

Prime Trader

and other lisp applications at Netfonds

Espen Vestre
Netfonds ASA
Oslo, Norway

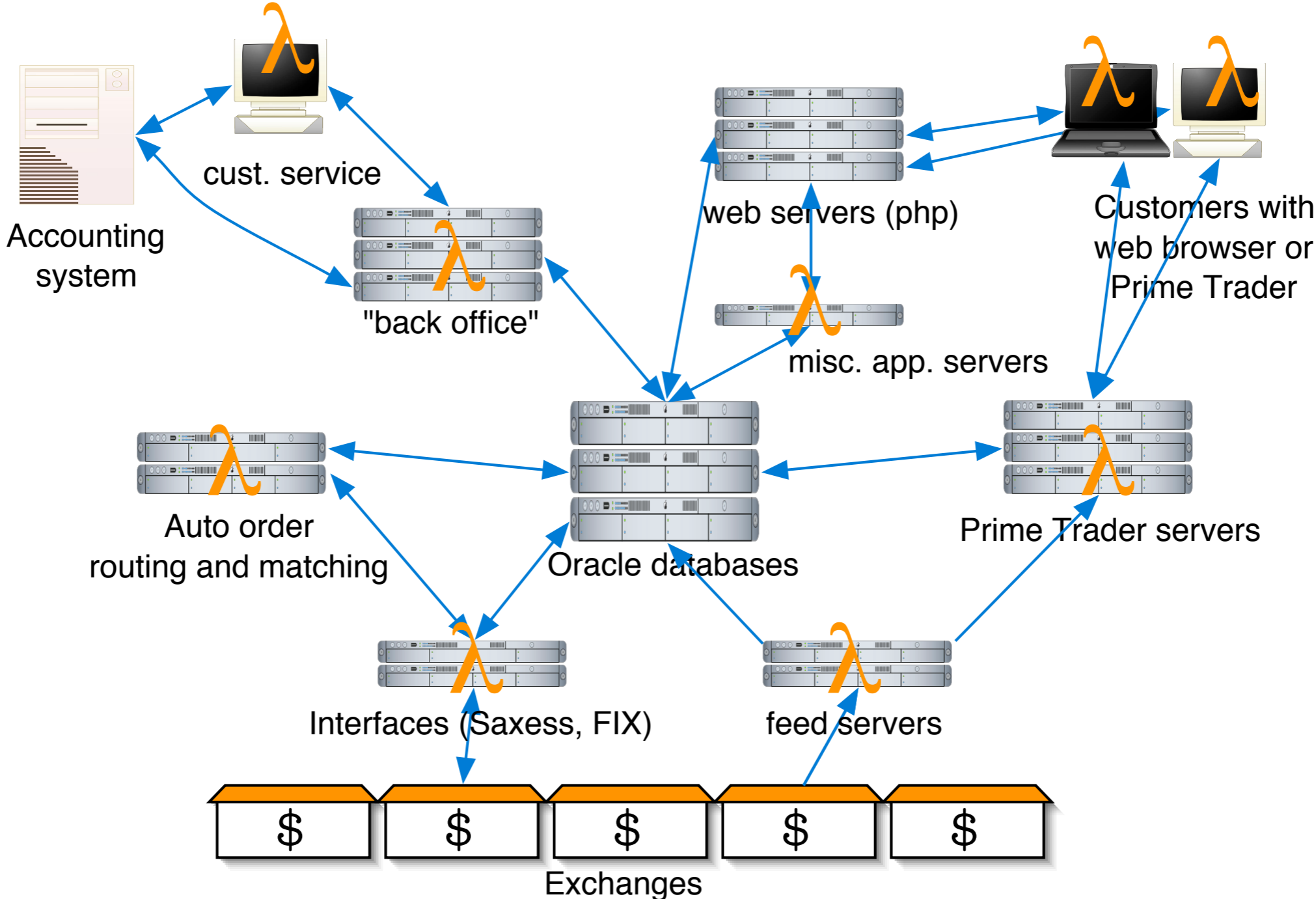
About Netfonds

- Established 1997
- Pioneered Internet-based (“self serve”) stock trading in Norway
- Offers trading and real time stock quotes on the stock exchanges of Oslo, Stockholm and New York
- 11 employees – 7000 active customers (1000 use Prime Trader)
- 3 (4) Lisp Developers / linux sysadmins

Talk Overview

- Lisp at Netfonds
- PrimeTrader
- A few problems and how we solved them
- System Demo

Simplified System Overview



and more...

- Interface to VPS (Norwegian central stock custody)
- Interface to banks (transfers to/from trading accounts)
- Auto-generated tax forms
- loans (of shares or money)
- risk management (portfolio VaR calculations)
- newsfeeds, real time exchange rates, etc.

PrimeTrader

- Specialized stock trading application
- Faster and more interactive than webapps
- Real time stock quotes
- Fast order entry and status
- Written in LispWorks, runs on Windows, OS X, linux
- Completely self-contained (even crypto)
- Lisp on the server side, too

PrimeTrader

- RSA-encryption for handshake
- All personal data blowfish-encrypted
- Automatic patch downloads
- Automatic patch generation from CVS
- HTTP-based protocol
- Subscription model ensures low bandwidth
- Automatic reconnect

CAPi experiences

- Extremely portable (and our competitors have Windows-only offerings!)
- Fairly easy in use
- More native look&feel than many other cross-platform tools
- ...but not 100%
- wish-list: E.g. html rendering panes, drag-and-drop

3x PrimeTrader

The screenshot shows a market data table with columns for ticker, time, price, and volume. An 'ORDER FORM (BUY NHY)' dialog box is open, displaying the following details:

- Settings: BUY NHY (Norsk Hydro)
- Ticker/Exchange: NHY / OSE
- Quantity (lot size: 20): 20
- Limit (NOK): 514
- Order type: Limit
- Buttons: Execute Order, Cancel
- Current Order Summary:
 - In account: 4 000
 - BUY NHY: 20@514
 - ON BEHALF OF: Espen Vestre (NFev)
 - Active buys: 0
 - Net Available: 20 000
 - Max shares: 20

The screenshot shows a market data table with columns for ticker, time, price, and volume. An 'ORDER FORM (BUY NHY)' dialog box is open, displaying the following details:

- Settings: BUY NHY (Norsk Hydro)
- Ticker/Exchange: NHY / OSE
- Quantity (lot size: 20): 20
- Limit (NOK): 514.00
- Order type: Limit
- Buttons: Execute Order, Cancel
- Current Order Summary:
 - In account: 4 000
 - BUY NHY: 20@514.00
 - Sales done: 0
 - Buyes done: 0
 - Active buys: 0
 - Net Available: 20 000
 - Max shares: 20

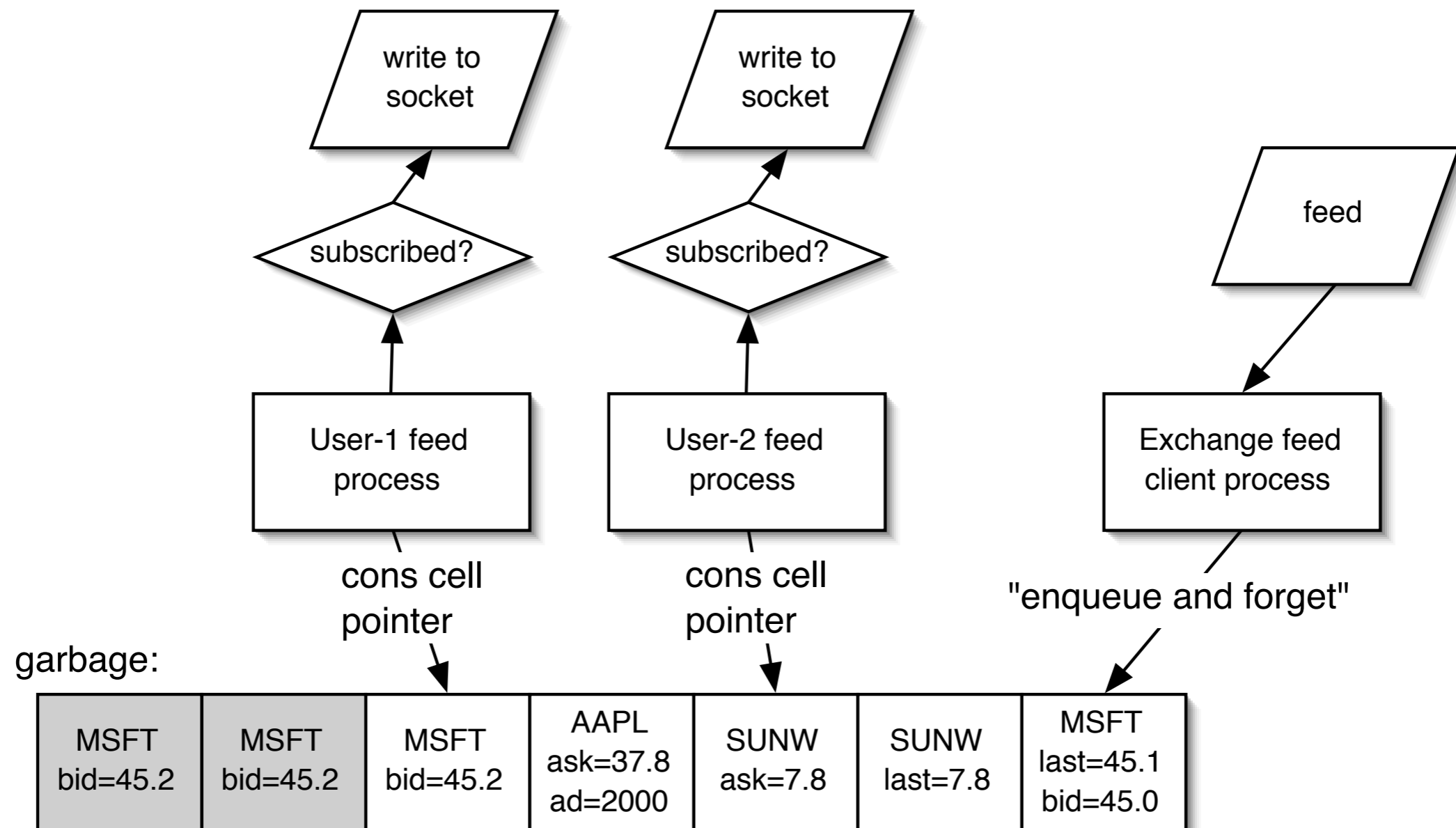
The screenshot shows a market data table with columns for ticker, time, price, and volume. An 'ORDER FORM (BUY NHY)' dialog box is open, displaying the following details:

- Settings: BUY NHY (Norsk Hydro)
- Ticker/Exchange: NHY / OSE
- Quantity (lot size: 20): 20
- Limit (NOK): 514
- Order type: Limit
- Buttons: Execute Order, Cancel
- Current Order Summary:
 - In account: 4 000
 - BUY NHY: 20@514
 - Sales done: 0
 - Buyes done: 0
 - Active buys: 0
 - Net Available: 20 000
 - Max shares: 20

Some GUI comments

- Table layout is done with pinboard layouts
- Table class has been successfully reused in a wide range of internal customer service and back office applications.
- Very object-oriented: Each table cell class object, knows how to draw itself, subscribes to data, gets redraw messages when data changes

Server feed model



Problems

- Not kind to GC
- full GC (mark-and-sweep 2) every hour (20 minutes for US servers) cleans up
- BUT NASDAQ feed speed often exceeds 2000 trans/second!
 - slow clients can quickly lag 1 million transactions behind!
- List of 1 million CLOS objects + 50 threads + dozens of pointers to list = GC hell

Session model

- Streaming is done by “slow http calls”
- Server session \neq Tcp session
- Session objects across tcp connections
- Need to kill idle sessions

IJK

- The Idle Job Killer looks for inactive sessions and removes them:

```
(defun kill-if-idle (cookie session)
  (with-slots (main-userid queue) session
    (when (idle-p session)
      (remhash cookie *sessions*)
      (logger:log-message
        (list
          :ijk
          (format nil "killed session ~a (userid ~a)"
                    cookie main-userid))))
      (when queue ;; then really logged in!
        (setf queue nil)
        (db-log-logout session))))))
```

IJK cont.

- First version of IJK (i.e. idle-p) solved garbage problem for servers for low-volume exchanges (Oslo)
- But *not* if clients were alive, but too slow (some clients of US servers)

```

(defparameter *max-queue-length* 100000)

(defmethod flush-queue ((session nftp-session))
  (with-slots (queue queue-emptied main-userid cookie init-ip-addr) session
    (with-slots (lq:lock) queue
      (pmp:with-lock (lq:lock)
        (let ((to-delete (- (lq:queue-length queue)
                            (/ *max-queue-length* 10))))
          (loop for i below to-delete
                do (lq:pop-queue queue))
          (logger:log-message
            (list :warning :flush to-delete main-userid cookie init-ip-addr)))
          (setf queue-emptied t))))))

(defun idle-p (session)
  (with-slots (queue last-event-time streamer-process) session
    ;; sessions without queue are not yet logged in and should
    ;; be timed out after a while:
    (let ((length (if queue (lq:queue-length queue) 0)))
      (when (> length *max-queue-length*)
        (flush-queue session))
      (and (or (not queue)
               (> length 500)
               (not streamer-process))
           (> (- (get-universal-time)
                  last-event-time)
                *idle-timeout*))))))

```

Result

- NASDAQ and NYSE servers now extremely stable (about 600MB each, runs 24x7 uninterrupted since january/february)
- mark-and-sweep every 15 min. Typical timings:

```
("2005-04-21 15:40:19" "at-in-lisp" :D0-GC :STATS "freed 258109K in 766 ms cpu time")  
("2005-04-21 15:55:19" "at-in-lisp" :D0-GC :STATS "freed 250749K in 778 ms cpu time")  
("2005-04-21 21:40:31" "at-in-lisp" :D0-GC :STATS "freed 187625K in 1872 ms cpu time")  
("2005-04-21 21:55:31" "at-in-lisp" :D0-GC :STATS "freed 256206K in 1947 ms cpu time")
```

- Happy sysadmins
- Happy customers, too: Slow clients now get near-real-time data. Only problem: Missing details in intraday graphs

Saving bandwidth

- Unique (we think) “data subscription” system:
- Client only requests as much data as it really needs
- Streaming data for a limited number of stocks possible via GPRS (9600bps)
- Full US feed may require > 10Mbps

Data subscriptions

- Client tells server what data it wants
- Server checks subscriptions before writing object to socket

```
(defmethod subscribe ((obj nftpc))
  (with-streamer-server (obj)
    (let ((args (make-key-arglist obj)))
      (unless (already-subscribed args)
        (do-bfec-command `(subscribe ,@args)))
      (incf (gethash args *subscriptions* 0)))))
```

;; subscribe to specific quote:

```
(subscribe
  (make-instance 'quotes
                 :paper "NHY" :exchange "OSE"))
```

;; subscribe to all trades on OSE

```
(subscribe
  (make-instance 'trade :exchange "OSE"))
```

Subscribe, cont.

- mapping facilities for subscribing to lists of objects
- one call to server subscribes list of objects
- distributed calls to several servers (when data with mixed sources)

```
(defun sync-all-subscriptions-with-server (sublist)
  (multiple-value-bind (subscribes unsubscribes)
    (analyze-subscriptions sublist)
    (when subscribes
      (nftp:map-bfec-command 'nftp:subscribe subscribes))
    (when unsubscribes
      (nftp:map-bfec-command 'nftp:unsubscribe unsubscribes))))
```

Versions and patches

- Automatic creation of sexp-level diff of two versions (including some cheap tricks for deleted methods and changed defvars).
- compile to patch (one for each platform)
- Customers get notifications
- SHA1 fingerprint for download integrity
- fast file loaded into running program (at startup - dynamic change of capi objects not always possible)

Server patches

- Same diff method
- Loaded into running servers through REPL server
- some scripts for mass loading (currently there are 19 PrimeTrader servers)

Bug Reporting

- catches all errors
- asks user for permission to submit bug report
- posts backtrace + LW bug form + other data to server, server sends mail to sysadmins
- gives the user a choice: continue or quit

Future Plans/wishes

- e.g.:
- better newsfeed
- more graph tools, technical analysis
- portfolio tools
- better mobile version (currently done in java, wish for common lisp :-))
- whatever the customer wants

System Demo

One moment please...

The screenshot shows a financial application window titled "PRICE Info (OSE OB)". The window contains a table of stock data with columns: Ticker, Last Update, Last, Change Today, Chg. % Today, Bld Depth, Bld, Ask, and t. A context menu is open over the table, listing options: Remove ticker, Market by price, Intraday chart, Last n trades, and Add alarm. Below the table, there is a section titled "ORDER" with a table of order details.

Ticker	Last Update	Last	Change Today	Chg. % Today	Bld Depth	Bld	Ask	t
DNBNOR	16:39:59	54.00	-0.25	-0.46	0	0	0	
EDBASA	16:40:08	42.60	-0.40	-0.93	0	0	0	
EKO	16:40:09	145.25	0.25	0.17	0	0	0	
ELT	16:40:08	50.00	0.00	0.00	0	0	0	
EME	16:40:09					0	0	
FAST	16:39:59					0	0	
FRO	16:39:59	3				0	0	
GOL	16:39:59	11				0	0	
KVI	16:39:59					0	0	
NER	16:39:59					0	0	
NHY	16:39:59	469.00	6.00	1.30	0	0	0	
NSG	16:39:59	116.00	-0.25	-0.22	0	0	0	

ORDER

Order ID	Paper	Order Type	Amount	Order Limit
----------	-------	------------	--------	-------------

Documents
Movies
Music

Netfonds