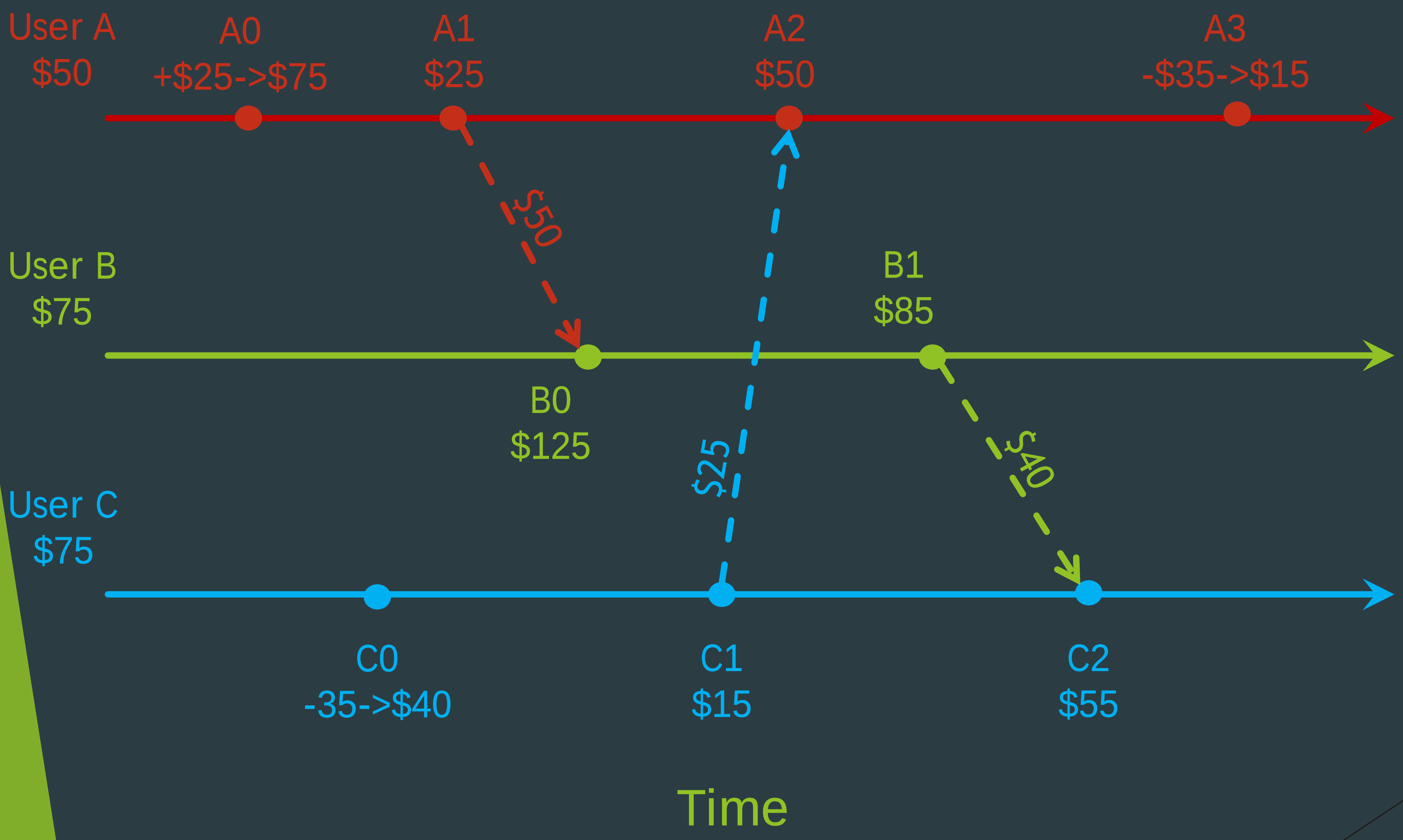
Distributed Snapshots – Lecture 1

Causal Consistency

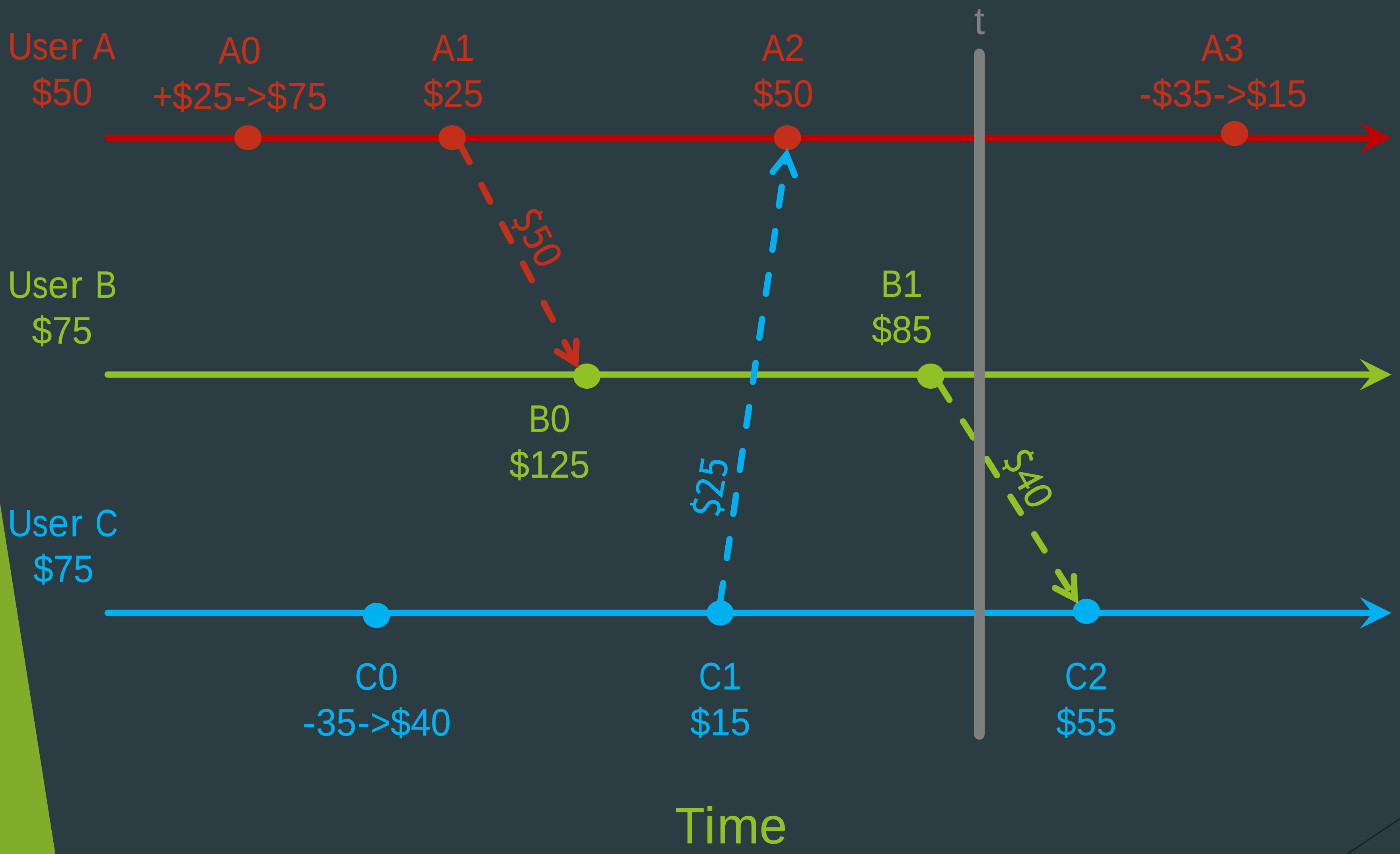
Assumptions

- Failures
  - No failures.
- Network
  - Asynchronous -> A message might take arbitrary time to be delivered.
  - Reliable -> Messages cannot be lost or duplicated while in transit.
  - FIFO -> A network channel maintains the order of messages (e.g. If node A sends message 1 and 2 in that order to B, then B is going to receive them in the same order).
- Clock Synchronization
  - Adjusted to the presentation needs.

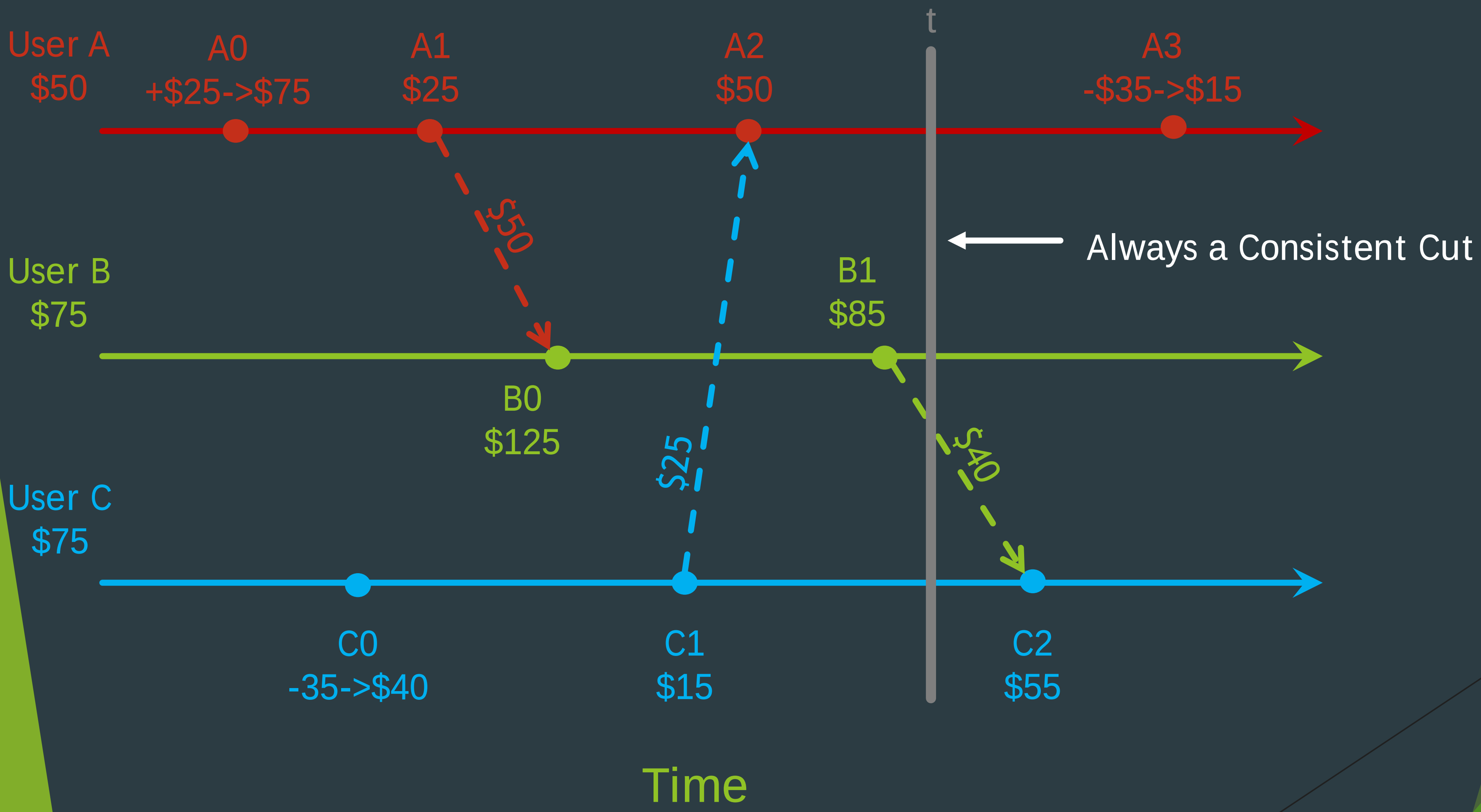
Example – Bank Accounts



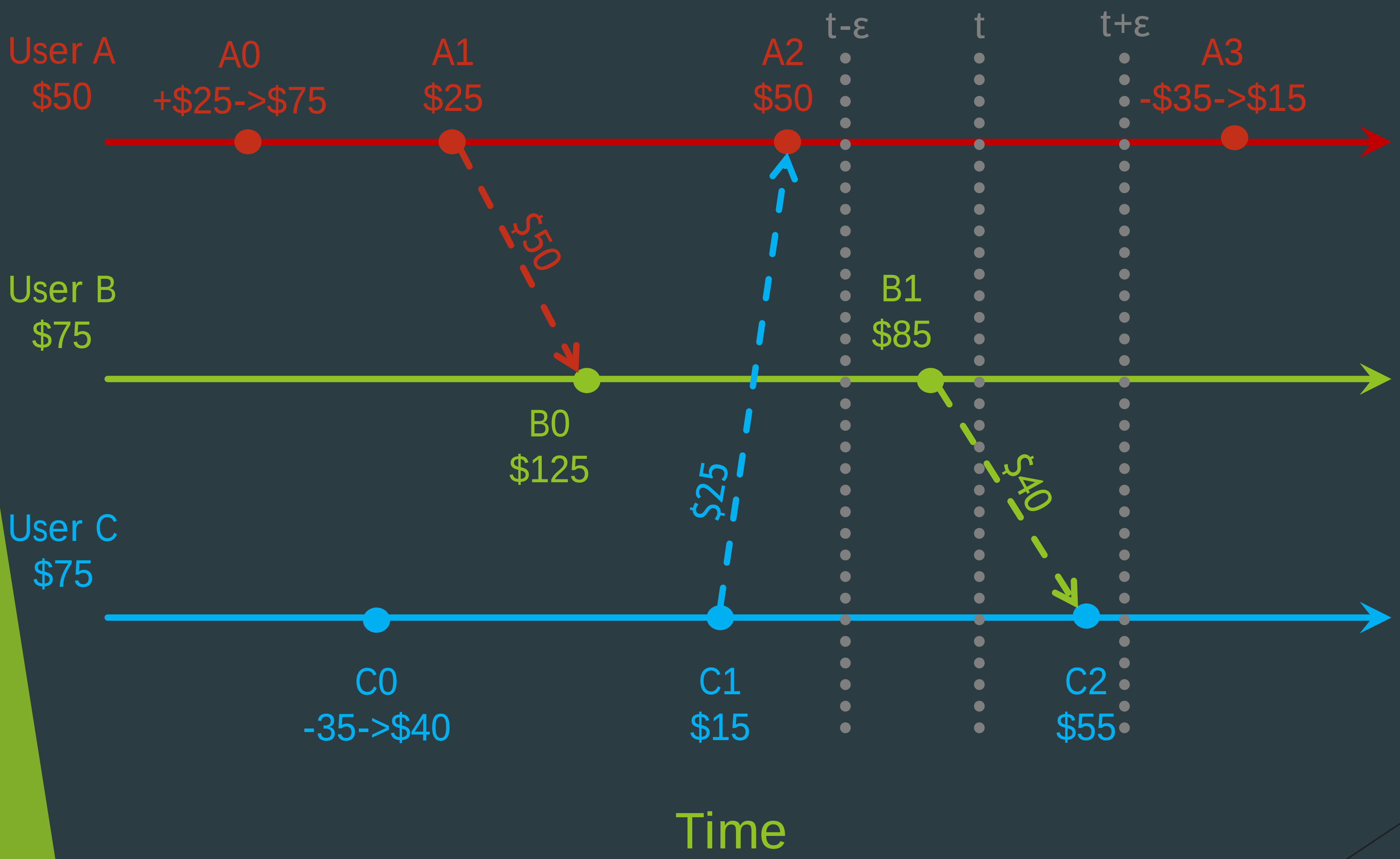
Universal Time



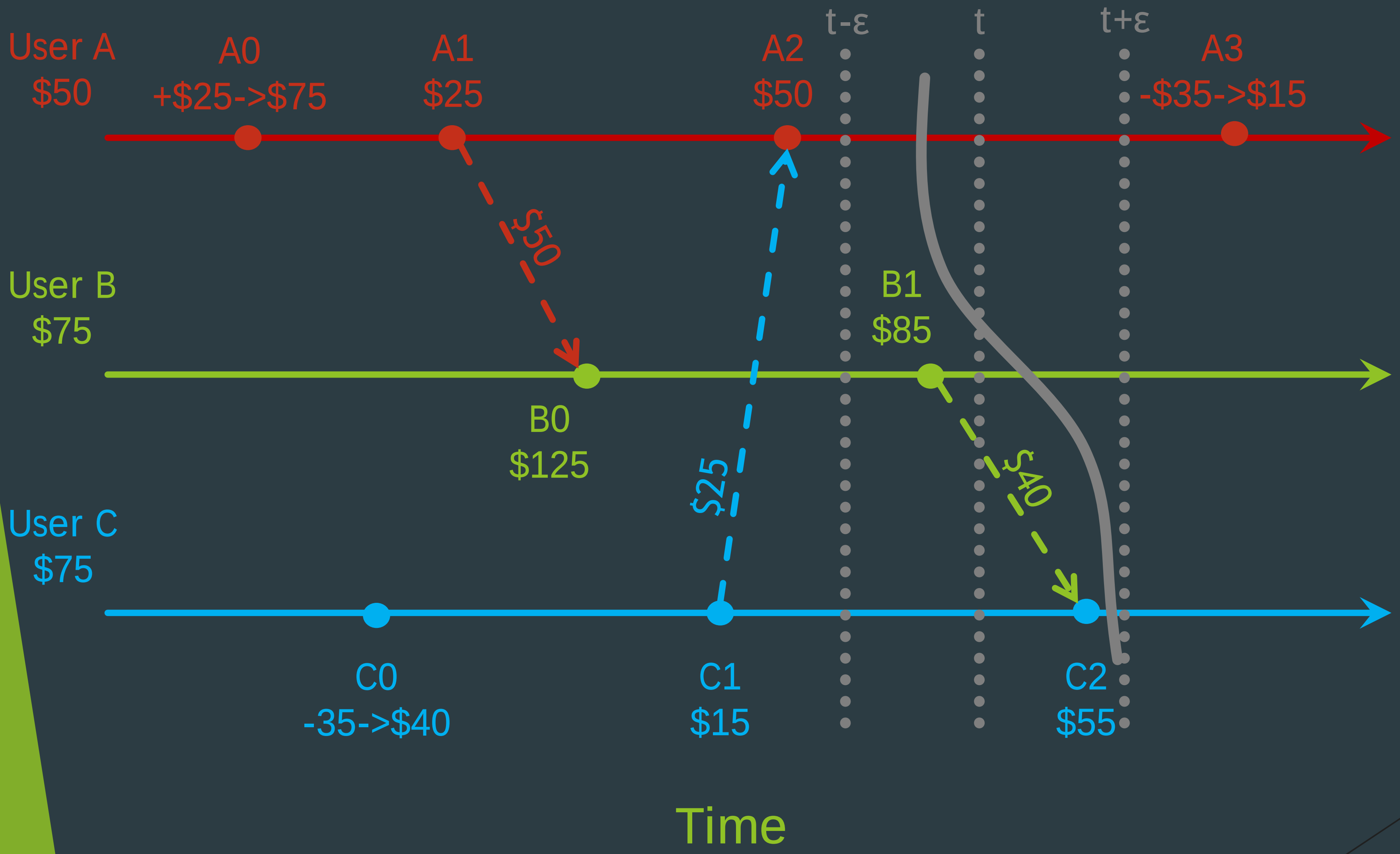
Universal Time



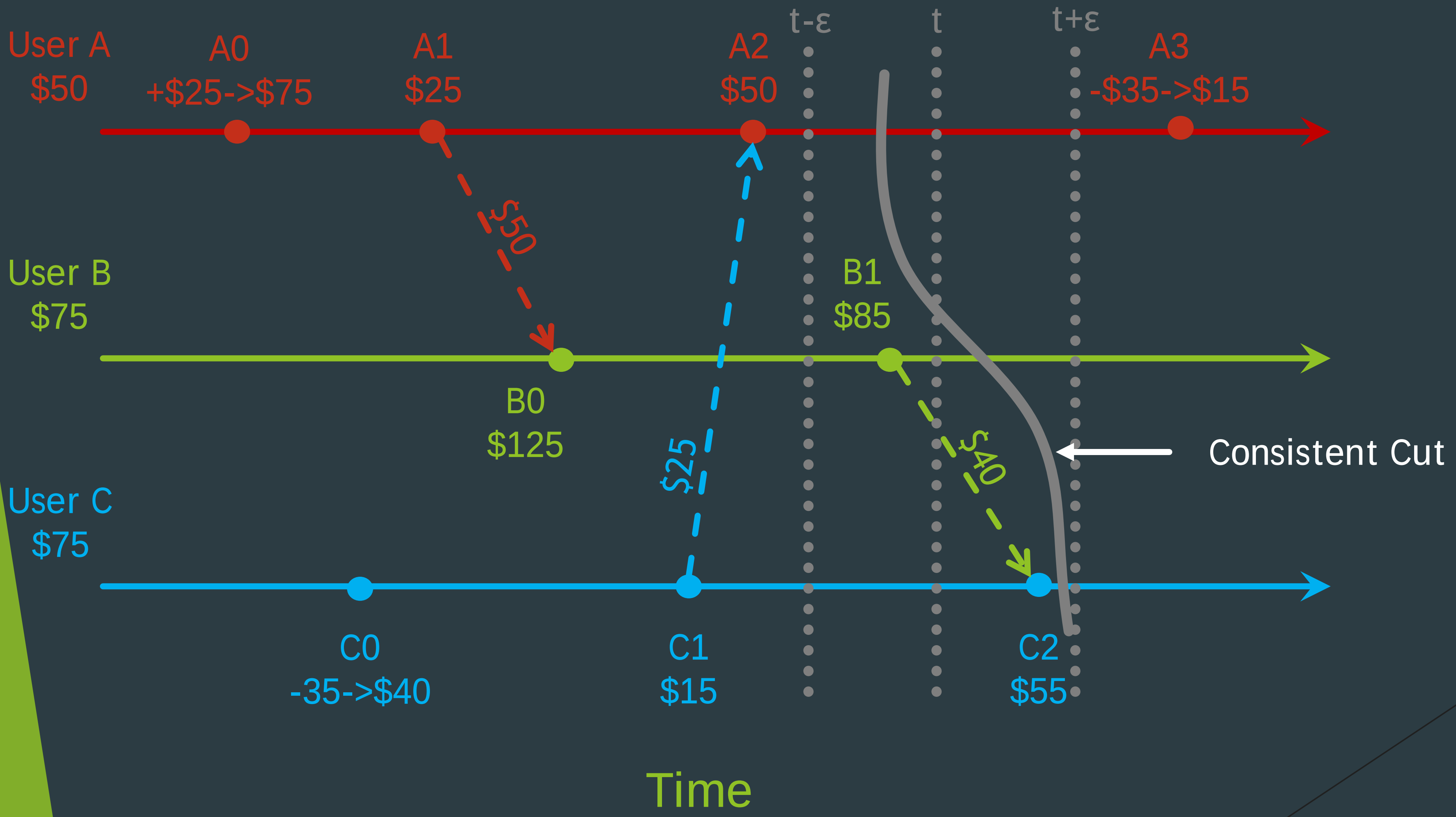
Strongly-Synchronized Clocks



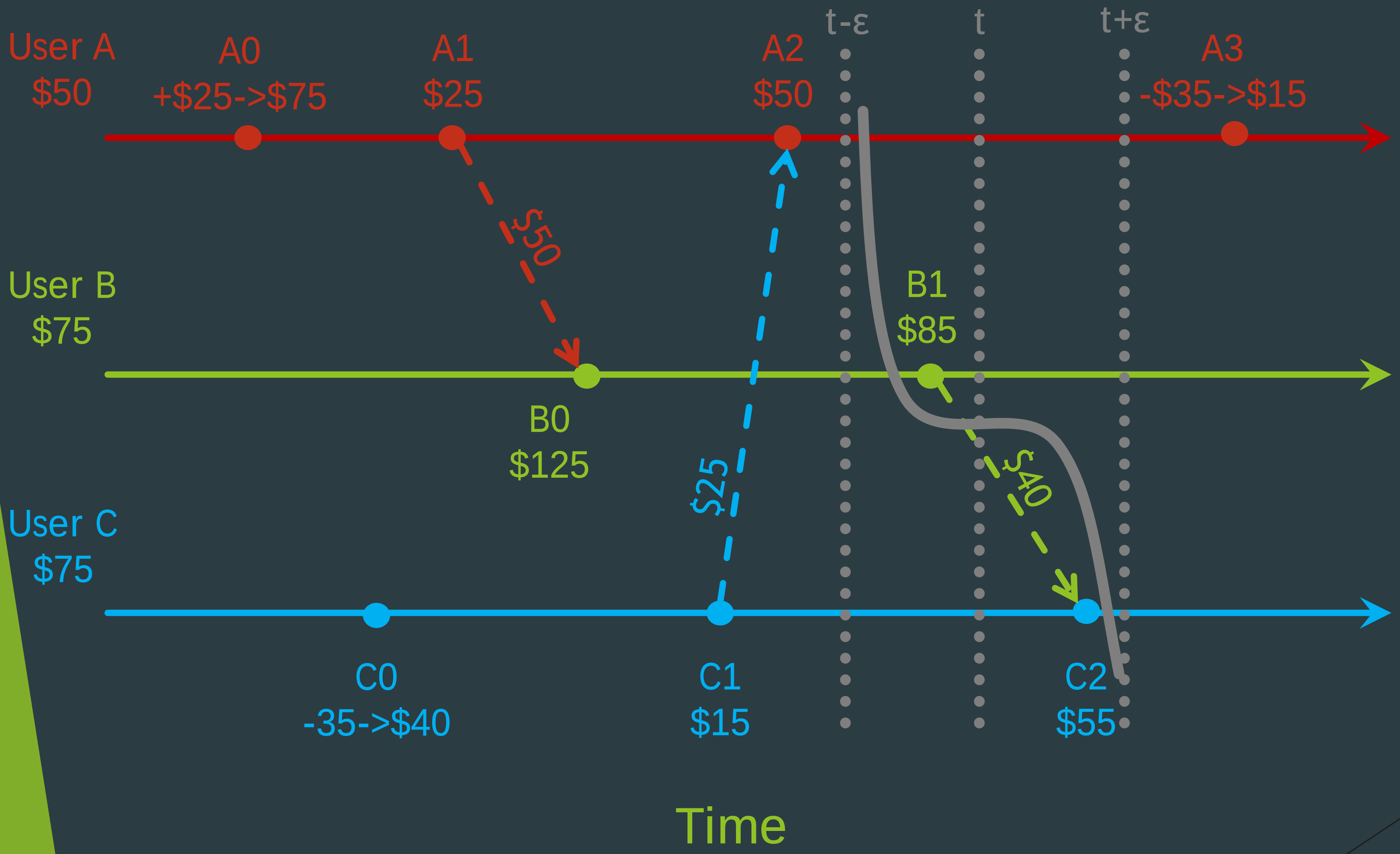
Strongly-Synchronized Clocks



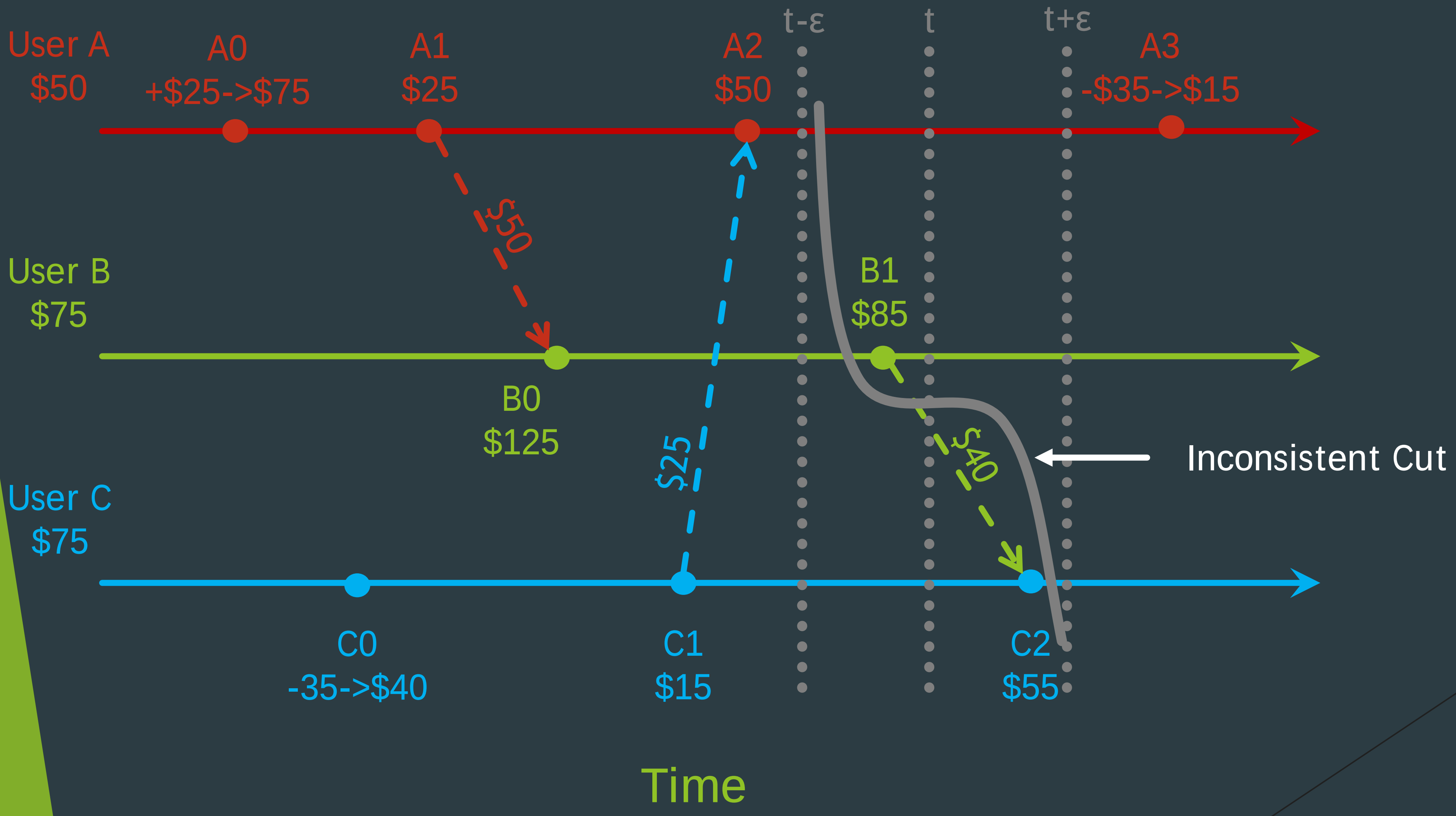
Strongly-Synchronized Clocks



Strongly-Synchronized Clocks



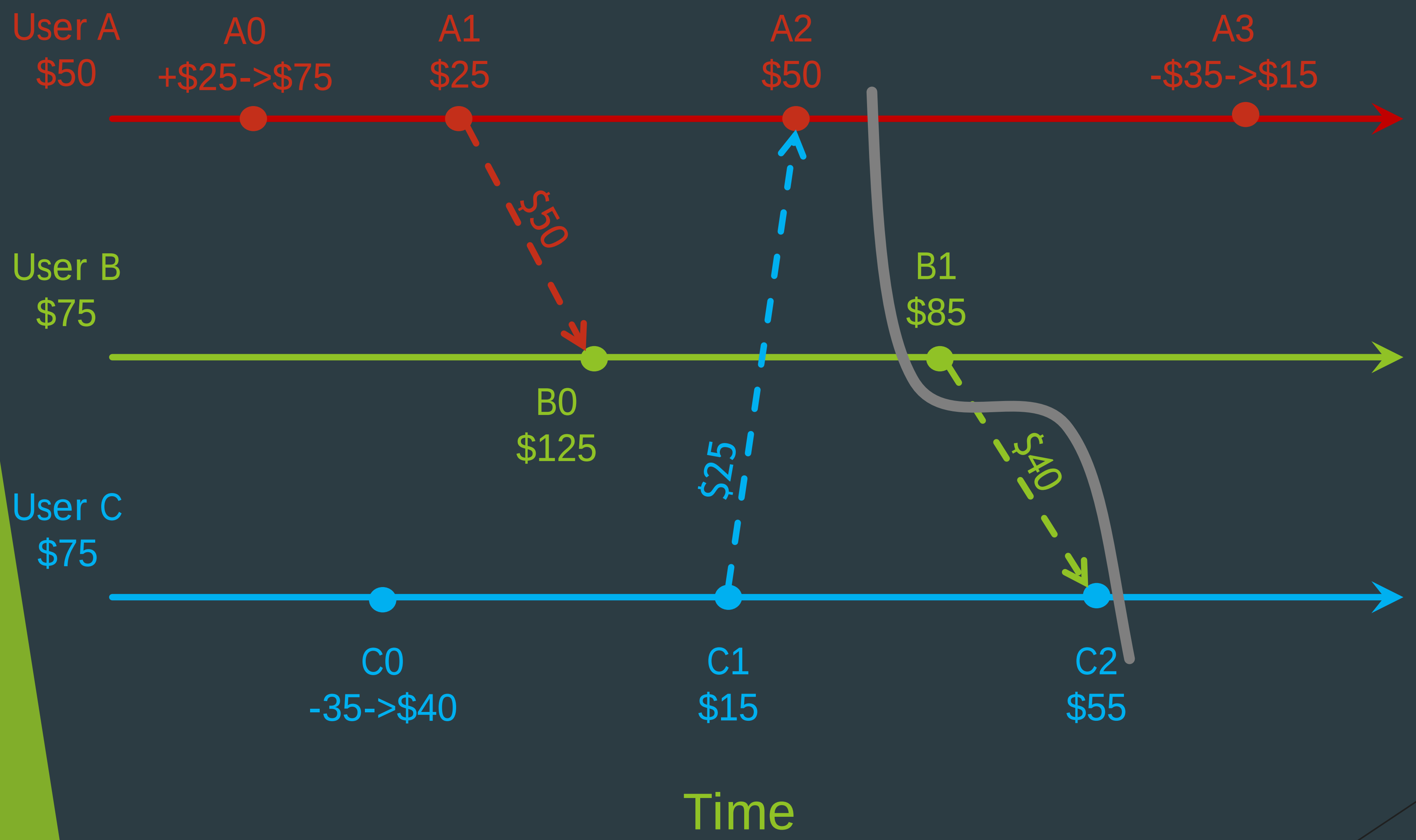
Strongly-Synchronized Clocks



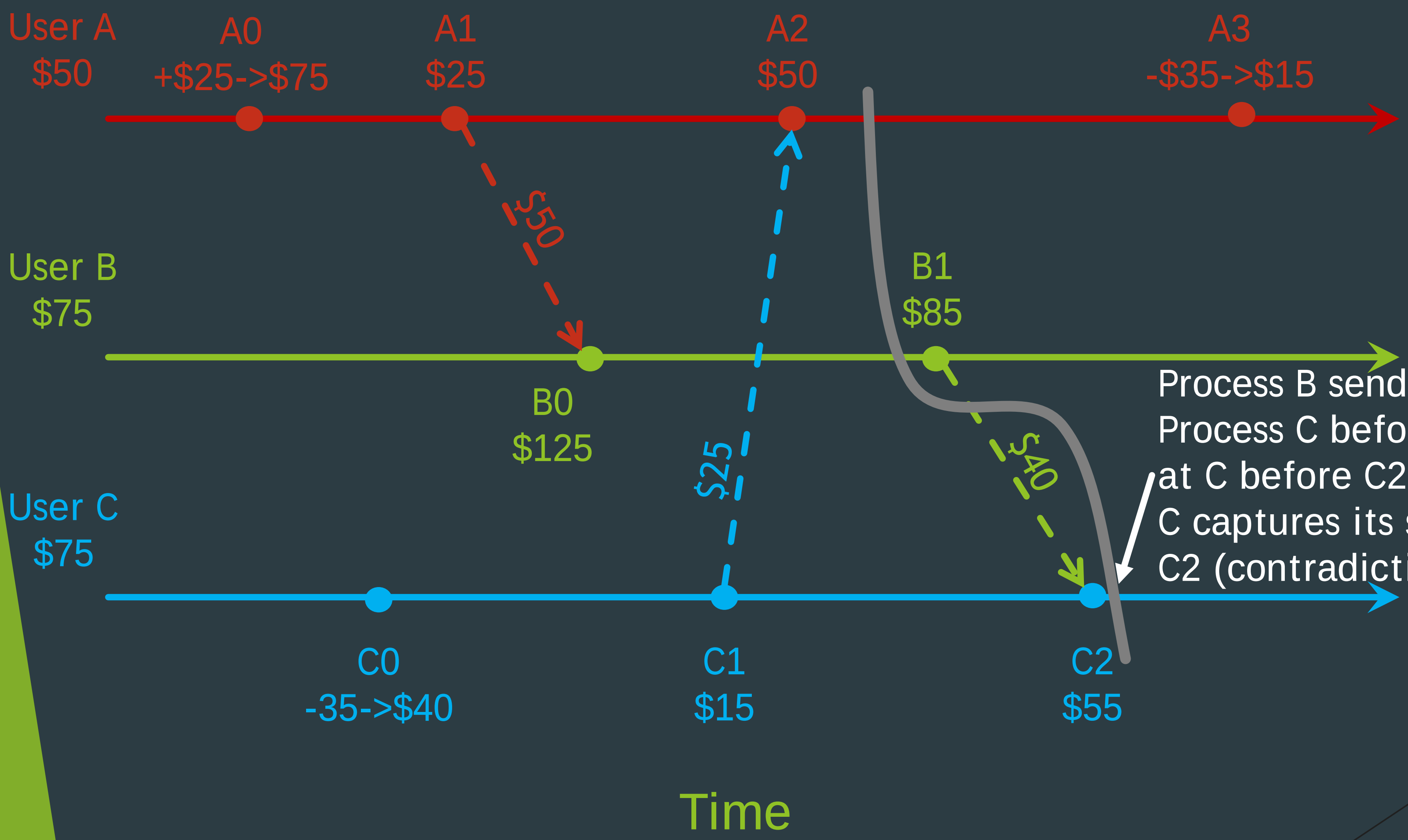
Consistent Cuts & Consistent Global States

- Process P starts taking a cut.
  - Take state on process P.
  - Send a token in each channel c adjacent to process P.
  - No message should be transmitted between taking the state and sending the tokens.
- On any process Q that receives token from channel c
  - If state has been captured, record channels c states as all the messages received from the point you have taken the state and received this token.
  - Else, record state and send token in each channel c adjacent to process Q.
  - No message should be transmitted between taking the state and sending the tokens.

Consistent Cuts – Is this possible?



Consistent Cuts – Is this possible?



Process B sends token to Process C before B1, arrives at C before C2 (FIFO), and C captures its state before C2 (contradiction).

Consistent Cut Issues

1. Cuts are not taken on demand. They should be taken pro-actively.
2. Might be slightly disruptive if the algorithm runs frequently.
3. What timestamp should be assigned to a cut?

Happened Before Relation

1. If an event e' happens after another event e in the same process P, then
$$e \rightarrow e'$$
2. If a process P sends a message m (event e) and another process Q receives message m (event e') then
$$e \rightarrow e'$$
3. Transitive Closure: If
$$e \rightarrow e' \text{ and } e' \rightarrow e''$$
then
$$e \rightarrow e''$$

Causal Consistency

A snapshot (or a cut) C is causally consistent iff

$$\forall e' \in \{e' | \exists e.\, e' \rightarrow e \land e \in C\}.\, e' \in \mathrm{C}$$

Causal Consistency – Universal Time

We know that:

$$e' \rightarrow e \Rightarrow UT(e') \leq UT(e)$$

Proof as exercise.

We take a snapshot C where we include every event e such that $UT(e) \leq t$. For all events $e' \rightarrow e$ such that $e \in C$, we have

$$UT(e') \leq UT(e) \Rightarrow$$
$$UT(e') \leq t \Rightarrow$$
$$e' \in C$$

Logical Clock

At process P:

1. On local event e:

$$LC(e) = LC^P + 1$$

2. When sending a message m to another process Q (e):

$$LC(e) = LC^P + 1$$

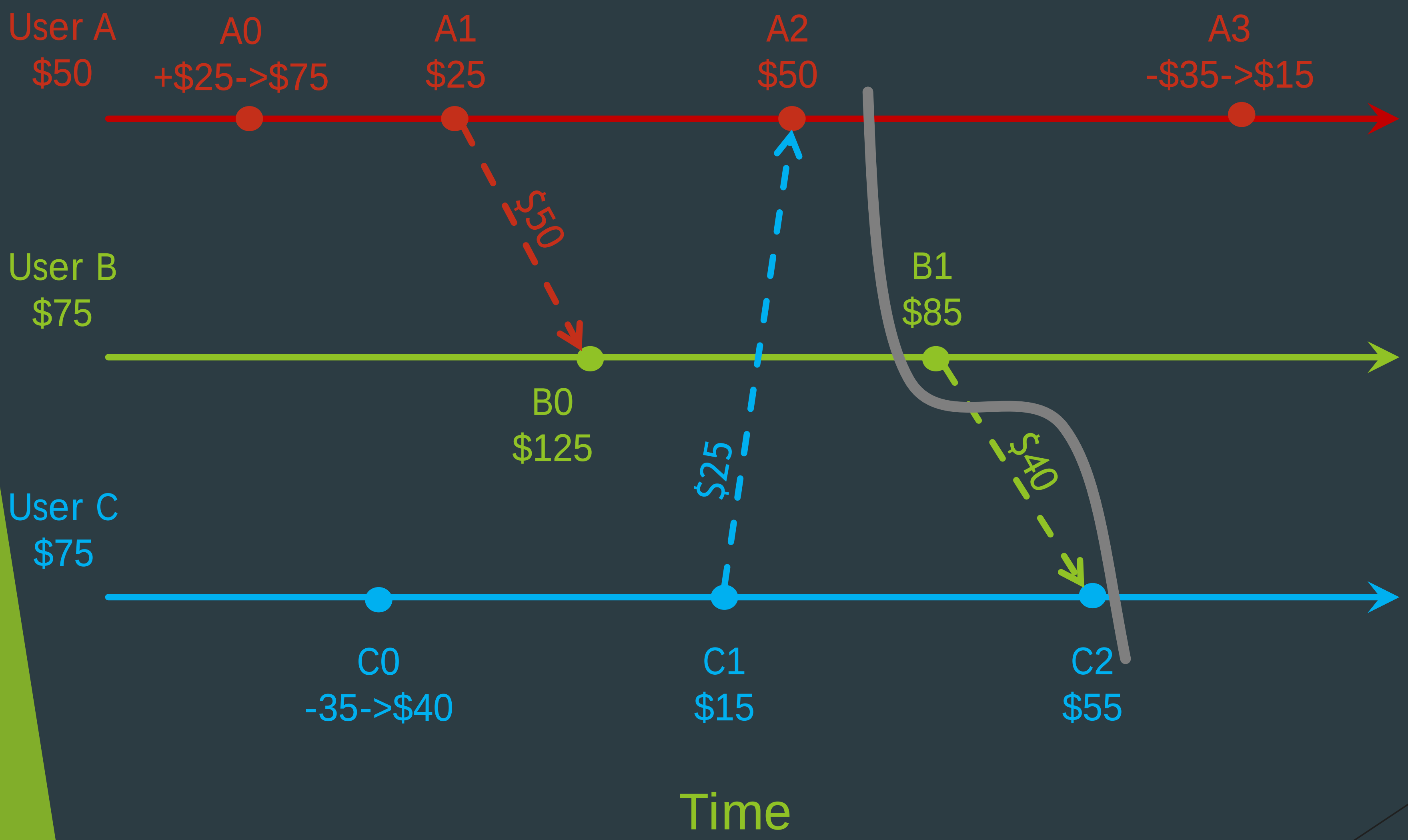
   Send message $m, LC(e)$ to process Q.

3. When receiving a message m, LC^m from process Q (e):

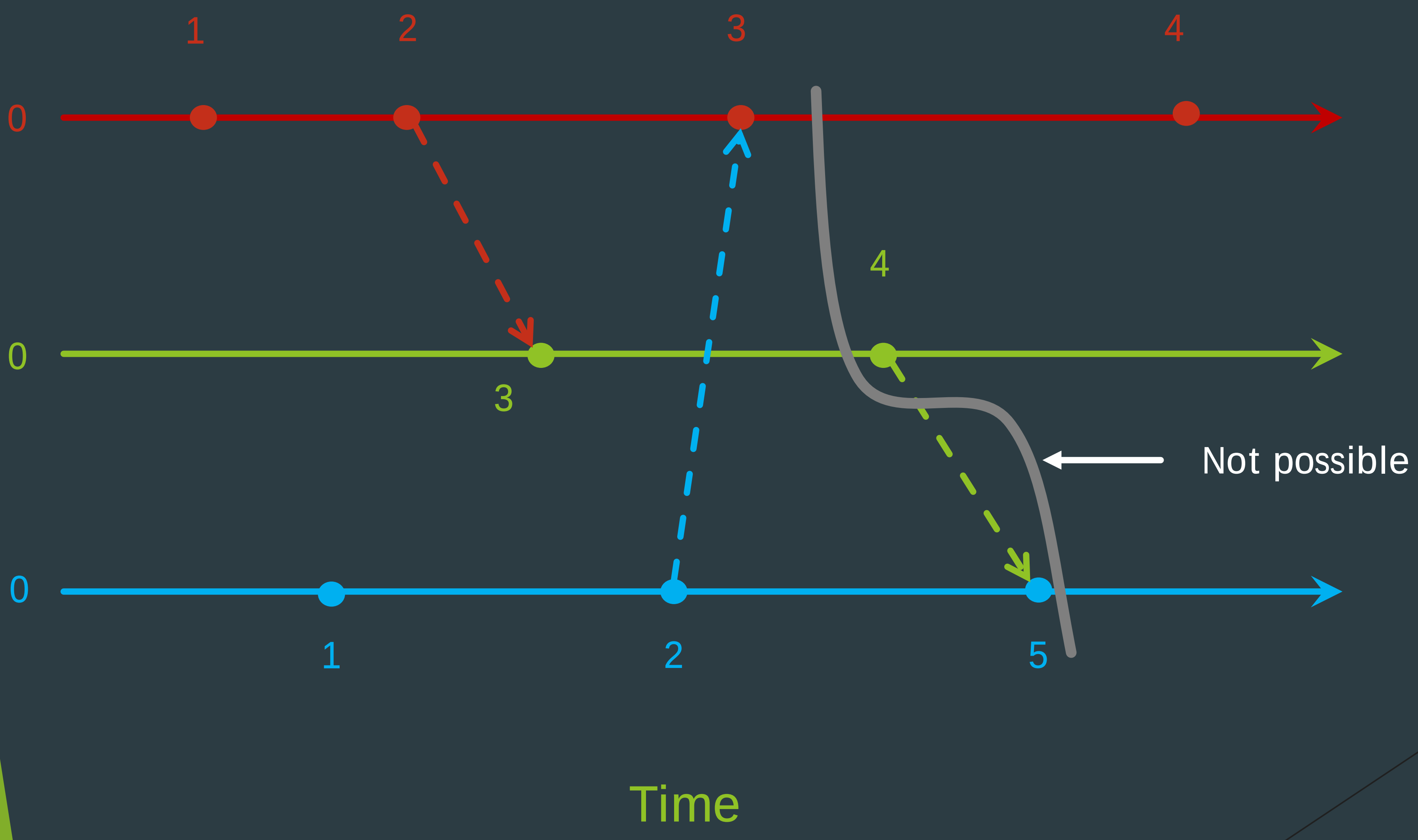
$$LC(e) = \max\{LC^P, LC^m\} + 1$$

4. We always set LC^P to $LC(e)$ after we finish executing the events.

Logical Clocks – Is this possible?



Logical Clocks – Is this possible?



Causal Consistency – Logical Clocks

We know that:

$$e' \rightarrow e \Rightarrow LC(e') \leq LC(e)$$

Proof as exercise.

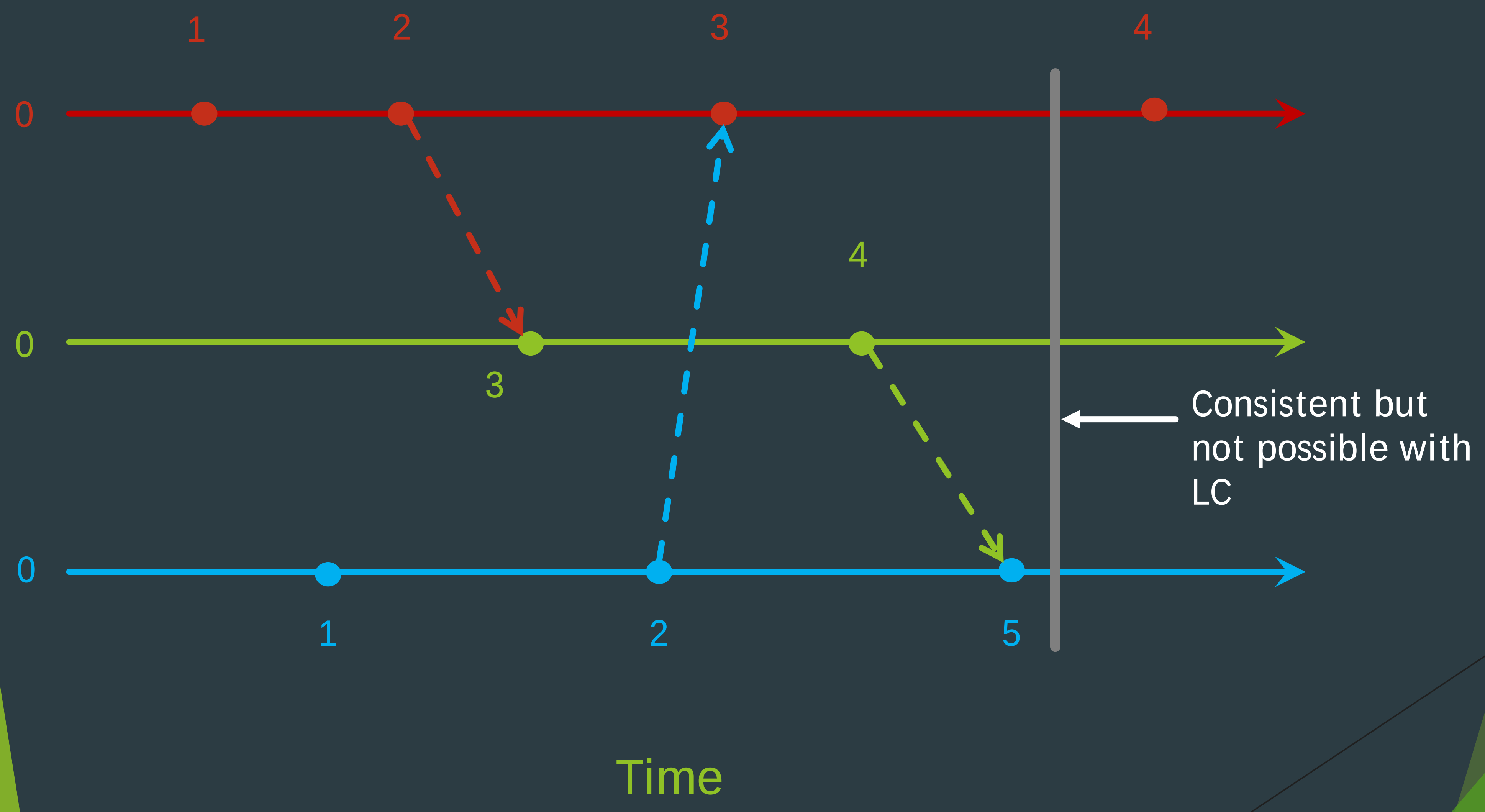
We take a snapshot C where we include every event e such that $LC(e) \leq t$. For all events $e' \rightarrow e$ such that $e \in C$, we have

$$LC(e') \leq LC(e) \Rightarrow$$
$$LC(e') \leq t \Rightarrow$$
$$e' \in C$$

Is the following true?

$$LC(e) < LC(e') \Rightarrow e \rightarrow e'$$

Logical Clocks – Is this possible?



Causal Consistency – Logical Clocks

We know that:

$$e' \rightarrow e \Rightarrow LC(e') \leq LC(e)$$

Proof as exercise.

We take a snapshot C where we include every event e such that $LC(e) \leq t$. For all events $e' \rightarrow e$ such that $e \in C$, we have

$$LC(e') \leq LC(e) \Rightarrow$$
$$LC(e') \leq t \Rightarrow$$
$$e' \in C$$

Is the following true?

$$LC(e) < LC(e') \Rightarrow e \rightarrow e'$$

NO!

Vector Clock

Assume we have n processes. Then VC is an n-tuple. We denote $VC^P[Q]$ as the VC value for process Q that is kept at process P. At process P:

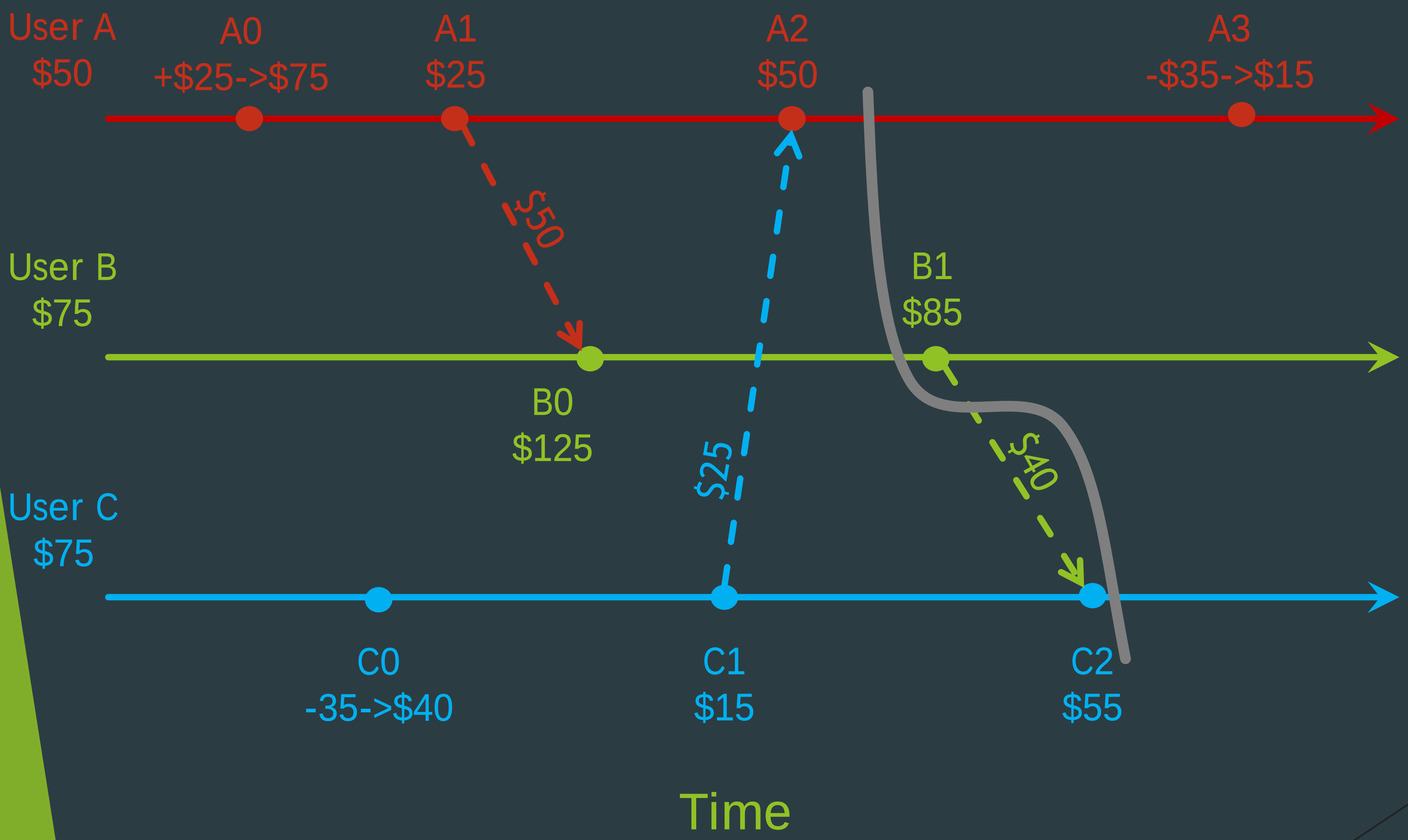
1. On local event e:
$$VC(e)[P] = VC^P[P] + 1$$
For all processes $Q \neq P$:
$$VC(e)[Q] = VC^P[Q]$$

2. When sending a message m to another process R (e):
$$VC(e)[P] = VC^P[P] + 1$$
For all processes $Q \neq P$:
$$VC(e)[Q] = VC^P[Q]$$
Send message $m, VC(e)$ to process Q.

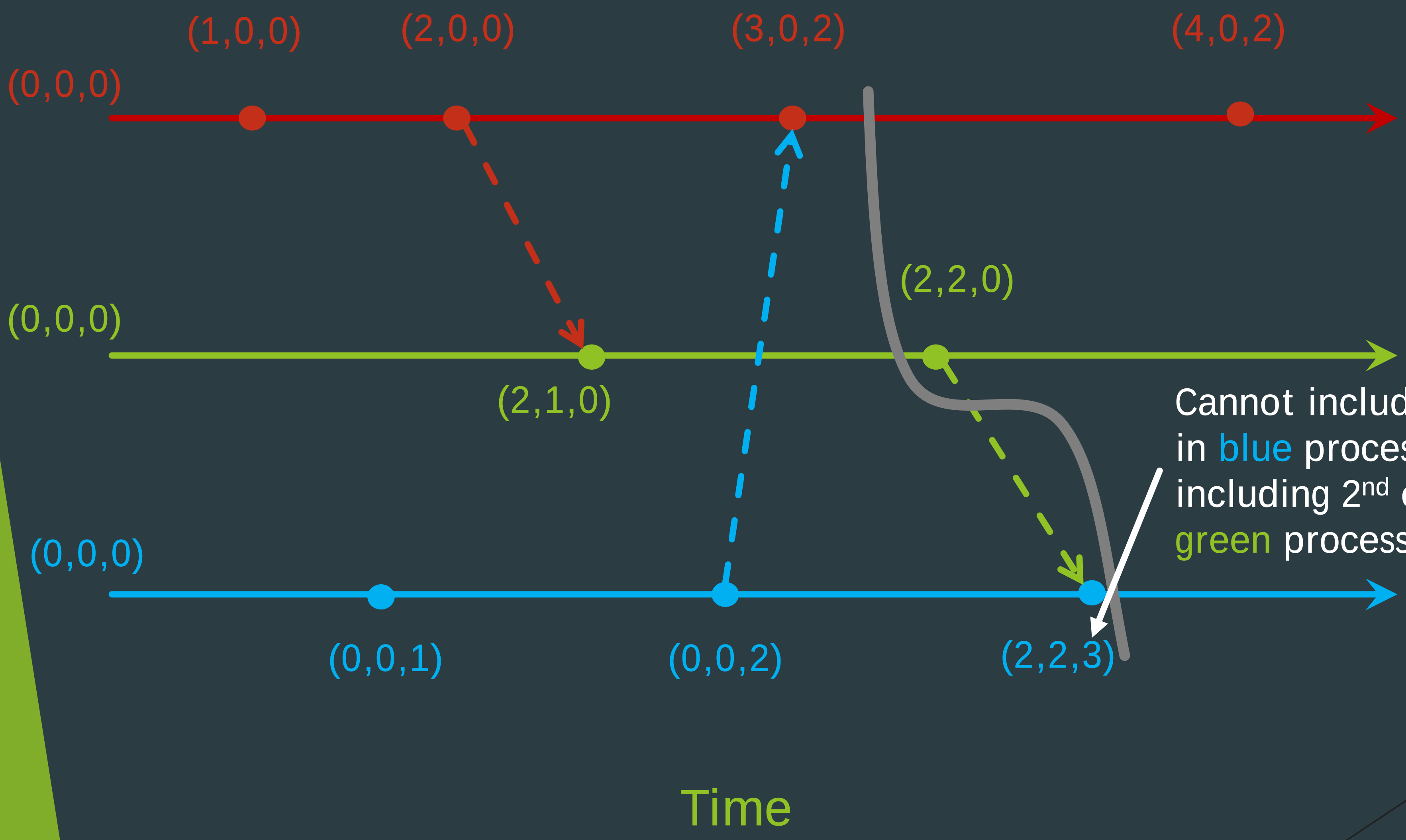
3. When receiving a message m, VC^m from process R (e):
$$VC(e)[P] = \max\{VC^P[P], VC^m[P]\} + 1$$
For all processes $Q \neq P$:
$$VC(e)[Q] = \max\{VC^P[Q], VC^m[Q]\}$$

4. We always set VC^P to $VC(e)$ after we finish executing the events.

Vector Clocks – Is this possible?

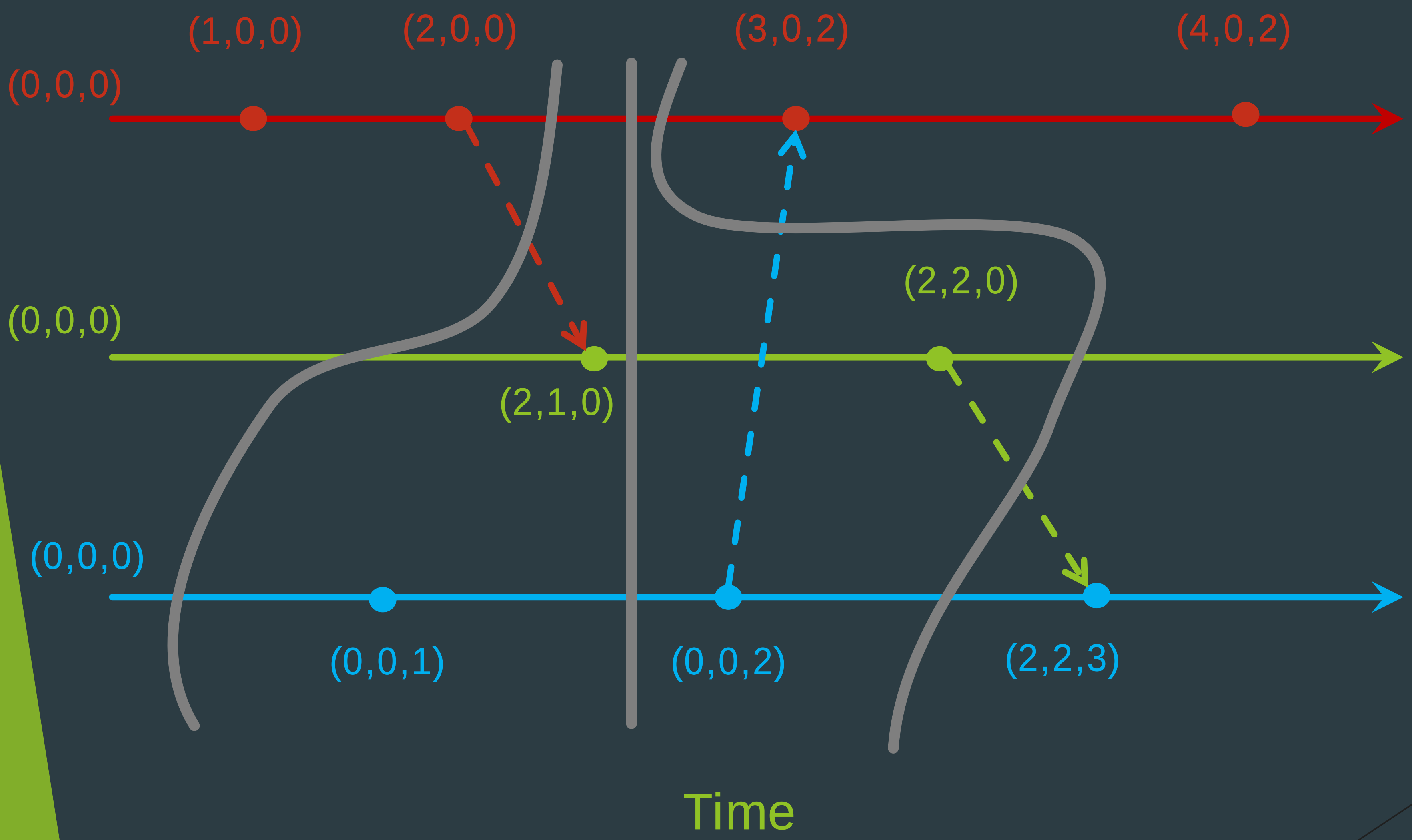


Logical Clocks – Is this possible?



Cannot include 3rd event in blue process without including 2nd event in green process.

Vector Clocks – Multiple Snapshots Possible



Causal Consistency – Vector Clocks

- We say that:

$$VC(e') < VC(e)$$

  iff for all processes P:
  $VC(e')[P] \leq VC(e)[P]$

$VC(e')[Q] < VC(e)[Q]$