UNIVERSITY OF TRENTO

Complexity, Cryptography, and Financial Technologies

Lecture 15 – FuturesMEX

Chan Nam Ngo

Futures market as illustrative of FinTech

- **A double auction market**
- **Bidders on both buy/sell side**
- **Futures contract**
  - standardized promise to buy/sell barrels of oil, bushels of corn, …
  - made today and to be fulfilled in a future date
  - with cash reserve to meet promises
- **Exchange platform for trading activities**
  - Chicago Mercantile Exchange → centralized

How futures trading works?

UNIVERSITY OF TRENTO

Trader	Promises	Cash
Alice	0	1200
Bob	0	1500

Market price = 10$

Alice sells 100 promises

Bob buys 80 promises

Trader	Promises	Cash at the exchange
Alice	Buy 100	2200=1200+**100***10
Bob	Sell 100	700=1500-**80***10

At end of (trading) da

Market price = 8$

Promises must be fulfilled at end of day price: Bob must sell and Alice must buy from the market

Trader	Promises	Cash at the exchange
Alice	~~Buy 100~~	1400=2200-**100***8
Bob	~~Sell 80~~	1360=700+**80***8

Alice made a profit of 200$, Bob lost.

Centralized futures trading (2)

Trader	Promises	Cash
Alice	0	1200
Bob	0	1500

Market price = 10$

Alice sells 100 promises

Bob buys 80 promises

Trader	Promises	Cash at the exchange
Alice	Buy 100	2200=1200+**100***10
Bob	Sell 80	700=1500-**80***10

At end of day Market price = 12$

Trader	Promises	Cash at the exchange
Alice	~~Buy 100~~	1000=2200-**100***12
Bob	~~Sell 80~~	1660=700+**80***12

Promises must be Fulfilled at current price

Bob made a profit but Alice lost 200$

Market price is volatile

Trader	Promises	Cash
Alice	Buy 100	2200
…	…	…

100 promises to buy when price was at 10$ looked a good idea but things change

Hi Alice, can you deposit more money?

Alice's cash reserve is now at 0$ → ***Exchange must do something***

No? → Alice is liquidated.

2200 > 12*100

2200 > 17*1000

2200 = 22*100!!!

12 17 22

If price rises further Alice's going broke

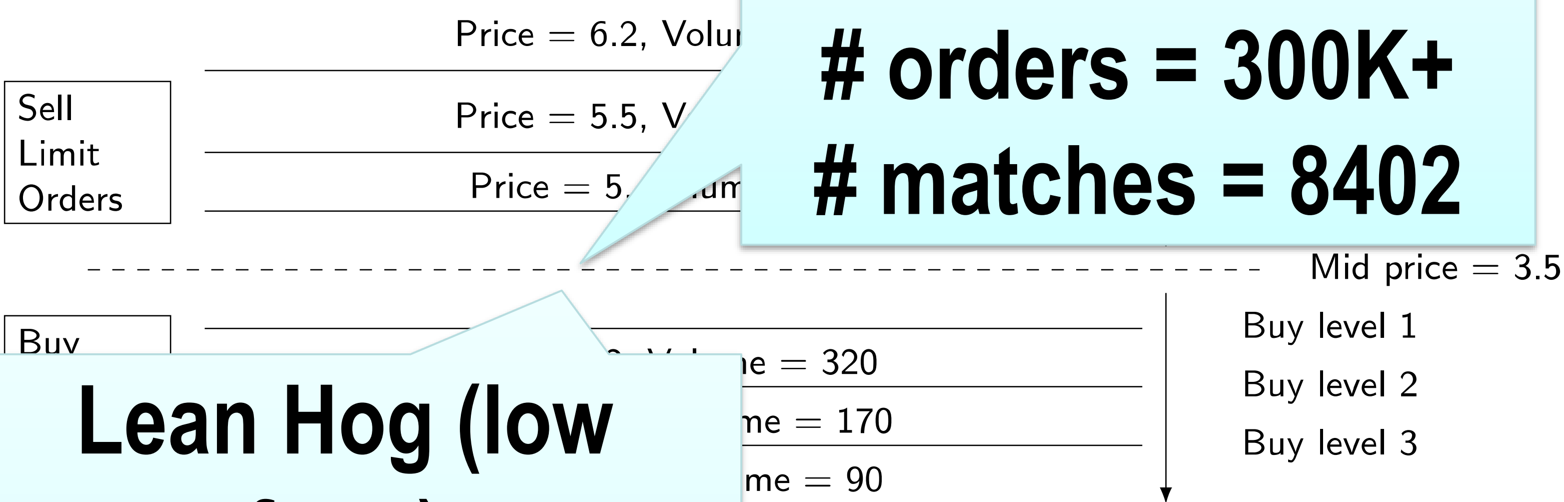
How does The Exchange Keep Track?

UNIVERSITY OF TRENTO

- **The Limit Order Book**
  - Auction mechanism that facilitates trade of assets and provides benchmark prices accessible to ALL members of the market.
  - Traders post buy and sell orders and the clearing of these orders forms part of the purpose of the limit order book.
- **Two types of orders:**
  - Limit orders (quotes), that specify a price and volume at which the trader is willing to buy or sell an asset.
  - Market orders, a request to buy or sell an asset
- **The limit order book displays current set of limit orders and records the execution of market orders as traded prices.**
- **A critical feature of the limit order book is that part of the order book is public (information visible to all traders) and part is private.**

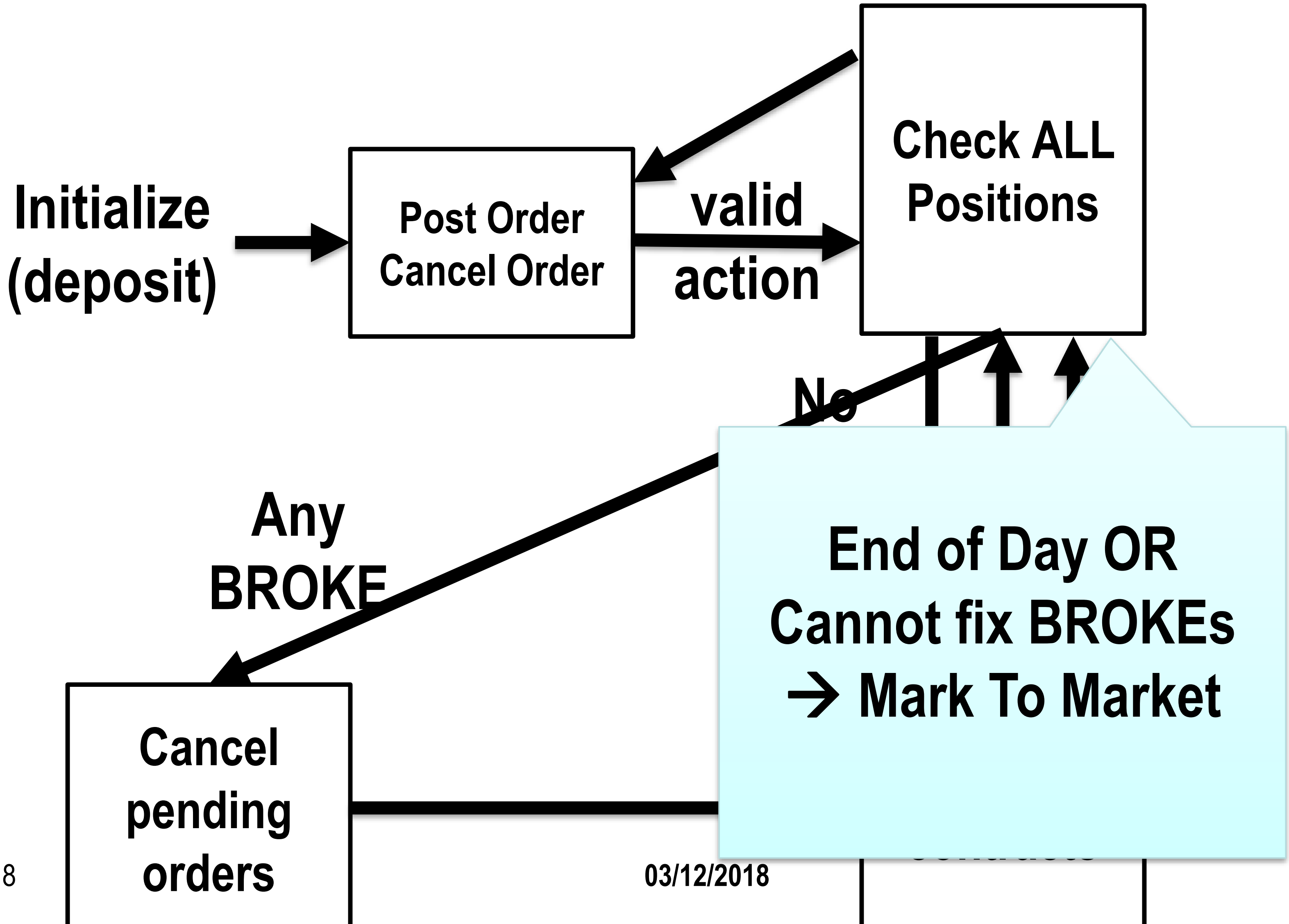
Example of the public part of the order book

Eurodollar (hi. freq.)
traders = 520
orders = 300K+
matches = 8402



Lean Hog (low freq.)
traders = 33
orders = 6709
matches = 536

Exchange as reactive functionality



Ideal Functionality - Storage

- **Trader Inventory P_i**
  - Cash available m_i
  - Volume holding v_i
  - Estimated cash em_i
  - Estimated volume ev_i
  - Counter c_i
  - Flag f_i
- **Order Book $O = \{o_1 \dots o_t\}$**
  - $O_j = (p, v, t, j)$ – P_j is the owner of this order o_j
  - Buy order $v > 0$
  - Sell order $v < 0$
- **Net Position**
  - $n_i = m_i + cash(v_i)$
  - $en_i = em_i + cash(v_i)$

Ideal Functionality - Initialize

- **Trader Inventory P_i**
  - m_i
  - $v_i = 0$
  - $em_i = m_i$
  - $ev_i = 0$
  - $c_i = 0$
  - $f_i = 0$
- **$O = \{\}$**

Ideal Functionality – Post Order

- **$O = \{\ldots o_t\}$**
  - $o_t = (p, v, t)$
- **Trader Inventory P_i**
  - m_i
  - v_i
  - $em_i = em_i - vp$
  - $ev_i = ev_i + v$
  - $c_i = c_i + 1$
  - f_i
- **Valid order?**
  - $en_i >= 0$

Ideal Functionality – (Estimated) Net Position

1. If $v_i > 0$, look at buy side, otherwise if $v_i < 0$ look at sell side
2. Take v_l from low to high on sell side and from high to low on buy side

   e.g. cash(500) = cash(320 + 170 + 10) = 2*320 + 1.5*170 + 0.5*10 = 900

Sell Limit Orders	Price = 6.2, Volume = 100	Sell level 3
	Price = 5.5, Volume = 120	Sell level 2
	Price = 5, Volume = 260	Sell level 1
		Mid price = 3.5
Buy Limit Orders	Price = 2, Volume = 320	Buy level 1
	Price = 1.5, Volume = 170	Buy level 2
	Price = 0.5, Volume = 90	Buy level 3

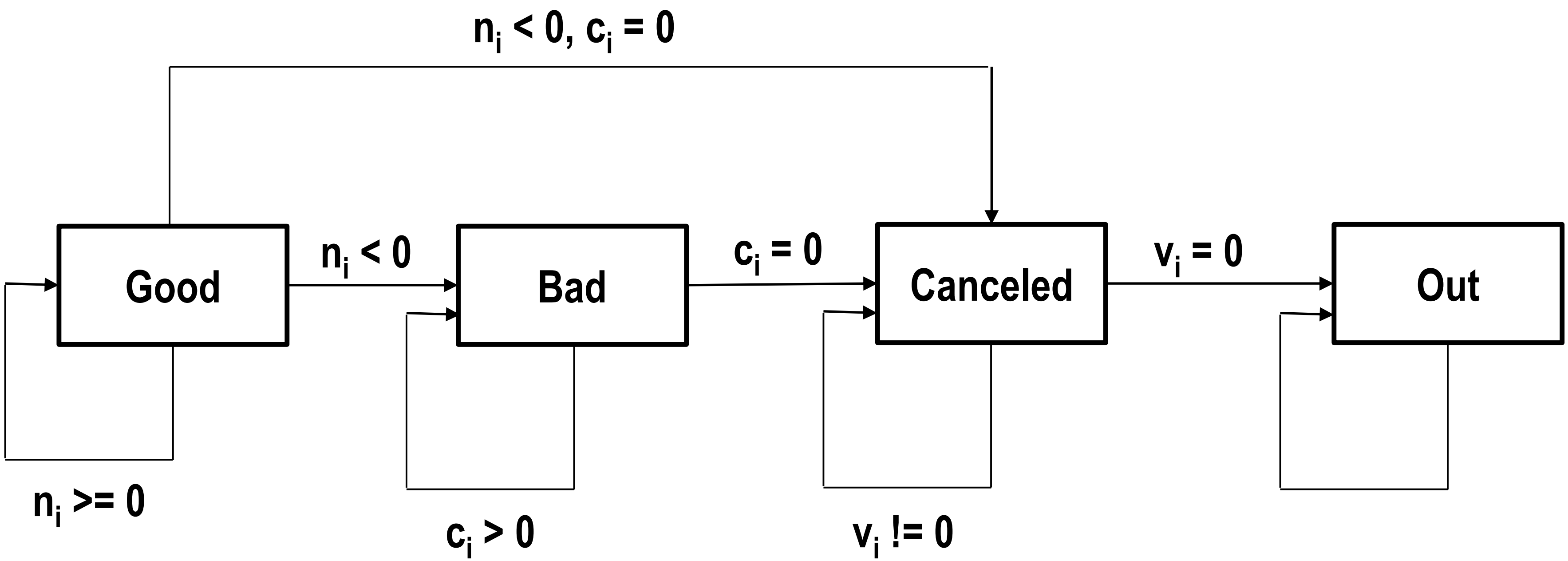
e.g. cash(-450) = cash(-260 - 120 - 70) = 5*260 + 5.5*120 + 6.2*70 = 2394

Ideal Functionality – Cancel Order

- $O = \{\ldots o_t\} - o_t$
  - $o_t = (p, v, t)$
- Trader Inventory P_i
  - m_i
  - v_i
  - $em_i = em_i + vp$
  - $ev_i = ev_i - v$
  - $c_i = c_i - 1$
  - f_i
- Valid order?
  - $en_i >= 0$

Ideal Functionality – Update Status

- **Compute n_i for all P_i, update f_i**



Ideal Functionality – Match Order

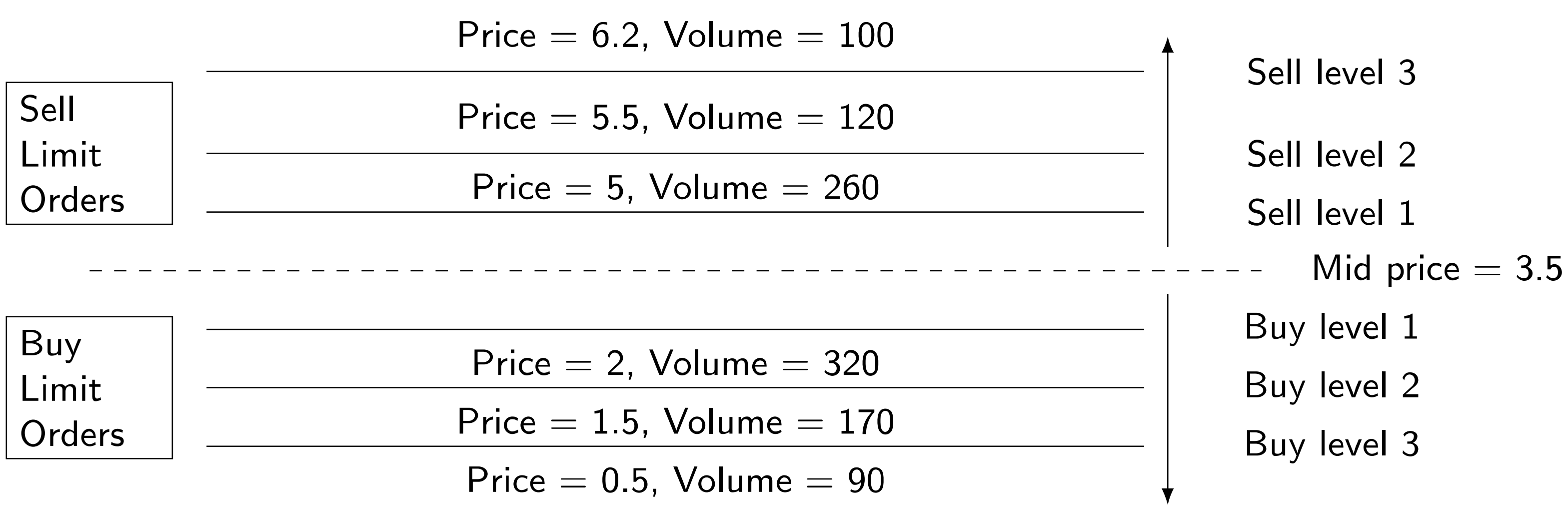
- $O = \{\ldots o_t\} - o_t$
  - $o_i = (p, v', t')$
  - $o_j = (p, v, t)$
  - Simple case: $-v' = v$
  - A bit more complicated, a single order will be matched with multiple orders
- Trader Inventory P_i
  - $m_i = m_i - vp$
  - $v_i = v_i + v$
  - em_i
  - ev_i
  - $c_i = c_i - 1$
  - f_i
- Do the reverse for P_j
  - $m_j = m_j + vp$
  - $v_j = v_j - v$
  - $c_j = c_j - 1$

Ideal Funcitonality - Examples

- **Alice inits (10000, 0)**
  - m_A = 10000, v_A = 0
- **Bob inits (9000, 0)**
  - m_B = 9000, v_B = 0
- **Alice posts (10, 100) – buy 100@10**
  - em_A = 10000 – 100*10 = 9000
  - ev_A = 100
- **Alice posts (11, 50) – buy 50@11**
  - em_A = 9000 – 50*11 = 8450
  - ev_A = 150
- **Bob posts (13, -90) – sell 90@13**
  - em_B = 9000 – (-90)*13 = 10170
  - ev_B = -90
- **Bob posts (11, -50) – sell 50@11**
  - em_B = 10170 – (-50)*11 = 10720
  - ev_B = -140
- **A match found**
  - Alice can now buy 50 and Bob can sell 50 at 11
  - m_A = 10000 – 50*11 = 9450
  - v_A = 50
  - m_B = 9000 + 50*11 = 9550
  - v_B = -50

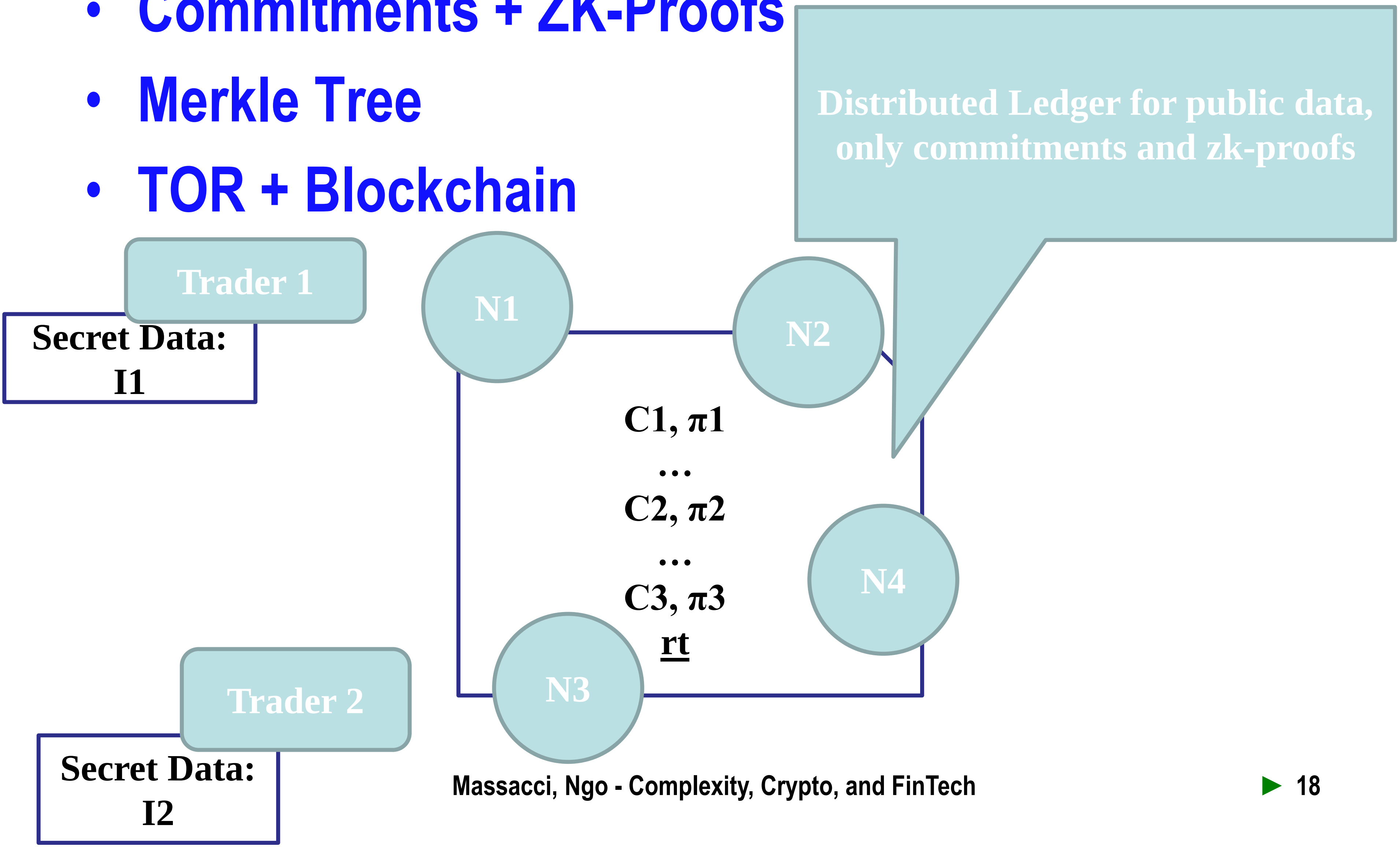
- **Alice inits (10000, 0)**
  - n_A = 10000 + cash(0) = 10000
- **Bob inits (9000, 0)**
  - n_B = 9000 + cash(0) = 9000
- **Alice posts (10, 100) – buy 100@10**
  - Assume max price is 20
  - en_A = 9000 + 100*20 = 11000
- **Alice posts (11, 50) – buy 50@11**
  - Assume max price is 20
  - en_A = 8450 + 150*20 = 11450
- **Bob posts (13, -90) – sell 90@13**
  - en_B = 10170 – 50*11 – 40*10 = 9220
  - n_A = 10000 + cash(0) = 10000
- **Bob posts (11, -50) – sell 50@11**
  - en_B = **10720** – 50*11 – 90*10 = 9270
  - n_A = 10000 + cash(0) = 10000
- **A match found**
  - Sell side: (13, -90)
  - Buy side: (10, 100)
  - n_A = 9450 + cash(50) = 9450 + 50*10 = 9950
  - n_B = 9550 + cash(-50) = 9550 – 50*13 = 8900

- **Trader Inventory P_i**
  - $m_i = m_i + v_i$*mid_price
  - $v_i = 0$



Distributed Protocol

- Commitments + ZK-Proofs
- Merkle Tree
- TOR + Blockchain



Distributed Protocol - Storage

- **Trader Inventory P_i**
  - Cash available $[m_i]$
  - Volume holding $[v_i]$
  - Estimated cash $[em_i]$
  - Estimated volume $[ev_i]$
  - Counter $[c_i]$
  - Flag $[f_i]$
  - Token $t_i = [m_i|v_i|em_i|ev_i|c_i|f_i]$
- **Order Book $O = \{o_1 \ldots o_t\}$**
  - $O_j = (p, v, t, [j])$
  - Prove that you know the randomness of [j] to claim that is your order
- **Net Position**
  - $[n_i] = [m_i + cash(v_i)]$
  - $[en_i] = [em_i + cash(v_i)]$

Distributed Protocol - Initialize

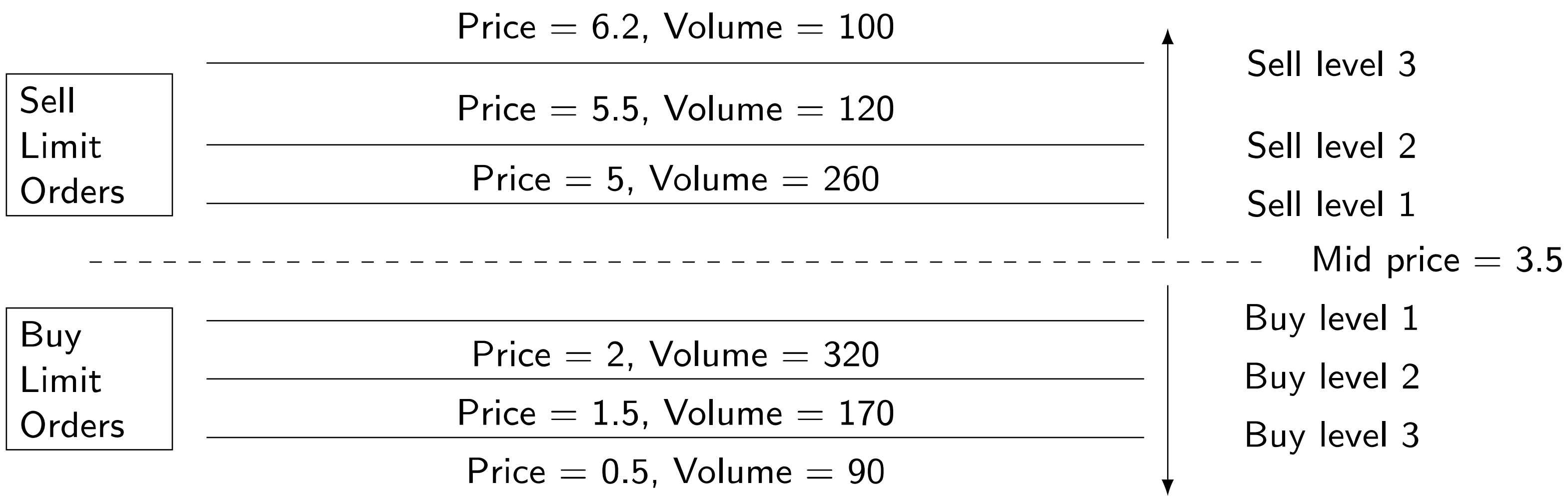
- **Trader Inventory P_i**
  - $[m_i]$, prove that $m_i > 0$
  - $[v_i] = [0]$, show randomness to prove $v_i = 0$
  - $[em_i] = [m_i]$, prove that $em_i = m_i$
  - $[ev_i] = [0]$, show randomness
  - $[c_i] = [0]$, show randomness
  - $[f_i] = 0$, show randomness
  - Token $t_i = [m_i|v_i|em_i|ev_i|c_i|f_i]$, prove with zk
  - $[t_i]$, to be added into Merkle Tree
- **$O = \{\}$**

- **$O = \{\ldots o_t\}$**
  - $o_t = (p, v, t, [i])$
- **Trader Inventory P_i**
  - Get $[m_i]$, $[v_i]$, $[em_i]$, $[ev_i]$, $[c_i]$, $[f_i]$ from MT using an unspent t_i
  - Update em_i ev_i and c_i
    - $em_i = em_i - vp$
    - $ev_i = ev_i + v$
    - $c_i = c_i + 1$
  - Commit to the new values and prove in zk
- **Valid order?**
  - $en_i >= 0$, compute the value, commit and prove in zk

Ideal Functionality – (Estimated) Net Position

1. If $v_i > 0$, look at buy side, otherwise if $v_i < 0$ look at sell side
2. Take v_l from low to high on sell side and from high to low on buy side

   e.g. cash(500) = cash(320 + 170 + 10) = 2*320 + 1.5*170 + 0.5*10 = 900



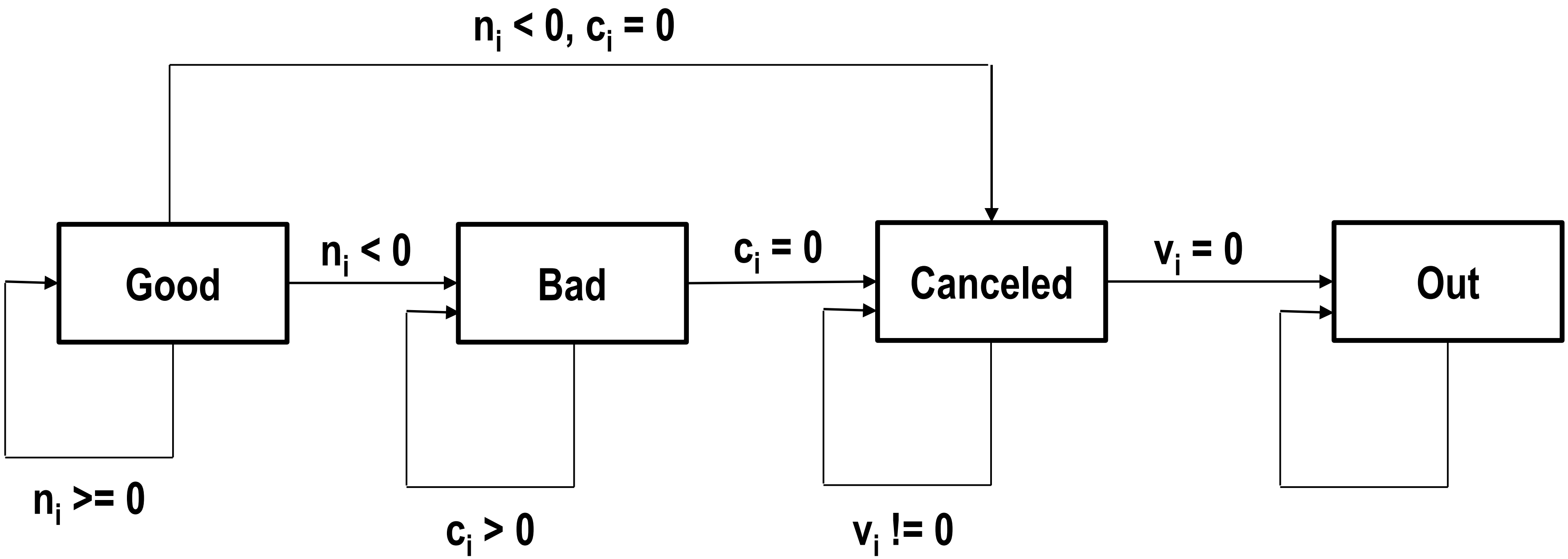
e.g. cash(-450) = cash(-260 - 120 - 70) = 5*260 + 5.5*120 + 6.2*70 = 2394

Ideal Functionality – Cancel Order

- **$O = \{\ldots o_t\} - o_t$**
  - $o_t = (p, v, t, [i])$
- **Trader Inventory P_i**
  - Get $[m_i]$, $[v_i]$, $[em_i]$, $[ev_i]$, $[c_i]$, $[f_i]$ from MT using an unspent t_i
  - Update em_i ev_i and c_i
    - $em_i = em_i + vp$
    - $ev_i = ev_i - v$
    - $c_i = c_i - 1$
  - Commit to the new values and prove in zk
- **Valid order?**
  - $en_i >= 0$
  - Knows the randomness of [i] to prove this is your order

Distributed Protocol – Update Status

- Get $[m_i]$, $[v_i]$, $[em_i]$, $[ev_i]$, $[c_i]$, $[f_i]$ from MT using an unspent t_i
- Compute $[n_i]$ for all P_i, update $[f_i]$



Distributed Protocol – Match Order & Mark To Market

UNIVERSITY OF TRENTO

- **Similar to the previous sub protocols**
  - Take inventory from MT
  - Update values
  - Commit, prove in zk, and put back into MT