initial state and set the RTS of q.0 to 1.
3) T1 performs a write of q. This creates a new version q.1 and sets the RTS and WTS of q.1 to 1.
4) T3 performs a read of q. It's going to read q.1 and set the RTS of q.1 to 3.
5) T3 performs a write of q. This creates a new version q.3 and sets the RTS and WTS of q.3 to 3.
6) T2 performs a read of q. It's going to read q.1 and leave RTS at 3.
7) T2 attempts a write of q. Unfortunately, T3 needed to have read this write, so T2 is aborted.
8) T3 performs a write of q. Since a q.3 version exists, and the RTS of q.3 is still 3, the value is updated.
9) T4 performs a read of q. It's going to read q.3 and set the RTS of q.3 to 4.
10) T4 performs a write of q. This creates q.4 and set RTS and WTS of q.4 to 4.
11) T1 commits. Version q.0 is no longer needed and released.
12) T4 attempts to commit – since it read q.3 and T3 is not yet committed, the commit is held.
13) T3 commits. Version q.1 is no longer needed and released.
14) T4 commits. Version q.3 is no longer needed and released.