STARKWARE

Bridging with StarkNet

Arturo Castañon

https://github.com/starknet-edu | @starkwareltd

https://github.com/ArturVargas | @0xVato

Octubre 2022

¿Qué son los Puentes?

Un puente o blockchain bridge conecta dos ecosistemas, esto facilita la comunicación entre cadenas a través de la transferencia de información y activos.

- Referencia: Blockchain bridges

Que tal si...

- Permisos.
- Poder Computacional.
- Instrucciones.

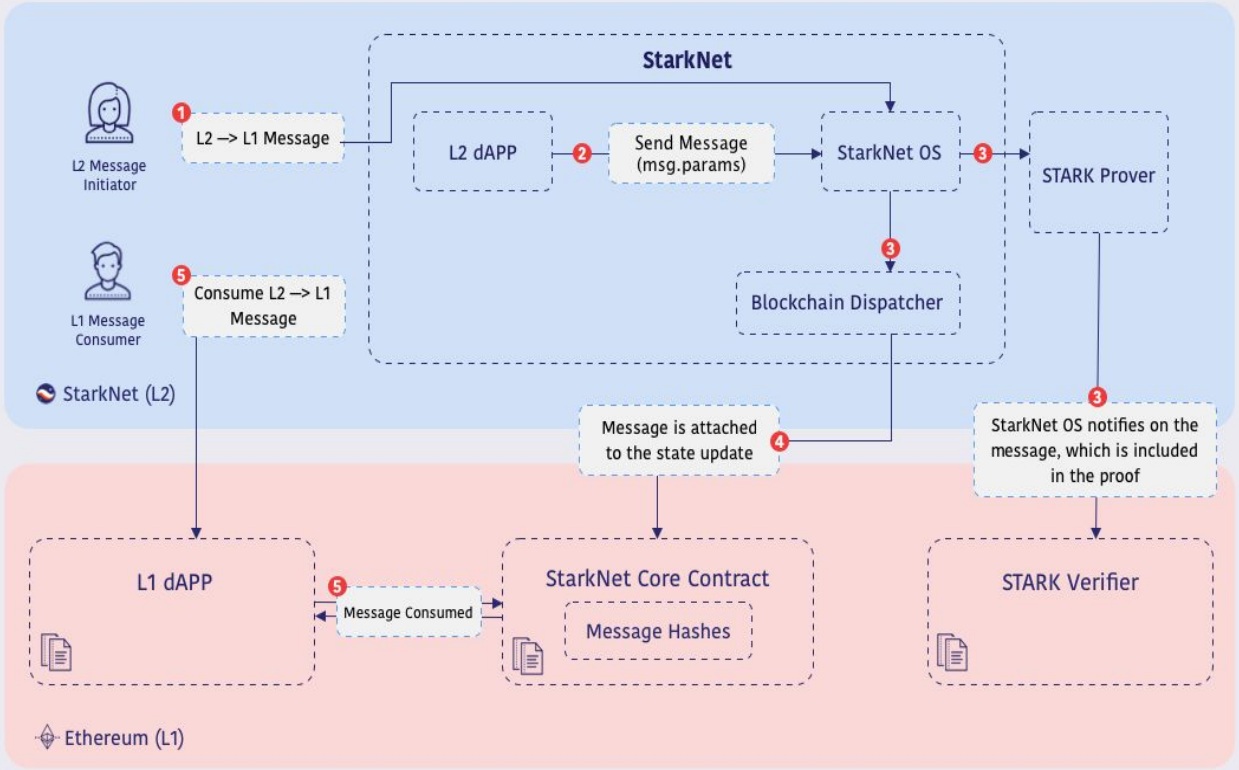
STARKWARE

Comunicandonos con Ethereum

Mensajes de L1 a L2

STARKWARE

L2 —> L1 Message Mechanism



Status	Description
Submit Tx	L2 Client sends TX
'send_message_to_l1'	TX initializes L2 Contract syscall to send message to L1 Contract
Proof	L2 TX including message included in Proof and verified on L1
State Update	Message attached to state update, Core Contract increases counter, and 'LogMessageToL1' event emit
Message Cleared	L1 Recipient Contract can consume message in L1 TX via 'consumeMessageFromL2'. Upon valid msg params counter decrement and message considered handled

@StarkNetEs

Ejemplo de un Token Bridge

STARKWARE

En este ejemplo veremos cómo construir un puente entre L2 y L1 para pasar assets de forma simple **código no apto para producción**.

Objetivo:
El usuario podrá depositar tokens en L1 y su balance se incrementará en L2.
También podrá retirar algunos tokens, lo que disminuirá su balance en L2 e incrementará en L1.

Primeras Lineas

STARKWARE

```
%lang starknet

from starkware.cairo.common.alloc import alloc
from starkware.cairo.common.cairo_builtins import HashBuiltin
from starkware.cairo.common.math import assert_nn
from starkware.starknet.common.messages import send_message_to_l1

const MESSAGE_WITHDRAW = 0;

@storage_var
func L1_CONTRACT_ADDRESS() -> (contract_address: felt) {
}
```

Usamos una variable storage para guardar el balance y creamos una función getter para recuperarlo

STARKWARE

```
13
14  // A mapping from a user (L1 Ethereum address) to their balance.
15  @storage_var
16  func balance(l1_user: felt) -> (amount: felt) {
17  }
18
19  @view
20  func get_balance{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, range_check_ptr}(
21      l1_user: felt
22  ) -> (balance: felt) {
23      let (res) = balance.read(l1_user=l1_user);
24      return (balance=res);
25  }
26
```

Creamos fondos para probar el cambio de balances.

STARKWARE

```
26
27  // set the contract address
28  @external
29  func set_l1_contract{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, range_check_ptr}(
30      l1_address: felt
31  ) {
32      L1_CONTRACT_ADDRESS.write(l1_address);
33      return();
34  }
35
36  // Function to print some money to play with
37  @external
38  func increase_balance{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, range_check_ptr}(
39      user: felt, amount: felt
40  ) {
41      let (res) = balance.read(l1_user=user);
42      balance.write(user, res + amount);
43      return ();
44  }
45
```

Enviando mensajes a L1

STARKWARE

```
45
46  // Sending a message to L1
47  @external
48  func withdraw{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, range_check_ptr}(
49      user: felt, amount: felt
50  ) {
51      // Make sure 'amount' is positive.
52      assert_nn(amount);
53
54      let (res) = balance.read(l1_user=user);
55      tempvar new_balance = res - amount;
56
57      // Make sure the new balance will be positive.
58      assert_nn(new_balance);
59
60      // Update the new balance.
61      balance.write(user, new_balance);
62
63      // Send the withdrawal message.
64      let (message_payload: felt*) = alloc();
65      assert message_payload[0] = MESSAGE_WITHDRAW;
66      assert message_payload[1] = user;
67      assert message_payload[2] = amount;
68      let (contract_address) = L1_CONTRACT_ADDRESS.read();
69      send_message_to_l1(to_address=contract_address, payload_size=3, payload=message_payload);
70
71      return ();
72  }
73
```

Recibiendo mensajes desde L1

STARKWARE

```
74
75  // Receiving messages from L1
76  @l1_handler
77  func deposit{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, range_check_ptr}(
78      from_address: felt, user: felt, amount: felt
79  ) {
80      // Make sure the message was sent by the intended L1 contract.
81      let (contract_address) = L1_CONTRACT_ADDRESS.read();
82      assert from_address = contract_address;
83
84      // Read the current balance.
85      let (res) = balance.read(l1_user=user);
86
87      // Compute and update the new balance.
88      tempvar new_balance = res + amount;
89      balance.write(user, new_balance);
90
91      return ();
92  }
93
```

STARKWARE

Contrato en L1

@StarkNetEs

STARKWARE

¿Por qué esto es cool?

- Ethereum tiene certeza de lo que está pasando en StarkNet.
- Ethereum tiene la prueba matemática de que ese mensaje fue enviado.
- Control Remoto completo: Enviar instrucciones desde L1 a L2 y viceversa.
- Bridge: Los Assets pueden alojarse en cualquier otro smart contract que use el bridge.
    - Por ejemplo, un porcentaje de la liquidez puede alojarse en AAVE para generar rendimientos.

STARKWARE

La frecuencia de verificación en testnet actualmente es de 2 horas.

Para llegar más lejos

STARKWARE

Recursos en StarkNet

- Documentación oficial de Cairo
- Links de la Comunidad - lista de ejemplos seleccionados por el equipo de StarkWare
- Awesome StarkNet - una lista curada de recursos por Georgios Konstantopoulos

Cosas geniales construidas en StarkNet

- Bridge Vault
- briq V1 contracts
- Qasr, an ETH <> StarkNet NFT bridge
- Tictactoe by @guiltygyoza

STARKWARE

Gracias!

Equipo educativo de StarkNet

@starkwareltd

Octubre 2022