501 lines (410 sloc) | 17 KB ...

```go
//go:build !race
// +build !race

// TODO remove build condition above to allow -race -short, after Arwen fix

package txsFee

import (
	"encoding/hex"
	"math/big"
	"testing"

	"github.com/ElrondNetwork/elrond-go-core/core"
	"github.com/ElrondNetwork/elrond-go-core/data/block"
	"github.com/ElrondNetwork/elrond-go/config"
	"github.com/ElrondNetwork/elrond-go/integrationTests/vm"
	"github.com/ElrondNetwork/elrond-go/integrationTests/vm/txsFee/utils"
	"github.com/ElrondNetwork/elrond-go/process"
	vmcommon "github.com/ElrondNetwork/elrond-vm-common"
	"github.com/stretchr/testify/require"
)

func TestAsyncESDTCallShouldWork(t *testing.T) {
	testContext, err := vm.CreatePreparedTxProcessorWithVMs(config.EnableEpochs{})
	require.Nil(t, err)
	defer testContext.Close()


	egldBalance := big.NewInt(100000000)
```

```
29          ownerAddr := []byte("12345678901234567890123456789010")
30          _, _ = vm.CreateAccount(testContext.Accounts, ownerAddr, 0, egldBalance)
31
32          // create an address with ESDT token
33          sndAddr := []byte("12345678901234567890123456789012")
34
35          esdtBalance := big.NewInt(100000000)
36          token := []byte("miiutoken")
37          utils.CreateAccountWithESDTBalance(t, testContext.Accounts, sndAddr, egldBalance, token, 0
38
39          // deploy 2 contracts
40          gasPrice := uint64(10)
41          ownerAccount, _ := testContext.Accounts.LoadAccount(ownerAddr)
42          deployGasLimit := uint64(50000)
43
44          argsSecond := [][]byte{[]byte(hex.EncodeToString(token))}
45          secondSCAddress := utils.DoDeploySecond(t, testContext, "../esdt/testdata/second-contract.
46
47          args := [][]byte{[]byte(hex.EncodeToString(token)), []byte(hex.EncodeToString(secondSCAddr
48          ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
49          firstSCAddress := utils.DoDeploySecond(t, testContext, "../esdt/testdata/first-contract.wa
50
51          testContext.TxFeeHandler.CreateBlockStarted(getZeroGasAndFees())
52          utils.CleanAccumulatedIntermediateTransactions(t, testContext)
53
54          gasLimit := uint64(500000)
55          tx := utils.CreateESDTTransferTx(0, sndAddr, firstSCAddress, token, big.NewInt(5000), gasP
56          tx.Data = []byte(string(tx.Data) + "@" + hex.EncodeToString([]byte("transferToSecondContra
57
58          retCode, err := testContext.TxProcessor.ProcessTransaction(tx)
59          require.Equal(t, vmcommon.Ok, retCode)
60          require.Nil(t, err)
61
62          _, err = testContext.Accounts.Commit()
63          require.Nil(t, err)
64
65          utils.CheckESDTBalance(t, testContext, firstSCAddress, token, big.NewInt(2500))
66          utils.CheckESDTBalance(t, testContext, secondSCAddress, token, big.NewInt(2500))
67
68          expectedSenderBalance := big.NewInt(95000000)
69          utils.TestAccount(t, testContext.Accounts, sndAddr, 1, expectedSenderBalance)
70
71          expectedAccumulatedFees := big.NewInt(5000000)
72          accumulatedFees := testContext.TxFeeHandler.GetAccumulatedFees()
73          require.Equal(t, expectedAccumulatedFees, accumulatedFees)
74
75          intermediateTxs := testContext.GetIntermediateTransactions(t)
76          testIndexer := vm.CreateTestIndexer(t, testContext.ShardCoordinator, testContext.Economics
77          testIndexer.SaveTransaction(tx, block.TxBlock, intermediateTxs)
```

```
78
79          indexerTx := testIndexer.GetIndexerPreparedTransaction(t)
80          require.Equal(t, tx.GasLimit, indexerTx.GasUsed)
81          require.Equal(t, "5000000", indexerTx.Fee)
82  }
83
84  func TestAsyncESDTCallSecondScRefusesPayment(t *testing.T) {
85          testContext, err := vm.CreatePreparedTxProcessorWithVMs(config.EnableEpochs{})
86          require.Nil(t, err)
87          defer testContext.Close()
88
89          egldBalance := big.NewInt(100000000)
90          ownerAddr := []byte("12345678901234567890123456789010")
91          _, _ = vm.CreateAccount(testContext.Accounts, ownerAddr, 0, egldBalance)
92
93          // create an address with ESDT token
94          sndAddr := []byte("12345678901234567890123456789012")
95
96          esdtBalance := big.NewInt(100000000)
97          token := []byte("miiutoken")
98          utils.CreateAccountWithESDTBalance(t, testContext.Accounts, sndAddr, egldBalance, token, 0
99
100         // deploy 2 contracts
101         gasPrice := uint64(10)
102         ownerAccount, _ := testContext.Accounts.LoadAccount(ownerAddr)
103         deployGasLimit := uint64(50000)
104
105         argsSecond := [][]byte{[]byte(hex.EncodeToString(token))}
106         secondSCAddress := utils.DoDeploySecond(t, testContext, "../esdt/testdata/second-contract.
107
108         args := [][]byte{[]byte(hex.EncodeToString(token)), []byte(hex.EncodeToString(secondSCAddr
109         ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
110         firstSCAddress := utils.DoDeploySecond(t, testContext, "../esdt/testdata/first-contract.wa
111
112         testContext.TxFeeHandler.CreateBlockStarted(getZeroGasAndFees())
113         utils.CleanAccumulatedIntermediateTransactions(t, testContext)
114         require.Equal(t, big.NewInt(0), testContext.TxFeeHandler.GetAccumulatedFees())
115
116         gasLimit := uint64(500000)
117         tx := utils.CreateESDTTransferTx(0, sndAddr, firstSCAddress, token, big.NewInt(5000), gasP
118         tx.Data = []byte(string(tx.Data) + "@" + hex.EncodeToString([]byte("transferToSecondContra
119
120         retCode, err := testContext.TxProcessor.ProcessTransaction(tx)
121         require.Equal(t, vmcommon.Ok, retCode)
122         require.Nil(t, err)
123
124         _, err = testContext.Accounts.Commit()
125         require.Nil(t, err)
126
```

```
127         utils.CheckESDTBalance(t, testContext, firstSCAddress, token, big.NewInt(5000))
128         utils.CheckESDTBalance(t, testContext, secondSCAddress, token, big.NewInt(0))
129
130         expectedSenderBalance := big.NewInt(95999990)
131         utils.TestAccount(t, testContext.Accounts, sndAddr, 1, expectedSenderBalance)
132
133         expectedAccumulatedFees := big.NewInt(4000010)
134         accumulatedFees := testContext.TxFeeHandler.GetAccumulatedFees()
135         require.Equal(t, expectedAccumulatedFees, accumulatedFees)
136
137         intermediateTxs := testContext.GetIntermediateTransactions(t)
138         testIndexer := vm.CreateTestIndexer(t, testContext.ShardCoordinator, testContext.Economics
139         testIndexer.SaveTransaction(tx, block.TxBlock, intermediateTxs)
140
141         indexerTx := testIndexer.GetIndexerPreparedTransaction(t)
142         require.Equal(t, uint64(400001), indexerTx.GasUsed)
143         require.Equal(t, "4000010", indexerTx.Fee)
144 }
145
146 func TestAsyncESDTCallsOutOfGas(t *testing.T) {
147         testContext, err := vm.CreatePreparedTxProcessorWithVMs(config.EnableEpochs{})
148         require.Nil(t, err)
149         defer testContext.Close()
150
151         egldBalance := big.NewInt(100000000)
152         ownerAddr := []byte("12345678901234567890123456789010")
153         _, _ = vm.CreateAccount(testContext.Accounts, ownerAddr, 0, egldBalance)
154
155         // create an address with ESDT token
156         sndAddr := []byte("12345678901234567890123456789012")
157
158         esdtBalance := big.NewInt(100000000)
159         token := []byte("miiutoken")
160         utils.CreateAccountWithESDTBalance(t, testContext.Accounts, sndAddr, egldBalance, token, 0
161
162         // deploy 2 contracts
163         gasPrice := uint64(10)
164         ownerAccount, _ := testContext.Accounts.LoadAccount(ownerAddr)
165         deployGasLimit := uint64(50000)
166
167         argsSecond := [][]byte{[]byte(hex.EncodeToString(token))}
168         secondSCAddress := utils.DoDeploySecond(t, testContext, "../esdt/testdata/second-contract.
169
170         args := [][]byte{[]byte(hex.EncodeToString(token)), []byte(hex.EncodeToString(secondSCAddr
171         ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
172         firstSCAddress := utils.DoDeploySecond(t, testContext, "../esdt/testdata/first-contract.wa
173
174         testContext.TxFeeHandler.CreateBlockStarted(getZeroGasAndFees())
175         utils.CleanAccumulatedIntermediateTransactions(t, testContext)
```

```
176
177         gasLimit := uint64(2000)
178         tx := utils.CreateESDTTransferTx(0, sndAddr, firstSCAddress, token, big.NewInt(5000), gasP
179         tx.Data = []byte(string(tx.Data) + "@" + hex.EncodeToString([]byte("transferToSecondContra
180
181         retCode, err := testContext.TxProcessor.ProcessTransaction(tx)
182         require.Equal(t, vmcommon.UserError, retCode)
183         require.Nil(t, err)
184
185         _, err = testContext.Accounts.Commit()
186         require.Nil(t, err)
187
188         utils.CheckESDTBalance(t, testContext, firstSCAddress, token, big.NewInt(0))
189         utils.CheckESDTBalance(t, testContext, secondSCAddress, token, big.NewInt(0))
190
191         expectedSenderBalance := big.NewInt(99980000)
192         utils.TestAccount(t, testContext.Accounts, sndAddr, 1, expectedSenderBalance)
193
194         expectedAccumulatedFees := big.NewInt(20000)
195         accumulatedFees := testContext.TxFeeHandler.GetAccumulatedFees()
196         require.Equal(t, expectedAccumulatedFees, accumulatedFees)
197
198         intermediateTxs := testContext.GetIntermediateTransactions(t)
199         testIndexer := vm.CreateTestIndexer(t, testContext.ShardCoordinator, testContext.Economics
200         testIndexer.SaveTransaction(tx, block.TxBlock, intermediateTxs)
201
202         indexerTx := testIndexer.GetIndexerPreparedTransaction(t)
203         require.Equal(t, tx.GasLimit, indexerTx.GasUsed)
204         require.Equal(t, "20000", indexerTx.Fee)
205 }
206
207 func TestAsyncMultiTransferOnCallback(t *testing.T) {
208         testContext, err := vm.CreatePreparedTxProcessorWithVMs(config.EnableEpochs{})
209         require.Nil(t, err)
210         defer testContext.Close()
211
212         ownerAddr := []byte("12345678901234567890123456789010")
213         sftTokenID := []byte("SFT-123456")
214         sftNonce := uint64(1)
215         sftBalance := big.NewInt(1000)
216         halfBalance := big.NewInt(500)
217
218         utils.CreateAccountWithESDTBalance(t, testContext.Accounts, ownerAddr, big.NewInt(10000000
219         utils.CheckESDTNFTBalance(t, testContext, ownerAddr, sftTokenID, sftNonce, sftBalance)
220
221         gasPrice := uint64(10)
222         ownerAccount, _ := testContext.Accounts.LoadAccount(ownerAddr)
223         deployGasLimit := uint64(1000000)
224         txGasLimit := uint64(1000000)
```

```
225
226         // deploy forwarder
227         forwarderAddr := utils.DoDeploySecond(t,
228                 testContext,
229                 "../esdt/testdata/forwarder-raw-managed-api.wasm",
230                 ownerAccount,
231                 gasPrice,
232                 deployGasLimit,
233                 nil,
234                 big.NewInt(0),
235         )
236
237         // deploy vault
238         ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
239         vaultAddr := utils.DoDeploySecond(t,
240                 testContext,
241                 "../esdt/testdata/vault-managed-api.wasm",
242                 ownerAccount,
243                 gasPrice,
244                 deployGasLimit,
245                 nil,
246                 big.NewInt(0),
247         )
248
249         // send the tokens to vault
250         ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
251         tx := utils.CreateESDTNFTTransferTx(
252                 ownerAccount.GetNonce(),
253                 ownerAddr,
254                 vaultAddr,
255                 sftTokenID,
256                 sftNonce,
257                 sftBalance,
258                 gasPrice,
259                 txGasLimit,
260                 "just_accept_funds",
261         )
262         retCode, err := testContext.TxProcessor.ProcessTransaction(tx)
263         require.Equal(t, vmcommon.Ok, retCode)
264         require.Nil(t, err)
265
266         _, err = testContext.Accounts.Commit()
267         require.Nil(t, err)
268
269         utils.CheckESDTNFTBalance(t, testContext, vaultAddr, sftTokenID, sftNonce, sftBalance)
270
271         lenSCRs := len(testContext.GetIntermediateTransactions(t))
272         // receive tokens from vault to forwarder on callback
273         // receive 500 + 500 of the SFT through multi-transfer
```

```
274         ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
275         tx = utils.CreateSmartContractCall(
276                 ownerAccount.GetNonce(),
277                 ownerAddr,
278                 forwarderAddr,
279                 gasPrice,
280                 txGasLimit,
281                 "forward_async_retrieve_multi_transfer_funds",
282                 vaultAddr,
283                 sftTokenID,
284                 big.NewInt(int64(sftNonce)).Bytes(),
285                 halfBalance.Bytes(),
286                 sftTokenID,
287                 big.NewInt(int64(sftNonce)).Bytes(),
288                 halfBalance.Bytes(),
289         )
290         retCode, err = testContext.TxProcessor.ProcessTransaction(tx)
291         require.Equal(t, vmcommon.Ok, retCode)
292         require.Nil(t, err)
293         require.Equal(t, 1, len(testContext.GetIntermediateTransactions(t))-lenSCRs)
294
295         _, err = testContext.Accounts.Commit()
296         require.Nil(t, err)
297
298         utils.CheckESDTNFTBalance(t, testContext, forwarderAddr, sftTokenID, sftNonce, sftBalance)
299 }
300
301 func TestAsyncMultiTransferOnCallAndOnCallback(t *testing.T) {
302         testContext, err := vm.CreatePreparedTxProcessorWithVMs(config.EnableEpochs{})
303         require.Nil(t, err)
304         defer testContext.Close()
305
306         ownerAddr := []byte("12345678901234567890123456789010")
307         sftTokenID := []byte("SFT-123456")
308         sftNonce := uint64(1)
309         sftBalance := big.NewInt(1000)
310         halfBalance := big.NewInt(500)
311
312         utils.CreateAccountWithESDTBalance(t, testContext.Accounts, ownerAddr, big.NewInt(10000000
313         utils.CheckESDTNFTBalance(t, testContext, ownerAddr, sftTokenID, sftNonce, sftBalance)
314
315         gasPrice := uint64(10)
316         ownerAccount, _ := testContext.Accounts.LoadAccount(ownerAddr)
317         deployGasLimit := uint64(1000000)
318         txGasLimit := uint64(1000000)
319
320         // deploy forwarder
321         forwarderAddr := utils.DoDeploySecond(t,
322                 testContext,
```

```
323                 "../esdt/testdata/forwarder-raw-managed-api.wasm",
324                 ownerAccount,
325                 gasPrice,
326                 deployGasLimit,
327                 nil,
328                 big.NewInt(0),
329         )
330
331         // deploy vault
332         ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
333         vaultAddr := utils.DoDeploySecond(t,
334                 testContext,
335                 "../esdt/testdata/vault-managed-api.wasm",
336                 ownerAccount,
337                 gasPrice,
338                 deployGasLimit,
339                 nil,
340                 big.NewInt(0),
341         )
342
343         // set vault roles
344         utils.SetESDTRoles(t, testContext.Accounts, vaultAddr, sftTokenID, [][]byte{
345                 []byte(core.ESDTRoleNFTAddQuantity),
346                 []byte(core.ESDTRoleNFTCreate),
347                 []byte(core.ESDTRoleNFTBurn),
348         })
349         // set lastNonce for vault
350         utils.SetLastNFTNonce(t, testContext.Accounts, vaultAddr, sftTokenID, 1)
351
352         // send the tokens to forwarder
353         ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
354         tx := utils.CreateESDTNFTTransferTx(
355                 ownerAccount.GetNonce(),
356                 ownerAddr,
357                 forwarderAddr,
358                 sftTokenID,
359                 sftNonce,
360                 sftBalance,
361                 gasPrice,
362                 txGasLimit,
363                 "deposit",
364         )
365         retCode, err := testContext.TxProcessor.ProcessTransaction(tx)
366         require.Equal(t, vmcommon.Ok, retCode)
367         require.Nil(t, err)
368
369         _, err = testContext.Accounts.Commit()
370         require.Nil(t, err)
371
```

```
372         utils.CheckESDTNFTBalance(t, testContext, forwarderAddr, sftTokenID, sftNonce, sftBalance)
373
374         // send tokens to vault, vault burns and creates new ones, sending them on forwarder's cal
375         ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
376         tx = utils.CreateSmartContractCall(
377                 ownerAccount.GetNonce(),
378                 ownerAddr,
379                 forwarderAddr,
380                 gasPrice,
381                 txGasLimit,
382                 "forwarder_async_send_and_retrieve_multi_transfer_funds",
383                 vaultAddr,
384                 sftTokenID,
385                 big.NewInt(int64(sftNonce)).Bytes(),
386                 halfBalance.Bytes(),
387                 sftTokenID,
388                 big.NewInt(int64(sftNonce)).Bytes(),
389                 halfBalance.Bytes(),
390         )
391         retCode, err = testContext.TxProcessor.ProcessTransaction(tx)
392         require.Equal(t, vmcommon.Ok, retCode)
393         require.Nil(t, err)
394
395         _, err = testContext.Accounts.Commit()
396         require.Nil(t, err)
397
398         utils.CheckESDTNFTBalance(t, testContext, forwarderAddr, sftTokenID, 2, halfBalance)
399         utils.CheckESDTNFTBalance(t, testContext, forwarderAddr, sftTokenID, 3, halfBalance)
400 }
401
402 func TestSendNFTToContractWith0Function(t *testing.T) {
403         testContext, err := vm.CreatePreparedTxProcessorWithVMs(config.EnableEpochs{})
404         require.Nil(t, err)
405         defer testContext.Close()
406
407         ownerAddr := []byte("12345678901234567890123456789010")
408         sftTokenID := []byte("SFT-123456")
409         sftNonce := uint64(1)
410         sftBalance := big.NewInt(1000)
411
412         utils.CreateAccountWithESDTBalance(t, testContext.Accounts, ownerAddr, big.NewInt(10000000
413         utils.CheckESDTNFTBalance(t, testContext, ownerAddr, sftTokenID, sftNonce, sftBalance)
414
415         gasPrice := uint64(10)
416         ownerAccount, _ := testContext.Accounts.LoadAccount(ownerAddr)
417         deployGasLimit := uint64(1000000)
418         txGasLimit := uint64(1000000)
419
420         vaultAddr := utils.DoDeploySecond(t,
```

```
421                 testContext,
422                 "../esdt/testdata/vault-managed-api.wasm",
423                 ownerAccount,
424                 gasPrice,
425                 deployGasLimit,
426                 nil,
427                 big.NewInt(0),
428         )
429
430         // send the tokens to vault
431         ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
432         tx := utils.CreateESDTNFTTransferTx(
433                 ownerAccount.GetNonce(),
434                 ownerAddr,
435                 vaultAddr,
436                 sftTokenID,
437                 sftNonce,
438                 sftBalance,
439                 gasPrice,
440                 txGasLimit,
441                 "",
442         )
443         tx.Data = append(tx.Data, []byte("@")...)
444         retCode, err := testContext.TxProcessor.ProcessTransaction(tx)
445         require.Equal(t, vmcommon.Ok, retCode)
446         require.Nil(t, err)
447
448         _, err = testContext.Accounts.Commit()
449         require.Nil(t, err)
450 }
451
452 func TestSendNFTToContractWith0FunctionNonPayable(t *testing.T) {
453         testContext, err := vm.CreatePreparedTxProcessorWithVMs(config.EnableEpochs{})
454         require.Nil(t, err)
455         defer testContext.Close()
456
457         ownerAddr := []byte("12345678901234567890123456789010")
458         sftTokenID := []byte("SFT-123456")
459         sftNonce := uint64(1)
460         sftBalance := big.NewInt(1000)
461
462         utils.CreateAccountWithESDTBalance(t, testContext.Accounts, ownerAddr, big.NewInt(10000000
463         utils.CheckESDTNFTBalance(t, testContext, ownerAddr, sftTokenID, sftNonce, sftBalance)
464
465         gasPrice := uint64(10)
466         ownerAccount, _ := testContext.Accounts.LoadAccount(ownerAddr)
467         deployGasLimit := uint64(1000000)
468         txGasLimit := uint64(1000000)
469
```

```
	vaultAddr := utils.DoDeployWithMetadata(t,
		testContext,
		"../esdt/testdata/vault-managed-api.wasm",
		ownerAccount,
		gasPrice,
		deployGasLimit,
		[]byte("0000"),
		nil,
		big.NewInt(0),
	)

	// send the tokens to vault
	ownerAccount, _ = testContext.Accounts.LoadAccount(ownerAddr)
	tx := utils.CreateESDTNFTTransferTx(
		ownerAccount.GetNonce(),
		ownerAddr,
		vaultAddr,
		sftTokenID,
		sftNonce,
		sftBalance,
		gasPrice,
		txGasLimit,
		"",
	)
	tx.Data = append(tx.Data, []byte("@")...)
	retCode, err := testContext.TxProcessor.ProcessTransaction(tx)
	require.Equal(t, vmcommon.UserError, retCode)
	require.Equal(t, process.ErrFailedTransaction, err)

	_, err = testContext.Accounts.Commit()
	require.Nil(t, err)
}
```