Sissiz Blokzincir

İlk işlemden
RpcNode ile kendi servisinize



Sade bir dille · çizimli ve örnekli

Sade bir dille

Sissiz Blokzincir

İlk işlemden RpcNode ile kendi servisinize



Bu kitap size göre, eğer

- "blokzincir" sözünü duydunuz ama nasıl çalıştığını bilmiyorsanız;
- sitenizde kripto parayla ödeme almak istiyorsanız;
- program yazıyor ama bir node'a henüz hiç istek göndermediyseniz.

Neye ihtiyacınız var

- meraka — önceden bir şey bilmeniz gerekmiyor;
- örnekleri tekrarlamak için terminali olan bir bilgisayara;
- ilk istekler kayıt ve anahtar olmadan çalışır.

Bölümler nasıl okunur



Yazar: rpcnode.dev · 2026

İçindekiler

Kısım 1. Neden sağlayıcı gerekli

1. Kripto kabul etmek için ne gerekir? 4
2. Her şeyi kendiniz yapmak kaça gelir? 9
3. Sağlayıcı mı, kendi node'unuz mu? 15

Kısım 2. Sıfırdan blokzincir

4. Aracısız para ve kayıt 20
5. Bloklar ve zincir 25
6. Anahtarlar, adresler ve imza 30
7. İşlemin yolculuğu 35
8. Ağ nasıl anlaşır 40
9. Coinler, tokenler ve akıllı sözleşmeler 46

Kısım 3. Node'lar ve RPC

10. Node çeşitleri 52
11. Node ile nasıl konuşulur 57
12. Kendi node'unuzun içi 63
13. Sağlayıcı nasıl seçilir 69

Kısım 4. Ağlar nasıl çalışır

14. Bitcoin ve akrabaları 74
15. Ethereum: hesaplar, gas ve olaylar 79
16. Ethereum'un akrabaları ve ikinci katlar 84
17. TRON: enerji ve stablecoinler 89
18. Solana: her şey hesaplarda 95
19. TON: telefon yerine mektup 101
20. Başka yaklaşımlar: Sui, XRP, Cardano ve diğerleri 106
21. Hangi ağı seçmeli 113

Kısım 5. RpcNode

22. RpcNode'da ilk beş dakika 119
23. İsteğin başına neler gelir 125
24. Geçmiş ve arşiv verileri 131
25. Planlar, çağrı bedeli ve limitler 137
26. Adres olayları: webhook ve soketler 143
27. Ödemeler ve işlem izleme 149

Kısım 6. Uygulama

28. Mağazayı kuruyoruz 155
29. İşler ters gidince 162
30. Güvenlik ve kontrol listesi 169

Kısım 7. Sırada ne var

31. DeFi: aracısız finans 175

Kitabın sonunda

Sonuç 182
Ek A — Sözlük 184
Ek B — Tek sayfada ağlar 186
Ek C — İstek kopya kâğıdı 188

Bölüm 1. Kripto kabul etmek istiyorum: bunun için ne gerekir?

Her şey bir müşteriyle başladı

Artyom'un küçük bir internet mağazası var: kahve kavurup kargoyla gönderiyor. Kart, havale, kapıda ödeme — hepsi çalışıyor. Derken bir mektup geliyor:

> "Merhaba! İki kilo sipariş vermek istiyorum. Kripto parayla ödeyebilir miyim?"

Artyom'un itirazı yok. Telefonuna bir cüzdan kuruyor, bir adres alıyor ve müşteriye gönderiyor. Birkaç dakika sonra para ekranda görünüyor.

Onuncu siparişte cüzdanı elle kontrol etmek sıkmaya başlıyor. İnsan, sitenin siparişin ödendiğini kendisi anlamasını istiyor.



İnceleyelim: node nedir

Blokzincir hakkında bir şey öğrenmek için onu saklayana sormak gerekir. İşte o saklayana **node** denir.

Kimin kime ne kadar gönderdiğinin yazıldığı ortak bir defter düşünün. Bu defter tek bir kasada durmaz. Kopyaları dünyanın dört bir yanındaki pek çok katılımcıdadır ve bunlar sürekli birbiriyle karşılaştırılır. Blokzincir budur; ayrıntılarını 5. bölümde inceleyeceğiz.

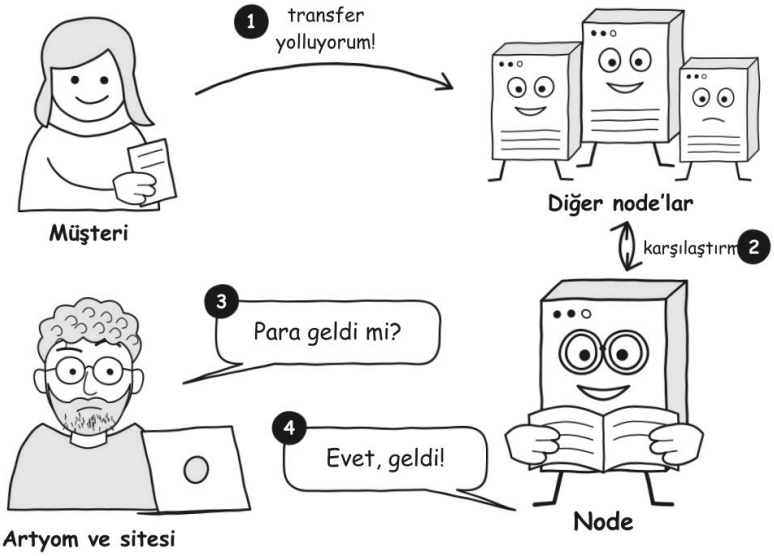
Node, üzerinde ağın programı çalışan bir bilgisayardır. Dört iş yapar:

1. Kayıtların kendi kopyasını saklar.
2. Diğer node'lardan yeni kayıtlar alır ve onları ağın kurallarına göre denetler.

3. Kendisine gönderilen transferleri başkalarına iletir.
4. Programların sorularını yanıtlar: “bakiye ne kadar?”, “transfer geçti mi?”.

Node’lar çeşit çeşittir: bazıları geçmişin tamamını saklamaz. Bunu 10. bölümde anlatacağız.

Artyom’un sitesi blokzincire kendisi bakmaz. Soruyu node’a sorar, node da yanıtlar.



Deneyelim: node’a ilk soru

Programlar node ile basit kurallara göre konuşur: istek gönderilir, yanıt alınır. Bu kurallardan birine **JSON-RPC** denir. İşte Ethereum ağındaki bir node’a sorulan soru: “bu adreste kaç coin var?”

```
curl https://NODE_ADRESİ \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getBalance","params":["CÜZDAN_ADRESİ","lates
t"]}'
```

Burada `NODE_ADRESİ` kapısını çaldığımız yer, `CÜZDAN_ADRESİ` ise bakiyesini sorduğumuz adrestir. `latest` sözcüğü “en güncel an itibarıyla” demektir.

Node şöyle yanıtlar:

```
{"jsonrpc":"2.0","id":1,"result":"0xde0b6b3a7640000"}
```

Rastgele harfler gibi görünüyor ama bu, on altılık sistemde yazılmış bir sayı: 1.000.000.000.000.000.000. Node, coinin **wei** denen en küçük parçalarıyla sayar. Bir ETH tam bu kadar wei eder. Demek ki adreste 1 ETH var.

Yani bütün numara bir soru ve bir yanıttan ibaret. Geriye `NODE_ADRESİ` kimin olacak, buna karar vermek kalıyor.

İki yol: kendi node'unuz veya sağlayıcı

Node'u kendiniz çalıştırabilirsiniz. Ya da birinin sizin için ayakta tuttuğu bir node'a bağlanabilirsiniz. Böyle bir hizmete **sağlayıcı** denir. Size hazır bir `NODE_ADRESİ` verir. Bu kitabın konusu olan RpcNode da bu hizmetlerden biridir.

Soru	Kendi node'unuz	Sağlayıcı
Ne zaman başlanır	Kurulum ve eşitlemeden sonra: ağa göre saatlerden haftalara	Kayıttan birkaç dakika sonra
Neye para ödenir	Sunucu, diskler, trafik ve uzman zamanı	Hizmet planı
Kim günceller ve onarır	Siz	Sağlayıcı
Birden fazla ağ	Her biri için ayrı node	Genellikle tek bir panelden
Riskiniz ne	Siz uyurken node geri kalmış veya bozulmuş olabilir	Başkasının hizmetine bağımlısınız

İki yolun da yeri var. Özel bir gizlilik gerekiyorsa, yük çok büyükse veya ağın işleyişine bizzat katılıyorsanız kendi node'unuz mantıklıdır.

> **Artyom**: "Ben kahve kavuruyorum, sunucu başında nöbet tutmuyorum. Peki yine de kendim yapsam kaça gelir?"

Kısa cevap: yedek sunucuları ve tam geçmişiyle birkaç ağ için hesap on binlerce doları bulur. Küçük tek bir node ise çok daha ucuza çalıştırılabilir. Ayrıntılı hesap 2. bölümde.

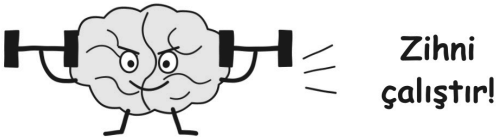
Tuzak: "sağlayıcıya cüzdanın anahtarını veririm"

Asla. Bakiyeyi öğrenmek veya ödemeyi görmek için node'a adres yeter. Özel anahtar ve gizli kurtarma ifadesi paraya hükmeder: bunları bilen her şeyi alabilir. Dürüst bir sağlayıcının bunlara ihtiyacı yoktur. Anahtarları 6. bölümde inceleyeceğiz.



Bir uyarı daha. Yukarıdaki istek yalnızca ağın ana coinini gösterir; burada bu ETH'dir. Müşteri bir token ile, örneğin USDT ile ödüyorsa bu soru onu görmez. Tokenler başka türlü sorulur; o da 9. bölümde.

Düşünün



1. Artyom ödemeyi kontrol ediyor ama node'u ağın bir saat gerisinde. Ne görür ve müşteriye ne yazar?
2. Mağazada günde üç sipariş var. Ya üç bin olsa? Kendi node'u ile sağlayıcı arasındaki seçimde ne değişir?

İlk soru için ipucu: geri kalmış node, bir saat önce ne varsa onu dürüstçe gösterir.

Özet

- Program, blokzincirde olanı node'a soru sorarak öğrenir.
- Node'u kendiniz de tutabilirsiniz, sağlayıcıdan da alabilirsiniz.
- Node'un adrese ihtiyacı vardır, özel anahtarınıza değil.

Sırada: 2. bölüm. Artyom hesap makinesini alıp kendi sunucularının kaça patlayacağını hesaplıyor.

Bölüm 2. Her şeyi kendiniz yapmak kaça gelir?

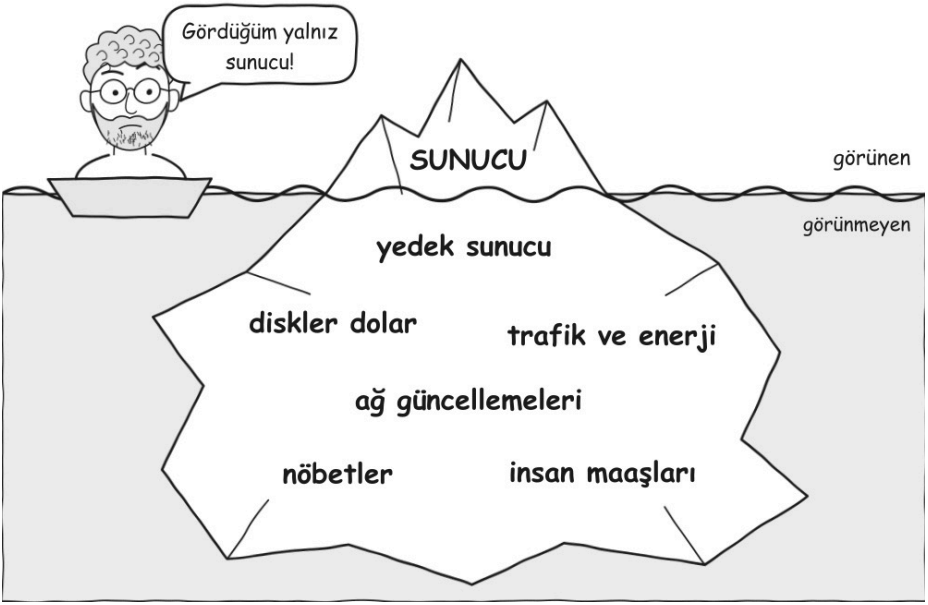
Artyom hesap makinesini alıyor

Artyom'un mantığı basit: "Bir sunucu alırım, node kurarım, ödemeler de kendiliğinden yürür. Parayı bir kez veririm." Hesap makinesini açıp yazmaya başlıyor.



İnceleyelim: hesap nelerden oluşur

Sunucu yalnızca görünen kısım. Gerisi ilk bakışta fark edilmiyor.



1. **Sunucu.** İşlemci, bellek ve hızlı NVMe diskler.
2. **Yedek sunucu.** Biri bozulunca, o onarılırken ikincisi yanıt verir.
3. **Diskler dolar.** Blokzincire hep yeni kayıt eklenir. Bitcoin zinciri Kasım 2026'ya kadarki bir yılda yaklaşık 700 GB'tan 786 GB'a büyüdü.
4. **Trafik.** Node sürekli veri alışverişi yapar. TON liteserver'ı yoğun dönemde ayda yaklaşık 16 TB ister.
5. **Güncellemeler.** Ağ kurallarını değiştirir, eski program ayak uyduramaz olur.
6. **İnsanlar.** Birinin bütün bunları kurması, izlemesi ve gece uyanması gerekir.

Her ağ ne kadar donanım ister

Bunlar resmi belgelerdeki sıradan node gereksinimleri; ağdaki oylamaya katılmak için olanlar değil.

Ağ	İşlemci	Bellek	Disk
Bitcoin	en az 1 GHz	2 GB	750 GB öneriliyor, oysa zincirin kendisi şimdiden yaklaşık 786 GB
Ethereum	4+ çekirdek	32 GB	4 TB NVMe
BNB Smart Chain	16 çekirdek	64 GB	en az 3 TB SSD; veritabanının tam anlık görüntüsü şimdiden yaklaşık 7,4 TB, budanmışı yaklaşık 1,9 TB
TRON	16 çekirdek	32 GB	en az 3,5–4 TB SSD
Solana (RPC node'u)	16 çekirdek ve üzeri	en az 512 GB	birkaç NVMe: hesaplar 1 TB+, günlük 1 TB+, anlık görüntüler 500 GB+

Tuhaflıkları fark ettiniz mi? Bitcoin için zincirin kendisinden küçük bir disk öneriliyor. BNB için asgari 3 TB deniyor, oysa tam anlık görüntü bunun iki katından fazla. Belgeler her zaman ağa yetişemiyor, o yüzden disk bol tutulur.

Ya geçmişin tamamı gerekiyorsa?

Sıradan node bugünü iyi bilir. "Bir yıl önce bakiye ne kadardı?" diye sormak için **arşiv** node'u gerekir; onu 10. bölümde anlatacağız. Arşiv node'u belirgin biçimde ağırdır. TON arşivi 16–20 TB ister. Ethereum'da boyut büyük ölçüde programa bağlıdır: 2 TB'tan 14 TB ve üzerine. Hesabımıza arşivleri katmıyoruz: katsak daha pahalı çıkar.

Parayı sayalım

Kendi sunucunuzu satın alabilirsiniz ama biz kiralamayı hesaplayacağız. Bu, barındırma firmalarının 27 Kasım 2026 tarihli açık fiyat listelerine dayanan, vergiler ve kurulum ücreti hariç bir tahmindir. Fiyatlar değişir.

Sunucu sınıfı	İçinde ne var	Ayda yaklaşık
Orta	8 çekirdek, 64 GB, 2×2 TB NVMe	$180
Büyük	16 çekirdek, 128 GB, 4–8 TB NVMe	$460–560
Çok büyük	512 GB ve üzeri bellek	$1.160–2.350

Şimdi beş ağ için. Fiyatları alt sınıra yakın alıyor ve her ağa iki sunucu koyuyoruz: biri çalışan, biri yedek.

Ağ	Sınıf	Tek sunucu, aylık	Yedekli, aylık
Bitcoin	orta	$180	$360
Ethereum	orta	$180	$360
BNB Smart Chain	büyük	$500	$1.000
TRON	büyük	$500	$1.000
Solana	çok büyük	$1.200	$2.400
Toplam		**$2.560**	**$5.120**

Artyom: "Ayda 5.120 dolar, yılda 61.440 eder. Üstelik bu yalnızca kira, öyle mi?!"

Evet. İlk tablodaki üst fiyatları alırsanız yılda yaklaşık $92.000 çıkar. "On binlerce dolar" sözü buradan geliyor.

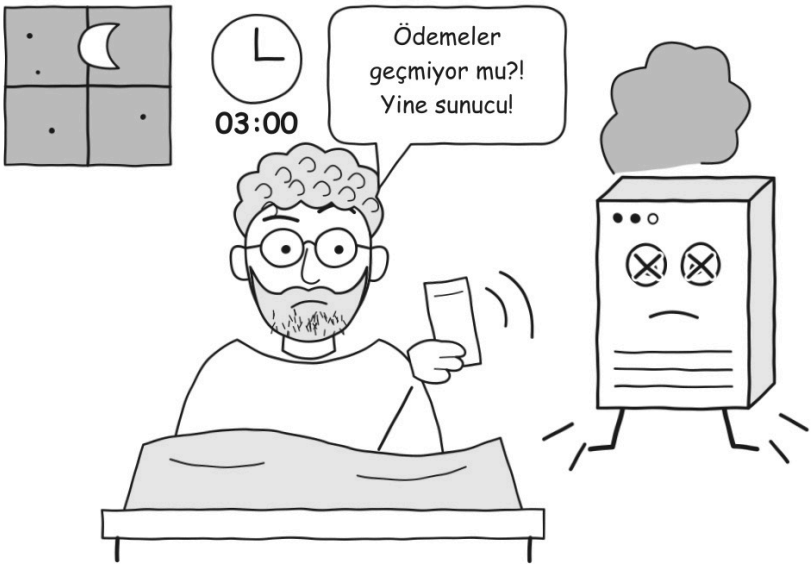
Tek bir ağı ayakta tutmak çok daha ucuz. Bitcoin kanaatkârdır: ona orta sınıfın altında bir sunucu bile yeter.

Peki bu işlerle kim uğraşacak?

Sunucular kendi bakımını yapmaz. "Habr Kariyer" verilerine göre Rusya'da orta düzey bir DevOps mühendisi ayda yaklaşık 160–290 bin ruble alıyor. Tek bir insan gece gündüz, hafta sonu ve izin olmadan nöbet tutamaz. İki kişi bile yılda yaklaşık 3,8–7 milyon ruble demektir.

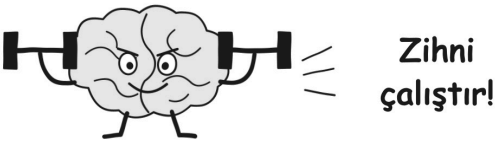
Tuzak: "bir sunucu alırım, iş biter"

Tek sunucu, tek arıza noktasıdır. Disk doldu, program takıldı, ağ güncellendi — site ödemeyi görmez olur. Müşteri parayı göndermiştir ama sipariş "ödenmedi" görünür.



Sık unutulan bir şey daha var. Ödeme yapmadan önce barındırma firmasının kurallarını okuyun: bazıları kripto parayla ilgili programları yasaklar ve sunucuyu kapatabilir.

Düşünün



1. Artyom'a yalnızca Bitcoin lazım; orta sınıf yedek sunucuyla birlikte. Yılda kaça gelir?
2. Beş ağa, $500'lık büyük sunucularda ve yine yedekli bir altıncısı eklendi. Yıllık hesap ne kadar artar?
3. Müşteriler günde üç kez ödeme yapıyor. Bu masrafa değer mi?

Cevaplar: 1) 360 × 12 = $4.320. 2) 1.000 × 12 = $12.000. 3) Karar sizin; 3. bölüm yardımcı olacak.

Özet

- Ödeme bir kez değil, her ay yapılır.
- Yedekli beş ağ, arşivler ve insanlar hariç, yalnızca kira olarak yılda $60.000'ı aşar.
- Diskler dolar, ağlar güncellenir, birinin nöbet tutması gerekir.

Sırada: 3. bölüm. Sağlayıcı neleri üstlenir ve kendi node'unuz ne zaman yine de gereklidir.

Rakamlar nereden

Tüm sayfalar 27 Kasım 2026'da açıldı.

- Bitcoin, gereksinimler: bitcoin.org/en/bitcoin-core/features/requirements
- Bitcoin, zincir boyutu: api.blockchain.info/charts/blocks-size
- Ethereum: ethereum.org/developers/docs/nodes-and-clients/run-a-node
- BNB Smart Chain: docs.bnbchain.org/bnb-smart-chain/developers/node_operators/node_best_practices; github.com/bnb-chain/bsc-snapshots
- TRON: tronprotocol.github.io/documentation-en/using_javatron/installing_javatron; developers.tron.network/docs/deploy-the-fullnode-or-supernode
- Solana: docs.anza.xyz/operations/requirements
- TON: docs.ton.org/nodes/cpp/run-liteserver; docs.ton.org/nodes/cpp/run-archive-liteserver
- Maaşlar: career.habr.com/professions/devops (2025–2026 verileri)

Sunucu fiyatları, yazarın birkaç barındırma firmasının açık fiyat listelerinden çıkardığı yuvarlanmış tahmindir.

Bölüm 3. Sağlayıcı ne sunar, kendi node'unuz ne zaman yine de gerekir

Artyom yol ayrımında

2. bölümdeki hesaplardan sonra hesap makinesi ekranı aşağı bakar halde duruyor. Yedekli beş ağ, yalnızca kira olarak yılda 60.000 doları aşıyor. Kahve o kadar kazandırmıyor.



İnceleyelim: sağlayıcı neleri üstlenir

Elektriği prizden alırsınız; çaydanlık için santral kurmazsınız. Node'larda da öyle: **sağlayıcı** onları kendisi tutar, size de soru sorabileceğiniz bir adres verir.



Genellikle onun tarafında beş iş vardır:

1. **Donanım ve diskler.** Satın almak, kiralamak, yer azalınca büyütmek.
2. **Eşitleme.** Node ağdan geri kalmamalı.
3. **Güncellemeler.** Ağ kural değiştirince program zamanında değiştirilir.
4. **Bozulanı değiştirmek.** İstekler sağlam bir node'a yönlendirilir.
5. **Nöbet.** Gece uyanan Artyom olmaz.

Sağlayıcılar farklıdır ve farklı şeyler vaat ederler. Bu listeden tam olarak neleri alacağınız ödemeden önce kontrol edilir. Nasıl seçileceğini 13. bölümde inceleyeceğiz.

Size ne kalır

- **Cüzdanın anahtarları.** Yalnızca sizde durur.
- **Uç nokta adresi.** Sağlayıcının verdiği adrese uç nokta (endpoint) denir. Çoğu zaman içinde kişisel erişim anahtarınız gömülüdür. Onu parola gibi koruyun: başkasının istekleri sizin limitinizi harcar.
- **Kodunuz.** Hataya dayanmalı, isteği tekrarlamalı ve aynı ödemeyi iki kez saymamalı.

Her planda geçen üç söz

Söz	Anlamı	Kendinize soru
Saniyede istek (RPS)	Ne sıklıkta soru sorulabileceği	Aynı anda kaç müşteri gelir?
Limit veya kredi	Bir dönemde toplam ne kadar sorulabileceği. Farklı soruların bedeli farklıdır	Ay sonuna kadar yeter mi?
Arşiv erişimi	Uzak geçmişin sorulup sorulamayacağı	Eski geçmişe ihtiyacım var mı?

Deneyelim: gerçek bir node'a soru

1. bölümde adres yerine `NODE_ADRESİ` yazıyordu. Şimdi gerçeğini koyalım. RpcNode'un ücretsiz herkese açık node'ları var; ne kayıt ister ne anahtar. Ethereum ağının son blok numarasını soralım:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_blockNumber","params":[]}'
```

Bu bölüm yazılırken node'un yanıtı şuydu:

```
{"jsonrpc":"2.0","id":1,"result":"0x1946393"}
```

`0x1946393`, 26.502.035 demektir. Sizde sayı daha büyük olacak: ağ yerinde durmuyor. İsteği bir dakika sonra tekrarlayın, kendiniz göreceksiniz.

Herkese açık node'lar herkesin ortak malıdır, bu yüzden sınırlıdır: 27 Kasım 2026 itibarıyla tek adresten saniyede 5 istek; en "pahalı" metotlar da kapalıdır. Tanışmak için yeter. Mağazaya ise kendi uç noktası gerekir.

Bu kaça gelir

Örnek olarak RpcNode'un 27 Kasım 2026 tarihli ilk iki planı. Güncel olanlara siteden bakın.

	Free	Starter
Aylık fiyat	$0	$19
Kredi	15 milyon	50 milyon
Saniyede istek	15	40
Uç nokta	3	5
Ağlar	Ethereum ve Solana	hepsi
Arşiv erişimi	yok	var

RpcNode'un node'ları üç coğrafyada bulunur: Frankfurt, ABD ve Japonya.

> **Artyom:** "Ayda 5.120'ye karşı 19 dolar mı? Bunun bir bit yeniği yok mu?"

İkisi farklı şeyler. Kendi sunucunuzun tamamı sizindir. Planda ise toplam güçten sınırlı bir pay alırsınız. Küçük bir mağazaya pay yeter; koca bir "santralin" parasını ödemeye gerek yoktur.

Kendi node'unuz ne zaman yine de gerekir

- **Gizlilik.** Sağlayıcı isteklerinizi görür: hangi adresleri kontrol ettiğinizi ve nereden sorduğunuzu.
- **Dev yük.** İstek o kadar çoktur ki plan, kendi sunucularınızdan pahalıya gelir.
- **Ağa katılım.** Validatörsünüz veya ağın kendisini geliştiriyorsunuz.
- **Özel ayarlar.** Sağlayıcının sunmadığı bir metot veya mod gerekiyor.
- **Yasanın veya müşterinin şartları.** Veriler yalnızca sizde işlenmeli.

Karma yol da mümkün: ana ağ için kendi node'unuz, diğerleri için ve yedek olarak sağlayıcı.

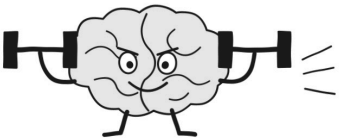
Tuzak: "sağlayıcı asla çökmez"

Her şey çöker: sunucular, iletişim hatları, sağlayıcıların hizmetleri de. Ödeme sizin için önemliyse, yanıt gelmediğinde sitenin ne yapacağına önceden karar verin.



- Sadece soramadıysanız müşteriye “ödenmedi” demeyin.
- İsteği sonra tekrarlayın; arka arkaya yüz kez değil.
- Yedek bir adres bulundurun: ikinci bir sağlayıcı veya kendi node’unuz.

Düşünün



1. Artyom’un sitesi, bekleyen 20 siparişin her biri için saniyede bir ödeme kontrolü yapıyor; her kontrol bir istek. Bu, saniyede 15 isteğe sığar mı?
2. Node on dakika yanıt vermezse müşteri ne görmeli?
3. Hangisine kendi node’u gerekir: kahve mağazasına mı, ağın kendisini geliştiren ekibe mi?

İlk soru için ipucu: 20, 15’ten büyük. Ya daha seyrek kontrol edin ya da daha akıllıca sorun: nasılı 26. bölümde.

Özet

- Sağlayıcı donanımı, eşitlemeyi, güncellemeleri ve nöbeti üstlenir.
- Anahtarlar, uç nokta adresi ve arıza anında sitenin davranışı size kalır.
- Kendi node’unuz gizlilik, büyük yük, ağa katılım veya özel şartlar için gerekir.

Birinci kısım bitti. Artyom bir sağlayıcı seçti ve gerçek bir node’dan yanıt aldı. Peki node’un söylediği 26.502.035 numaralı o “blok” ne? **Sırada:** 2. kısım, 4. bölüm. Aracısız para ve kayıt.

Bölüm 4. Aracısız para ve kayıt

Paranız nerede duruyor?

Artyom'un kartına bir havale geldi. Tek bir banknot bile yer değiştirmedi. Peki ne oldu? Biri müşterinin adının karşısındaki sayıyı azalttı, Artyom'un adının karşısındakini artırdı.

Kaydi para bir kayıttır. Kayıtları tutana da aracı denir: banka veya ödeme hizmeti.



İnceleyelim: aracı neden gerekli ve neden zahmetli

Aracı pek çok faydalı iş yapar. Paranın yetip yetmediğine bakar. Aynı paranın iki kez harcanmasını engeller. Anlaşmazlığı inceleyip ödemeyi geri çevirebilir.

Ama bütün kayıtlar onun elindedir. Transferi durdurabilir, komisyon alabilir, hesabı bloke edebilir veya hafta sonu kapalı olabilir. Müşteri başka bir ülkede yaşıyorsa bu daha da belli olur.

> **Artyom:** "O zaman müşteri bana parayı dosya olarak göndersin. Fotoğraf gibi. Aracı falan olmadan!"

Neden bir dosya göndermek yetmez

Bir fotoğrafı yüz kez gönderebilirsiniz, yine de sizde kalır. Parada bu olmaz: yoksa aynı coinle iki kişiye ödeme yapılır. Buna **çifte harcama** denir.



Uzun süre tek çözüm vardı: yetkili biri listeyi tutar ve hangi coinin artık kimde olduğunu söyler. Yani yine aracı.

Fikir: kayıtları herkes tutsun

2008'de Satoshi Nakamoto takma adlı bir yazar başka bir yol anlattı ve Ocak 2009'da Bitcoin ağı çalışmaya başladı. Fikir şu:

1. Transfer listesi tek kişide saklanmaz. **Kopyası pek çok kişide vardır.**
2. Katılımcılar her yeni transferi **ortak kurallara göre denetler**: para var mı, daha önce harcanmış mı.
3. Katılımcılar hangi kayıtların doğru olduğu konusunda, başlarında bir yetkili olmadan **anlaşır**.



Birlikte tutulan böyle ortak bir listeye **blokzincir** (blockchain) denir. Kayıtların nasıl zincire dizildiğini 5. bölümde, katılımcıların nasıl anlaştığını 8. bölümde inceleyeceğiz.

Deneyelim: ortak defter oyunu

Kod gerekmiyor. İki arkadaş bulun, herkesin kendi kâğıdı olsun. Başta herkesin kâğıdında şu yazılı: Anya'da 10 coin, Boris'te 5, Vera'da 0. Tek kural var: **elinde olandan fazlasını veremezsin**.

No	Kim transfer etmek istiyor	Kabul mü?	Anya	Boris	Vera
başlangıç			10	5	0
1	Anya → Vera'ya 4				
2	Vera → Boris'e 1				
3	Boris → Anya'ya 7				
4	Anya → Boris'e 6				

Boş kutuları doldurun, sonra kâğıtları karşılaştırın. Herkeste aynı şey yazıyorsa, az önce başınızda bir yetkili olmadan anlaştınız demektir. Cevap bölümün sonunda.

Blokzincir neyi verir, neyi vermez

Verir	Vermez
Her gün, her saat doğrudan transfer	"Geri al" düğmesi
Farklı ülkelerden erişilen tek bir hesap	Sabit coin fiyatı: kur çok oynayabilir
Herkesin denetleyebileceği kayıtlar	Saklılık: her adresin transferlerini herkes görür
Listeyi saklaması için kimseye güvenmek gerekmez	Kayıp anahtarı geri getirecek destek hattı

Her şey zıplamaz: bir dolar civarında kalmaya çalışan coinler var, bunlara **stablecoin** denir. Ama en yaygın olanlarının arkasında bir ihraççı şirket durur ve o şirket bir adresteki parayı dondurabilir. Ayrıntılar 9. bölümde.

Aracılar ortadan kalkmadı: borsalar, cüzdanlar, sağlayıcılar var. Ama hepsi aynı değil. Borsa coinlerinizi kendisinde tutabilir. İlk kısımdaki sağlayıcı ise para tutmaz: yalnızca sorularınızı node'a iletir.

Tuzak: "beğenmezsem iptal ederim"

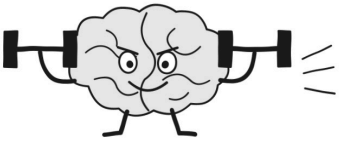
Onaylanmış bir transfer iptal edilemez. Adreste hata yaptıysanız para gitti. Onu ancak alıcı geri verebilir; o da isterse ve o adres gerçekten birine aitse.



Bu yüzden adres iki kez kontrol edilir; büyük bir tutarda da önce küçük bir parça göndererek denenir.

Küçük bir hikâye. Mayıs 2010'da iki pizza için 10.000 bitcoin ödendi. O transfer hâlâ ortak defterde yazılı.

Düşünün



1. Oyunun sonunda Vera'nın 3 coini var. Aynı anda Anya'ya "sana 3 veriyorum", Boris'e de "sana 3 veriyorum" yazıyor. Katılımcılar ne yapmalı?
2. Ya bir katılımcı birinci transferi, diğeri ikincisini kabul ettiyse? Kim haklı?
3. Artyom için hangisi daha rahat: ödemenin iptal edilebilmesi mi, edilememesi mi? Ya müşterisi için?

İpuçları: 1) yalnızca biri kabul edilebilir. 2) En zor soru bu; 8. bölüm tamamen ona ayrılmış.

Özet

- Kaydi para kayıtlardan ibarettir ve birinin onları tutması gerekir.
- Blokzincir, pek çok kişinin ortak kurallarla tuttuğu ortak bir listedir.
- Aracıdan kurtulmak sorumluluk da demektir: transferi iptal edecek ve anahtarı geri getirecek kimse yoktur.

Oyunun cevabı: 1. transfer kabul (6, 5, 4); 2. kabul (6, 6, 3); 3. ret, Boris'te yalnızca 6 var (6, 6, 3); 4. kabul (0, 12, 3). Toplam coin yine 15.

Sırada: 5. bölüm. Node'un söylediği "blok" neydi ve eski bir kayıt neden sessizce değiştirilemez.

Bölüm 5. Bloklar ve zincir: eskisi neden sessizce değiştirilemez

O meşhur numara

3. bölümde node Artyom'a şöyle demişti: son blok 26.502.035 numaralı. Blok nedir? Neden numarası var? Birinin defterdeki bir kaydı gizlice düzeltmesine ne engel olur?

İnceleyelim: blok, defterin bir sayfasıdır

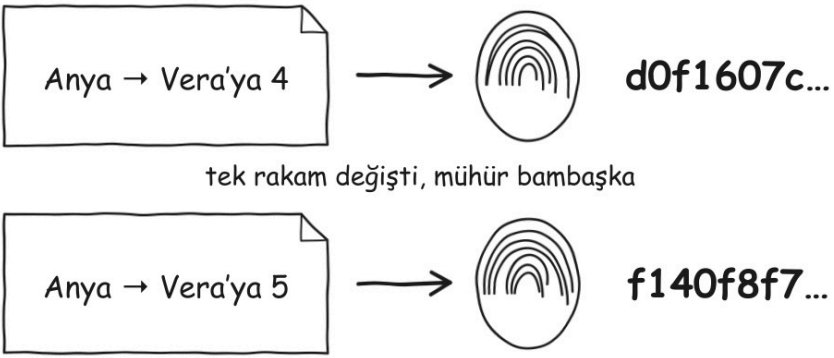
Transferler tek tek yazılmaz. Satırların bir sayfada toplanması gibi demetler halinde toplanır. Böyle bir sayfaya **blok** denir. Üzerinde şunlar vardır:

1. **Transferler** ve başka eylemler; bunlara işlem denir.
2. Sayfanın sıra **numarası**.
3. Oluşturulma **zamanı**.
4. **Önceki sayfanın mührü.** Bütün sır burada.

Mühür, yani hash

Her metin için rakam ve harflerden oluşan kısa bir dizi hesaplanabilir. Bu **hash**'tir; biz ona mühür diyeceğiz. Parmak izine benzer:

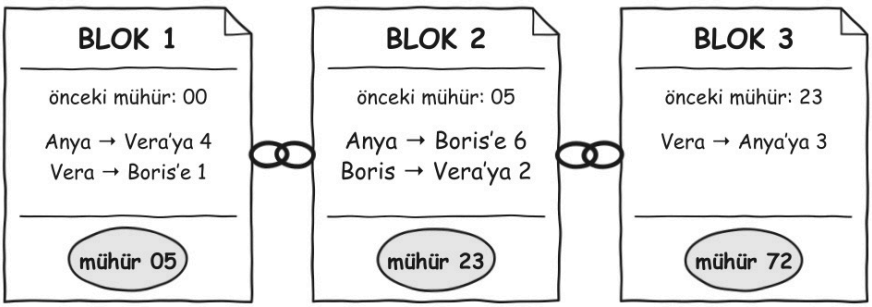
- aynı metnin mührü hep aynıdır;
- tek bir harf bile değişirse mühür bambaşka olur;
- mühürden metin geri elde edilemez;
- aynı mührü veren başka bir metin bulmak pratikte imkânsızdır.



Bunlar resimdeki iki satırın gerçek SHA-256 hash'lerinin baş kısmı. SHA-256'yı Bitcoin kullanır; başka ağlar mührü başka yöntemlerle hesaplar.

Zincir

Her sayfada bir öncekinin mührü yazılıdır. Sayfa kendi mührünü, üzerindeki her şeyden alır; **o önceki mühür de buna dahildir**.

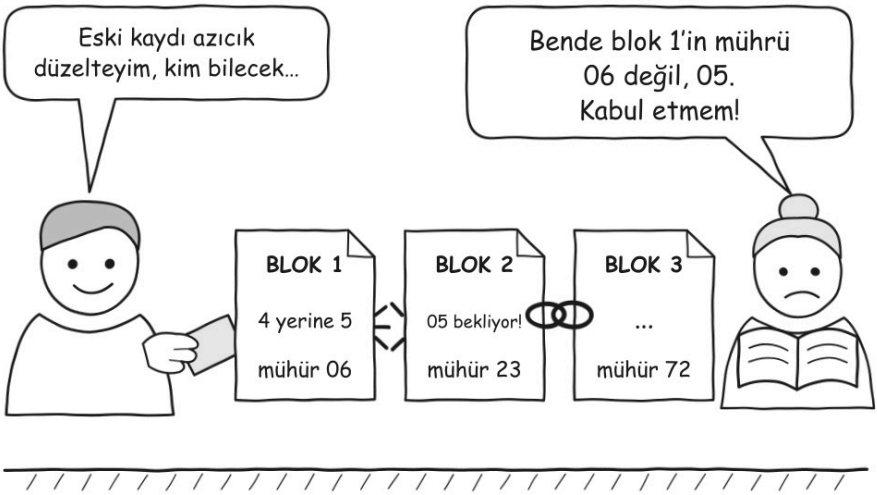


Böylece sayfalar bir **blok zinciri** halinde birbirine kenetlenir. İngilizcesi blockchain. Adın bütün hikâyesi bu.

Resimdeki mühürler iki rakamlı oyuncak mühürler: böylece akıldan hesaplanabiliyorlar.

Eskiyi düzeltmeye kalkarsanız

Dolandırıcı birinci bloktaki 4'ü 5 yapıyor. Bloğun mührü değişiyor. Oysa ikinci blokta eskisi yazılı. Zincir kopuyor ve bu hemen görülüyor.



Sonraki bütün mühürleri de yeniden yazabilir. Ama diğer katılımcıların kendi kopyaları vardır ve onun sürümü bunlarla uyuşmaz. Sürümünün kabul edilmesi için ağın geri kalanını yenmesi gerekirdi. Bu çok pahalıdır. Ağın hangi zincirin doğru olduğuna tam olarak nasıl karar verdiğini 8. bölümde inceleyeceğiz.

Gerçek bir zincire bakın

Bunlar Ethereum ağının 26 Kasım 2026 tarihli üç bloğu. Mühürler uzun olduğu için yalnızca başı ve sonu gösterildi.

Blok numarası	Mührü	Öncekinin mührü	İşlem sayısı
26.502.033	`0xbe58…d764`	`0x2844…c876`	229
26.502.034	`0x7bb7…b7a4`	`0xbe58…d764`	229
26.502.035	`0xeab2…c904`	`0x7bb7…b7a4`	515

Parmağınızla izleyin: bir satırdaki mühür bir sonrakinde tekrar ediyor. Bu blokların arasında tam 12'şer saniye geçmiş. Bitcoin'de yeni blok daha seyrek çıkar, ortalama yaklaşık 10 dakikada bir.

İlk bloğun öncesi yoktur. Bitcoin'in ilk bloğu 3 Ocak 2009'da oluşturuldu.

Deneyelim: kendi mührünüz

Oyuncak bir kural uyduralım:

Mühür = (önceki mühür × 3 + sayfadaki tüm tutarlar), sonucun son iki rakamını alıyoruz.

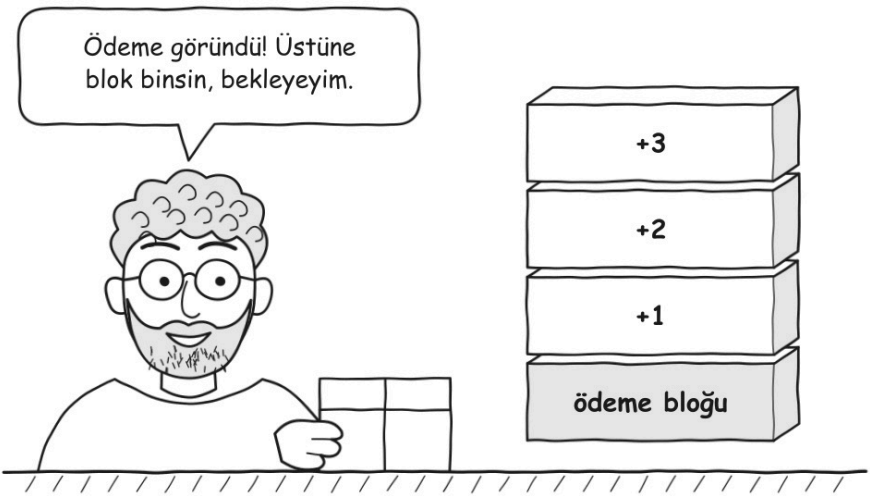
Blok	Transfer tutarları	Mühür	1. blokta 4 yerine 5 yazılırsa
1	4 ve 1		
2	6 ve 2		
3	3		

İlk blok için önceki mühür 0'dır. İki sütunu da doldurun. Kaç mühür değişti?

Tuzak: "blokta gördüm, artık sonsuza kadar orada"

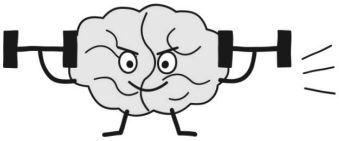
En yeni bloklar bazen başkalarıyla değiştirilir: ağın farklı kesimleri henüz anlaşamamıştır. Üzerine ne kadar çok blok binerse kaydınızı kaldırmak o kadar zorlaşır.

Buna **onay** sayısı denir: ödemenin bulunduğu blok birinci onaydır, sonraki her blok bir tane daha ekler.



Ne kadar bekleneceği ağa ve tutara bağlıdır. Bir fincan kahve için daha az, büyük bir parti için daha uzun. Ayrıntılar 8. bölümde.

Düşünün



1. Mührümüzü kandırın: 1. bloktaki 4 ve 1 tutarlarını, mühür değişmeyecek şekilde başkalarıyla değiştirin. Bu, kuralımız hakkında ne söylüyor?
2. Üç yıl önceki bir bloğu değiştirmek neden dünkü bloğu değiştirmekten daha zor?
3. 12 saniyede bir blok çıkıyorsa bir saatte kaç blok olur?

Cevaplar: 1) örneğin 3 ve 2: toplam aynı. Gerçek bir hash bunu fark ederdi. 2) Üzerine binen her şeyi yeniden yapmak gerekirdi. 3) 300.

Özet

- Blok; işlemleri, numarası, zamanı ve önceki sayfanın mührü olan bir sayfadır.
- Eski bir kaydı değiştirmek mührü değiştirir ve zinciri koparır.
- Yeni bloğa daha az güvenilir: onay beklenir.

Oyunun cevabı: mühürler 05, 23, 72. Değişiklikten sonra 06, 26, 81: üçü de değişti.

Sırada: 6. bölüm. Ağ, transferi gerçekten paranın sahibinin gönderdiğini nereden bilir? Anahtarlar, adresler ve imza.

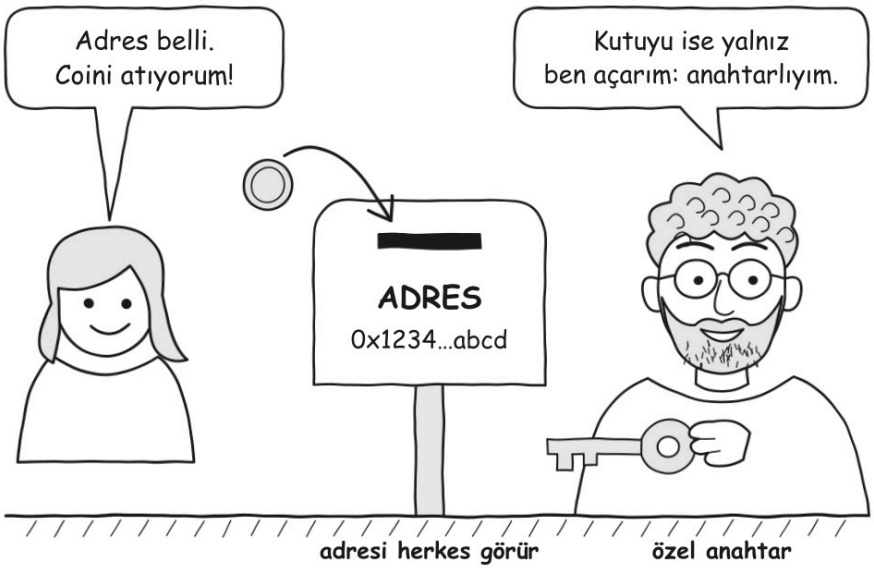
Bölüm 6. Anahtarlar, adresler ve imza

Ağ bunun sizin paranız olduğunu nereden bilir?

Ortak defterde kimlik kartı yoktur. Kimse adınızı sormaz, fotoğrafınıza bakmaz. O halde herhangi birinin "Artyom her şeyini bana veriyor" yazmasına ne engel olur?

İnceleyelim: posta kutusu ve anahtar

Sokakta bir posta kutusu düşünün. Kutunun numarasını herkes bilir ve isteyen yarıktan içeri bir coin atabilir. Ama kutuyu açıp onu alabilen yalnızca anahtarın sahibidir.



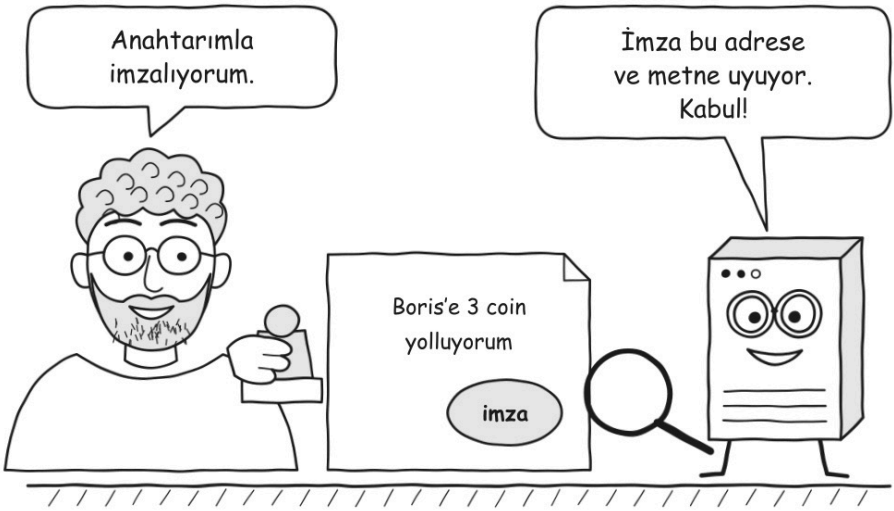
Ne	Neye benzer	Kime gösterilir
Adres	Kutunun numarası	Para almak için herkese
Açık anahtar	İmzanın karşılaştırıldığı örnek	Herkese olur
Özel anahtar	Kutunun anahtarı	**Hiç kimseye**

Aralarındaki bağ tek yönlüdür. Özel anahtardan açık anahtar, açık anahtardan adres elde edilir. Geriye doğru hesaplamak pratikte imkânsızdır. Bazı ağlarda adres, özel bir biçimde yazılmış açık anahtarın ta kendisidir.

Cüzdan, anahtarların saklandığı program veya cihazdır. İçinde coin yoktur: coinler ortak defterde kayıt olarak kalır.

İmza

Sahibi coin vermek için bir transfer hazırlar ve onu özel anahtarıyla **imzalar**. Bu sırada anahtar hiçbir yere gönderilmez. Ağa transfer ile imza gider.



Node iki şeyi denetler: imza tam da bu adresin anahtarıyla mı atılmış ve metin imzadan sonra değiştirilmiş mi. Bir transferin altındaki imza başka bir transfere uymaz. Aynı transferi de ağ ikinci kez işlemez.

Deneyelim: oyuncak imza

Her şey hesap makinesiyle yapılabilir. Özel anahtar **7**, açık anahtar **3** sayısı. Gizli olan yalnızca ilki.

> Kural: seçilen sayıyla çarpın ve her adımdan sonra 33'e bölümünden kalanı tutun.

"4" mesajını imzalıyoruz. 4'ü özel anahtarın söylediği kadar, yani 7 kez alıyoruz:

Kaç tane dört	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Kalan	4	16	31	25	1	4	**16**

Her adım, önceki kalanın 4 ile çarpımıdır. Örneğin üçüncüsü: 16 × 4 = 64, 33'e bölümünden kalan 31. Yedinci adımdan sonra imzayı elde ederiz: **16**.

Doğruluyoruz. 16 imzasını açık anahtarın söylediği kadar, yani 3 kez alıyoruz: 16, sonra 16 × 16 = 256, kalan 25, sonra 25 × 16 = 400, kalan **4**. Mesajımız çıktı. İmza gerçek!

Şimdi biri imzayı 16 bırakıp mesajı "5" diye değiştirmiş olsun. Doğrulama yine 4 verir, 5 değil. Sahtecilik ortada.

Bu yalnızca işin fikri. Böyle küçük sayıları bir lise öğrencisi kırar. Gerçek ağlarda anahtarlar devasa sayılardır, matematik de farklıdır.

Gizli ifade

Özel anahtarı bir yere yazmak zahmetlidir: uzun bir dizidir. Bu yüzden pek çok cüzdan 12 veya 24 sözcükten oluşan bir **gizli ifade** (kurtarma ifadesi) verir. Anahtarlar ve bütün adresler ondan geri oluşturulur.



- İfadeyi bilen, paranın sahibidir.
- Hem ifadeyi hem cihazı kaybederseniz geri getirecek kimse yoktur.
- Fotoğrafını çekmeyin, notlarda ve yazışmalarda saklamayın.
- Sitelere yazmayın, "destek ekibi"ne okumayın. Gerçek destek onu sormaz.

Parola değiştirilebilir, özel anahtar değiştirilemez. Sızdıysa tek yol, parayı hemen yeni bir adrese taşımaktır.

Mağaza için bunun anlamı

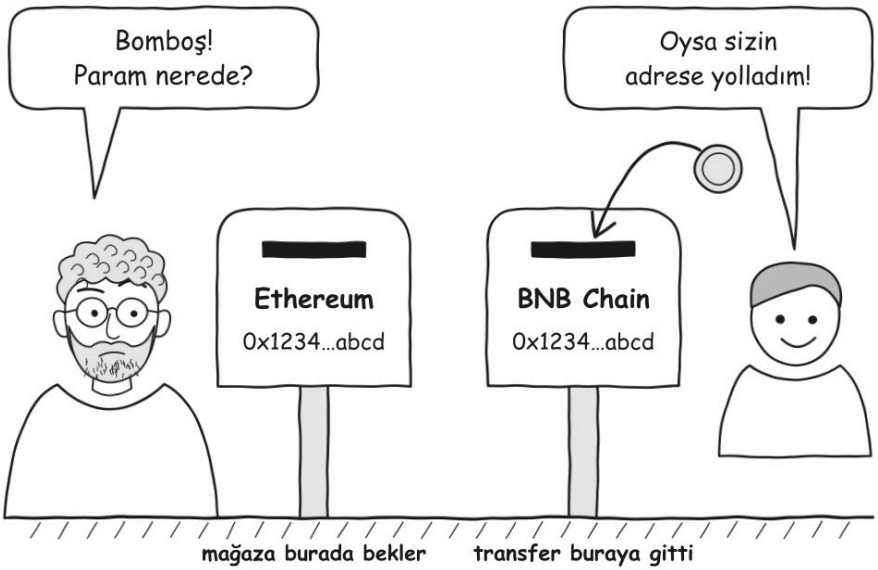
Para **almak** ve ödemeyi görmek için Artyom'un sitesine yalnızca adres gerekir. Mağaza sunucusunda özel anahtara ihtiyaç yoktur ve orada tutmamak gerekir: site ele geçirilirse para da gider. Her siparişe ayrı adres vermek kullanışlıdır; bunu 28. bölümde anlatacağız.

Tuzak: "adres aynı, demek ki ulaşır"

Farklı ağların adresleri farklı görünür:

Ağ	Adres şununla başlar
Bitcoin	`1`, `3` veya `bc1`
Ethereum ve benzer ağlar	`0x`
TRON	`T`
TON	genellikle `EQ` veya `UQ`
Solana	özel bir başlangıcı yok

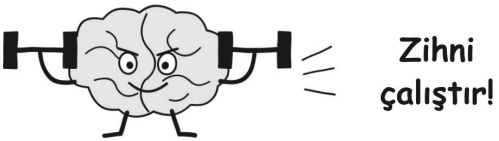
Ama Ethereum, BNB Smart Chain, Polygon ve daha pek çok ağda adres **aynı görünür**. Defterleri ise farklıdır. Müşteri yanlış ağdan gönderirse mağaza ödemeyi görmez.



Bu yüzden faturaya yalnızca adres ve tutar değil, **ağ** da yazılır.

Dolandırıcıların bir numarası daha var: tanıdığınız bir adrese benzeyen bir adresten minicik bir transfer gönderirler. Onu geçmişten kopyalayacağınızı umarlar. Adresin ilk ve son karakterlerini değil, tamamını karşılaştırın.

Düşünün



1. "2" mesajını 7 anahtarıyla imzalayın ve imzayı 3 anahtarıyla doğrulayın.
2. Artyom mağazanın adresini kartvizitine bastırdı. Bu tehlikeli mi?
3. "Cüzdan destek ekibi", takılan bir transferi geri getirmek için gizli ifadeyi istiyor. Ne yapmalı?

Cevaplar: 1) imza 29, doğrulama yine 2 verir. 2) Parayı bu yolla çalamazlar ama bu adrese gelenleri herkes görür. 3) Hiçbir şey söylemeyin: bunlar dolandırıcı.

Özet

- Adres herkese gösterilir; özel anahtar ve gizli ifade hiç kimseye.
- İmza, transfere anahtarın sahibinin izin verdiğini ve transferin değiştirilmediğini kanıtlar.
- Ağ da adres kadar önemlidir.

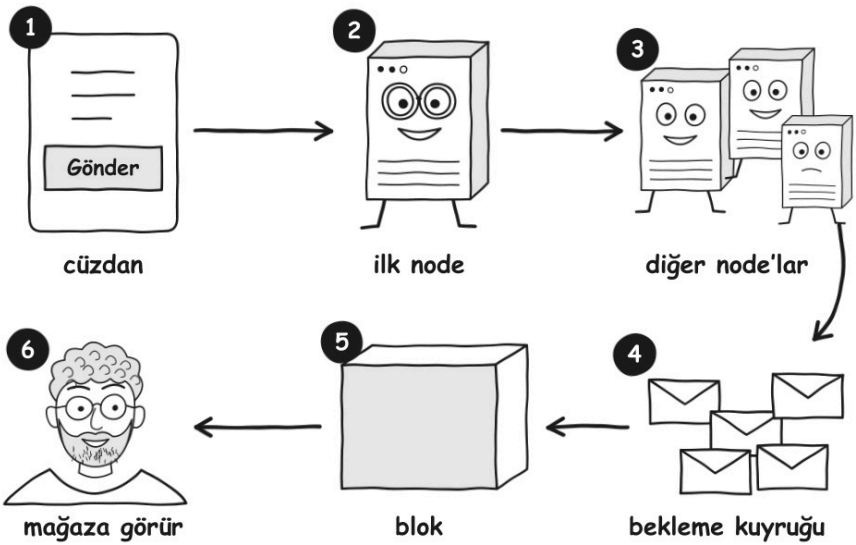
Sırada: 7. bölüm. Transfer imzalandı. "Gönder" düğmesinden bloktaki kayda kadar başına neler gelir?

Bölüm 7. İşlemin yolculuğu: “Gönder” düğmesinden bloğa

“Gönderdim!” — “Bende bir şey yok”

Müşteri düğmeye basıyor ve yazıyor: “Gönderdim”. Artyom bakıyor, para yok. Biri yalan mı söylüyor? Büyük olasılıkla hayır. Transfer henüz yolda.

İnceleyelim: altı durak



1. **Cüzdan** transferi hazırlar ve anahtarla imzalar.
2. **İlk node** transferi alır ve denetler: imza doğru mu, para yetiyor mu.
3. Onu **diğer node’lara** iletir, onlar da daha ileriye.
4. Transfer **kuyrukta** bekler. İngilizcede bu kuyruğa mempool denir.
5. Sayfayı toplama sırası kimdeyse, transferi **bloğa** koyar.
6. **Mağaza** onu görür ve onayları bekler.

Ağdan ağa adlar ve ayrıntılar değişir: örneğin bazılarında ortak kuyruk yoktur, transfer doğrudan bloğu toplayana gider.

Her transferin kendi “mührü” vardır: **işlem hash’i**. Kargo takip numarası gibidir: onunla transferin şu an nerede olduğunu öğrenebilirsiniz.

Ağ ücreti: otobüsteki koltuklar

Bloğa her şey sığmaz. Üç koltuklu bir otobüs düşünün; durakta ise beş kişi var.



Bloğa girebilmek için gönderen, ağa bir **ücret** öder. İsteyen ne kadar çoksa koltuk o kadar pahalıdır. Az teklif eden sonraki otobüsü bekler, bazen de hiç binemez.

- Ücreti gönderen öder; mağazaya bir şey kalmaz.
- Ücret transferin tutarına değil, "ağırlığına" ve ağın yoğunluğuna bağlıdır.
- Bu "ağırlığın" nasıl hesaplandığı her ağda farklıdır: kitabın dördüncü kısmı bunu anlatır.

Gerçek bir transfere bakın

Bu, Ethereum ağında 26.502.085 numaralı bloktaki sıradan bir transfer. Node'a işlem hash'iyle soruldu:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,
       "method":"eth_getTransactionReceipt",
       "params":["TRANSFER_HASH_DEĞERİ"]}'
```

Yanıt uzun. İşte önemli kısımları, sıradan sayılara çevrilmiş haliyle:

Alan	Değer	Anlamı
`status`	`0x1`	Başarıyla yürütüldü
`blockNumber`	26.502.085	Hangi blokta olduğu
`gasUsed`	21.000	Basit bir transferin "ağırlığı"
ücret	0,000043 ETH	"Ağırlık" × birim fiyatı
transfer tutarı	0,000041 ETH	Transferin kendisinden alınır

Fark ettiniz mi? Bu kez göndermek için, **gönderilenden daha fazlası** ödendi. Ücretin, içinde ne kadar para olduğu umurunda değildir.

Bu rakam ise sürekli değişir. Kendiniz sorun:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_gasPrice","params":[]}'
```

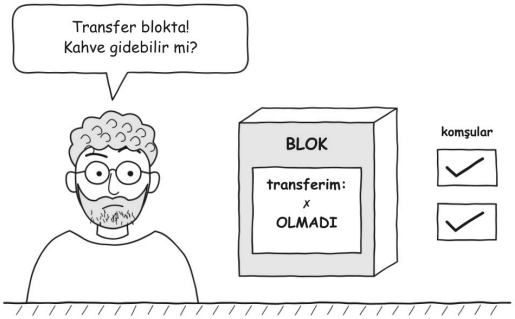
Node, bir birim "ağırlığın" wei cinsinden güncel fiyatını söyler. RpcNode'un herkese açık Bitcoin node'una göre aynı gün kuyrukta neredeyse 83.000 transfer bekliyordu.

Transferin başına neler gelebilir

Durum	Anlamı	Mağaza ne yapmalı
Bekliyor	Hâlâ kuyrukta	Beklemeli, malı göndermemeli
Blokta	Yazıldı ama onayı az	Başarılı olduğunu kontrol edip beklemeli
Onaylandı	Üzerine yeterince blok bindi	Siparişi göndermeli
Yürütülemedi	Blokta ama transfer gerçekleşmedi	Ödeme saymamalı
Kuyruktan düştü veya değiştirildi	Bloğa giremedi	Müşteriden cüzdanını kontrol etmesini istemeli

Tuzak: "bloğa girdiyse ödenmiştir"

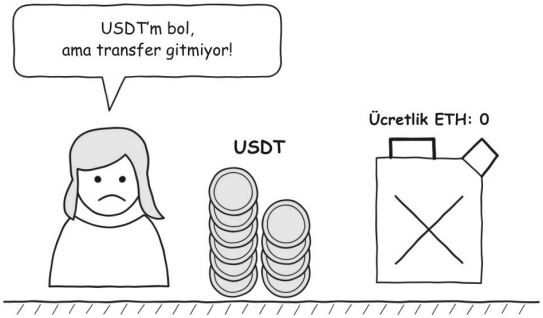
Ethereum ve benzer ağlarda bir transfer bloğa girip yine de gerçekleşmeyebilir. Örneğin müşterinin tokeni yetmemiştir. Ücret yine kesilir ama para size gelmez.



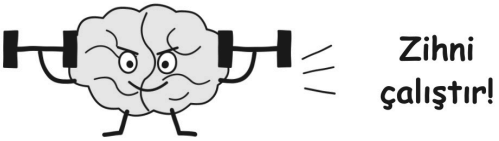
Bu yüzden yalnızca bloğa değil, sonuca da bakılır: yukarıdaki örnekte bu `status` alanıdır. Ayrıca alıcı, tutar, coin ve ağ da karşılaştırılır.

İkinci tuzak: boş depo

Ücret, ağın ana coiniyle alınır. USDT gönderseniz bile Ethereum'da bu ETH'dir. TRON'da ödeme TRX ve ağın özel kaynaklarıyla yapılır.



Düşünün



1. Bloğa üç transfer sığıyor. Kuyruktakiler koltuk için 4, 9, 1, 7 ve 2 teklif etti. İlk blokla kimler gider?
2. Müşteri işlem hash'ini ve "başarılı" yazan bir ekran görüntüsü gönderdi. Kahveyi yollamak için bu yeterli mi?
3. Müşterinin Ethereum ağında 100 USDT'si var ama hiç ETH'si yok. Ne yapmalı?

Cevaplar: 1) 9, 7 ve 4. 2) Hayır: görüntü sahte olabilir, hash'i kendi tarafınızda kontrol edin. 3) Önce ücret için biraz ETH edinmeli.

Özet

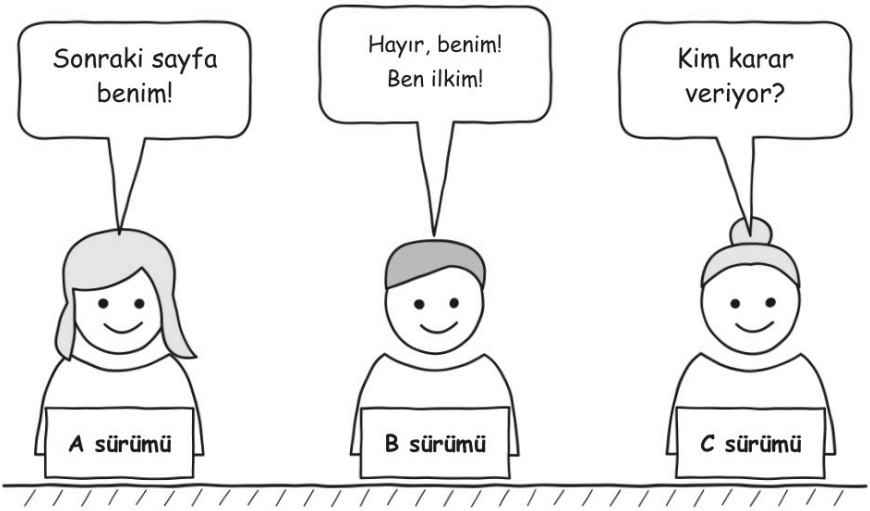
- "Gönderdim" ile "aldım" arasında bir yol var: node, kuyruk, blok, onaylar.
- Ücret bloktaki yeri satın alır ve tutara bağlı değildir.
- Ödeme; doğru coinin doğru ağda, yeterli onayla yapılmış başarılı transferidir.

Sırada: 8. bölüm. Blokları kim toplar ve herkes doğru olanın tam da bu zincir olduğunda nasıl anlaşır?

Bölüm 8. Ağ nasıl anlaşır

Sonraki sayfayı kim yazar?

Ortak defterin başında bir yetkili yoktur. Binlerce katılımcıda kopyası vardır ve her biri kendi sonraki sayfasını önerebilir. Hangisi gerçek sayılacak?



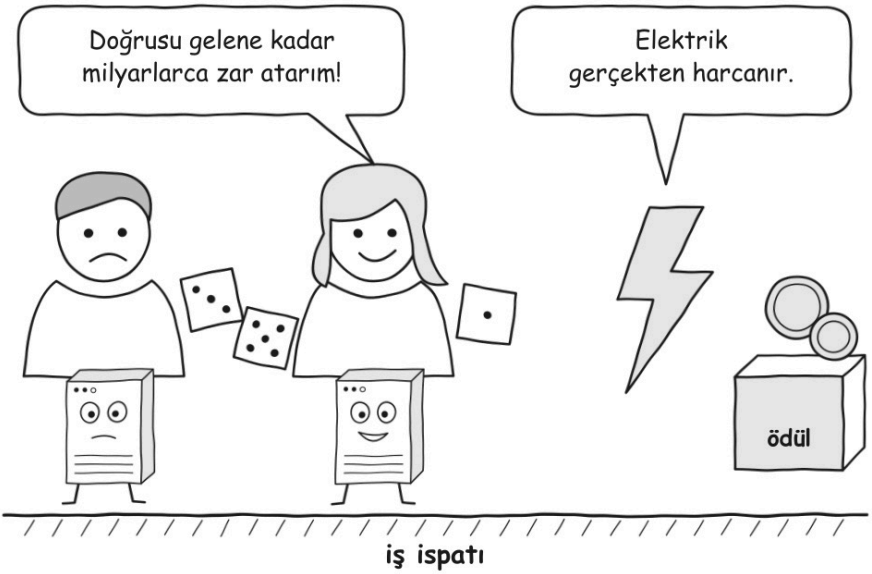
İnceleyelim: mutabakat

Katılımcıların ortak bir yanıta varmasını sağlayan kurallara **mutabakat** (konsensüs) denir. Amaçları, hile yapmayı dürüst oynamaktan daha pahalı kılmaktır.

Basitçe oy saymak olmaz: dolandırıcı bir milyon program çalıştırır ve bir milyon oy alır. Bu yüzden oyun bir bedeli olmalıdır. Başlıca iki yöntem vardır.

1. yöntem: iş ispatı

İngilizcesi Proof of Work; sürecin kendisine **madencilik** (mining) denir. Katılımcılar, akılla değil yalnızca deneme yanılmayla çözülebilen bir problemi yarışarak çözer. Nadir bir kombinasyon gelene kadar zar atmak gibi.

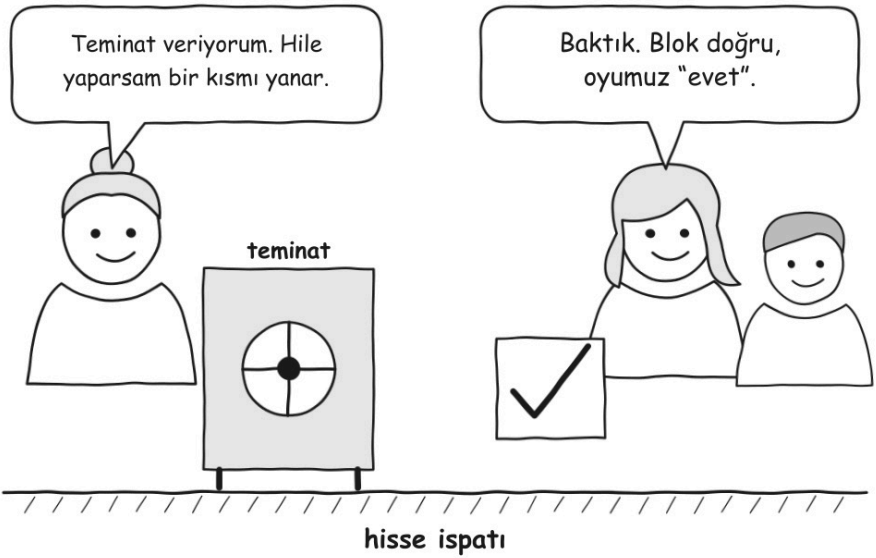


Kazanan bloğu ekler ve ödül alır: yeni coinler ve işlem ücretleri. Cevabını denetlemek kolay, bulmak ise pahalıdır. Bitcoin böyle çalışır:

- blok ortalama 10 dakikada bir çıkar;
- bu tempo korunsun diye problemin zorluğu 2016 blokta bir ayarlanır;
- blok başına verilen yeni coin şu an 3,125 BTC'dir ve ödül yaklaşık dört yılda bir yarıya iner.

2. yöntem: hisse ispatı

İngilizcesi Proof of Stake, yani **staking**. Katılımcı elektrik harcamak yerine coinlerini teminat olarak yatırır. Ona **validatör** (doğrulayıcı) denir. Validatörler sırayla blok önerir, diğerleri denetleyip oy verir.

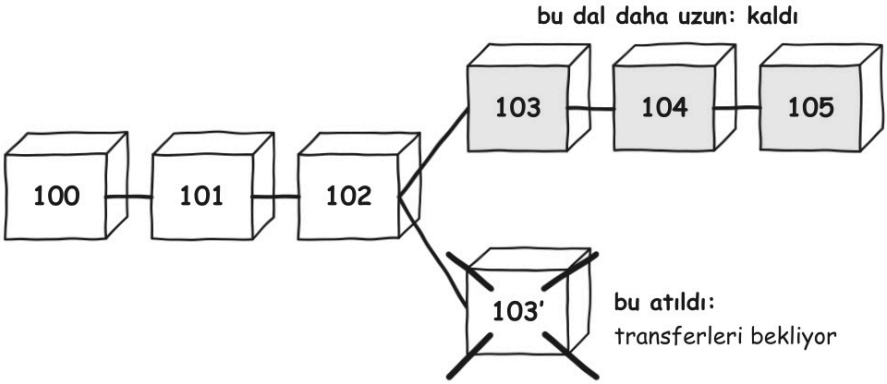


Ciddi bir ihlalde teminatın bir kısmı yanar; buna slashing denir. Ethereum bu yönteme 15 Eylül 2022'de geçti ve ağın elektrik tüketimi yaklaşık %99,95 azaldı. Kendiniz validatör olmak için 32 ETH gerekir.

Başka biçimler de var. TRON'da blokları, coin sahiplerinin seçtiği 27 temsilci yazar. BNB Smart Chain'de de sınırlı sayıda validatör çalışır.

İki sayfa birden çıkarsa

Ağ büyüktür, haberler anında yayılmaz. Bazen aynı numaralı iki blok ortaya çıkar ve zincir bir süreliğine çatallanır. Sonra bir dal kazanır, diğeri atılır. Buna **yeniden düzenlenme** (reorg) denir.



Atılan bloktaki transfer iz bırakmadan kaybolmaz: genellikle kuyruğa döner ve başka bir bloğa girer. Ama girmeyebilir de. 5. bölümdeki onaylar işte bunun için gerekli.

Resimde uzun dal kazanıyor. Daha doğrusu, Bitcoin'de üzerine daha çok iş harcanmış dal, staking kullanan ağlarda ise daha çok teminat oyu alan dal seçilir.

Kesinlik: artık geri alınamayacağı an

Kesinlik (finality), bloğun artık değiştirilmeyeceği andır. Bazı ağlarda açıkça ilan edilir, bazılarında ise güven her blokla birlikte artar.

Ağ	Kayıt yaklaşık ne zaman kesin sayılır
Bitcoin	Açık bir işaret yok. Büyük tutarlar için 6 onay önerilir, bu yaklaşık bir saattir
Ethereum	Genellikle 13–15 dakika, bazen daha uzun
TRON	Genellikle yaklaşık bir dakika
BNB Smart Chain	Yaklaşık bir saniye veya daha hızlı
Solana	2026 güncellemesinden sonra saniyenin kesirlerine göre tasarlandı
TON	Yaklaşık bir saniye

Ağların belgelerine göre 27 Kasım 2026 verileri. Bu sayılar güncellemelerle değişir.

Deneyelim: üç ayrı "tazelik" sorun

Ethereum node'larından yalnızca en yeni blok istenmez. 1. bölümdeki `latest` sözcüğünün iki akrabası vardır: `safe` (artık güvenilir) ve `finalized` (kesinleşmiş).

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,
       "method":"eth_getBlockByNumber",
       "params":["finalized",false]}'
```

`finalized` yerine `safe` ve `latest` yazıp numaraları karşılaştırın. Bu bölüm yazılırken node şöyle yanıt verdi:

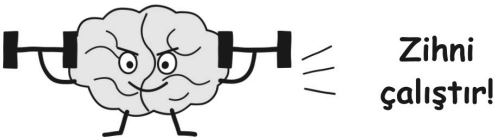
Ne soruldu	Blok numarası	En yeniye göre gerilik
`latest`	26.502.104	—
`safe`	26.502.070	34 blok, yaklaşık 7 dakika
`finalized`	26.502.038	66 blok, yaklaşık 13 dakika

Tuzak: "bütün ağlarda aynıdır"

"Bir onay beklerim, olur biter" tehlikeli bir kuraldır. Bir onay, farklı ağlarda bambaşka güvenilirlik demektir. Ağa ve tutara göre karar verin:

- sipariş ne kadar pahalıysa o kadar uzun bekleyin;
- büyük tutarlarda, kesinlik varsa onu bekleyin;
- geri alınamayacak bir şeyi yeni bir bloktaki transfere güvenerek teslim etmeyin.

Düşünün



1. Defterde neden "bir program, bir oy" ilkesiyle oylama yapılamaz?
2. Bitcoin blokları 10 dakikada bir geliyor. 6 onay ortalama ne kadar sürer? Ya 3 onay?
3. Bir müşteri bir fincan kahveyi alıp çıkıyor, diğeri 200 kg çekirdek alıyor. Artyom hangisinde kesinliği beklemeli?

Cevaplar: 1) programların bir bedeli yoktur, istendiği kadar çalıştırılabilir. 2) Yaklaşık bir saat ve yaklaşık yarım saat. 3) 200 kg alanda.

Özet

- Mutabakat, katılımcıların bir yetkili olmadan tek bir zinciri seçmesini sağlayan kurallardır.
- Oyun bedeli ya iştir (madencilik) ya da teminat (staking).
- Yeni bloklar değişebilir; kesinlik daha sonra ve her ağda kendine özgü biçimde gelir.

Sırada: 9. bölüm. Coinler, tokenler ve akıllı sözleşmeler: USDT neden ETH ile aynı şey değil.

Bölüm 9. Coinler, tokenler ve akıllı sözleşmeler

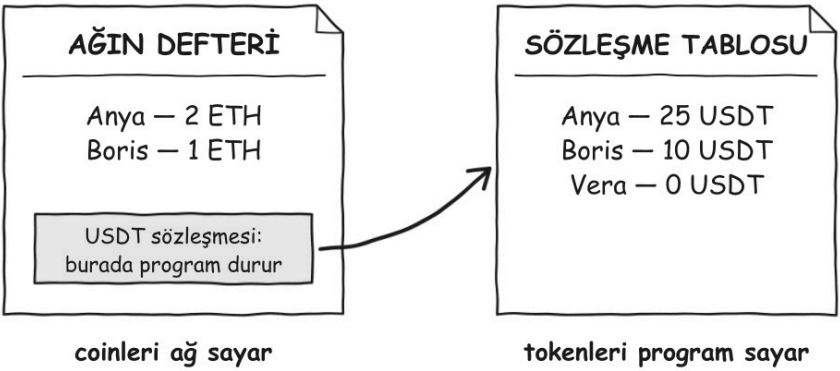
“Ödedim ama sizde görünmüyor”

Müşteri Ethereum ağında 25 USDT gönderdi. Artyom 1. bölümdeki gibi node’a soruyor ama bakiye değişmemiş. Para kayboldu mu? Hayır. Sadece başka türlü sormak gerekiyordu.

İnceleyelim: coin ve token

Ana coin, ağın bizzat kendisinin tuttuğu coindir: Ethereum’da ETH, TRON’da TRX, BNB Smart Chain’de BNB. İşlem ücreti onunla ödenir.

Token ise ağın içinde yaşayan bir programın tuttuğu kayıtlardır. USDT bir tokendir.



	Ana coin	Token
Kayıtları kim tutar	ağın kendisi	akıllı sözleşme
Ethereum’da örnek	ETH	USDT, USDC
Ücret neyle ödenir	kendisiyle	ağın ana coiniyle
Bakiye nasıl sorulur	`eth_getBalance`	sözleşmeye `eth_call`

Farklı ağlarda tokenler farklı kurulmuştur. Burada Ethereum’a ve ona benzeyen ağlara bakıyoruz.

Akıllı sözleşme: satıcısız otomat

Akıllı sözleşme, blokzincire yazılmış bir programdır. Yiyecek otomatına benzer: parayı atar, düğmeye basarsınız, önceden belirlenmiş kurallara göre sonucu alırsınız. Her node onu aynı şekilde çalıştırır.



- İmzalı bir anlaşma değil, düpedüz bir programdır.
- Genellikle değiştirilemez ama bazı yazarlar kendilerine bu imkânı bırakır.
- Programın özel yetkilere sahip bir "sahibi" olabilir.
- Programdaki bir hata paraya mal olabilir ve onu geri alacak kimse yoktur.

Deneyelim: token bakiyesi

`eth_call` metodu "hiçbir şeyi değiştirmeden sözleşmeye sor" demektir. Ağ böyle bir soru için ücret almaz: deftere bir şey yazılmaz. USDT sözleşmesine, kendi adresinde ne kadar USDT durduğunu soralım:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_call","params":[
  {"to":"0xdac17f958d2ee523a2206206994597c13d831ec7",
  "data":"0x70a08231000000000000000000000000dac17f958d2ee523a2206206994597c13d831ec7"},
  "latest"]}'
```

`to` hangi sözleşmeye sorduğumuzdur. `data` ise üç parçanın birleşimidir:

Parça	Anlamı
`0x70a08231`	hangi soru: "bakiye ne kadar?" (`balanceOf`)
24 sıfır	gereken uzunluğa tamamlama
40 karakter	kimin bakiyesi: `0x` olmadan adres

Bu bölüm yazılırken node şöyle yanıt verdi:

```
{"jsonrpc":"2.0","id":1,"result":"0x0000000000000000000000000000000000000000000000000000000
014d0a9aa6c5"}
```

Sıfırlardan sonra `14d0a9aa6c5` geliyor; bu 1.430.402.016.965 demektir. Ethereum'daki USDT'nin virgülden sonra altı basamağı vardır; demek ki sözleşmenin adresinde **1.430.402,016965 USDT** duruyor.

Nereden gelmişler? Anlaşılan çoğu "yanlış yere" yapılmış transferler: biri alıcının adresi yerine sözleşmenin adresini yapıştırmış. Gönderenin bunları geri almasının olağan bir yolu yoktur.

Virgül nereye konur

Sözleşme yalnızca tam sayı saklar. Kaç basamağın ayrılacağını ayrı bir özellik söyler: `decimals`.

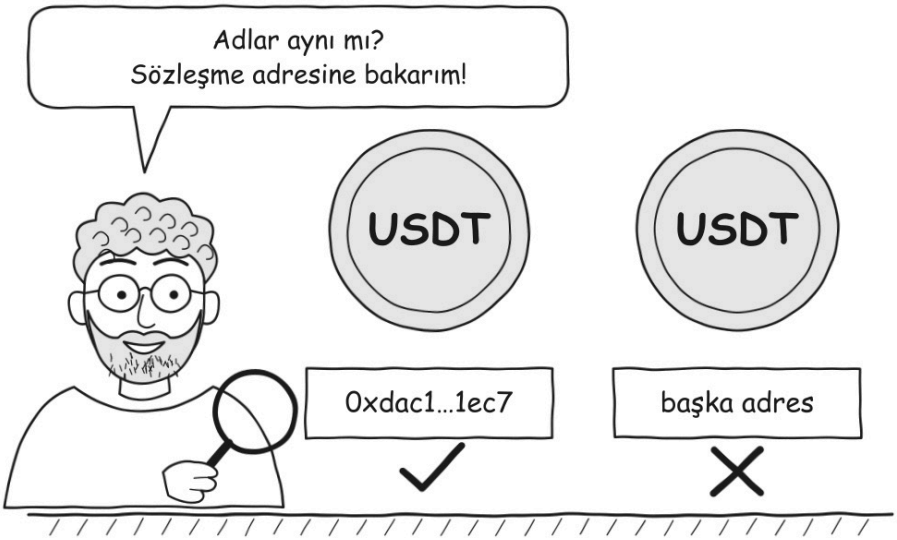
Ne	Virgülden sonraki basamak sayısı
ETH, ana coin	18
Ethereum'daki USDT	6
BNB Smart Chain'deki "Tether USD" (USDT) tokeni	18

Ad aynı ama basamak sayısı üç kat fazla. Karıştırırsanız bir trilyon kat yanılırsınız. Üstelik BNB Smart Chain'deki bu tokenin adresi, Tether şirketinin sitesindeki USDT sözleşmeleri arasında yok: aynı ad, aynı ihraç anlamına gelmez.

27 Kasım 2026'da herkese açık node'lara yapılan isteklerle ve tether.to/en/supported-protocols sayfasından kontrol edildi.

Tuzak: sahte token

Herkes tokeninin adını "USDT" koyabilir. Ad hiçbir şeyi kanıtlamaz. Gerçek USDT bir ikiliyle tanınır: **ağ ve sözleşme adresi**.



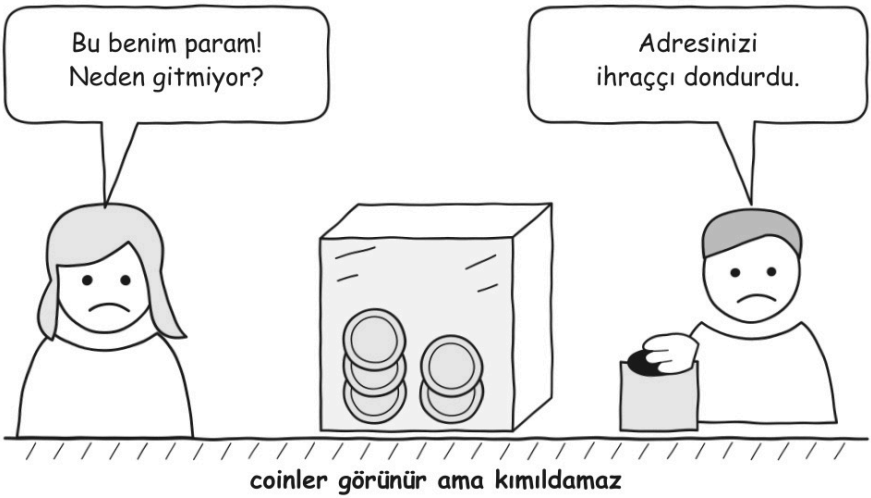
Bu yüzden mağaza her şeyi birden kontrol eder:

- ağı;
- token sözleşmesinin adresini;
- alıcıyı;
- doğru basamak sayısıyla tutarı;
- başarıyı ve onayları.

Stablecoinler: kullanışlı ama bir sahibi var

Stablecoin bir dolar civarında kalmaya çalışır. USDT'de bu bir doğa yasası değil, ihraççı şirketin sözüdür.

İhraççının yetkisi de vardır: Tether'in kullanım koşullarında tokenleri dondurma hakkı açıkça yazılıdır (26 Şubat 2026 tarihli sürüm, tether.to/en/legal).

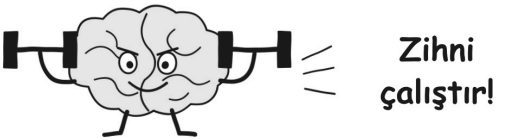


İkinci tuzak: "harcamaya izin verin"

Bir site sizden bir **izin** (`approve`) imzalamanızı isteyebilir: "bu sözleşme tokenlerimi harcayabilir". Dolandırıcıya sınırsız tutarda izin verirseniz, tokenleri sonradan yeni bir imza gerekmeden alır. Gizli ifadeyi bilmesine gerek yoktur.

> **Kural:** neyi imzaladığınızı anlamıyorsanız imzalamayın.

Düşünün



1. Ethereum'daki USDT için node 25.000.000 sayısını döndürdü. Bu kaç USDT?
2. Müşteri adı USDT olan ama sözleşme adresi farklı bir token gönderdi. Sipariş ödendi mi?
3. Artyom'un Ethereum'da 100 USDT'si ve 0 ETH'si var. Bunları tedarikçiye gönderebilir mi?

Cevaplar: 1) 25. 2) Hayır. 3) Hayır: ücreti ödeyecek bir şey yok, 7. bölüme bakın.

Özet

- Coini ağ tutar, tokeni akıllı sözleşme.
- Token adıyla değil, ağı ve sözleşme adresiyle tanınır.
- Tutar, virgülden sonraki basamaklar hesaba katılarak okunur.

İkinci kısım bitti. Artyom blok, anahtar, ücret ve token nedir biliyor. Ama node onun için hâlâ bir kara kutu. **Sırada:** 3. kısım, 10. bölüm. Node çeşitleri.

Bölüm 10. Node çeşitleri

"Peki o zaman orada ne kadar vardı?"

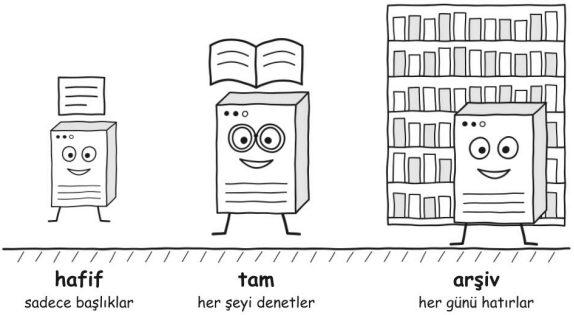
Artyom muhasebesini denkleştiriyor ve birkaç ay önce mağazanın adresinde ne kadar para olduğunu öğrenmek istiyor. Node bugünü gayet güzel yanıtlıyor ama geçmişi söylemeyi reddediyor. Blokzincir her şeyi saklamıyor muydu?

İnceleyelim: kayıtlar ve bakiyeler

Burada iki farklı şey gizli:

- **Geçmiş** — kayıtların kendisi: bloklar ve transferler.
- **Durum** — belirli bir anda kimin neye sahip olduğu: bakiyeler ve sözleşme verileri.

Bankadaki gibi: işlem listesi vardır ama geçen yılın her dakikasının bakiyesi ayrıca saklanmaz. Onu yeniden hesaplamak gerekir. Node'lar bunların ne kadarını hatırladıklarıyla birbirinden ayrılır.

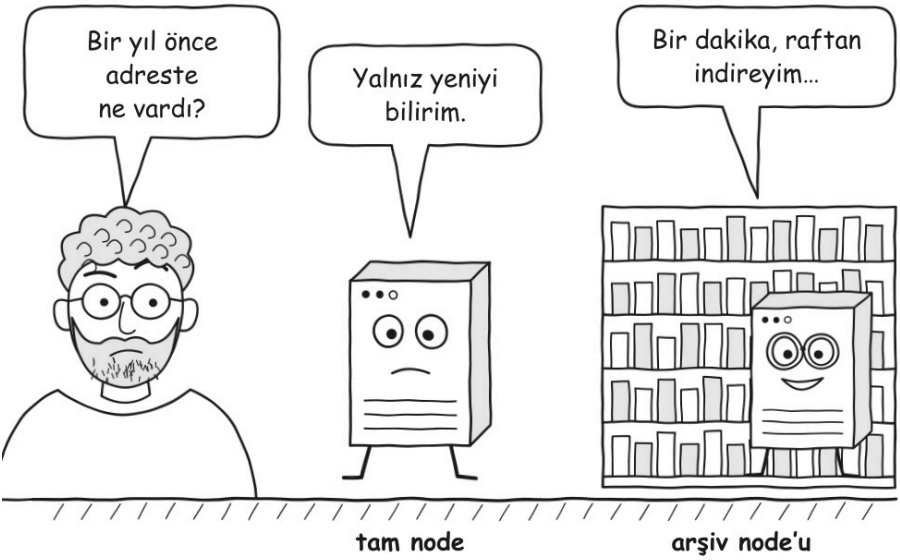


Üç ana tür

Tam node blokları ağın kurallarına göre denetler ve güncel durumu saklar. Ethereum'da bu genellikle son 128 blok kadar, yani yaklaşık 25 dakikadır. Kesin derinlik programa ve ayarlara bağlıdır.

Arşiv node'u her şeyi en baştan denetler ve indirdiğini silmez. Herhangi bir bloktaki durumu hatırlar ama çok daha fazla disk ister.

Hafif node yalnızca blok başlıklarını indirir, ayrıntıları başkalarına sorar. Fikir güzel, ancak 27 Kasım 2026 itibarıyla Ethereum'da günlük kullanıma hazır bir hafif node yok.



Budanmış node: denetledi ve attı

Bitcoin'de disk tasarrufu sağlayan bir mod vardır. **Budanmış node** (pruned) her şeyi denetler, sonra eski blokları siler. Yine de tam yetkili bir tam node'dur; yalnızca sildiğini veremez. Bitcoin Core geliştiricilerine göre ona 10 GB bile yeter.

Ethereum'a da böyle bir imkân geldi: Temmuz 2025'ten itibaren node programları en eski blokların, yani Eylül 2022'deki staking'e geçişten önce çıkanların içeriğini saklamayabiliyor. Bu bir ayardır ve her yerde açık değildir. Bu yüzden "tam" bir node bile çok eski kayıtları hatırlamayabilir.

Deneyelim: geçmişi sorun

Aynı adresin bakiyesini iki kez soralım. Önce şu an için:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getBalance",
  "params":["0xdac17f958d2ee523a2206206994597c13d831ec7","latest"]}'
```

```
{"jsonrpc":"2.0","id":1,"result":"0x2a"}
```

`0x2a` , 42 wei demek; kırıntı bile sayılmaz. Şimdi `"latest"` yerine `"0xe4e1c0"` , yani 15.000.000 numaralı bloğu yazın. Herkese açık node şöyle yanıtlar:

```
{"jsonrpc":"2.0","id":1,"error":{"code":-32601,
 "message":"archive data is not available on the public endpoint,
 use a personal endpoint"}}
```

Türkçesi: “arşiv verileri herkese açık uç noktada yok, kişisel uç nokta kullanın”. Bu bir arıza değil, ücretsiz hizmetin sınırı. RpcNode’da arşiv erişimi ücretli planlarda vardır, Free planında yoktur (3. bölüm).

Bunlar ne kadar yer tutar

Node	Disk, yaklaşık
Bitcoin, budanmış	en az 10 GB
Bitcoin, tam	yaklaşık 900 GB
Ethereum, tam	yaklaşık 2 TB ve üzeri
Ethereum, arşiv	programa göre 2 TB’tan 14 TB ve üzerine

Ethereum verileri, ethereum.org’un node çalıştırma sayfasından, 27 Kasım 2026.

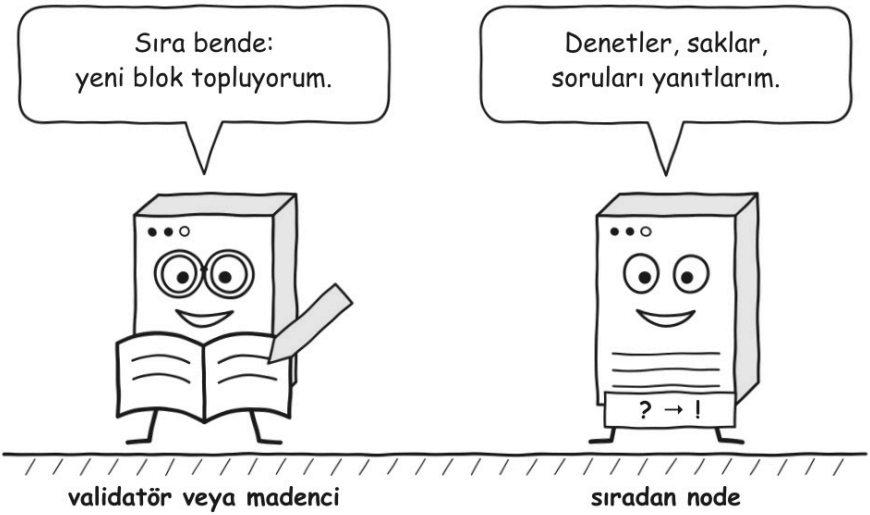
İkinci satırı kendiniz kontrol edebilirsiniz: Bitcoin node’u boyutunu dürüstçe söyler.

```
curl https://public-bitcoin-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"getblockchaininfo","params":[]}'
```

Uzun yanıtta iki satırı bulun. Bu bölüm yazılırken şöyleydi: `"size_on_disk":896027659764` (yardımcı dosyalarla birlikte yaklaşık 896 GB) ve `"pruned":false` — node budanmamış.

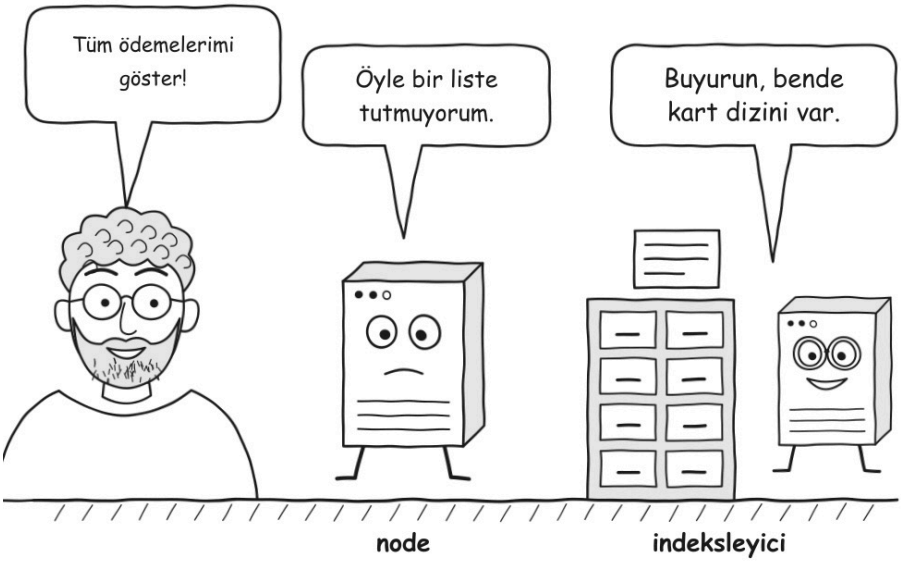
Peki blokları kim yazar?

Bu ayrı bir görev. **Validatör** veya **madenci** blok oluşturmaya katılır (8. bölüm). Ödeme kontrol etmek için onlardan biri olmanız gerekmez: okuyan, denetleyen ve yanıtlayan bir node yeter.



Tuzak: "bütün ödemelerimi node'a sorarım"

Ethereum node'una sorulabilen standart sorular arasında "bu adresin bütün transferlerini göster" diye bir şey yoktur. Arşiv node'u bile bunu yapamaz. `eth_getTransactionCount` metodu yalnızca adresten **gönderilen** işlemlerin sayısını söyler.

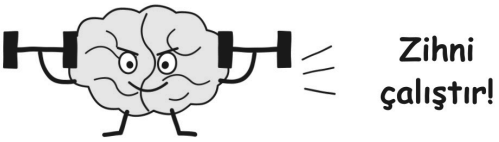


Böyle listeler için **indeksleyici** gerekir: blokları okuyup kayıtları kart dizinine yerleştiren bir program. Blok gezgini siteleri de böyle çalışır. Mağaza için en kolayı, kendi ödemelerini en baştan kaydetmek ve yenilerini izlemektir; bunu 26. bölümde anlatacağız.

Hangi iş için ne gerekir

İş	Uygun olan
Yeni bir ödemeyi kontrol etmek	sıradan tam node
Geçmiş bir tarihteki bakiyeyi öğrenmek	arşiv node'u
Bir adresin tüm transfer listesini almak	indeksleyici
Blok oluşturmaya katılmak	validatör veya madenci

Düşünün



1. Artyom mağazanın bir yıllık bakiye grafiğini, her güne bir nokta olacak şekilde çizmek istiyor. Hangi node gerekir?
2. Budanmış bir Bitcoin node'u eski blokları sildi. Yenilerini denetlemesine güvenilebilir mi?
3. Müşteriye geçmişteki tüm siparişlerini ödemeleriyle göstermek için ne gerekir: arşiv node'u mu, mağazanın kendi veritabanı mı?

Cevaplar: 1) arşiv node'u. 2) Evet, silmeden önce hepsini denetledi. 3) Kendi veritabanı: node böyle listeler tutmaz.

Özet

- Geçmiş kayıtlardır, durum ise belirli bir andaki bakiyelerdir.
- Tam node güncel durumu, arşiv node'u her andaki durumu hatırlar.
- Node bir adresin transfer listesini vermez; bunun için indeksleyici gerekir.

Sırada: 11. bölüm. Node'a şimdiden onlarca soru sorduk. Hangi dilde sorduğumuzu anlamanın zamanı: JSON-RPC, REST, gRPC ve WebSocket.

Bölüm 11. Node ile nasıl konuşulur

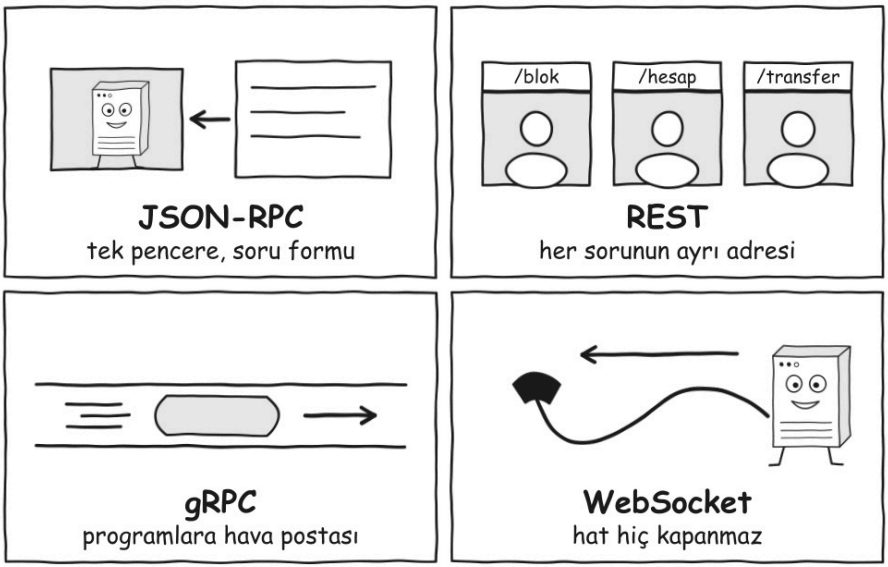
İsteklerdeki bu harfler ne?

Artyom bu kitaptaki komutları on kez kopyaladı, hepsi de çalıştı. İçlerinde ne yazdığını anlamanın zamanı geldi: RPC nedir, bu kadar parantez neden var ve farklı ağların belgelerinde istekler neden farklı görünüyor.

İnceleyelim: API ve RPC nedir

API, bir programın başka bir programa önceden bilinen kurallarla başvurduğu bir penceredir. **RPC** (uzak yordam çağrısı), başka bir bilgisayardaki programdan, yanı başınızdaymış gibi bir şey yapmasını istemenin yoludur. Node bu tür istekleri kabul eder.

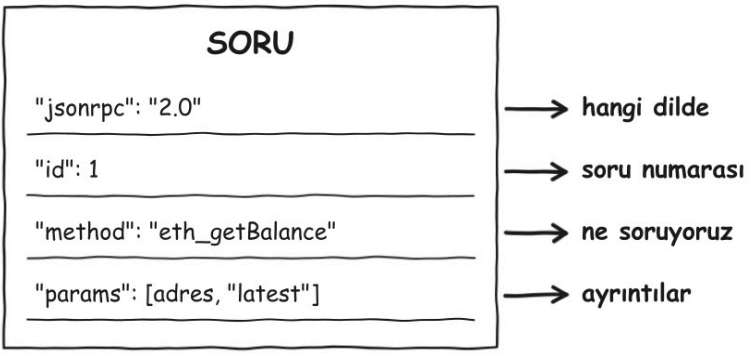
Konuşmanın birkaç yolu vardır:



Yöntem	Nasıl görünür	Nerede karşılaşırsınız
JSON-RPC	Tek adres; istekte metot ve parametreler belirtilir	Ethereum ve benzerleri, Solana, Bitcoin
REST	Her sorunun adreste kendi yolu vardır	TRON, TON, Aptos
gRPC	Katı bir şemaya uyan kompakt ikili mesajlar	Programlar arası alışveriş, örneğin Sui'de
WebSocket	Kapanmayan bir bağlantı	Node'un yenilikleri kendisi bildirmesi gerektiğinde

WebSocket bir dil değil, bir **iletişim hattıdır**: Ethereum'da bu hattan aynı JSON-RPC mesajları gider. Bazı ağlarda aynı anda birkaç yöntem bulunur.

JSON-RPC'nin anatomisi



JSON, veri yazmanın basit bir yoludur: süslü parantezler, tırnak içinde adlar ve değerler. İşte 3. bölümdeki komut, satır satır:

Komutun parçası	Anlamı
`curl`	terminalden istek gönderen program
`https://…`	uç noktanın adresi
`-H '…'`	"içinde JSON var" notu
`-d '{…}'`	sorunun kendisi

Yanıtta `result` (ne elde edildi) ve `error` (ne olmadı) alanlarına bakılır. Bazı node'lar bu alanlardan yalnızca birini gönderir, bazıları ikisini de; o zaman boş olan `null` olur. Yanıttaki `id` sorudakiyle aynıdır: eşleştirme böyle yapılır.

Tek seferde birkaç soru

Sorular köşeli parantez içinde bir listeye konabilir. Buna **toplu istek** (batch) denir:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '[{"jsonrpc":"2.0","id":1,"method":"eth_blockNumber","params":[]},
       {"jsonrpc":"2.0","id":2,"method":"eth_chainId","params":[]}]'
```

```
[{"jsonrpc":"2.0","id":1,"result":"0x1946420"},
 {"jsonrpc":"2.0","id":2,"result":"0x1"}]
```

İlk yanıt 26.502.176 numaralı blok, ikincisi ağın numarası: Ethereum için 1. JSON-RPC kurallarına göre toplu istekteki yanıtlar herhangi bir sırayla gelebilir, bu yüzden `id` alanına bakın. Şunu da unutmayın: toplu istek gidiş gelişten tasarruf ettirir, limitten değil. RpcNode'da on soruluk bir toplu istek on istek sayılır ve bütün olarak ya kabul edilir ya reddedilir.

Bir de REST

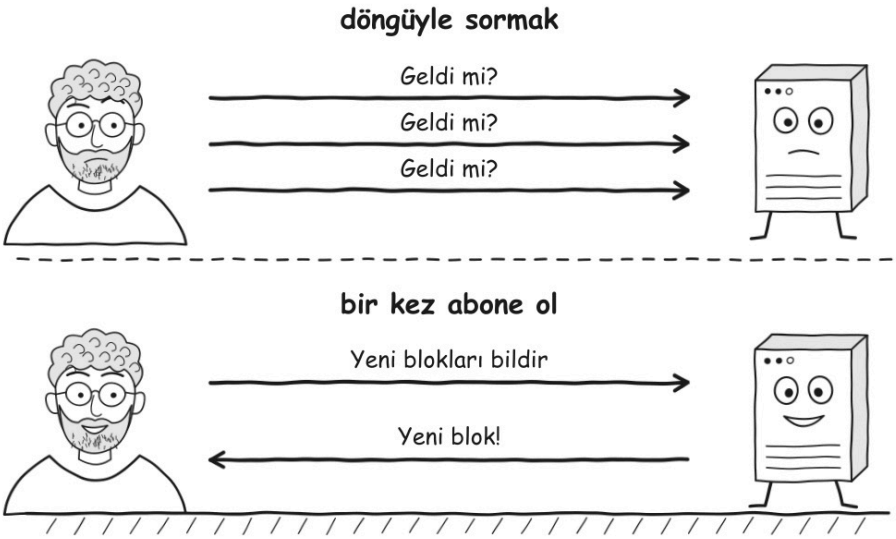
Burada soru adresin içine gömülüdür. TRON ağından en yeni bloğu isteyelim:

```
curl https://public-tron-mainnet.rpcnode.dev/wallet/getnowblock
```

Bloğu ve transferlerini anlatan uzun bir yanıt gelir. Bu bölüm yazılırken bu 88.351.889 numaralı bloktu ve içinde 194 işlem vardı. Başka bir soru için adresin sonunda başka bir yol yazılır.

WebSocket: sormak değil, dinlemek

Her saniye "geldi mi?" diye sormak israftır. Açık bir hat üzerinden bir kez abone olursunuz, node haberleri kendisi gönderir.



RpcNode'un herkese açık node'ları WebSocket'e yanıt vermez; kişisel uç nokta gerekir. Onun adresi `wss://` ile başlar. İşte JavaScript ile bir taslak:

```
const ws = new WebSocket('wss://UÇ_NOKTANIZ')

ws.onopen = () => ws.send(JSON.stringify({
  jsonrpc: '2.0', id: 1,
  method: 'eth_subscribe', params: ['newHeads'],
}))

ws.onmessage = (event) => console.log(event.data)
ws.onclose = () => console.log('bağlantı koptu')
```

Hatlar kopar. Program yeniden bağlanmayı, tekrar abone olmayı ve neleri kaçırdığını kontrol etmeyi bilmelidir. Abonelikler de kredi harcar.

gRPC

Bu, hıza ve katı biçime önem veren programlar için bir yöntemdir. Mesajlar gözle okunmaz, alanları ayrı bir dosyada önceden tanımlanır. İşe yeni başlayan bir mağazanın genellikle buna ihtiyacı olmaz. RpcNode'da gRPC her ağda yoktur: uç noktanın taşıma yöntemleri listesine bakın.

Tuzak: "200 geldiyse her şey yolunda"

200 teslimat kodudur: zarf yerine ulaştı. İçinde ise bir hata olabilir. "hello" adresinin bakiyesini isteyelim:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getBalance","params":["hello","latest"]}'
```

Yanıt kodu 200, içinde ise `-32602` numaralı bir `error` var: geçersiz parametreler.



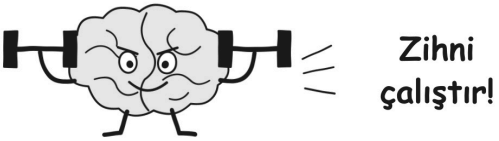
Bu yüzden program her zaman hem kodu hem de `error` alanını kontrol eder.

Sık görülen hatalar

Kod	Ne oldu	Ne yapmalı
`-32700`	İstek metni çözümlenemiyor	Parantez ve tırnakları kontrol edin
`-32600`	İstek kurallara uygun değil	Alanları kontrol edin
`-32601`	Metot bulunamadı veya kapalı	Adı ve planı kontrol edin
`-32602`	Geçersiz parametreler	Değerleri kontrol edin
`-32005` (HTTP 429)	RpcNode: çok sık istek	Bekleyin ve daha seyrek sorun
`-32006` (HTTP 402)	RpcNode: kredi yetmiyor	Harcamayı ve planı kontrol edin

İlk dört kod JSON-RPC 2.0 standardından, son ikisi RpcNode'un kodları.

Düşünün



1. Toplu isteğin ilk sorusuna `eth_blockNumber` yerine `eth_blockNumbr` yazın. Hangi kod gelir? İkinci, doğru soru yanıtlanır mı?
2. Kendiliğinden güncellenmesi gereken bir "son blok" panosu için hangisi daha iyi: saniyede bir sorgulamak mı, abonelik mi?
3. Program HTTP 200 aldı ve siparişi hemen ödendi diye işaretledi. Sorun ne?

Cevaplar: 1) `-32601`*, üstelik hepsi için tek bir tane: RpcNode toplu isteğin tamamını reddeder, ikinci soru da yanıtsız kalır. 2) Abonelik. 3) Ne* `error` *alanı ne de transferin kendisi kontrol edilmiş (7. bölüm).*

Özet

- JSON-RPC metodu isteğin içinde, REST ise adreste belirtir.
- WebSocket, node haberleri kendisi bildirsin diye hattı açık tutar.
- Başarı 200 kodu değil, `error` içermeyen ve beklediğinizi taşıyan yanıttır.

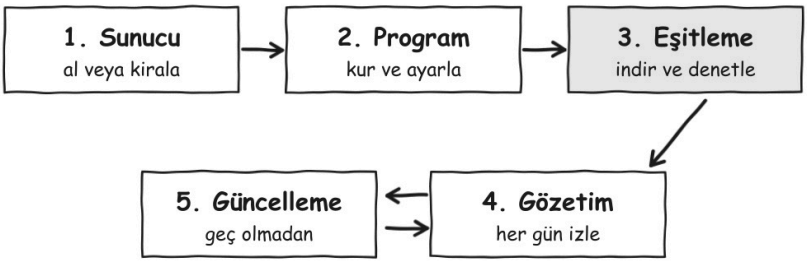
Sırada: 12. bölüm. Pencerenin öbür tarafına bakalım: kendi node'unuz nasıl yaşar, eşitleme nedir ve neden bozulur.

Bölüm 12. Kendi node'unuzun içi: kurulum, eşitleme ve arızalar

"Bir akşamlık iş bu!"

Tanıdık bir programcı Artyom'u ikna etmeye çalışıyor: "İndir, çalıştır, al sana kendi node'un." Çalıştırmak gerçekten uzun sürmez. Uzun süren, sonrasında başlayanlardır.

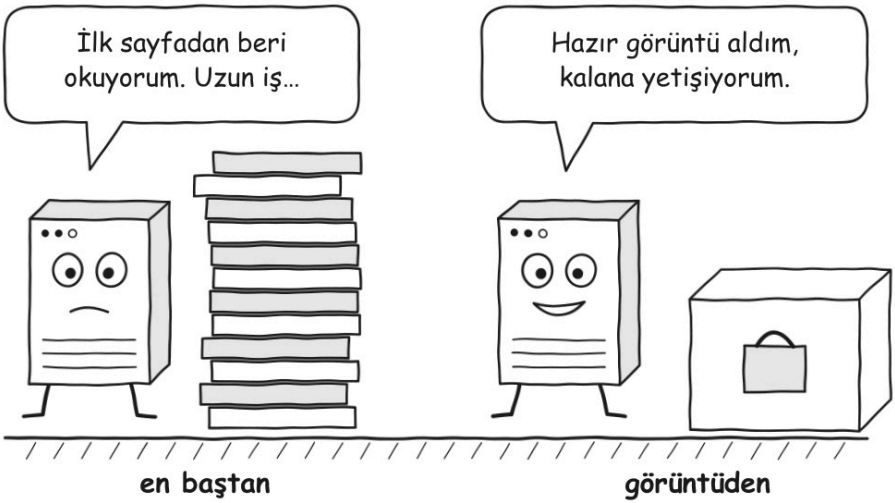
İnceleyelim: bir node'un hayatı



1. **Sunucu.** Gereksinimlere 2. bölümde baktık.
2. **Program.** Her ağın kendisininki vardır, çoğu zaman da seçebileceğiniz birkaç tane. Ethereum'da aynı anda iki tane gerekir: **yürütme istemcisi** (transferleri ve sözleşmeleri çalıştırır) ve **mutabakat istemcisi** (hangi zincirin doğru olduğunu izler).
3. **Eşitleme.** Node ağa yetişir.
4. **Gözetim.** Geri kalmış mı, yer yetiyor mu.
5. **Güncellemeler.** Ağ kural değiştirecekse program önceden değiştirilir.

Eşitleme: ağa yetişmek

Yeni bir node hiçbir şey bilmez. Diğerlerine yetişene kadar ona bakiye sormak boşunadır: geçmişi söyler.



- **En baştan.** Node her bloğu kendisi denetler. Güvenilir ve uzun sürer. Bitcoin'in ilk indirmesi yaklaşık 740 GB'tır.
- **Anlık görüntüden.** Veritabanının hazır bir kopyası alınır, kalanı tamamlanır. Daha hızlıdır ama görüntüyü hazırlayana güvenirsiniz.
- **Ara yollar.** Ethereum programları genellikle güncel durumu indirip blok başlıklarıyla karşılaştırır.

Süre birkaç saatten pek çok güne uzanır: ağa, yönteme, diske ve bağlantıya bağlıdır.

Deneyelim: iki satırlık ayar

`bitcoin.conf` ayar dosyasının başı şöyle görünebilir:

```
server=1
prune=5000
```

Satır	Anlamı
`server=1`	RPC üzerinden soru kabul et; başka bir izin verilmedikçe yalnızca aynı bilgisayardan
`prune=5000`	budanmış node (10. bölüm): eski blokları sil, yaklaşık 5 GB'lık kısmını tut; yardımcı veriler bunun üstüne yer kaplar

Bu eksiksiz bir talimat değildir. Kurmadan önce kendi programınızın belgelerini okuyun.

Node'un yetiştiği nasıl anlaşılır

Node'a doğrudan sorulabilir. Ethereum'da:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_syncing","params":[]}'
```

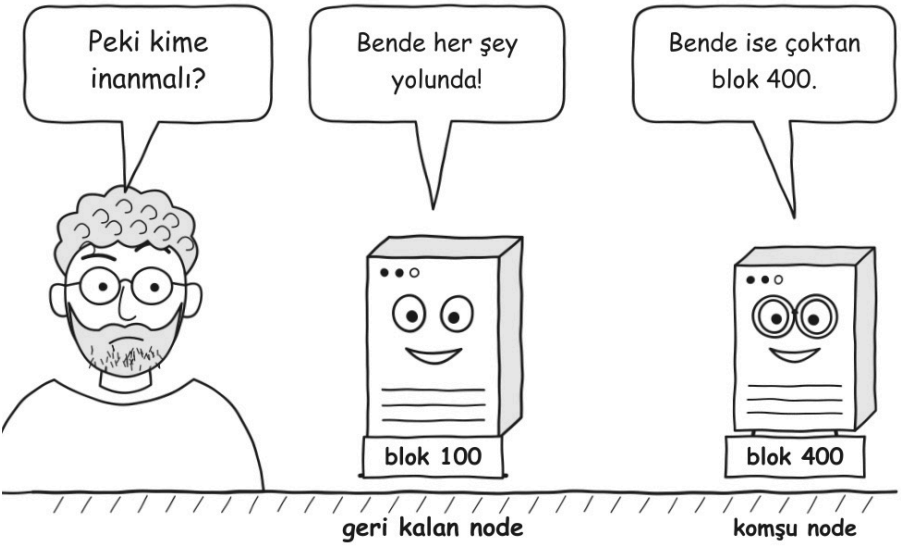
```
{"jsonrpc":"2.0","id":1,"result":false}
```

`false`, "şu an arayı kapatmaya çalışmıyorum" demektir. Node hâlâ yetişiyorsa bunun yerine numaralar gelir: nerede olduğu ve nereye gittiği. Bitcoin'de 10. bölümdeki `getblockchaininfo` yanıtına bakılır:

Alan	Kontrol anındaki değer	Anlamı
`blocks`	977.715	kaç blok denetlendiği
`headers`	977.715	kaç bloktan haberdar olunduğu
`initialblockdownload`	`false`	ilk indirme geride kaldı

Tuzak: "her şey yolunda dedi ya"

Node bildiklerine göre hüküm verir. Diğerleriyle bağlantısı koptuysa geri kaldığından haberi bile olmayabilir.



Bu yüzden iki şey kontrol edilir: **son bloğun tazeliği** ve **başka bir kaynakla karşılaştırma**. Tazeliği öğrenmek kolaydır. Son bloğun zamanını isteyin:

```
curl -s https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getBlockByNumber",
  "params":["latest",false]}' | grep -o '"timestamp":"0x[0-9a-f]*"'
```

```
"timestamp":"0x6b08aacb"
```

Bu, 1970 yılından beri geçen saniyedir. Onu şimdiki zamandan çıkarın (kendi sayınızı koyun):

```
echo $(( $(date +%s) - 0x6b08aacb ))
```

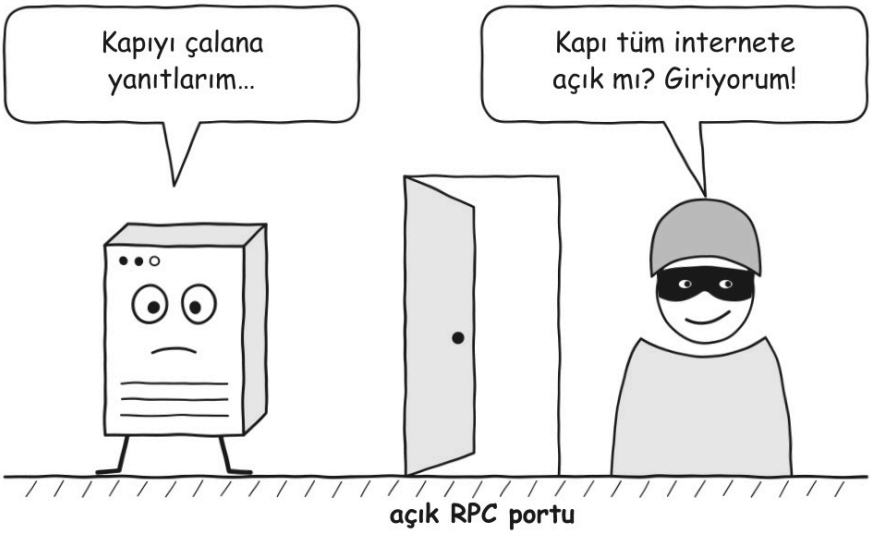
Bloğun saniye cinsinden yaşı çıkar. Bu bölüm yazılırken 7 saniyeydi: 12 saniyede bir blok üreten Ethereum için bu normaldir. Sizin node'unuzda binlerce çıkıyorsa geri kalmış demektir. 1. bölümdeki, bir saat geride kalan node sorusunu hatırlıyor musunuz?

Neler bozulur

Görünen	Sık sebebi	Ne yapmalı
Node durdu	Diskte yer bitti	Yeri izleyin ve önceden büyütün
Sürekli geride	Zayıf disk veya bağlantı, az sayıda komşu	Donanımı ve ağ bağlantısını kontrol edin
Ağ güncellemesinden sonra "takıldı"	Programın eski sürümü	İlan edilen tarihten önce güncelleyin
Elektrik kesintisinden sonra açılmıyor	Veritabanı bozuldu	Yedekten geri yükleyin veya yeniden eşitleyin
Görünür bir sebep yokken yavaş	Yabancıların istekleri yük bindiriyor	Yabancılara erişimi kapatın

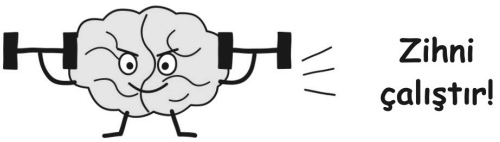
İkinci tuzak: açık kapı

Bitcoin Core ve Geth, başlangıçta RPC sorularını yalnızca kendi bilgisayarlarından kabul eder. Bu boşuna değildir. Portu bütün internete açarsanız herkesi içeri alırsınız; node üzerinde bir cüzdan varsa paranızı da riske atarsınız.



- Yalnızca kendi adreslerinize erişim izni verin (güvenlik duvarı).
- Node'un önüne anahtar kontrolü koyun.
- Siteye yanıt veren node üzerinde para dolu cüzdan tutmayın.

Düşünün



1. `eth_syncing` `false` diyor ama son blok 40 dakikalık. Node sağlıklı mı?
2. 2000 GB'lık diskin 1900 GB'ı dolu, günde 5 GB ekleniyor. Kaç gün sonra her şey durur?
3. Site başka bir sunucuda. "Site sorabilsin diye" RPC portu herkese açılabilir mi?

Cevaplar: 1) Hayır. 2) En geç 20 gün sonra; onarıma ise bugün başlanmalı. 3) Hayır: yalnızca sitenin adresine izin verin.

Özet

- Node'u çalıştırmak işin sonu değil, başlangıcıdır.
- Sağlık, bloğun tazeliğiyle ve başka bir kaynakla karşılaştırarak kontrol edilir.
- RPC portu bütün internete açılmaz.

Sırada: 13. bölüm. Bütün bunlar en sevdiğiniz uğraş değilse: sağlayıcı nasıl seçilir ve ona neler sorulur.

Bölüm 13. Sağlayıcı nasıl seçilir

Herkes en iyisini vaat ediyor

Artyom kararını verdi: kendi node'u ona göre değil. Sağlayıcıların sitelerini açıyor, her yer ünlem işareti.



İnceleyelim: önce kendinizden başlayın

İhtiyacınızı bilmeden sağlayıcıları karşılaştırmak boşunadır. Önce şunları yanıtlayın:

- Hangi ağlar gerekli, test ağları da lazım mı?
- Saniyede ve ayda kaç istek?
- Eski geçmiş (arşiv) gerekli mi?
- Olaylardan anında haberdar olmak gerekiyor mu (WebSocket, bildirimler)?

İsteklerinizi sayın

Artyom'un ödeme bekleyen 20 siparişi var ve site her birini 10 saniyede bir kontrol ediyor:

> 20 sipariş × dakikada 6 kez = dakikada 120 istek, yani saniyede 2.
> 30 günde: 5.184.000 istek.

Sonra bunu metodun kredi cinsinden fiyatıyla çarpın: her metodun fiyatı farklıdır. Şimdi de bu kadar sık sormaya gerek var mı, onu düşünün. 30 saniyede bir sorarsanız istek sayısı üçte bire iner.

Soru listesi



Sorun	Şüphe uyandıran
Ağlarım var mı? Gereken metotlar da içlerinde mi?	Metot listesi olmayan uzun bir ağ listesi
Limitler: saniyede ve toplamda kaç istek; aşınca ne olur?	Rakamsız "sınırsız"; belirsiz ek ücretler
Arşiv, WebSocket, gRPC: hangi planda ve hangi ağlarda?	"Her şey var" ama ayrıntı yok
Node arızası: başkasına geçiliyor mu? Sunucular nerede?	Cevap yok
Erişilebilirlik: ne vaat ediliyor, nasıl ölçülüyor, ihlalde ne olur?	Dönemi ve koşulu belli olmayan bir yüzde
Koruma: adresler ve siteler sınırlanabiliyor mu? İsteklerle ilgili neler kaydediliyor?	Anahtar kimin elinde olsa çalışıyor
Destek: nereye yazılır, hangi dilde yanıt verilir?	Yalnızca isimsiz bir form
Ayrılış: metotlar standart mı? Aylık ödenebiliyor mu?	Uzun zorunlu süre

Yüzdenin sakladığı

"%99 erişilebilirlik" neredeyse "her zaman" gibi duyulur. Bunun 30 günde ne kadar kesintiye izin verdiğini hesaplayalım:

Vaat	Çalışmayabileceği süre
%99	7 saat 12 dakika
%99,9	yaklaşık 43 dakika
%99,99	yaklaşık 4 dakika

Bir de **neyin** çalışıyor sayıldığını sorun: sağlayıcının sitesi mi, sizin ağınızdaki uç noktanız mı.

Deneyelim: hızı kendiniz ölçün



Başkalarının ölçümleri başka bir yerden yapılmıştır. Sitenizin yaşayacağı sunucudan beş istek gönderin:

```
for i in 1 2 3 4 5; do
  curl -s -o /dev/null -w '%{time_total} s\n' \
    https://public-ethereum-mainnet.rpcnode.dev \
    -H 'Content-Type: application/json' \
    -d '{"jsonrpc":"2.0","id":1,"method":"eth_blockNumber","params":[]}'
  sleep 1
done
```

Beş sayı göreceksiniz: saniye cinsinden yanıt süresi. Yalnızca en iyisine bakmayın: müşteriye en kötüsü de denk gelir. Burada bağlantı her seferinde yeniden açılıyor; sürekli bağlantı kullanan bir program genellikle daha hızlıdır. Yalnızca hızı değil, bloğun tazeliğini de karşılaştırın (12. bölüm).

Bu soruları bize de sorun

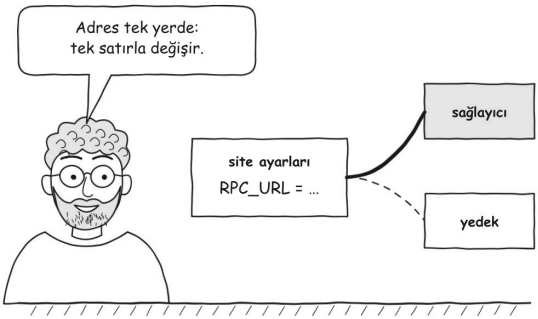
Kitap RpcNode hakkında, öyleyse onu da sınayalım. Veriler 27 Kasım 2026 tarihli.

Soru	RpcNode
Ağlar	sitedeki listede 25
Ücretsiz deneme	Free planı ve anahtarsız herkese açık node'lar
Limitler	Free: saniyede 15 istek ve 15 milyon kredi; Starter: 40 ve 50 milyon
Aşım halinde	aşım için ek ücret yok: istekler reddedilir (429 veya 402, 11. bölüm)
Arşiv	ücretli planlarda
WebSocket ve gRPC	kişisel uç nokta üzerinden, her ağda değil
Sunucular nerede	Frankfurt, ABD, Japonya
Koruma	uç nokta IP adreslerine ve sitelere göre sınırlanabilir
Gizlilik	sağlayıcı isteklerinizi görür (3. bölüm)
Erişilebilirlik ve destek	rpcnode.dev sitesindeki güncel koşullara bakın

Tuzak: en büyük rakama göre seçmek

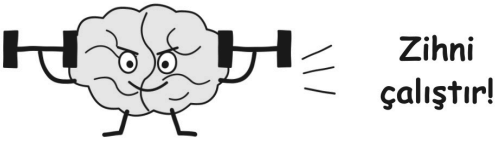
Sizin iki ağınız kötü çalışıyorsa bin ağın faydası olmaz. İhtiyacınız olan metot pahalıysa veya kapalıysa en düşük fiyatın da faydası olmaz. Ücretsiz veya en ucuz planı alın ve birkaç gün boyunca gerçek isteklerinizi çalıştırın.

Kök salmayın



- Uç nokta adresini kodun içinde değil, tek bir ayarda tutun.
- Mümkün olduğunca ağın standart metotlarını kullanın.
- Bir yedek seçenek hazırlayın ve bir kez deneme için ona geçin.

Düşünün



1. 20 sipariş, 30 saniyede bir kontrol. 30 günde kaç istek eder?
2. %99,9 vaat edildi ama uç nokta ay içinde iki saat çalışmadı. Söz tutuldu mu?
3. Uç nokta adresi kodun 14 yerine yazılmış. Bunun nesi kötü?

Cevaplar: 1) 1.728.000. 2) Hayır: izin verilen yaklaşık 43 dakika. 3) Değiştirirken bir yeri unutmak kolaydır; koddaki anahtar da daha kolay sızar.

Özet

- Önce ihtiyacınızı hesaplayın, sonra karşılaştırın.
- Vaatleri kendi ölçümleriniz ve kendi isteklerinizle sınayın.
- Kendinize ayrılma imkânı bırakın.

Üçüncü kısım bitti. Artyom node nedir, onunla nasıl konuşulur ve kimden alınır biliyor. Ama ağlar birbirinden farklı! **Sırada:** 4. kısım, 14. bölüm. Bitcoin ve akrabaları.

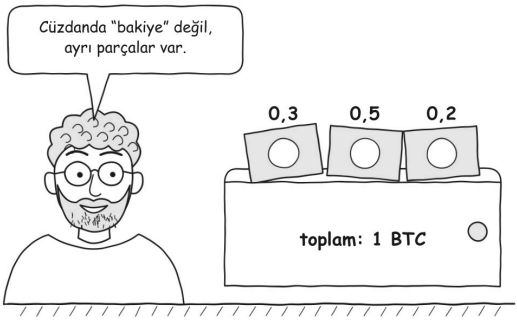
Bölüm 14. Bitcoin ve akrabaları

“Peki bunun bakiyesi nerede?”

Müşteri bitcoin ile ödemek istiyor. Artyom artık tecrübeli: Ethereum’daki gibi node’a adresin bakiyesini soracak. Ama Bitcoin node’unda bu kadar basit bir soru yok. Neden?

İnceleyelim: hesap değil, para parçaları

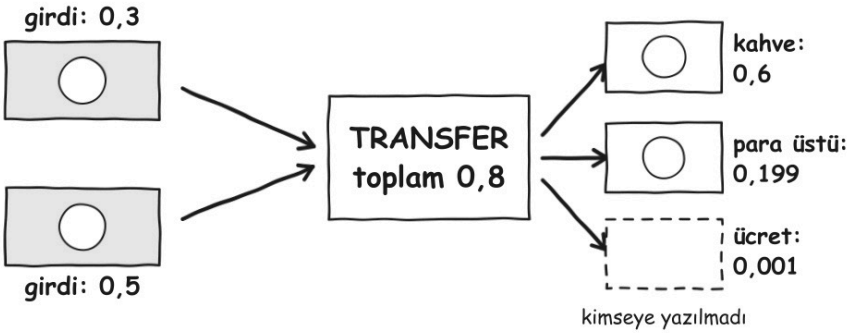
Ethereum’da adresin, banka hesabındaki gibi bir bakiyesi vardır. Bitcoin’de yoktur. Orada farklı değerde banknotlar gibi ayrı “para parçaları” bulunur.



Bu parçaların her birine **UTXO** (harcanmamış çıktı) denir. Bakiye, harcanmamış parçalarınızın toplamıdır. Onu cüzdan veya indeksleyici hesaplar (10. bölüm).

Transfer: girdiler, çıktılar ve para üstü

Banknot yırtılıp bir kısmı verilemez. Tamamını verip para üstü almak gerekir. Burada da öyle: transfer eski parçaları tümüyle harcar (**girdiler**) ve yenilerini oluşturur (**çıktılar**).



- Bir çıktı alıcıya gider.
- İkincisi kendinize **para üstüdür**.
- Hiçbir çıktıya yazılmayan her şey madenciye kalır. Bu **ücrettir**: girdiler eksi çıktılar.

> 0,8 − 0,6 − 0,199 = 0,001 BTC ücret

Ücretin fiyatı tutara göre değil, transferin boyutuna göre hesaplanır: **sanal bayt başına satoshi** olarak. Satoshi, bitcoinin yüz milyonda biridir. Girdide ne kadar çok parça varsa transfer o kadar büyük ve o kadar pahalı olur.

İnsanlar arasındaki ilk transfer

12 Ocak 2009'da 170 numaralı blokta 50 BTC'lik bir parça ikiye ayrıldı: 10 BTC ve 40 BTC. On bitcoini alan programcı Hal Finney, bunu Satoshi'nin deneme için gönderdiğini bizzat anlattı. 40 BTC'lik para üstü gönderene döndü. Ücret yoktu.

Blokzincirde isim yoktur: Finney'i kendi anlattıklarından biliyoruz (bitcointalk.org, 2013).

Bu transfer bugün hâlâ sorgulanabilir:

```
curl https://public-bitcoin-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"getrawtransaction","params":
  ["f4184fc596403b9d638783cf57adfe4c75c605f6356fbc91338530e9831e9e16",true]}'
```

Yanıtta `vin` (girdiler, burada bir tane) ve `vout` (çıktılar, burada iki tane: `10.0` ve `40.0`) alanlarını arayın.

Deneyelim: parça harcanmış mı?

`gettxout` metodu "bu parça hâlâ duruyor mu?" diye sorar. Parça, transfer numarası ve sıfırdan sayılan çıktı numarasıyla belirtilir. O 10 BTC'yi soralım:

```
curl https://public-bitcoin-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"gettxout","params":
  ["f4184fc596403b9d638783cf57adfe4c75c605f6356fbc91338530e9831e9e16",0]}'
```

```
{"jsonrpc":"2.0","result":null,"id":1}
```

`null`, böyle harcanmamış bir parça yok demektir: çoktan harcanmış. Şimdi de başlangıç bloğundan sonraki ilk bloğun ödülüne bakalım:

```
curl https://public-bitcoin-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"gettxout","params":
  ["0e3e2357e806b6cdb1f70b54c3a3a17b6714ee1f0e68bebb44a74b1efd512098",0]}'
```

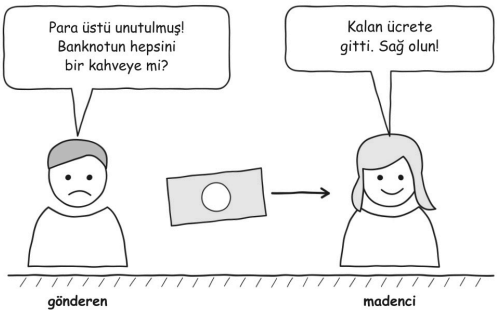
Burada `"value":50.00000000` ve neredeyse bir milyon onay gelir. Bu bölüm yazılırken o 50 BTC'ye hâlâ kimse dokunmamıştı.

Bitcoin hakkında üç bilgi daha

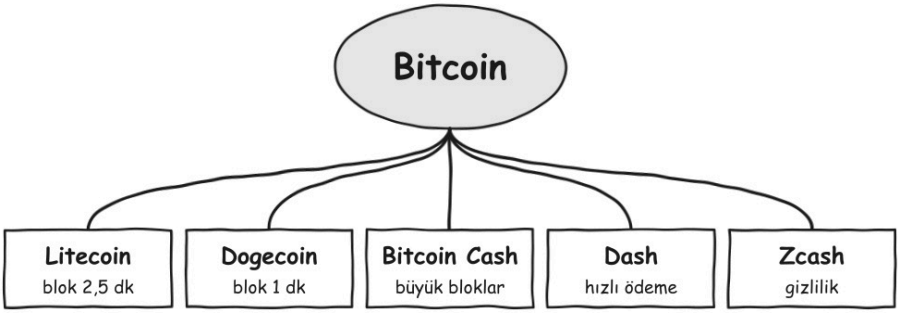
- Toplam coin sayısı **21 milyonu** geçmeyecek.
- Madencinin ödülü, üzerine 100 blok binmeden harcanamaz.
- Büyük tutarlarda 6 onay, yaklaşık bir saat beklenir (8. bölüm).

Tuzak: unutulan para üstü

Sıradan bir cüzdan para üstünü kendisi ekler. Ama bir program transferi elle kuruyorsa ve para üstü çıktısı unutulmuşsa, kalanın tamamı ücret olur.



Akrabalar



yapıları akraba ama her birinin ağı ve coini ayrı

Ağ	Blok, yaklaşık	Farkı	Adres şununla başlar
Bitcoin	10 dk	ailenin büyüğü	`1`, `3`, `bc1`
Litecoin	2,5 dk	farklı madencilik algoritması, Scrypt	`L`, `M`, `ltc1`
Dogecoin	1 dk	arzın üst sınırı yok; Litecoin ile birlikte çıkarılabiliyor	`D`
Bitcoin Cash	10 dk	çok daha büyük bloklar	`bitcoincash:q…`
Dash	2,5 dk	hızlı ödemeler (InstantSend) ve blokların değiştirilmesine karşı koruma (ChainLocks)	`X`
Zcash	75 sn	transferler gizlenebiliyor	açık `t1`, gizli `zs`, `u1`

Yapıları benzer, bu yüzden soruları da neredeyse aynı. Beş ağın zincir yüksekliğini birden soralım:

```
for net in bitcoin litecoin doge bch dash; do
  echo -n "$net: "
  curl -s https://public-$net-mainnet.rpcnode.dev \
    -H 'Content-Type: application/json' \
    -d '{"jsonrpc":"2.0","id":1,"method":"getblockcount","params":[]}'
  echo; sleep 1
done
```

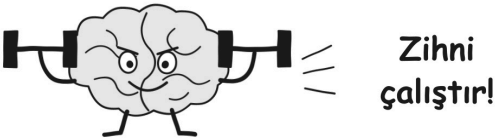
```
bitcoin: {"jsonrpc":"2.0","result":977717,"id":1}
litecoin: {"result":3219941,"error":null,"id":1}
doge: {"result":6474162,"error":null,"id":1}
bch: {"result":979125,"error":null,"id":1}
dash: {"result":2578936,"error":null,"id":1}
```

İkinci tuzak: "error alanı varsa hata vardır"

Yanıtlara bakın: dört ağda `error` alanı hep var. `null` değeri "her şey yolunda" demektir. 11. bölümün öğütlediği gibi alanın varlığını değil, içinde ne olduğunu kontrol edin.

Ağları da karıştırmayın. Akrabalar birbirinin coinini kabul etmez. Bitcoin Cash'in eski adres biçimi Bitcoin'inkiyle aynı görünüyordu; bu yüzden orada `bitcoincash:` ön ekli yeni bir biçim getirildi.

Düşünün



1. Cüzdanda 0,3, 0,5 ve 0,2 BTC'lik parçalar var. 0,4 ödenecek, ücret 0,001. Tek parça yeter mi, para üstü ne olur?
2. Yüz küçük parçadan toplanan 1 BTC'yi göndermek, tek parça halindeki 1 BTC'yi göndermekten neden daha pahalı?
3. Girdi 1,0 BTC, tek çıktı 0,4 BTC. 0,6 nereye gitti?

Cevaplar: 1) evet, 0,5'lik parça; para üstü 0,099. 2) Girdi çoksa transferin boyutu büyür. 3) Ücret oldu. Büyük olasılıkla para üstü unutuldu.

Özet

- Bitcoin'de hesap bakiyesi yoktur, harcanmamış parçalar (UTXO) vardır.
- Ücret girdiler eksi çıktılardır; para üstünü unutmayın.
- Akrabaların yapısı benzer ama bunlar ayrı ağlardır.

Sırada: 15. bölüm. Ethereum: hesaplar, gas ve olaylar.

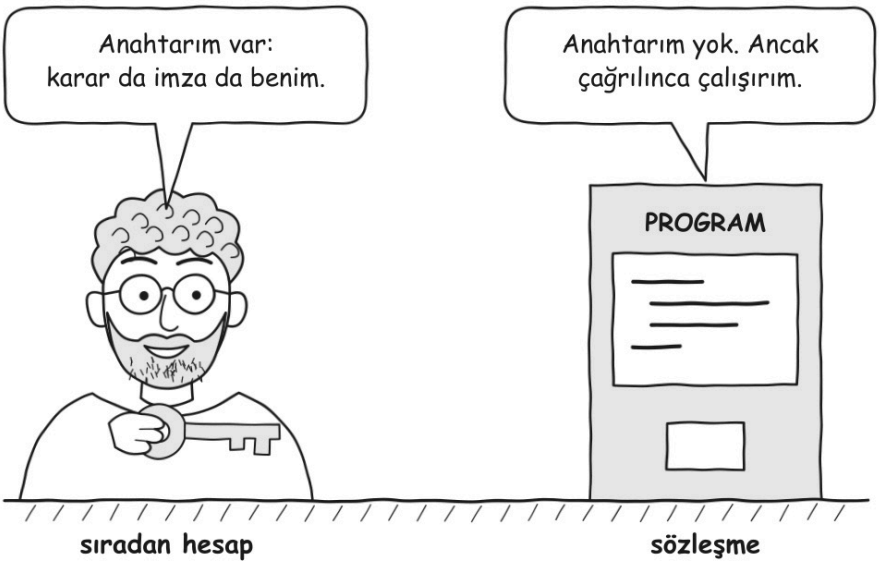
Bölüm 15. Ethereum: hesaplar, gas ve olaylar

“Dolar göndermek neden daha pahalı?”

Artyom tedarikçiye önce biraz ETH, sonra USDT gönderdi. Tutarlar benzer, ama USDT için iki kattan fazla “gas” alındı. Bu gas ne ve neden böyle?

İnceleyelim: iki tür hesap

Ethereum’da adresin, bankadaki gibi bir bakiyesi vardır. Bitcoin’deki parçalar gibi değil (14. bölüm). Hesaplar da iki türlüdür.



	Sıradan hesap	Sözleşme
Kim yönetir	anahtarın sahibi	kendi programı
Kendiliğinden transfer başlatabilir mi	evet	hayır, yalnızca çağrıldığında
Kodu var mı	genellikle yok	var

Programları **EVM**, yani Ethereum sanal makinesi çalıştırır. Her node’un içinde bir oyun konsolu var gibi düşünün: aynı program her yerde aynı sonucu verir.

Deneyelim: program nerede duruyor

`eth_getCode` metodu bir adresteki kodu gösterir. USDT sözleşmesini soralım ve yanıtın yalnızca başını alalım:

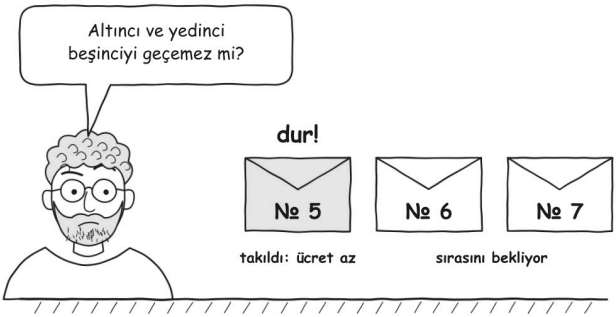
```
curl -s https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getCode","params":
  ["0xdac17f958d2ee523a2206206994597c13d831ec7","latest"]}' | head -c 80; echo
```

Uzun bir karakter dizisinin başını göreceksiniz: bu programın kendisi, 11 bin bayttan fazla. Adresi `0x0000000000000000000000000000000000000dEaD` ile değiştirin; `"0x"` gelir, yani kod yok.

2025'ten itibaren sıradan bir hesap da eylemlerini bir programa devredebiliyor (Pectra güncellemesi). O zaman yanıt `0xef0100` ile başlar. Yani "kod var" artık her zaman "bu bir sözleşme" demek değil.

Sıra numarası: nonce

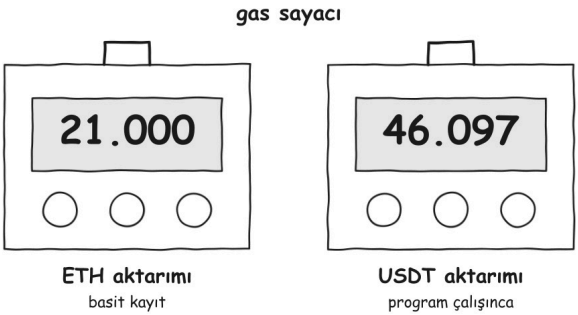
Hesabın her işlemi bir numara alır: 0, 1, 2 ve böyle devam eder. Buna **nonce** denir. Ağ işlemleri kesinlikle bu sırayla yürütür ve aynı numarayı ikinci kez kabul etmez.



Düşük ücretli bir transfer takılırsa sonrakiler onun arkasında bekler. Cüzdanlar "hızlandırmayı" önerir: aynı numarayla, daha yüksek ücretli yeni bir transfer gönderilir.

Gas: iş sayacı

Gas, işlemin ne kadar iş gerektirdiğini ölçer. Basit bir "ETH aktar" kaydı bugün 21.000 tutar. Token transferi bir programın çalıştırılmasıdır ve daha çok gas ister.



Ücret = harcanan gas × (taban fiyat + bahşiş)

- **Taban fiyatı** ağın kendisi belirler. Bu para **yakılır**.
- **Bahşişi** validatör alır: ne kadar yüksekse sizi o kadar istekle alırlar.
- Gas fiyatı **gwei** ile hesaplanır; bu, ETH'nin milyarda biridir.

Bir bloğa sınırlı miktarda gas sığar. Bloklar yarıdan fazla doluysa taban fiyat artar, daha az doluysa düşer. Örneğin 26.502.250 numaralı blok 60 milyon gas alıyordu ve %93 doluydu.

Cüzdan bir **gas sınırı** koyar. Yalnızca harcanan kadarı kesilir. Ama sınır yetmezse transfer gerçekleşmez ve harcanan gas geri verilmez.

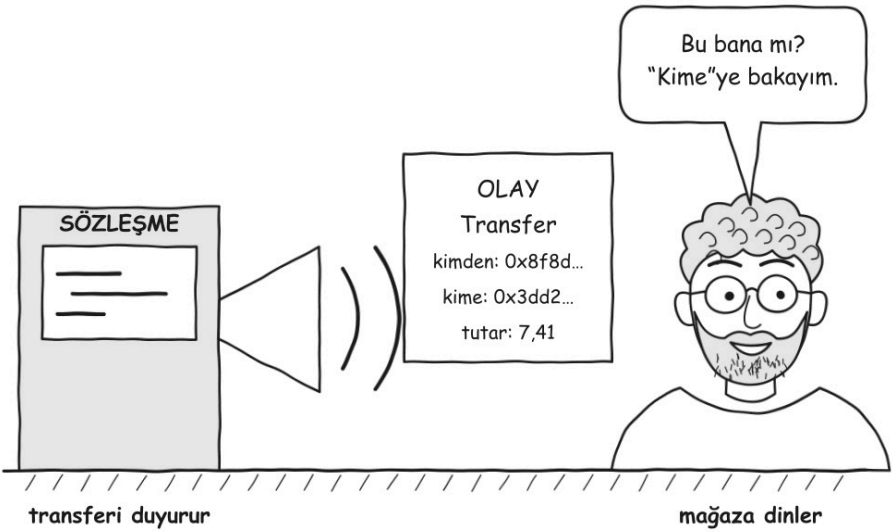
Gerçek bir USDT transferi

Bu, sözünü ettiğimiz bloktaki 7,41 USDT'lik bir transfer. Gönderen o ana kadar çoktan 423.464 işlem yapmıştı.

Ne	Değer
harcanan gas	46.097
taban fiyat	0,058 gwei
bahşiş	2 gwei
ücret	yaklaşık 0,000095 ETH
bunun yakılan kısmı	yaklaşık 0,000003 ETH

Olaylar: sözleşme transferi nasıl duyurur

Token bakiyeleri sözleşmenin içinde durur. Dünya transferden haberdar olsun diye sözleşme bir **olay** (log) duyurur. Tokenlerde bunun adı `Transfer`'dir: kimden, kime, ne kadar.



Olaylar, 7. bölümdeki işlem makbuzunda durur. Bu transferin makbuzunu isteyelim:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getTransactionReceipt","params":
  ["0xc0015ff5178d72a68831b133187c3f9f1fbee2e440d3af311d0b32110e2cf016"]}'
```

Yanıtta `logs` alanını bulun:

Alan	Başı ve sonu	Anlamı
`address`	`0xdac1…1ec7`	kim duyurdu: USDT sözleşmesi
`topics[0]`	`0xddf2…b3ef`	olayın türü: Transfer
`topics[1]`	`…8f8d…884e`	kimden
`topics[2]`	`…3dd2…a9c0`	kime
`data`	`…711150`	7.410.000, yani 7,41 USDT

`eth_getLogs` metodu ise olay arayabilir. En yeni bloktaki USDT transferlerini sayalım:

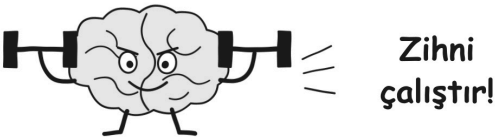
```
curl -s https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getLogs","params":[{
  "address":"0xdac17f958d2ee523a2206206994597c13d831ec7",
  "topics":["0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef"],
  "fromBlock":"latest","toBlock":"latest"}]}' | grep -o '"logIndex"' | wc -l
```

Tek bir bloktaki transfer sayısı çıkar. Herkese açık node geçmiş bloklar için yanıt vermez: bu artık arşivdir (10. bölüm).

Tuzak: "yalnızca olayları dinlerim"

- **ETH'nin kendisinin transferi Transfer olayı oluşturmaz.** Yalnızca olayları bekleyen mağaza böyle bir ödemeyi kaçırır. Ağın gelecekteki güncellemeleri bunu değiştirebilir.
- **ETH bir sözleşmeden gelebilir**, örneğin bazı cüzdanlardan. O zaman işlemin kendisinde adresiniz görünmez.
- **Olayı herhangi bir sözleşme duyurabilir.** `address` alanını kontrol edin: gerçek token mu (9. bölüm).

Düşünün



1. ETH transferi, 21.000 gas, bahşiş dahil fiyat 2 gwei. Ücret kaç gwei ve kaç ETH?
2. Art arda birkaç blok %30 dolu. Taban fiyata ne olur?
3. Mağaza yalnızca USDT sözleşmesinin Transfer olaylarını izliyor. Müşteri ETH ile ödedi. Mağaza bunu görür mü?

Cevaplar: 1) 42.000 gwei, yani 0,000042 ETH. 2) Düşer. 3) Hayır.

Özet

- Hesapları ya bir anahtar ya da bir program yönetir.
- Ücret, gas ile fiyatın (taban ve bahşiş) çarpımıdır.
- Token transferleri olaylardan öğrenilir; ETH'yi ise ayrıca yakalamak gerekir.

Sırada: 16. bölüm. Ethereum'un akrabası çok: diğer EVM ağları ve "ikinci katlar".

Bölüm 16. Ethereum'un akrabaları ve "ikinci katlar"

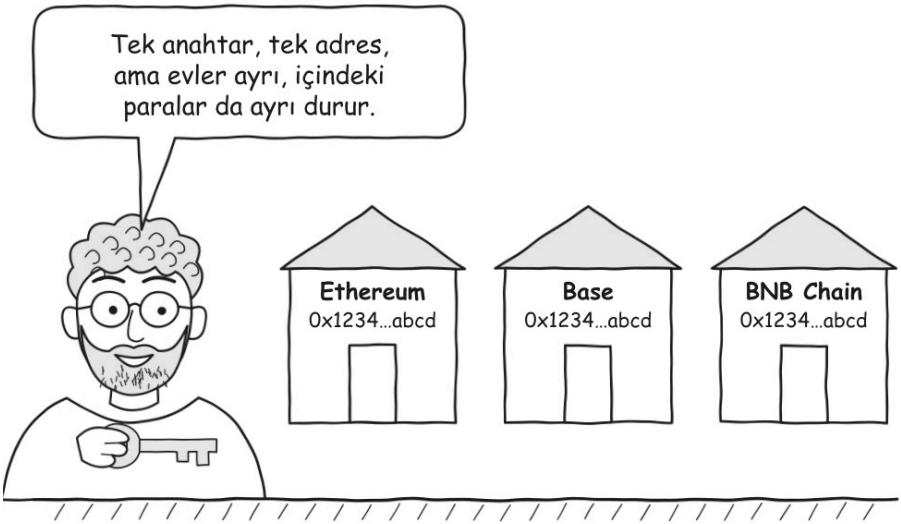
"Base'den gönderdim"

Müşteri yazıyor: "USDC'yi Base'den gönderdim, adres aynı." Artyom'un kafası karışıyor. Bu başka bir token mı? Başka bir ağ mı? Adres neden aynı?

İnceleyelim: aynı konsol, ayrı evler

15. bölümdeki EVM'yi pek çok ağ benimsedi. Bu yüzden adresleri aynı, `eth_…` metotları aynı ve tek bir sözleşme programı hepsinde çalışabiliyor. Böyle ağlara **EVM uyumlu** denir.

Ama defterleri **ayrıdır**. Kendi blokları, kendi bakiyeleri, kendi ücretleri vardır.



Ağlar karışmasın diye her birinin bir numarası vardır: **chain ID**. Bu numara imzaya girer, bu yüzden bir ağ için imzalanan transferi başka bir ağ kabul etmez.

Deneyelim: altı ağa numarasını soralım

```
for net in ethereum base arb bsc mantle monad; do
  echo -n "$net: "
  curl -s https://public-$net-mainnet.rpcnode.dev \
    -H 'Content-Type: application/json' \
    -d '{"jsonrpc":"2.0","id":1,"method":"eth_chainId","params":[]}'
  echo; sleep 1
done
```

On altılık sistemde sayılar gelir. İşte sıradan sayılar halinde, kontrol sırasında blokların ne sıklıkta çıktığıyla birlikte:

Ağ	Chain ID	Blok yaklaşık kaç saniyede bir	Ücret neyle ödenir
Ethereum	1	12 sn	ETH
Base	8453	0,2 sn	ETH
Arbitrum One	42161	0,3 sn	ETH
BNB Smart Chain	56	0,45 sn	BNB
Mantle	5000	2 sn	MNT
Monad	143	0,3 sn	MON

Birinci tür akrabalar: bağımsız olanlar

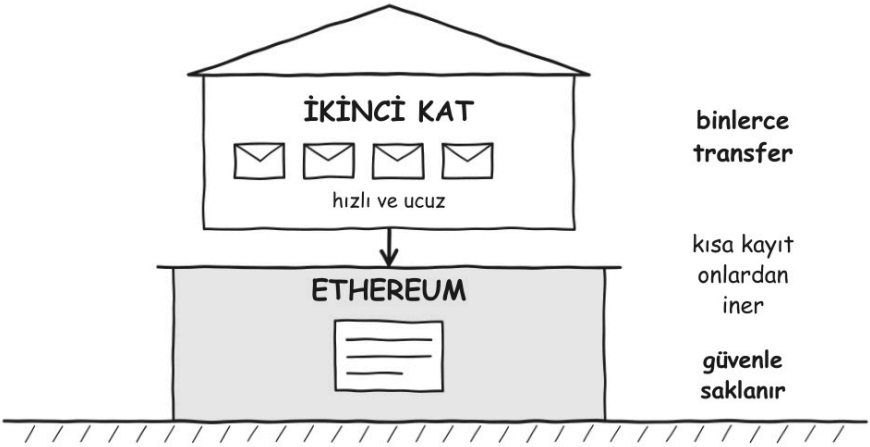
Kendi validatörleri vardır, bloklar üzerinde kendi aralarında anlaşırlar.



Polygon PoS bunun yanında Ethereum'a düzenli olarak kendi bloklarıyla ilgili işaretler bırakır. Kasım 2025'te çalışmaya başlayan Monad da bağımsızlar arasındadır.

İkinci tür akrabalar: ikinci katlar (L2)

L2, transferleri kendi üzerinde hızlı ve ucuz biçimde yürütür, bunların sıkıştırılmış kayıtlarını ise Ethereum'a gönderir. En yaygın türüne **rollup**, yani "dürüm" denir.



- Transferlerin sırasını genellikle tek bir işletmeci belirler: **sequencer** (sıralayıcı). Her şeyin bu kadar hızlı olması bundandır. Bu noktada ona güvenmeniz de bundandır.
- **İyimser** rollup'lar (Arbitrum One, OP Mainnet) kayıtları doğru sayar ama itiraz için süre tanır.
- **ZK** rollup'ları (örneğin Mantle) matematiksel bir kanıt ekler.
- Base 2026'da kendi koduna geçti ve birkaç kanıt türünü birlikte kullanıyor.

Denetim süresi yüzünden parayı resmi köprüyle Ethereum'a hızlıca geri almak mümkün değildir:

Hangi ağdan	Ethereum'a çekim için bekleme süresi
Arbitrum One	en az 7 gün
OP Mainnet	7 gün
Base	5 gün, bazı durumlarda 1 gün

Ağların 27 Kasım 2026 tarihli belgelerine göre. Bu süreler değişir: OP Mainnet'te Ocak 2027 için bir değişiklik planlanıyor.

"Hemen gördüm" demek "kesinleşti" demek değil

8. bölümdeki `latest`, `safe` ve `finalized` sözcüklerini hatırlıyor musunuz? Base node'u şöyle yanıt verdi:

Ne soruldu	En yeni bloğa göre gerilik
`latest`	—
`safe`	yaklaşık 17 saniye
`finalized`	yaklaşık 19 dakika

Blok saniyenin kesirlerinde ortaya çıkar; ona ait kayıt Ethereum'un kendisinde kesinleştiğinde ise kesin olur.

İkinci katta ücret

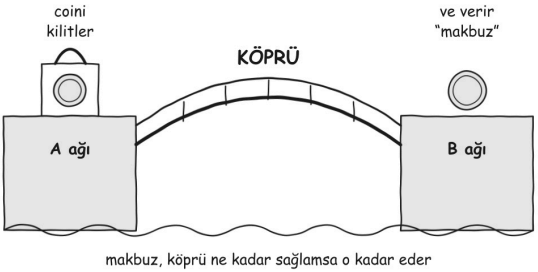
Esas olarak iki parçadan oluşur: L2'nin kendisindeki iş ve Ethereum'a yazma bedeli. Küçük ekler de olabilir. Base makbuzunda ikinci parça `l1Fee` alanındadır:

```
curl https://public-base-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getTransactionReceipt","params":
  ["0xa1d4398ce62b570a2810ad2b2ed347f388f9ad85aee4b62ba3215917cf3ac7e5"]}'
```

Burada Base'deki iş için yaklaşık 0,0000035 ETH, Ethereum'a yazma için yaklaşık 0,0000000006 ETH alınmış. Her zaman bu kadar ucuz olmaz.

Köprüler

Bir ağdan diğerine "taşınmak" için **köprü** kullanılır. Köprü çoğu zaman coinleri bir ağda kilitler, diğerinde onların yerini tutan bir token çıkarır.



makbuz, köprü ne kadar sağlamsa o kadar eder

Bu yüzden **USDC** ile **USDC.e**, sözleşme adresleri farklı iki ayrı tokendir. İlkini Circle şirketi kendisi çıkarır, ikincisi köprü yoluyla ortaya çıkar.

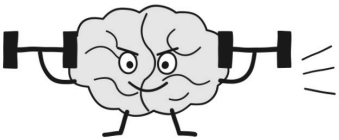
Tuzak: yanlış ağdan gönderildi

6. bölümde mağazanın böyle bir ödemeyi görmeyeceğini söylemiştik. Doğru. Peki para kayboldu mu?

- **Adresin anahtarı sizde** — para genellikle alınabilir: cüzdanı o ağa geçirin. Parayı başka yere göndermek için ücret olarak o ağın coini gerekir.
- **Bu bir borsa adresi** — anahtar sizde değil. Borsa o ağı desteklemiyorsa para büyük olasılıkla geri alınamaz.
- **Bu bir sözleşme cüzdan** — diğer ağda var olmayabilir. İşi bilen birinin yardımı gerekir.

> **Kural:** gizli ifadeyi kimseye söylemeyin; transferi "kurtaracağına" söz verene bile (6. bölüm).

Düşünün



1. Mağaza Ethereum'da USDT bekliyordu, para ise BNB Smart Chain'de aynı adrese geldi. Anahtar Artyom'da. Ne yapmalı?
2. Base'de blok 0,2 saniyede çıktı. Büyük bir ödeme kesinleşmiş sayılabilir mi?
3. Müşteri USDC yerine USDC.e gönderdi. Bu aynı şey mi?

Cevaplar: 1) cüzdanı BNB Smart Chain'e geçirmeli; parayı göndermek için BNB gerekir. 2) Hayır, finalized'ı bekleyin. 3) Hayır, bu başka bir token: kabul edip etmeyeceğine mağaza karar verir.

Özet

- EVM ağları benzer ama ayrı defterlerdir: chain ID'yi kontrol edin.
- Bağımsız ağlar da vardır, Ethereum'un üstündeki ikinci katlar da.
- Hızlı blok henüz kesinleşmiş değildir.

Sırada: 17. bölüm. TRON: gas yerine enerji ve stablecoinlerin gözde yolu.

Bölüm 17. TRON: gas yerine enerji ve stablecoinlerin gözde yolu

"TRC-20 ile olur mu?"

Müşteri soruyor: "TRC-20 ile USDT kabul ediyor musunuz?" Artyom bunu bu hafta üçüncü kez duyuyor. TRC-20 nedir ve neden herkes onu istiyor?

İnceleyelim: bağımsız bir ağ

TRON, kendi coini **TRX** olan ayrı bir blokzincirdir. Coinin en küçük parçasına **sun** denir: bir TRX'te bir milyon sun vardır.

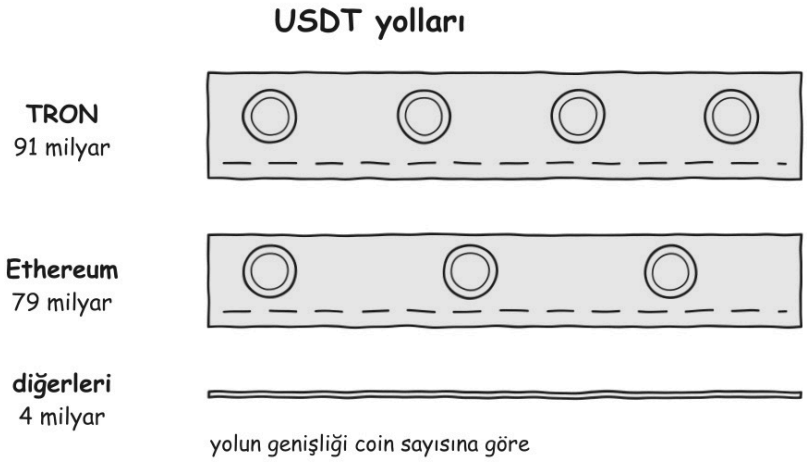
TRON'un yapısı pek çok yönden Ethereum'a benzer: hesaplardaki bakiyeler, akıllı sözleşmeler, olaylar. Programları kendi sanal makinesi çalıştırır; 15. bölümdeki EVM'nin yakın akrabasıdır.

TRC-20, TRON'da tokenlerin yapılma kurallarıdır; Ethereum'daki ERC-20 standardının kardeşi. "TRC-20 ile USDT", TRON ağındaki USDT tokeni demektir.

Bloklar 3 saniyede bir çıkar. Onları seçilmiş 27 temsilci yazar (8. bölüm).

Stablecoinlerin gözde yolu

USDT pek çok ağda yaşar ama çok dengesiz dağılır. İşte bu coini çıkaran Tether şirketinin kendi rakamları, 27 Kasım 2026'da açıldı:



Bütün USDT'lerin yarısından fazlası TRON'da. Müşterilerin bu ağı bu kadar sık seçmesi bundan.

Kaynak: app.tether.to/transparency.json. İhraç edilen coinlerden henüz dolaşıma sokulmamış yedek çıkarılarak hesaplandı.

Adres T ile başlar

İçeride TRON adresi, Ethereum'daki gibi 20 bayttır ve başında `41` ön eki vardır. İnsanlara ise farklı gösterilir, `T` harfiyle başlayarak:

İnsanların gördüğü	İçerideki hali (USDT sözleşmesinin adresi)
`TR7NHqjeKQxGTCi8q8ZY4pL8otSzgjLj6t`	`41a614f803b6fd780986a42c78ec9c7f77e6ded13c`

`T` ile yazılan biçimin içinde bir sağlama toplamı gizlidir: yazım hatası fark edilebilir. Node'a gerçek adresi soralım, sonra son harfini değiştirelim:

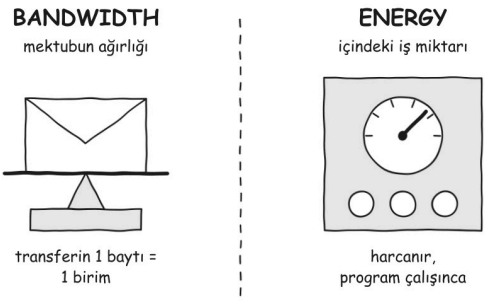
```
curl https://public-tron-mainnet.rpcnode.dev/wallet/validateaddress \
  -d '{"address":"TR7NHqjeKQxGTCi8q8ZY4pL8otSzgjLj6t","visible":true}'

curl https://public-tron-mainnet.rpcnode.dev/wallet/validateaddress \
  -d '{"address":"TR7NHqjeKQxGTCi8q8ZY4pL8otSzgjLj6u","visible":true}'
```

```
{"result":true,"message":"Base58check format"}
{"result":false,"message":"Invalid address"}
```

`"visible":true`, "adresleri T'li biçimde yazıyorum" demektir.

Gas yerine iki ayrı bedel

7. bölümde TRON'un transferin "ağırlığını" nasıl hesapladığını anlatacağımıza söz vermiştik. İki ölçüyle hesaplar.



- **Bandwidth** — kapladığı yer için. Harcama yaklaşık olarak transferin bayt cinsinden boyutuna eşittir. Her transfer öder.

- **Energy** (enerji) — programın işi için. Bir sözleşme çağrıldığında, dolayısıyla USDT transferinde de gerekir.

Üzerine en az bir kez TRX gelmiş adresin bir hediyesi vardır: **günde 600 birim bandwidth**. Harcanan kısım 24 saat içinde yavaş yavaş yenilenir. Ücretsiz enerji yoktur.

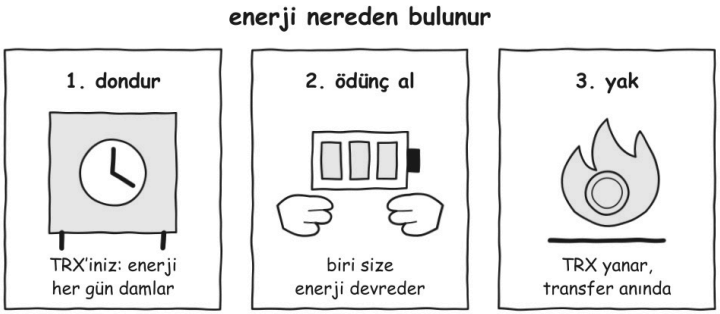
Bir şey yetmezse ağ TRX alıp yakar. Kaça? Node'a kendimiz soralım:

```
curl -s https://public-tron-mainnet.rpcnode.dev/wallet/getchainparameters |
  python3 -c "import json,sys
for p in json.load(sys.stdin)['chainParameter']:
    if p['key'] in ('getFreeNetLimit','getTransactionFee','getEnergyFee'):
        print(p['key'], p['value'])"
```

```
getTransactionFee 1000
getEnergyFee 100
getFreeNetLimit 600
```

Yani bir birim bandwidth 1000 sun, bir birim enerji 100 sun eder. Bu fiyatlar temsilcilerin oylamasıyla değiştirilir.

Enerji nereden bulunur

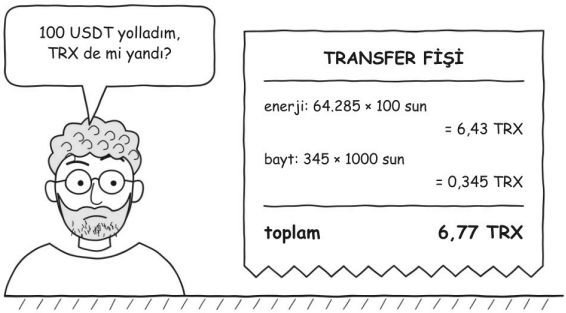


- **TRX dondurmak** (İngilizcesi staking). Coinler sizin kalır, adres de her gün toplam enerjiden bir pay alır. Geri almak için dondurma çözüldükten sonra 14 gün beklenir.
- **Ödünç almak**: dondurulmuş TRX'in sahibi enerjisini başka bir adrese devreder.
- **Hiçbir şey yapmamak**: eksik kısmı yakılan TRX öder.

Ne kadar dondurmalı? Bu, diğer herkesin ne kadar coin dondurduğuna bağlıdır. Kontrol günü, günde bir USDT transferi için yaklaşık 6600 dondurulmuş TRX yeterdi.

Gerçek bir transferin makbuzu

İşte 88.352.467 numaralı bloktan 100 USDT'lik bir transfer. Gönderenin ne enerjisi ne de ücretsiz bandwidth'i vardı:



Makbuz `gettransactioninfobyid` metoduyla alınır. Herkese açık node'da bu metot kapalıdır, kişisel uç nokta gerekir; o yüzden bu bir taslak:

```
curl https://UÇ_NOKTANIZ/walletsolidity/gettransactioninfobyid \
  -d '{"value":"0eb7b372f5a268ce56430392de8db2fd970455bb0970c67b47a7b2673abb513c"}'
```

Yanıttaki alan	Değer	Anlamı
`energy_usage_total`	64.285	toplam harcanan enerji
`energy_penalty_total`	49.635	bunun popülerlik zammı olan kısmı
`energy_fee`	6.428.500	enerji için yakılan, sun cinsinden
`net_fee`	345.000	baytlar için yakılan, sun cinsinden
`fee`	6.773.500	toplam, yani 6,7735 TRX
`result`	`SUCCESS`	program başarıyla çalıştı

Bu zam ne? Bir sözleşme çok fazla kullanıldığında ağ onun çağrılarını pahalılaştırır. O gün USDT'de, zam olmasa harcanacak enerjinin neredeyse 4,4 katı harcanıyordu.

Bir şey daha: alıcının daha önce hiç USDT'si yoksa iki kat enerji gerekir, 130.285. Sözleşme ona yeni bir satır açmak zorundadır.

Yeni blok ve kesinleşmiş blok

TRON node'larında iki adres takımı vardır. `/wallet/…` en yeni verilerle yanıt verir. `/walletsolidity/…` yalnızca kesinleşmiş verilerle. Bir blok, 27 temsilcinin 19'u o yükseklikte veya daha yukarıda kendi bloklarını çıkardığında kesinleşir. Bu genellikle yaklaşık bir dakika sürer.

Kontrol günü en yeni blok 88.352.763, kesinleşmiş olan ise 88.352.746 idi.

Deneyler için Nile test ağı da var; onun da herkese açık bir node'u bulunuyor: `public-tron-nile.rpcnode.dev`.

Tuzak: "blokta olduğuna göre gelmiştir"

Son kesinleşmiş bloktaki doğrudan USDT transferlerine bakalım:

```
curl -s https://public-tron-mainnet.rpcnode.dev/walletsolidity/getnowblock > block.json
python3 <<'EOF'
import json
usdt = '41a614f803b6fd780986a42c78ec9c7f77e6ded13c'
rows = []
for t in json.load(open('block.json'))['transactions']:
    v = t['raw_data']['contract'][0]['parameter']['value']
    data = v.get('data', '')
    if v.get('contract_address') == usdt and data.startswith('a9059cbb'):
        amount = int(data[72:136], 16) / 1_000_000
        rows.append((t['txID'][:8], t['ret'][0]['contractRet'], amount))
print('USDT transferi:', len(rows))
print('gerçekleşmeyen:', sum(r[1] != 'SUCCESS' for r in rows))
for r in rows[:3]:
    print(*r)
EOF
```

Sizde sayılar farklı olacak. 88.352.467 numaralı blokta ise böyle 41 transferden ikisi gerçekleşmemişti. Biri 100.000 USDT'likti:

```
fbfec9fa OUT_OF_ENERGY 100000.0
```

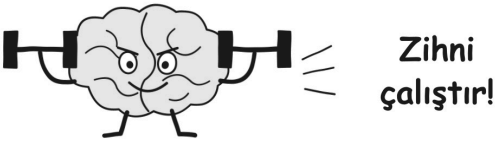
`OUT_OF_ENERGY` : enerji yetmedi, program durdu, USDT hiçbir yere gitmedi. Ama blokta kayıt var ve ekran görüntüsünde tutar pek heybetli duruyor.

Ne kadar enerji harcanabileceğini gönderen belirler: `fee_limit` alanı, sun cinsinden sınır. Sınır yetmezse transfer gerçekleşmez ve yakılan geri gelmez.

Mağazanın aklında kalması gerekenler

- Kesinleşmiş bloğu bekleyin ve makbuzdaki `result` alanını kontrol edin.
- Makbuzda `TR7NHq…Lj6t` sözleşmesinin `Transfer` olayını arayın. Böylece başka programlardan yapılan transferleri de görürsünüz. USDT'nin virgülden sonra altı basamağı vardır.
- Kendi adresinizden USDT göndermek için o adreste TRX veya enerji gerekir. Her siparişe ayrı adres kullanışlıdır, ama yüz adresten para toplamak ücreti ödenen yüz transfer demektir.
- Yeni bir adrese yapılan ilk TRX transferi gönderene en az 1 TRX daha pahalıya gelir.

Düşünün



1. Adreste 600 ücretsiz bandwidth ve bol enerji var. Art arda 345'lik iki transfer gönderiyor. İkincisine ne olur?
2. Enerji yok, alıcının da daha önce hiç USDT'si olmamış: birimi 100 sun olan 130.285 birim gerekiyor. Enerji için kaç TRX yanar?
3. Müşteri hash gönderdi. İşlem blokta, içinde `OUT_OF_ENERGY` yazıyor. Kahve gönderilmeli mi?

Cevaplar: 1) ücretsiz hakkın 255'i kaldı, 345'e yetmez; 345'in tamamı için 0,345 TRX yanar, parçalar birleştirilmez. 2) 13,0285 TRX. 3) Hayır, transfer gerçekleşmedi.

Özet

- TRON'da iki ölçü vardır: baytlar için bandwidth, programın işi için energy.
- Enerji yoksa TRX yanar; üstelik USDT için, basit bir gönderime göre belirgin biçimde daha çok.
- Ödeme, gerçek USDT'nin kesinleşmiş bir bloktaki başarılı transferidir.

Sırada: 18. bölüm. Solana: her şey hesaplara dağıtılmış.

Bölüm 18. Solana: her şey hesaplara dağıtılmış

"USDC'lerim nerede?"

Müşteri USDC ile ödedi. Artyom node'a cüzdanının bakiyesini soruyor ve yalnızca SOL kırıntıları görüyor. USDC yok. Kandırıldı mı?

İnceleyelim: hesaplardan oluşan bir ağ

Solana, coini **SOL** olan bağımsız bir ağdır. En küçük parçasına **lamport** denir: bir SOL'da bir milyar lamport vardır.

Solana adresi, harf ve rakamlarla yazılmış 32 bayttır. Özel bir başlangıcı yoktur (6. bölüm), içinde sağlama toplamı da yoktur.

Ağın sakladığı her şey **hesaplarda** (accounts) durur. Buradaki hesap yalnızca para demek değil, adresi olan herhangi bir göz demektir. Ethereum'da sözleşme verilerini kendi içinde tutar. Solana'da ise **program ayrı, veriler ayrı durur**.



İngilizcesi: wallet, program, mint account, token account

Her hesabın sahibi bir program vardır (`owner` alanı). Hesabın verilerini yalnızca o değiştirebilir.

Deneyelim: tokenin kimliği

Her tokenin bir kimlik hesabı vardır; ona **mint** denir. USDC'nin kimliğini soralım:

```
curl https://public-solana-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"getAccountInfo","params":
  ["EPjFWdd5AufqSSqeM2qN1xzybapC8G4wEGGkZwyTDt1v",{"encoding":"jsonParsed"}]}'
```

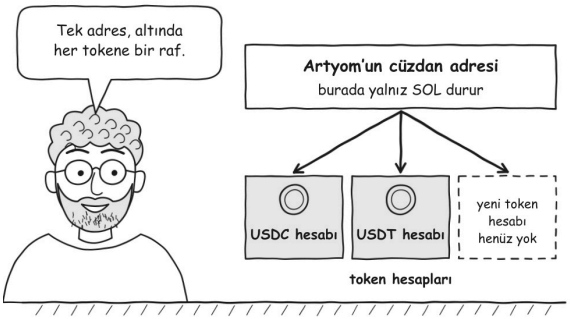
Yanıtın önemli kısımları:

Alan	Değer	Anlamı
`owner`	`Tokenkeg…Q5DA`	hesabı token programı yönetiyor
`supply`	`8513455604386747`	ihraç edilen miktar, küçük parçalarla
`decimals`	6	demek ki yaklaşık 8,5 milyar USDC
`mintAuthority`	`BJE5…5ruG`	yenilerini kimin basabileceği
`freezeAuthority`	`7dGb…Crar`	kimin dondurabileceği (9. bölüm)

USDC'nin kendine ait bir programı yoktur. Tek bir ortak token programı binlerce tokene hizmet eder; tokenler kimlikleriyle ayrılır. Böyle iki ortak program vardır: sıradan **Token** (USDC ve USDT buradadır) ve ek imkânları olan daha yeni **Token-2022**.

Token hesabı: ayrı bir raf

Artyom'un ne kadar USDC'si olduğu cüzdanda veya kimlikte değil, **token hesabında** yazılıdır. Her token için ayrı bir tane vardır.



Asıl token hesabına **ATA** (associated token account) denir. Adresi üç şeyden hesaplanır: cüzdan adresi, token kimliği ve token programı. Bu yüzden müşteriye **cüzdan adresi** verilir, rafı ise onun programı kendisi bulur.

Gerçek bir token hesabına bakalım. Önce üzerinde ne kadar token olduğunu, sonra ne kadar SOL olduğunu soralım:

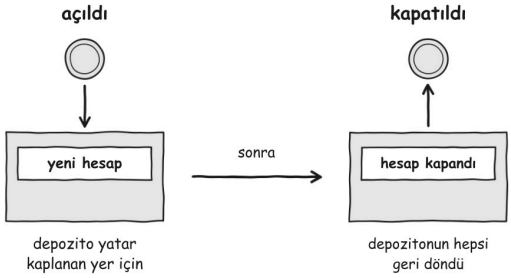
```
curl https://public-solana-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"getTokenAccountBalance","params":
  ["BnGihgNUu4X1JdBMNhbWnXUcpib8UQQC154isHTYwDNK"]}'

curl https://public-solana-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"getBalance","params":
  ["BnGihgNUu4X1JdBMNhbWnXUcpib8UQQC154isHTYwDNK"]}'
```

İlk yanıttaki `"uiAmountString":"207637.336362"` USDT miktarıdır. İkincideki `"value":2039280` lamport cinsindendir, toplam 0,00203928 SOL. `getBalance` metodu **tokenleri görmez**. Artyom'un "kaybolan" USDC'leri işte oraya gitti.

Bu arada, bu rafın sahibi USDT kimliğinin adresi. İnsanlar 9. bölümdeki gibi yanlışlıkla "USDT adresine" USDT göndermiş gibi görünüyor.

Yer için depozito

Her hesap bütün node'larda bellek kaplar. Bu yüzden üzerinde SOL olarak bir **depozito** durur (İngilizcesi rent, yani "kira", ama kira değildir).



Token hesabının boyutu 165 bayttır. Bu kadarı için ne kadar depozito gerekir?

```
curl https://public-solana-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,
  "method":"getMinimumBalanceForRentExemption","params":[165]}'
```

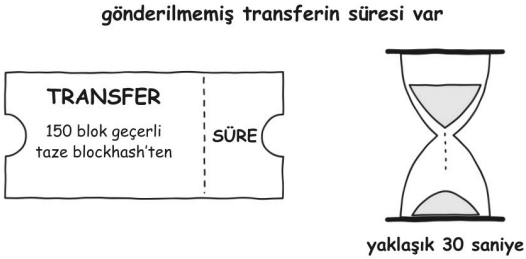
```
{"jsonrpc":"2.0","result":1488440,"id":1}
```

Bu yaklaşık 0,0015 SOL. 2026'da depozito aşama aşama düşürülüyor; bu yüzden programa sayı yazılmaz, sorulur. Alıcının o token için henüz rafı yoksa gönderen genellikle aynı transferde rafı oluşturur ve depozitoyu kendisi yatırır.

Ücret ve son kullanma süresi

Taban ücret **her imza için 5000 lamporttur**: tek imzalı sıradan bir transferde bu 0,000005 SOL eder. Yarısı yakılır. Üstüne öncelik için bir ödeme eklenebilir; onun tamamını validatör alır. Transfer başarısız olsa bile ücret alınır.

Her transfere bir **blockhash**, yani yakın tarihli bir bloğun parmak izi konur. Son kullanma süresini de o belirler.




```
curl https://public-solana-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"getLatestBlockhash",
  "params":[{"commitment":"finalized"}]}'

curl https://public-solana-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"getBlockHeight",
  "params":[{"commitment":"finalized"}]}'
```

İlk yanıtta `"lastValidBlockHeight":452330099`, bir saniye sonraki ikincide ise 452.329.960 yüksekliği geldi. Geriye 139 blok kalmış. Süre dolduysa ve transfer bloklarda yoksa artık geçmez. O zaman yenisi imzalanabilir. İstisna, özel ertelenmiş transferlerdir (durable nonce); onlarda böyle bir süre yoktur.

2026: daha da hızlı

Ne değişti	Önce	Sonra
Slot uzunluğu, yani bir blokluk sıra (Ekim)	0,4 sn	0,2 sn
Kesinlik, yeni oylama düzeni Alpenglow (Ekim)	yaklaşık 13 sn	yaklaşık 0,15 sn
En büyük işlem boyutu, yeni v1 biçimi (Eylül)	1232 bayt	4096 bayt

Hızı kendimiz ölçelim. Bu metot bir dakikada kaç slot geçtiğini döndürür:

```
curl https://public-solana-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"getRecentPerformanceSamples","params":[1]}'
```

Bize 60 saniye için `numSlots` = 277 geldi: slot başına yaklaşık 0,22 sn. Slotlar boş kalabilir, bu yüzden slot numarası ile blok yüksekliği farklı sayılardır; karıştırmayın.

Node farklı katılık düzeylerinde yanıt verebilir: `processed` — blok görüldü ama hâlâ atlanabilir; `finalized` — kesinleşti. Bir de ortadaki `confirmed` sözcüğü var: Alpenglow'dan sonra `finalized` ile aynı anlama geliyor. Sipariş `finalized` üzerine gönderilir.

Tuzak: "işlem okunmuyor"

Yeni v1 biçimi yüzünden `getTransaction` ve `getBlock` metotlarına, değeri 1 olan `maxSupportedTransactionVersion` eklemek gerekiyor. Eklenmezse yeni işlemlerde hata döner. `getTransaction` metodu herkese açık node'da kapalıdır; bu yüzden bu, kişisel uç nokta için bir taslak:

```
curl https://UÇ_NOKTANIZ \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"getTransaction","params":
  ["TRANSFER_İMZASI",{"encoding":"jsonParsed","commitment":"finalized",
  "maxSupportedTransactionVersion":1}]}'
```

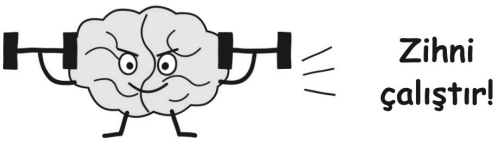
Solana'da "transfer imzası", işlem hash'i yerine transferin numarası görevini görür.

Mağaza neyi kontrol eder

- İşlem başarılı: `meta.err` değeri `null`.
- **Sizin cüzdanınızın**, doğru token kimliğine ait token hesabı doğru tutar kadar arttı: `preTokenBalances` ile `postTokenBalances` alanlarını karşılaştırın.
- Ödeme tam da bu siparişe ait. Bunun için transfere bir **reference**, yani siparişe özel tek kullanımlık bir etiket adresi eklenir.
- Bu imza daha önce sayılmadı.

Sıradan SOL ile de dikkatli olun: yanlış adrese yapılan transfer yine de gerçekleşir. Böyle bir parayı büyük olasılıkla geri verecek kimse çıkmaz.

Düşünün



1. Cüzdan transferi imzaladı ama ağ onu iki dakika sonra gördü. Ne olur?
2. Mağaza USDC ödemesini yakalamak için kendi cüzdanına `getBalance` soruyor. Hata nerede?
3. Artyom, henüz böyle bir rafı olmayan tedarikçiye USDC gönderiyor. Ücretten başka ne ödemesi gerekecek?

Cevaplar: 1) süre dolmuştur, transfer geçmez. 2) `getBalance` *yalnızca SOL gösterir, USDC token hesabında durur. 3) Yeni token hesabının depozitosunu, yaklaşık 0,0015 SOL.*

Özet

- Solana'da her şey hesaptır; programlar ve verileri ayrı durur.
- Tokenler token hesaplarında durur: her token için ayrı ve depozitolu.
- İşlemleri `maxSupportedTransactionVersion` ile okuyun ve `finalized` bekleyin.

Sırada: 19. bölüm. TON: telefon yerine mektup.

Bölüm 19. TON: telefon yerine mektup

“Gönderdim, sonrası ne?”

Müşteri TON ağında USDT gönderdi. Artyom blok gezgininde tek bir kayıt yerine koca bir zincir görüyor: beş işlem, yabancı adresler ve küçük bir para üstü. Bu bir ödeme mi, beş mi?

İnceleyelim: herkes sözleşme, herkes mektup yazıyor

TON (The Open Network) bağımsız bir ağdır. Coininin adı **Gram** (GRAM). Eskiden Toncoin deniyordu ve bu ada hâlâ sık rastlanır. Bir Gram’da bir milyar **nanogram** vardır.

Nisan 2026’dan beri bloklar yaklaşık 0,4 saniyede bir çıkıyor, kayıt da yaklaşık bir saniyede kesinleşiyor.

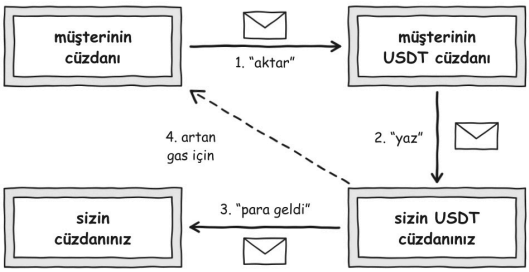
Asıl fark şu. Buradaki her hesap bir akıllı sözleşmedir, **cüzdan da öyle**. Sözleşmeler birbirini doğrudan çağırmaz, **mesaj** gönderir.

Ethereum’da tek bir transferin başlattığı her şey tek bir işlemin içinde olur: ya tamamı ya hiçbiri. Telefon görüşmesi gibidir. TON’da ise mektuplaşma vardır: mektubu gönderirsiniz, hatta beklemezsiniz. Her mektubu alıcı sonradan, ayrı bir işlemle ele alır. Bu düzene **eşzamansız** denir.

Tek bir talimattan doğan bütün yazışmaya **trace** (iz) denir. İçindeki bir işlemin başarılı olması, işin tamamının başarıldığı anlamına gelmez.

USDT nasıl gider

TON’da tokenlere **jetton** denir. Her birinin bir ana sözleşmesi (**master**), her sahibin de kendine ait ayrı bir **jetton cüzdanı** vardır. 18. bölümdeki token hesaplarına benzer, yalnız bu bağımsız bir sözleşmedir. USDT için ana sözleşme `EQCxE6mUtQJKFnGfaROTKOt1lZbDiiX1kCixRv7Nw2Id_sDs` adresidir.



her mektup ayrı bir işlemle ele alınır

Bu uydurma değil; 27 Kasım 2026'ya bağlayan gece yapılmış gerçek bir 11,99 USDT transferi:

İşlem kimin	Ne aldı	Ne gönderdi
gönderenin cüzdanı	imzalı talimat	"aktar" ve masraf için 0,05 Gram
onun USDT cüzdanı	"aktar"	"hesaba geçir"
alıcının USDT cüzdanı	"hesaba geçir"	bildirim ve para üstü
alıcının cüzdanı	bildirim ve 1 nanogram	hiçbir şey
gönderenin cüzdanı	para üstü, yaklaşık 0,0494 Gram	hiçbir şey

Hepsi bir saniyeye sığdı. Masraf parası bol tutularak eklenir, artanı geri döner.

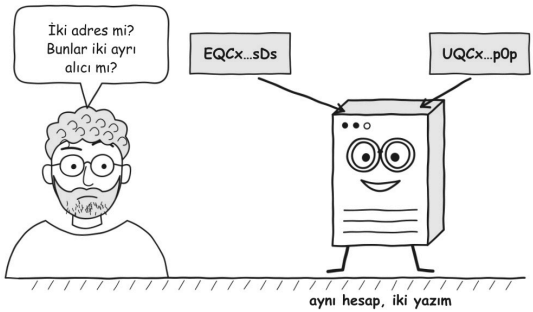
İlginç bir ayrıntı: dördüncü işlem yarıda kesilmiş (`aborted`) diye işaretli. Alıcının programı çalıştıracak gas'ı yetmemiş: gelen yalnızca bir nanogram. Oysa USDT daha üçüncü işlemde hesaba geçmiş. Tek bir kayda bakarak ne o yönde ne bu yönde hüküm verilebilir.

Zincirin tamamı indeksleyiciye istek göndererek görülebilir. Herkese açık node'da bu kapalıdır; o yüzden bu, kişisel uç nokta için bir taslak:

```
curl -G https://UÇ_NOKTANIZ/api/v3/traces \
  --data-urlencode "trace_id=7rg6DNoAZeR1KPVJHd1FWIc/dobTITAsXqK1Y2kuQlM="
```

Bir hesap, iki adres

Hesabın gerçek adı "ham" biçimdir: ağın hangi parçasında olduğunu gösteren numara ve 64 karakter. İnsanlara ise sağlama toplamı içeren 48 karakterlik başka bir yazım gösterilir. Üstelik tek bir hesabın böyle birkaç yazımı vardır.



Node'a USDT'nin adresini soralım:

```
curl -s "https://public-ton-mainnet.rpcnode.dev/api/v2/detectAddress?address=EQCxE6mUtQJKFnGfaROTKOt1lZbDiiX1kCixR
v7Nw2Id_sDs" |
  python3 -c "import json,sys
r = json.load(sys.stdin)['result']
print(r['raw_form'])
print(r['bounceable']['b64url'])
print(r['non_bounceable']['b64url'])"
```

```
0:b113a994b5024a16719f69139328eb759596c38a25f59028b146fecdc3621dfe
EQCxE6mUtQJKFnGfaROTKOt1lZbDiiX1kCixRv7Nw2Id_sDs
UQCxE6mUtQJKFnGfaROTKOt1lZbDiiX1kCixRv7Nw2Id_p0p
```

- `EQ…` — geri dönebilir (bounceable) adres; sıradan sözleşmeler böyle yazılır.
- `UQ…` — geri dönmez adres; cüzdanların böyle yazılması önerilir.
- Test ağında yazımlar `kQ` ve `0Q` ile başlar.

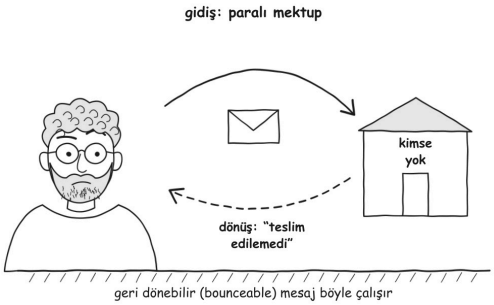
Bu yüzden mağazanın veritabanında **ham biçim karşılaştırılır**, `EQ` ve `UQ` dizileri değil.

Sağlama toplamı yazım hatalarını yakalar. Sondaki `sDs` yerine `sDt` yazılırsa node 422 koduyla yanıt verir:

```
curl -s -o /dev/null -w '%{http_code}\n' \
  "https://public-ton-mainnet.rpcnode.dev/api/v2/detectAddress?address=EQCxE6mUtQJKFnGfaROTKOt1lZbDiiX1kCixRv7Nw2I
d_sDt"
```

Boş eve giden mektup

Alıcı mektubu ele alamazsa veya sözleşmesi henüz yoksa ne olur? Bu, mesajın üzerindeki işarete bağlıdır.



Geri dönebilir mesaj, dönüş yoluna yetecek para varsa kalan parayla birlikte gönderene döner. Geri dönmez mesaj alıcıda kalır. Yeni bir cüzdan henüz "yayına alınmamış" olabilir, yani programı ağa yazılmamıştır. Cüzdanlara geri dönmez adresle ödeme yapılması bundandır.

Deneyelim: hızı ölçelim

En yeni ana bloğun numarasını ve hem onun hem de bin blok öncekinin çıkış zamanını öğrenelim:

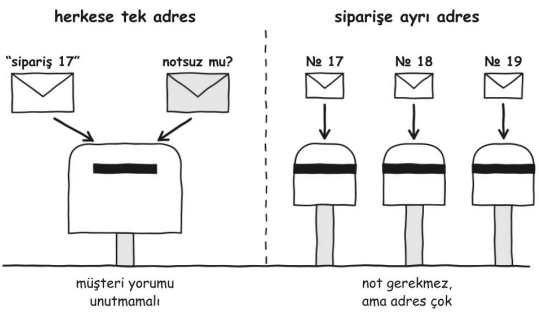
```
B=https://public-ton-mainnet.rpcnode.dev
last=$(curl -s $B/api/v2/getMasterchainInfo |
  python3 -c "import json,sys; print(json.load(sys.stdin)['result']['last']['seqno'])")
for s in $last $((last-1000)); do
  sleep 2
  curl -s "$B/api/v2/getBlockHeader?workchain=-1&shard=-9223372036854775808&seqno=$s" |
    python3 -c "import json,sys
r = json.load(sys.stdin)['result']
print(r['id']['seqno'], r['gen_utime'])"
done
```

```
108011078 1795732480
108010078 1795732076
```

İkinci sayılar saniye cinsinden zaman. 1000 blokta fark 404 saniye, yani blok başına yaklaşık 0,4 saniye.

Ayrı adres yerine yorum

TON'da transfere bir metin **yorumu** eklenebilir. Bu yüzden ödemenin kime ait olduğunu anlamanın iki yolu vardır:



Ortak adresin zahmeti azdır ama müşteri yorumu unutabilir veya yanlış yazabilir. Üstelik her cüzdan yorum eklemeyi bilmez. Böyle ödemeler elle incelenir.

Tuzak: "bildirim geldi, gönderiyorum"

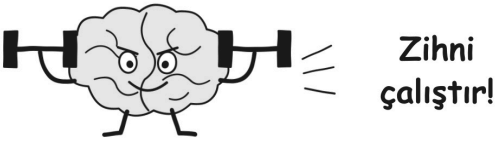
Herkes cüzdanınıza aynı bildirimi gönderen bir sözleşme yazabilir. Ya da adı "USDT" olan kendi jetton'unu çıkarabilir. Bu yüzden şunlar kontrol edilir:

- bildirim **sizin** USDT cüzdanınızdan gelmiş. Onun adresi önceden gerçek ana sözleşmeye sorulur (`get_wallet_address`);

- tutar tutuyor (USDT'nin virgülden sonra altı basamağı vardır);
- yorum veya adres bu siparişe ait ve bu ödeme henüz sayılmamış;
- kayıt ana zincire (masterchain) girmiş, yani kesinleşmiş.

Bildirim, ancak gönderen onun için en azından biraz Gram eklediyse gelir. Genellikle eklenir ama elde yapılmış bir program unutabilir. O zaman tokenler bildirimsiz gelir.

Düşünün



1. Veritabanında `UQ` ile başlayan bir adres var, blok gezgini ise ortası aynı olan `EQ` ile başlayan bir adres gösteriyor. Aynı hesap olup olmadığı nasıl anlaşılır?
2. Zincirdeki ilk işlem başarılı. Demek ki USDT ulaştı mı?
3. Müşteri ortak adrese ödedi ama yorumu unuttu. Mağaza ne yapmalı?

Cevaplar: 1) ikisini de ham biçime çevirip (`detectAddress`) karşılaştırın. 2) Henüz değil: alıcının USDT cüzdanının onu hesaba geçirmesi gerekir. 3) Kendiliğinden saymayıp elle incelemeli: tutarı, göndereni, transfer numarasını.

Özet

- TON'da sözleşmeler mektuplaşır: tek bir talimat bir işlem zinciri doğurur.
- Bir hesabın adresinin birkaç yazımı vardır; ham biçimi karşılaştırın.
- Jetton, ana sözleşmeye ve sizin jetton cüzdanınıza göre doğrulanır.

Sırada: 20. bölüm. Başka yaklaşımlar: Sui ve Aptos, XRP Ledger ve Stellar, Cardano, Polkadot, Cosmos.

Bölüm 20. Başka yaklaşımlar: Sui ve Aptos, XRP Ledger ve Stellar, Cardano, Polkadot, Cosmos

“XLM de alır mısınız?”

Bir müşteri Stellar üzerinden ödeyip ödeyemeyeceğini soruyor. Bir diğeri Sui’de USDC göndermek istiyor. Artyom Bitcoin ve Ethereum’u çoktan öğrendi. Bu bilgi yeter mi? Pek değil: burada düzen başka.

İnceleyelim: yeni bir ağa beş soru

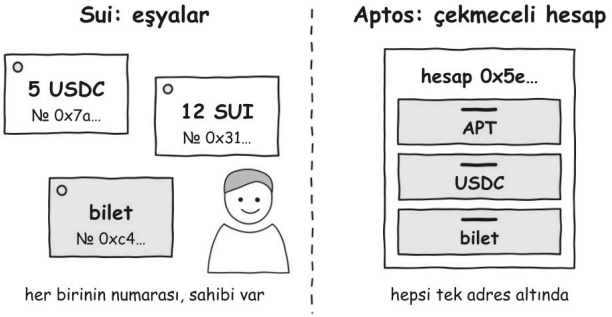
Tanımadığınız bir ağı baştan sona öğrenmeniz gerekmez. Başlangıç için beş cevap yeter:

1. Para nerede durur: hesapta mı, ayrı parçalar halinde mi?
2. Transfer alabilmek için önceden ne yapılmalı?
3. Gelen paranın hangi siparişe ait olduğu nasıl anlaşılır?
4. Ne zaman kesinleşir?
5. Parayı ileride başka yere göndermek gerektiğinde ücret neyle ödenir?

Sui ve Aptos: ortak dil, farklı yapı

İki ağda da programlar **Move** diliyle yazılır. Bu dil, değerli bir şeyin öylece kopyalanamamasını gözetir. Para ise farklı biçimlerde saklanır.

- **Sui** her şeyi **nesneler** halinde saklar: bunlar her birinin kendi numarası olan eşyalardır. Para, Bitcoin’deki parçalar gibi ayrı coin nesneleri halinde durabilir. 2026’dan itibaren adresin ortak bir bakiyesi de olabiliyor. Tam bakiye ikisinin toplamıdır.
- **Aptos** alışılmış hesaba daha yakındır. Mal varlığı adresin altında toplanır; ona **kaynak** (resource) denir. Önceden hesap açmak gerekmez.



İki ağ da transferi genellikle bir saniyeden kısa sürede kesinleştirir. Ücret kendi coiniyle ödenir: SUI veya APT. Sui'de bir istisna var: 2026'dan itibaren USDC'nin de aralarında olduğu bazı stablecoinler ücretsiz gönderilebiliyor (en az 0,01).

İkisinde de adres `0x` ve genellikle 64 karakterdir. Başı Ethereum'daki gibi ama uzunluğu farklı: orada `0x` ardından 40 karakter gelir.

Deneyelim: iki ağda USDC'yi bulalım

Bu bölümdeki ağlar için RpcNode'un ücretsiz herkese açık node'u yok. Bu yüzden bu ağları geliştirenlerin açık adreslerine soracağız. Bunlar ortak ve sınırlı adreslerdir: tanışmak için uygundur, mağaza için değil.

Sui; yöntemin adı GraphQL (hangi alanları istediğinizi sayarsınız):

```
curl -s https://graphql.mainnet.sui.io/graphql \
  -H 'Content-Type: application/json' \
  -d '{"query":"{ coinMetadata(coinType: \"0xdba34672e30cb065b1f93e3ab55318768fd6fef66c15942c9f7cb846e2f900e7::usd
c::USDC\") { symbol decimals } }"}'
```

```
{"data":{"coinMetadata":{"symbol":"USDC","decimals":6}}}
```

Aptos; 11. bölümdeki sıradan REST:

```
curl -s https://api.mainnet.aptoslabs.com/v1/accounts/0xbae207659db88bea0cbead6da0ed00aac12edcdda169e591cd41c94180
b46f3b/resource/0x1::fungible_asset::Metadata
```

```
{"type":"0x1::fungible_asset::Metadata","data":{"decimals":6,"icon_uri":"https://circle.com/usdc-icon","name":"USD
C","project_uri":"https://circle.com/usdc","symbol":"USDC"}}
```

İkisinde de token "USDC" sözcüğüyle değil, ihraççı Circle şirketinin listesindeki uzun adresle bulunuyor. Bu 9. bölümün kuralı. Virgülden sonraki basamak sayısı iki durumda da altı.

Eski örneklere dikkat. Eskiden Sui'ye JSON-RPC ile soru sorulurdu. Temmuz 2026'da Sui vakfının node'larında bu kapatıldı; artık orada gRPC ve GraphQL var. Bir makaledeki örneği kopyalamadan önce tarihine bakın.

XRP Ledger ve Stellar: transfer ağları

İki ağ da uzun zamandır yalnızca kendi coinini aktarmakla kalmıyor, başkalarının tokenlerini de tutabiliyor. Burada zincirin sayfasına **ledger** denir. XRP Ledger 3–5 saniyede bir, Stellar yaklaşık 5 saniyede bir yenisini onaylar. Onaylanan yeniden yazılmaz.

Ama alışılmadık iki kural var.

Birincisi: yer için teminat. Üzerinde asgari bir tutar, yani **rezerv** durmadıkça hesap yoktur. Bir tokeni, örneğin USDC'yi alabilmek için hesap sahibi önceden bir **güven hattı** (trust line) açar:

“evet, bunu tutmayı kabul ediyorum”. Böyle her kayıt rezervi artırır.

	XRP Ledger	Stellar
Adres şununla başlar	`r`	`G`
Hesapta asgari tutar	1 XRP	1 XLM
Her trust line için ek	0,2 XRP (ilk ikisi bundan muaf olabilir)	0,5 XLM
En düşük ücret	0,00001 XRP	işlem kalemi başına 0,00001 XLM
Sipariş numarası	destination tag	memo veya `M` ile başlayan adres

Ağ bu sayıları oylamayla değiştirebilir; bu yüzden node’a sorulurlar:

```
curl -s https://s1.ripple.com:51234/ -H 'Content-Type: application/json' \
  -d '{"method":"server_info","params":[{}]}' | grep -o '"validated_ledger":{[^}]*}'
```

```
"validated_ledger":{"age":10,"hash":"AD65FEE0E665A2CA5D08D3757A9E5A65047ECC2391B5E118F3912F25C76AB64F","seq":10859
3026,"base_fee_xrp":1E-5,"reserve_base_xrp":1E0,"reserve_inc_xrp":2E-1}
```

`1E-5` 0,00001; `1E0` 1; `2E-1` 0,2 demektir. Şimdi Stellar:

```
curl -s 'https://horizon.stellar.org/ledgers?order=desc&limit=1' \
  | grep -E '"(base_fee|base_reserve)_in_stroops"'
```

```
       "base_fee_in_stroops": 100,
       "base_reserve_in_stroops": 5000000,
```

Bir XLM’de on milyon **stroop** vardır. Demek ki ücret 0,00001 XLM, rezervin bir payı da 0,5 XLM (hesaba en az iki pay gerekir).

Herkese tek adres

İkinci kural. Burada her siparişe ayrı adres açmak masraflıdır: her birinde rezerv yatardı. Bu yüzden mağaza tek bir adres gösterir ve bir numara yazılmasını ister: XRP Ledger’da **destination tag**, Stellar’da **memo**.

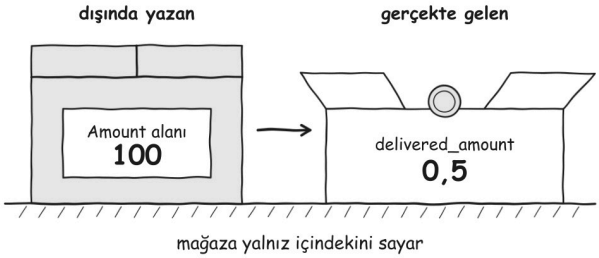


Müşteri numarayı unutursa para gelir ama ne için geldiği belirsiz kalır. Şunlar işe yarar:

- XRP Ledger'da `RequireDest` ayarı açılabilir: ağ numarasız transferi geçirmez;
- müşterinin cüzdanı anlıyorsa numara adresin kendisine gizlenebilir: XRP Ledger'da X adresi, Stellar'da `M` ile başlayan adres.

Tuzak: etikette 100, içinde 0,5

XRP Ledger'da gönderen **kısmi ödemeye** (partial payment) izin verebilir: daha azı ulaşsa bile transfer başarılı sayılır. O zaman `Amount` alanı "gelen" değil, "en fazla" demektir. Gerçekte ne kadar geldiği `delivered_amount` alanında yazılıdır.



İşte ağdan gerçek bir kayıt; hash'iyle soralım:

```
curl -s https://s2.ripple.com:51234/ -H 'Content-Type: application/json' \
  -d '{"method":"tx","params":[{"transaction":"18F797F64D0255522223B1A404BA9CFF7EAE28E2ABB74CC91DD986E162D42DE
C"}]}' \
  | grep -o '"Amount":{[^}]*}\|"delivered_amount":{[^}]*}\|"TransactionResult":"[^"]*"'
```

```
"Amount":{"currency":"XXX","issuer":"rPXeYRpA9Ne3H7fwcM26ZNe98M3rkzTeg6","value":"9.552474017069"}
"TransactionResult":"tesSUCCESS"
"delivered_amount":{"currency":"XXX","issuer":"rPXeYRpA9Ne3H7fwcM26ZNe98M3rkzTeg6","value":"4.5344917829"}
```

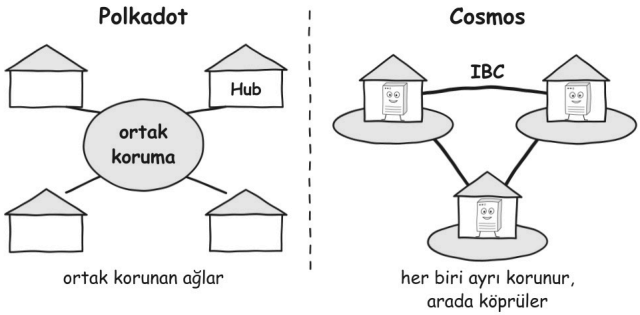
Sonuç `tesSUCCESS`, `Amount` yaklaşık 9,55, ulaşan ise yaklaşık 4,53. Böyle bir kayıt kendi başına dolandırıcılık değildir. Ama bir dolandırıcı büyük bir tutar yazıp kırıntı gönderebilir ve malı bekleyebilir. **Yalnızca `delivered_amount` değerini sayın**; tokenin hangisi olduğunu ve kimin ihraç ettiğini de kontrol edin.

Cardano: ekleri olan parçalar

Cardano'nun yapısı Bitcoin'e daha yakındır (14. bölüm): bakiye değil, harcanmamış parçalar vardır. Yalnız parça daha fazlasını yapabilir; modele **genişletilmiş UTXO** (eUTXO) denir.

- Parçada ADA coininin yanında başka tokenler de durur. Onlar için ayrı bir sözleşme gerekmez.
- Token tek başına yolculuk etmez: aynı parçada bir miktar ADA bulunmak zorundadır. Bu yüzden tokenle birlikte birazcık coin de gelir.
- Ücret, transferin bayt cinsinden boyutuna bağlıdır.
- Güven yavaş yavaş artar: üstte ne kadar çok blok varsa o kadar sağlamdır.

Polkadot ve Cosmos: tek ağ değil, pek çok ağ



Polkadot, kendisine bağlı ağları koruyan merkezi bir zincirdir. Önemli bir değişiklik: Kasım 2025'ten beri kullanıcı bakiyeleri artık merkezi zincirde değil, **Polkadot Hub** ağında (Asset Hub da denir) duruyor.

- USDT (1984 numaralı varlık) ve USDC (1337 numara) de orada.
- Hesabın bir yaşam eşiği vardır, **existential deposit**: 0,01 DOT. Ne yeterli DOT ne de yeterli miktarda USDT veya USDC varsa hesap silinir.
- Adres, Bitcoin'deki eski adresler gibi `1` ile başlar. Karıştırmayın.

Cosmos, ortak parçalardan kurulmuş bağımsız ağlardır. Her birinin kendi güvenliği, coini ve adres ön eki vardır (Cosmos Hub'da `cosmos1…`). Bloklar hızla kesinleşir. Transferde sipariş numarası için bir **memo** alanı bulunur.

Ağları **IBC** protokolü bağlar. Komşu ağdan gelen tokenin adı alışılmış sözcük değil, `ibc/` ve uzun bir mühürdür, örneğin `ibc/7F1D3FCF…`. Mühür, tokenin hangi yoldan geldiğini söyler.

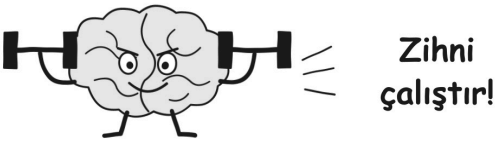
Stablecoin de çekip gidebilir. Circle, Cosmos ailesinden Noble ağında USDC ihracını sonlandıracağını duyurdu: yeni ihraç 13 Ekim 2026'dan beri kapalı, 12 Ocak 2027'de ise oradaki USDC durdurulacak. Elinde bulunduranlara önceden başka yere aktarmaları öneriliyor.

Hepsi bir arada

Ağ	Adres	Önceden gereken	Kesinleşme
Sui	`0x…`	hiçbir şey	bir saniyeden kısa
Aptos	`0x…`	hiçbir şey	bir saniyeden kısa
XRP Ledger	`r…`	rezerv, trust line	3–5 saniye
Stellar	`G…`	rezerv, trust line	yaklaşık 5 saniye
Cardano	`addr1…`	hiçbir şey	yavaş yavaş
Polkadot Hub	`1…`	eşiğin üstünde bakiye	işaretle
Cosmos Hub	`cosmos1…`	hiçbir şey	hızlı

Trust line yalnızca tokenler için gerekir. 27 Kasım 2026'da kontrol edildi; kurallar değişir.

Düşünün



1. Artyom Stellar'da yeni, boş bir adres oluşturdu ve hemen USDC faturası kesti. Sorun ne ve en az kaç XLM gerekir?
2. Bir transfer geldi: `Amount` 100 USDC, `delivered_amount` 0,5, sonuç `tesSUCCESS`. 100'lük kahve gönderilmeli mi?
3. XRP Ledger'da müşteriler destination tag'i unutuyor. Ne açılmalı?

Cevaplar: 1) hesap henüz yok ve trust line açılmamış; en az 1,5 XLM rezerv ve ücret için de bir miktar gerekir. 2) Hayır: 0,5 geldi. 3) RequireDest ayarı.

Özet

- Müşteriye adres göstermeden önce tanımadığınız ağa beş soru sorun.
- XRP Ledger ve Stellar'da rezerv, trust line ve sipariş numarası gerekir; XRP Ledger'da yalnızca `delivered_amount` değerini sayın.
- Ethereum alışkanlıkları ve eski örnekler başka bir ağa taşınmaz.

Sırada: 21. bölüm. Hangi ağı seçmeli.

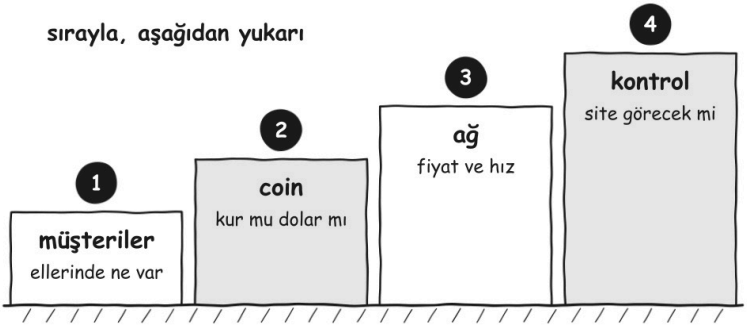
Bölüm 21. Hangi ağı seçmeli

"Bunca ağdan hangisi bana göre?"

Yedi bölümde pek çok ad birikti: Bitcoin, Ethereum, TRON, Solana, TON ve on tane daha. Artyom'un bir internet mağazası ve bir kahve kavurma atölyesi var. Hepsini mi bağlamalı? Birini mi seçmeli? Neye göre?

İnceleyelim: dört adım

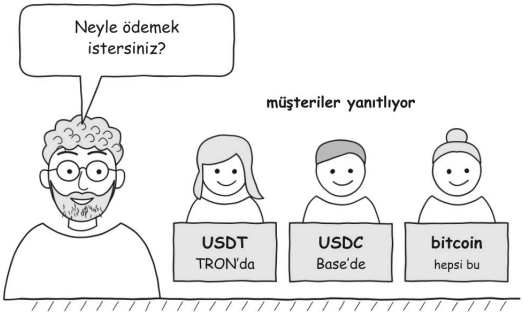
"En iyi ağ" yoktur. Müşterilerinize ve siparişinize uyan ağ vardır. Sırayla ilerlemek kolaylık sağlar.



Burada konu, ödemeyi neyle almanın kullanışlı olduğu. Bu, paranızı neye yatıracağınıza dair bir tavsiye değildir.

1. adım. Müşterilere sorun

Elinde zaten coin olan öder. Coinleri bir ağdaysa ve siz başka bir ağda kabul ediyorsanız ödeme büyük olasılıkla gelmeyecektir.



E-posta bülteninde, sohbette, sipariş sırasında sorun. İki soru yeter: “Elinizde ne var?” ve “Hangi ağda?”

2. adım. Hangi coin

	Ağın kendi coini	**Stablecoin (USDT, USDC)**
Dolar fiyatı	değişir, bazen sert biçimde	dolar civarında kalmaya çalışır
Söz kimde	ağın kurallarında	bir ihraççı var, dondurabilir (9. bölüm)
Ücret neyle ödenir	kendisiyle	genellikle stablecoinle değil, ağın coiniyle

Artyom’un kahve fiyatları dolarla veya dolara bağlıysa, stablecoinle daha az sürpriz yaşanır. Ama bu da bir güvendir: onu çıkaran şirkete.

3. adım. Hangi ağ

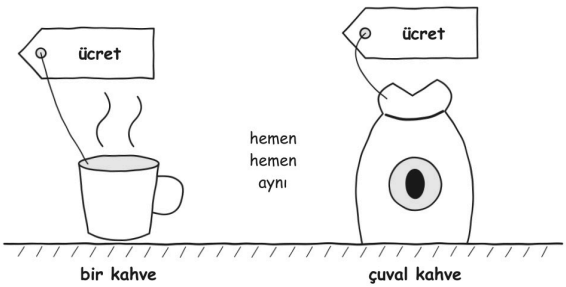
Aynı stablecoin her yerde yaşamaz. İşte 14–20. bölümlerdeki ağlar. "Var", ihraççının sitesinde bu ağ için tam da bu coinin adresinin belirtildiği anlamına gelir.

Ağ	Ücret	Tether'in USDT'si	Circle'ın USDC'si	Unutmayın
Bitcoin	BTC	yok	yok	onayları
Litecoin, Doge vb.	kendi coini	yok	yok	ayrı ağlar
Ethereum	ETH	var	var	gas'ı
Base, Arbitrum	ETH	yok	var	finalized'ı
BNB Smart Chain	BNB	yok	yok	ihraççıyı
TRON	TRX	var	yok	enerjiyi
Solana	SOL	var	var	token hesaplarını
TON	Gram	var	yok	mesajları
Aptos	APT	var	var	uzun adresi
Sui	SUI	yok	var	gRPC, GraphQL
Polkadot Hub	DOT	var	var	hesap eşiğini
XRP Ledger	XRP	yok	var	rezervi, etiketi
Stellar	XLM	yok	var	rezervi, etiketi
Cardano	ADA	yok	yok	tokenin yanında ADA'yı

tether.to/en/supported-protocols ve developers.circle.com sayfalarına göre, 27 Kasım 2026. "Yok", orada bu adda bir token bulunmadığı anlamına gelmez: kimin çıkardığını kontrol edin (9. bölüm). Cosmos'ta her şey belirli ağa bağlıdır.

Transfer bedeli ve sipariş büyüklüğü

Ücret tutara değil, transferin "ağırlığına" ve ağın yoğunluğuna bağlıdır (7. bölüm).



Bir çuval çekirdek için bu devede kulaktır. Bir fincan kahvede ise ücret fincanın kendisinden pahalı olabilir. Bu yüzden küçük siparişlerde transferin nerede ucuz olduğuna bakılır. Fiyatlar değişir; eski bir tablodan değil, göndermeden önce cüzdandan öğrenilir.

Dönüş yolunu da unutmayın: aldığınızı başka yere göndermek için adreste ücret coini gerekir. Yüz adres, ücreti ödenen yüz transfer demektir.

Ne kadar beklenir

Kesinleşme süresi	Ağlar
saniyeler ve daha hızlı	BNB Smart Chain, Solana, TON, Aptos, Sui, XRP Ledger, Stellar
yaklaşık bir dakika	TRON
onlarca dakika	Ethereum, Base, Arbitrum One
yaklaşık bir saat	Bitcoin (6 onay)

Yaklaşık değerler, 8. ve 17–20. bölümlere göre.

Deneyelim: kesinleşmiş blok ne kadar eski

EVM ağlarına `finalized` bloğunu (8. bölüm) soralım ve kaç saniye önce çıktığını hesaplayalım:

```
for net in ethereum base arb bsc; do
  t=$(curl -s https://public-$net-mainnet.rpcnode.dev \
    -H 'Content-Type: application/json' \
    -d '{"jsonrpc":"2.0","id":1,"method":"eth_getBlockByNumber","params":["finalized",false]}' \
    | grep -o '"timestamp":"0x[0-9a-f]*"' | cut -d'"' -f4)
  echo "$net: $(( $(date +%s) - t )) sn"
  sleep 1.5
done
```

```
ethereum: 929 sn
base: 1343 sn
arb: 990 sn
bsc: 1 sn
```

Ethereum'da bu yaklaşık 15 dakika, Base'de yaklaşık 22 dakika. BNB Smart Chain'de bir saniye. Sizin sayılarınız farklı olacak: işaret sıçrayarak ilerler. Küçük bir sipariş için en güvenilir işareti beklemek şart değildir ama o zaman riski siz üstlenirsiniz.

4. adım. Siteniz bunu görecek mi

Adresi siteye yazmadan önce kontrol edin:

- sağlayıcının bu ağ için uç noktası var mı (13. bölüm);

- özellikle bu ağ için ödeme bildirimleri var mı. İkisi aynı şey değildir. RpcNode'da uç nokta alınabilen ağların sayısı, olaylar için blokları taranan ağlardan fazladır (26. ve 27. bölümler);
- gerçek para olmadan denemek için bir test ağı var mı;
- cüzdanınız ve coin bozdurduğunuz yer bu coini **tam da bu ağda** destekliyor mu.

Plana da bir göz atın: RpcNode'un Free planında uç noktalar yalnızca Ethereum ve Solana için vardır (22. bölüm).

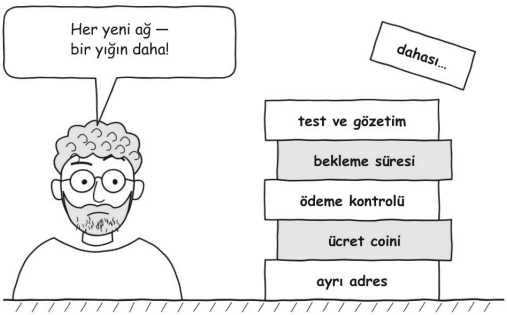
Örnek: Artyom nasıl karar veriyor

Diyelim 20 müşteriye sordu. Rakamlar alıştırma amaçlı; sizin mağazanızınkiler farklı olacak.

Neyle ödemek istiyorlar	Kaç kişi	Karar
TRON'da USDT	9	hemen bağlanacak
Base'de USDC	5	hemen bağlanacak
BTC	3	isteyen olursa sonra
ETH	2	sonra
XLM	1	şimdilik hayır

İki ağ 20 kişiden 14'ünü karşılıyor. Artyom diğerlerine şimdilik her şeyi kabul etmediğini dürüstçe yazıyor.

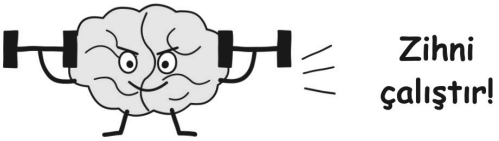
Tuzak: "hepsini birden bağlarım"



Ödeme sayfasındaki fazladan bir satır bedava gibi görünür. Ama arkasında ayrı bir adres, ücret için ayrı bir coin, ayrı kontrol kuralları ve güncellemelerden sonra ayrı sürprizler vardır. Bir iki ağla başlayın, istek geldikçe ekleyin.

İkinci tuzak: ödeme sayfasına yalnızca "USDT" yazmak. Müşteri ağı kendisi seçer ve yanlışını seçebilir (16. bölüm). Coin ile ağı birlikte ve büyük yazın: "TRON ağında USDT".

Düşünün



1. Sipariş 5 dolar, seçilen ağda bugünkü ücret 2 dolar. Bu kimin işine gelmez ve ne yapmalı?
2. Müşteri Base'de USDT kabul etmenizi istiyor. Tether'in sitesindeki liste ne gösteriyor?
3. Artyom ilk günden sekiz ağ kabul etmek istiyor. Ona nasıl itiraz edersiniz?

Cevaplar: 1) müşterinin: ücret siparişin üstüne %40 ekliyor; transferin şu an daha ucuz olduğu bir ağ önerin. 2) Orada Base yok: bu adı taşıyan bir tokeni kabul etmeden önce kimin çıkardığını öğrenin. 3) Sekiz yığın iş demek; müşterilerin en çok istediği bir iki ağla başlamalı.

Özet

- Seçim sırayla yapılır: müşteriler, coin, ağ, kontrol.
- Ücreti sipariş büyüklüğüyle karşılaştırın ve bekleme süresini hesaba katın.
- Bir iki ağla başlayın ve coini her zaman ağıyla birlikte söyleyin.

Dördüncü kısım bitti. Artyom ağların neyle ayrıldığını biliyor ve nereden başlayacağına karar verdi. **Sırada:** 5. kısım, 22. bölüm. RpcNode'da ilk beş dakika.

Bölüm 22. RpcNode'da ilk beş dakika

"Kendi kapımı istiyorum"

Artyom kitap boyunca herkese açık node'ların kapısını çaldı. Kayıt istemeden yanıt veriyorlardı ama o kapı herkesin. Mağazaya kendi kapısı gerekiyor. Bunu yapmak uzun sürer mi? Neleri doldurmak gerekecek?

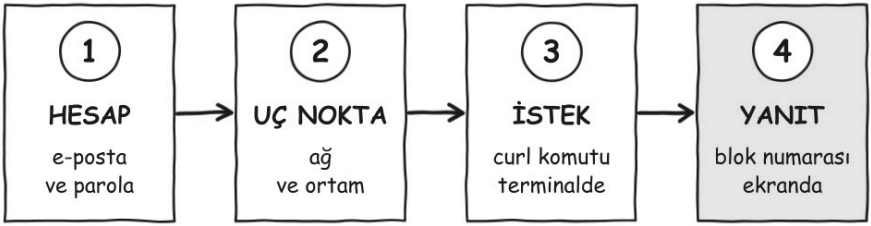
İnceleyelim: ortak kapı ve kendi kapınız

Programın node'a soru sorduğu adrese **uç nokta** (endpoint) denir (3. bölüm). RpcNode'da iki türü vardır.



	Herkese açık adres	Kişisel uç nokta
Kayıt	gerekmez	gerekir
Anahtar	yok	adrese gömülü
Limit	IP başına saniyede 5 istek	plana göre
Ağır metotlar	kapalı	açık
Arşiv (24. bölüm)	kapalı	ücretli planlarda
WebSocket ve gRPC	yok	ağ destekliyorsa

Kendi uç noktanız dört adımda hazır olur.



1. adım. Kayıt

`cabinet.rpcnode.dev/register` adresini açın veya rpcnode.dev sitesinde "Başla" düğmesine basın. Panel, hizmetteki kendi sayfanızdır: uç noktalar, harcama ve plan burada.

1. **E-posta** yazın, bir **Parola** belirleyip tekrar girin. Ad yazmak zorunlu değil.
2. "Kabul ediyorum: hizmet şartları" kutusunu okuyup işaretleyin. "Robot değilim" kontrolü çıkarsa geçin.
3. "Devam" düğmesine basın.
4. E-postanıza tek kullanımlık bir kod gelir. Kodu girin ve "Doğrula ve devam et" düğmesine basın.

Kısa yol da var: Google veya GitHub ile giriş yapmak.

Yeni hesap ücretsiz **Free** planını alır; kart gerekmez. Bu planda ayda 15 milyon kredi, saniyede 15 istek, 3 uç nokta ve iki ağ vardır: Ethereum ve Solana. Kredinin ne olduğunu 25. bölümde inceleyeceğiz.

2. adım. Uç nokta

Panel menüsünden "Uç noktalar" bölümünü seçip "Yeni uç nokta" düğmesine basın. Üç adımlı bir pencere açılır:

Penceredeki adım	Ne yapılır
Ağ seçin	örneğin Ethereum
Ortam seçin	ana ağ veya test ağı (aşağıda anlatılıyor)
Onayla	kontrol edip "Oluştur" düğmesine basın

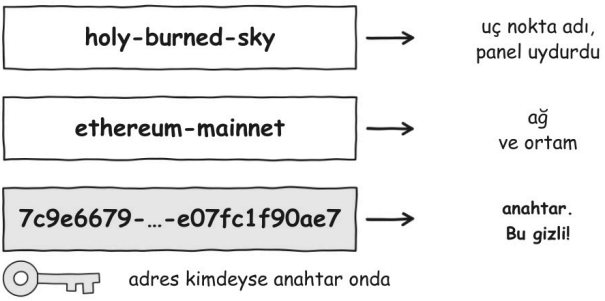
Ortam, ağın çeşididir: gerçek veya alıştırma amaçlı. Uç noktanın adını panel kendisi, üç İngilizce sözcükten uydurur.

Free planında oluşturmadan önce arşiv node'larıyla ilgili bir uyarı görürsünüz. Bu bir hata değildir; 24. bölümde anlatılıyor.

Adres nelerden oluşur

İşte RpcNode belgelerinden bir örnek. Tek bir satırdır; burada eğik çizginin önünden alt satıra geçirildi:

```
https://holy-burned-sky-ethereum-mainnet.rpcnode.dev
/7c9e6679-7425-40de-944b-e07fc1f90ae7
```



Eğik çizgiden sonraki uzun kuyruğa **UUID** denir: tahmin edilmesi pratikte imkânsız bir numaradır. Anahtar odur. İsteğe ayrıca parola eklemek gerekmez.

Adresi elle yazmayın, paneldeki düğmeyle kopyalayın. Ekranda anahtarın bir kısmı gizlidir ama tamamı kopyalanır.

Menüde bir de "API anahtarları" var. Bu başka bir şeydir: hizmetin kendisine verilen komutlar, örneğin "ödeme oluştur", onlarla imzalanır. Node'a sorulan sıradan sorular için gerekmezler.

3. adım. Deneyelim: ilk istek

Adresi bir ayara koyup son blok numarasını soruyoruz. Bu bir **taslaktır**: yazının yerine kendi adresinizi koyun.

```
export RPC_URL='PANELDEKİ_ADRESİNİZ'
curl -s "$RPC_URL" \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_blockNumber","params":[]}'
```

Kişisel adres yalnızca sizde olduğu için kitapta aynı satırların herkese açık adresle verdiği yanıtı gösteriyoruz: `RPC_URL='https://public-ethereum-mainnet.rpcnode.dev'`.

```
{"jsonrpc":"2.0","id":1,"result":"0x19464e4"}
```

`0x19464e4`, 26.502.372 numaralı bloktur. 3. bölümde node 26.502.035 demişti: ağ bu arada 337 blok eklemiş.

Solana için `eth_blockNumber` yerine `getSlot` yazın.

Panelin "Genel bakış" sayfasında bir "Başlarken" listesi vardır. İçinde adresinizin yazılı olduğu hazır bir komut ve "İlk istek bekleniyor…" yazısı durur. Uç nokta sayfasının "Canlı" sekmesinde ise isteklerin kendisi görünür: zaman, metot, yanıt ve gecikme.

Olmadıysa

Bilerek, sıfırlardan oluşan uydurma bir anahtarla soralım:

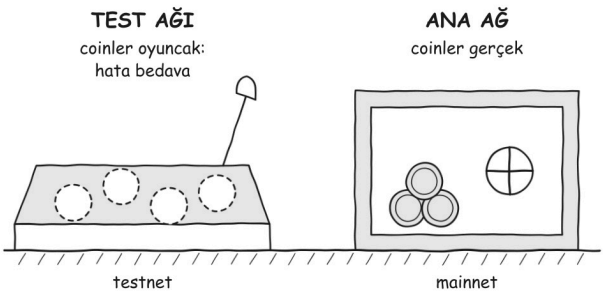
```
HOST=https://holy-burned-sky-ethereum-mainnet.rpcnode.dev
KEY=00000000-0000-0000-0000-000000000000
curl -s -w '  HTTP %{http_code}\n' $HOST/$KEY \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_blockNumber","params":[]}'
```

```
{"jsonrpc":"2.0","id":1,"error":{"code":-32000,
"message":"endpoint not found"}}  HTTP 404
```

`endpoint not found`, "böyle bir uç nokta yok" demektir. Adresin eksiksiz kopyalandığını kontrol edin. Yanlış adresi döngüde tekrarlamayın: ısrarla deneme yaparsanız sizi bir süreliğine dinlemeyi bırakırlar.

Test ağları: oyuncak paralı deneme sahası

Test ağı (testnet) gerçeğiyle aynı kurallarla çalışır ama coinleri beş para etmez. Gerçeğine **mainnet** (ana ağ) denir. Burada hata yapmak bedavadır.



Test coinleri bir "musluktan" (faucet) alınır: belirttiğiniz adrese coin gönderen bir sayfa.

Ağ	RpcNode'daki test ortamlarından örnekler
Ethereum	Sepolia, Hoodi
Solana	testnet, devnet
Bitcoin	testnet4, signet
TRON	Nile, Shasta

Ethereum uygulamaları için ethereum.org sitesi Sepolia'yı önerir. Hoodi, ağın kendisini ve validatörleri sınayanlar içindir.

Kayıtsız test ağları da var: 27 Kasım 2026 itibarıyla herkese açık node listesinde bunlar Bitcoin testnet4, TRON Nile ve TON testnet. TRON'un hem gerçeğine hem test ağına ağ numarasını soralım (16. bölüm):

```
for host in public-tron-mainnet public-tron-nile; do
  echo -n "$host: "
  curl -s https://$host.rpcnode.dev \
    -H 'Content-Type: application/json' \
    -d '{"jsonrpc":"2.0","id":1,"method":"eth_chainId","params":[]}'
  echo; sleep 2
done
```

```
public-tron-mainnet: {"jsonrpc":"2.0","id":1,"result":"0x2b6653dc"}
public-tron-nile: {"jsonrpc":"2.0","id":1,"result":"0xcd8690dc"}
```

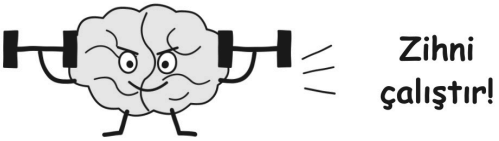
Numaralar farklı: bunlar iki ayrı defter. Uç nokta da tek bir ağa ve tek bir ortama bağlıdır. Test ağından gerçeğine geçmek için yenisi oluşturulur.

Tuzak: "adresi doğrudan sitenin sayfasına koyarım"

Sayfanın koduna giren her şeyi her ziyaretçi görür. Adresinizi alıp limitinizi harcar.

- Node'a müşterinin tarayıcısından değil, kendi **sunucunuzdan** sorun.
- Adresi kodda veya ortak kod deposunda değil, sunucu ayarlarında tutun.
- Uç nokta sayfasındaki "Ayarlar" bölümünde "IP izin listesi" belirleyin: böylece yabancı bir bilgisayar ret alır.
- "İzin verilen Origin'ler", uç noktanın başka sitelerden tarayıcı aracılığıyla çağrılmasını engeller. Ama bu liste tarayıcı dışındaki bir programa karşı korumaz.
- Adres sızdı mı? Yenisini oluşturun, siteyi ona geçirin, eskisini silin: silineni geri getirmek mümkün olmaz.

Düşünün



1. Artyom blok numarasını göstermek için uç nokta adresini mağaza sayfasındaki bir betiğe yazdı. Anahtarı kim görür?
2. Her şey Sepolia'da sınandı. Aynı uç noktayla gerçek ödeme alınabilir mi?
3. `endpoint not found` geliyor. Aramaya nereden başlamalı?

Cevaplar: 1) her ziyaretçi. 2) Hayır: mainnet'te bir uç nokta gerekir. 3) Adresi paneldekiyle karşılaştırın: baştan, eksiksiz kopyalayın.

Özet

- Kişisel uç nokta, içinde anahtar bulunan bir adrestir: onu parola gibi koruyun.
- Uç nokta tek bir ağa ve tek bir ortama bağlıdır.
- Önce test ağı, sonra gerçeği.

Sırada: 23. bölüm. İsteğin başına neler gelir: ağ geçidi, node seçimi ve üç coğrafya.

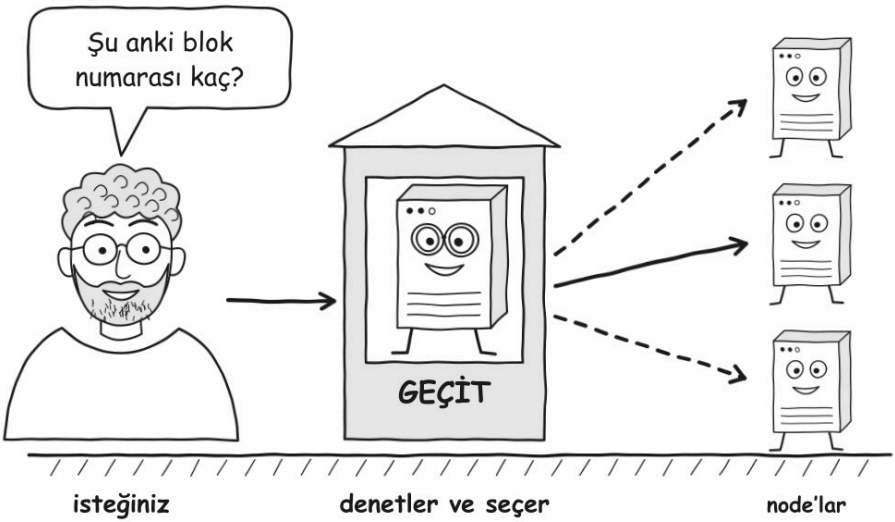
Bölüm 23. İsteğin başına neler gelir

“Peki bana kim yanıt verdi?”

Artyom istek gönderdi ve blok numarasını aldı. Ama soru nereye gitti? Hangi node yanıtladı? Ya o node indirim kampanyasının tam ortasında bozulursa?

İnceleyelim: girişte ağ geçidi durur

İsteğiniz doğrudan node’a gitmez. Önce **ağ geçidine** (gateway) gelir. Bu bir dağıtıcı programdır: istekleri alır, denetler ve uygun node’a iletir.



Ağ geçidi yolda neleri denetler

Düzen, bir binanın güvenlik girişine benzer. Bir kontrolden geçemeyen istek daha ileri gidemez.

Kontrol	Geçemezse
Böyle bir uç nokta var mı, adres doğru mu	HTTP 404, `-32000`
Kapatılmış mı; IP'nize ve sitenize izin var mı	HTTP 403, `-32000`
Çok sık mı soruyorsunuz	HTTP 429, `-32005`
Bu ağda böyle bir metot var mı	HTTP 404, `-32601`
Kredi yetiyor mu	HTTP 402, `-32006`

Her ağın kendi metot listesi vardır. Listede olmayan bir şey soralım:

```
curl -s -w '  HTTP %{http_code}\n' https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_noSuchMethod","params":[]}'
```

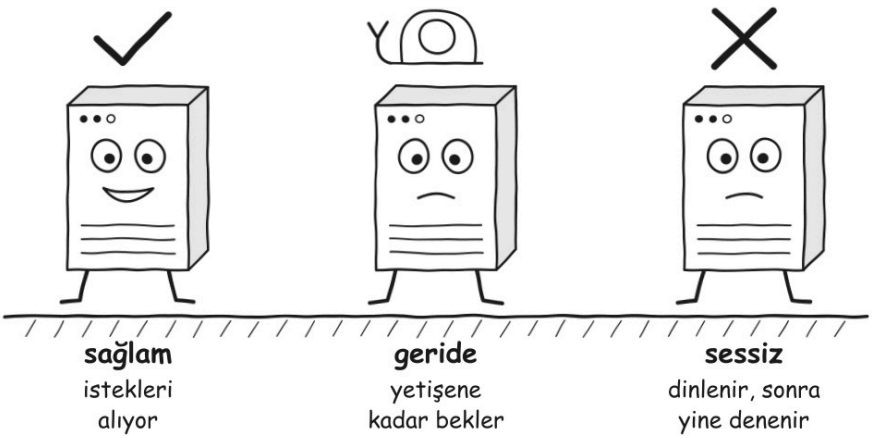
```
{"jsonrpc":"2.0","id":1,"error":{"code":-32601,
"message":"method not found: eth_noSuchMethod"}}  HTTP 404
```

Bu soru node'a ulaşmadı bile: ağ geçidi kendisi yanıtladı.

Node seçimi: yük dengeleme

Bir ağ için birkaç node bulunabilir. İstekleri aralarında paylaştırmaya **yük dengeleme** denir. Marketteki kasalar gibi: yeni müşteri daha boş olana yönlendirilir.

Ağ geçidi önce node'un sağlam olup olmadığına bakar.



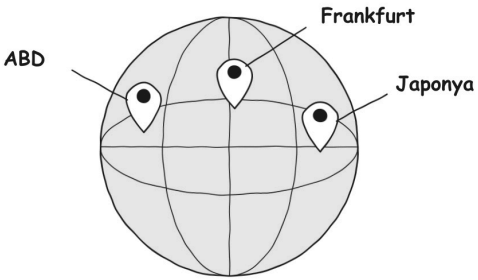
- **Geri kalan** node neşeyle yanıt verir ama dünü anlatır (12. bölüm). Güncel node'lar varken istekler onlara gider.

- Sağlam node'lar arasında meşgul olana az, boş olana çok iş düşer.
- Bağlantı koptuysa veya node bir iç hata döndürdüyse aynı istek sıradakine gider. Siz bunu fark etmezsiniz.
- Üst üste birkaç kez beceremeyen node bir süre rahat bırakılır, sonra yeniden denenir.

Bir kayıt düşelim. Hiç sağlam node kalmadıysa ağ geçidi yine de geri kalanları dener: bazen eski bir yanıt, hiç yanıt almamaktan iyidir. Bu yüzden tazeliğin önemli olduğu yerde blok zamanını kendiniz kontrol edin (12. bölüm).

Üç coğrafya

RpcNode üç coğrafyada çalışır: Frankfurt, ABD ve Japonya. Node ne kadar yakınsa gidiş dönüş yolu o kadar kısadır. Ağ geçidi kendisine en yakın sağlam node'u almaya çalışır, uzaktakileri yedekte tutar.



RpcNode'un üç bölgesi (çizim, harita değil)

Her ağın üç yerde birden node'u yoktur. İhtiyacınız olan ağın nereden ve nasıl yanıt verdiğini `rpcnode.dev/status` sayfası gösterir.

Deneyelim: zarfın üstündeki notları okuyalım

Yanıtla birlikte **başlıklar** da gelir: zarfın üstündeki damgalar gibi yardımcı notlar. İkisi, ağ geçidinin ve node'un hangi ülkede olduğunu söyler:

```
for net in ethereum base arb; do
  echo "== $net"
  curl -s -D - -o /dev/null https://public-$net-mainnet.rpcnode.dev \
    -H 'Content-Type: application/json' \
    -d '{"jsonrpc":"2.0","id":1,"method":"eth_blockNumber","params":[]}' |
    grep -iE 'x-gateway-region|x-upstream-country'
  sleep 2
done
```

```
== ethereum
x-gateway-region: DE
x-upstream-country: DE
== base
x-gateway-region: DE
x-upstream-country: JP
== arb
x-gateway-region: DE
x-upstream-country: JP
```

`DE` Almanya, `JP` Japonya. Bu kez bütün istekleri Almanya'daki ağ geçidi karşıladı, Base ve Arbitrum için ise Japonya'ya gitti. Sizde harfler farklı olabilir. Bunlar yardımcı notlardır: sitenizin işleyişini bunlara dayandırmayın.

Node'un hatası da bir yanıttır

USDT sözleşmesine soralım: boş bir adresten dev bir tutar aktarılırsa ne olur? `eth_call` metodu yalnızca dener, hiçbir şey göndermez. `data` alanında transfer fonksiyonunun numarası, alıcı (`TO`) ve tutar (`SUM`) birleştirilmiştir.

```
TO=000000000000000000000000000000000000000000000000000000000000dead
SUM=0000000000000000000000000000000000000000000000000fffffffffffffff
curl -s -w '  HTTP %{http_code}\n' https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_call","params":[{
  "from":"0x0000000000000000000000000000000000000dEaD",
  "to":"0xdac17f958d2ee523a2206206994597c13d831ec7",
  "data":"0xa9059cbb'$TO$SUM'"},"latest"]}'
```

```
{"jsonrpc":"2.0","id":1,"error":{"code":-32000,
"message":"invalid opcode: INVALID"}}  HTTP 200
```

Bu eski sözleşme "olmaz" demeyi böyle yapıyor. Node görevini yaptı: programın durduğunu dürüstçe bildirdi. Başka bir node da aynı şeyi söyleyeceği için ağ geçidi bu yanıtı size olduğu gibi, HTTP 200 ile verir.

Kimse yanıt vermezse

Daha kötüsü de olur: ağ geçidi node'ları tek tek dener ama yanıt yoktur. O zaman `-32002` kodu, `upstream unavailable` metni ("node erişilemiyor") ve "beş yüzlü" bir HTTP kodu gelir. Böyle bir çağrının kredisi geri verilir.

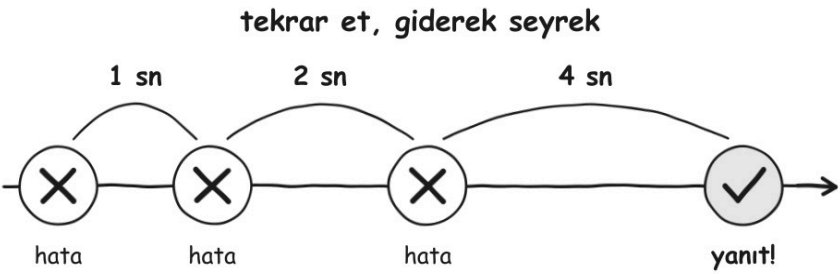
Ya bu bir para gönderimiyse?

Node işlem gönderimini **reddettiyse**, yanıt budur. Ama hiç **ses vermediyse**, ağ geçidi aynı imzalı transferi başka bir node'a iletebilir.

Bu yüzden para iki kez gitmez: imzalı işlemin hash'i aynıdır ve ağ onu bir kez kabul eder. Ama yanıtı alamayan **sizseniz**, körlemesine yeni bir işlem imzalamayın. Önce eskisini hash'iyle arayın (7. bölüm).

Deneyelim: akıllıca tekrarlayalım

Ağ geçidi node'ları sırayla dener ama sizinle geçit arasındaki bağlantı da aksayabilir. Kendiniz tekrarlayın, giderek uzayan aralarla: 1, 2, 4, 8 saniye. İngilizcede buna **backoff** denir.



`ask.js` dosyası, `node ask.js` ile çalıştırılır (Node.js gerekir):

```
const url = 'https://public-ethereum-mainnet.rpcnode.dev'
const sleep = (ms) => new Promise((ok) => setTimeout(ok, ms))

async function ask(method, params = []) {
  for (const pause of [1, 2, 4, 8]) {
    try {
      const res = await fetch(url, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ jsonrpc: '2.0', id: 1, method, params }),
        signal: AbortSignal.timeout(10000),
      })
      if (res.status !== 429 && res.status < 500) return await res.json()
      console.log('HTTP', res.status)
    } catch {
      console.log('bağlantı yok')
    }
    await sleep(pause * 1000)
  }
  throw new Error('sorulamadı')
}

ask('eth_blockNumber').then(console.log)
```

```
{ jsonrpc: '2.0', id: 1, result: '0x19464ee' }
```

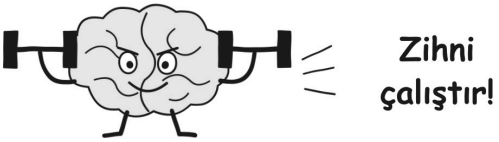
Yalnızca ikinci denemede geçebilecek şeyler tekrarlanır: "çok sık", bağlantı kopması, "beş yüzlüler". İsteğin kendisindeki bir hatayı tekrarlamak boşunadır. Yanıt gelince `error` alanını kontrol etmeyi de unutmayın (11. bölüm).

Tuzak: "yanıt yok" demek "ödeme yok" demek değil

Sorulamadıysa site müşterinin ödeyip ödemediğini **bilmiyor** demektir. "Ödenmedi" değil, "ödeme kontrol ediliyor" yazın ve sonra tekrar sorun.

Unutmayın: birkaç node'u olan sağlayıcı tek bir node'dan daha seyrek çöker, ama "asla" değil (3. bölüm).

Düşünün



1. HTTP 200 ve `execution reverted` geldi. İsteği beş kez tekrarlamak işe yarar mı?
2. Program `eth_blokNumber` sordu. İstek node'a ulaştı mı?
3. Artyom bir transfer gönderdi ve yanıt alamadı. İlk iş ne yapmalı?

Cevaplar: 1) hayır: bu bir arıza değil, gerçek yanıt. 2) Hayır, ağ geçidi tanımadığı metodu reddetti. 3) İşlemi hash'iyle aramalı.

Özet

- Ağ geçidi isteği denetler ve sağlam bir node seçer; biri yanıt vermezse diğerini dener.
- Node'un isteğinizle ilgili kendi hatası size olduğu gibi ulaşır.
- Programınız yine de istekleri aralıklı olarak tekrarlamalıdır.

Sırada: 24. bölüm. Geçmiş ve arşiv verileri.

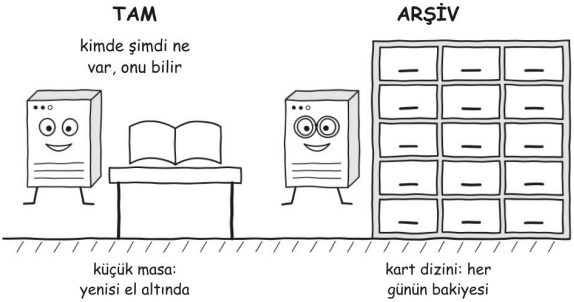
Bölüm 24. Geçmiş ve arşiv verileri

“Bir Ocak’ta ne kadar vardı?”

Muhasebeci Artyom’dan istiyor: “Yıl başında mağazanın adresinde ne kadar vardı, söyleyin.” Node bugünü anında yanıtlıyor. Ya o günü?

İnceleyelim: kayıtlar ve bakiyeler

10. bölümü hatırlayalım. **Geçmiş** bloklar ve transferlerdir. **Durum** ise belirli bir anda kimin neye sahip olduğudur.



Tam node güncel bakiyeleri el altında tutar. **Arşiv node’u** her bloktaki bakiyeyi hatırlar ama çok daha fazla disk ister.

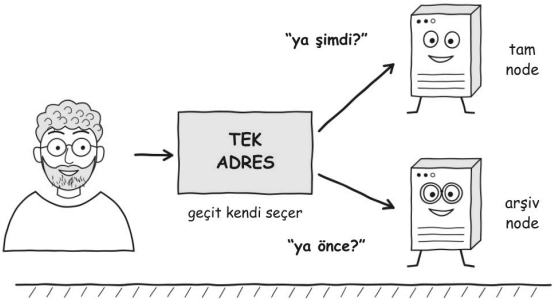
Hangi sorular arşiv ister

Soru	Örnek
Eski bir bloktaki bakiye veya sözleşme yanıtı	blok numarasıyla `eth_getBalance`, `eth_call`
Geçmiş blokların olayları	`eth_getLogs`
İşlemi adım adım “geri sarmak” (izleme)	`trace_…`, `debug_…`
Durumla ilgili kanıt	eski blokta `eth_getProof`
Solana’nın eski slotları	eski numarayla `getBlock`

“Hesapta şu an ne kadar var” sorusu ve sıradan, yeni bir ödeme ise arşiv gerektirmez. Mağazanın günlük işinde genellikle arşive ihtiyacı olmaz.

RpcNode'da bu nasıl işler

Uç noktanın adresi tektir. Node'u sizin seçmeniz gerekmez: ağ geçidi soruya bakıp kime ileteceğine karar verir.



Arşiv ücretli planlara dahildir, Free planında yoktur. Free'de uç nokta oluştururken panelde şunu görürsünüz:

> "Ücretsiz planda arşiv node'ları kullanılamaz. İstekleriniz normal node'lar tarafından karşılanır; sakladıkları her şeyi yanıtlarlar. Yalnızca arşiv node'unda bulunan veriler (eski durum, uzak geçmiş, bazı trace'ler) sunulmaz: node kendi hatasını döndürür."

- "Eski" soru için ayrı bir fiyat yoktur: metodun bedeli ödenir (25. bölüm).
- Arşivin derinliği ağdan ağa değişir. Üzerine önemli bir şey kurmadan önce, en eski sorunuzu kendi ağınızda deneyin.
- Herkese açık adresler arşiv sorularına hizmet vermez.

Deneyelim: herkese açık node katıdır

Bakiyeyi "şu an" için değil, yalnızca on blok, yaklaşık iki dakika önceki bir blok numarasıyla soralım:

```
E=https://public-ethereum-mainnet.rpcnode.dev
H='Content-Type: application/json'
N=$(curl -s $E -H "$H" \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_blockNumber","params":[]}' |
  grep -o '0x[0-9a-f]*')
OLD=$(printf '0x%x' $((N - 10)))
sleep 2
curl -s -w '  HTTP %{http_code}\n' $E -H "$H" \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getBalance",
  "params":["0xdac17f958d2ee523a2206206994597c13d831ec7","'$OLD'"]}'
```

```
{"jsonrpc":"2.0","id":1,"error":{"code":-32601,"message":
"archive data is not available on the public endpoint,
use a personal endpoint"}}  HTTP 403
```

Herkese açık adres bu kadar yakın geçmişi bile arşiv sayıyor. Ona 8. bölümün sözcükleriyle sorun: `latest`, `safe` veya `finalized`.

İkinci deneme: node atmış da olabilir

Blok durum değil, geçmiştir. Ama onu da herkes saklamaz. Temmuz 2025'ten itibaren Ethereum node programları, staking'e geçişten önce çıkan blokların içeriğini tutmayabiliyor (10. bölüm). Ağustos 2015'ten 46.147 numaralı bloğu (`0xb443`) isteyelim:

```
curl -s -w '  HTTP %{http_code}\n' https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getBlockByNumber",
  "params":["0xb443",false]}'
```

```
{"jsonrpc":"2.0","id":1,"error":{"code":4444,
"message":"pruned history unavailable"}}  HTTP 200
```

"Budanmış geçmiş mevcut değil." Node dürüstçe bunun kendisinde olmadığını söylüyor. Ücretli planda ağ geçidi, sıradan node'dan "bende yok" yanıtını alınca, ağın arşiv node'u varsa aynı soruyu ona da sorabilir.

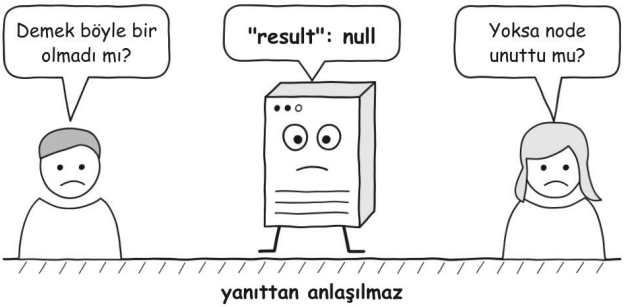
Tuzak: "null, demek ki böyle bir şey olmadı"

O bloğun içinde bir transfer var. Onu hash'iyle soralım, hemen ardından da az önce uydurduğumuz bir hash'i:

```
for hash in \
  0x5c504ed432cb51138bcf09aa5e8a410dd4a1e204ef84bfed1be16dfba1b22060 \
  0x0000000000000000000000000000000000000000000000000000000000000001
do
  curl -s https://public-ethereum-mainnet.rpcnode.dev \
    -H 'Content-Type: application/json' \
    -d '{"jsonrpc":"2.0","id":1,"method":"eth_getTransactionByHash",
    "params":["'$hash'"]}'
  echo; sleep 2
done
```

```
{"jsonrpc":"2.0","id":1,"result":null}
{"jsonrpc":"2.0","id":1,"result":null}
```

Yanıtlar aynı! Oysa birinci transfer gerçekten yapılmıştı, ikincisi ise hiç var olmadı.



Tek bir `null` yanıtından, özellikle eski kayıtlar hakkında, sonuç çıkarmayın. Bazen "olmadı" demektir, bazen "saklamıyorum", yeni bir transfer için de "henüz bloğa girmedi".

Solana node'u ne kadarını hatırlar

Solana saniyede birkaç blok üretir ve kayıtlar dağ gibi birikir. En yeni slot numarasını ve node'un hâlâ tuttuğu en eski bloğu soralım:

```
for method in getSlot getFirstAvailableBlock; do
  echo -n "$method: "
  curl -s https://public-solana-mainnet.rpcnode.dev \
    -H 'Content-Type: application/json' \
    -d '{"jsonrpc":"2.0","id":1,"method":"'$method'"}'
  echo; sleep 2
done
```

```
getSlot: {"jsonrpc":"2.0","result":474316453,"id":1}
getFirstAvailableBlock: {"jsonrpc":"2.0","result":473916707,"id":1}
```

Fark yaklaşık 400.000 slot. Kontrol sırasında bu yalnızca bir gün ediyordu. 1000 numaralı blok ise çoktan temizlenmiş:

```
curl -s https://public-solana-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"getBlockTime","params":[1000]}'
```

```
{"jsonrpc":"2.0","error":{"code":-32001,"message":"Block 1000 cleaned up,
does not exist on node. First available block: 473916707"},"id":1}
```

Muhasebeciye dönelim

"Yıl başındaki" bakiye, eski yılın son bloğundan sonraki bakiyedir (zamanı eşgüdümlü evrensel zamanla, UTC ile sayıyoruz). Node bir tarihe karşılık gelen blok numarasını doğrudan söylemez. Ama bloklar sırayla gider ve **ikili arama** yapılabilir: ortadaki bloğu alır, zamanına bakar, gereksiz yarıyı atarız.

2026'nın ilk bloğunu bulalım. Betik yaklaşık bir dakika çalışır:

```
E=https://public-ethereum-mainnet.rpcnode.dev
ask() {  # method, params
  curl -s $E -H 'Content-Type: application/json' \
    -d '{"jsonrpc":"2.0","id":1,"method":"'$1'","params":'$2'}'
  sleep 2
}
WHEN=1767225600   # 1 Ocak 2026, 00:00 UTC
low=20000000
high=$(( $(ask eth_blockNumber '[]' | grep -o '0x[0-9a-f]*') ))
steps=0
while [ $low -lt $high ]; do
  mid=$(( (low + high) / 2 ))
  hex=$(printf '0x%x' $mid)
  t=$(ask eth_getBlockByNumber '["'$hex'",false]' |
    grep -o '"timestamp":"0x[0-9a-f]*"' | grep -o '0x[0-9a-f]*')
  if [ -z "$t" ]; then echo "yanıt yok"; exit 1; fi
  if [ $(( t )) -lt $WHEN ]; then low=$(( mid + 1 )); else high=$mid; fi
  steps=$(( steps + 1 ))
done
echo "blok $low, soru sayısı: $steps"
```

```
blok 24136053, soru sayısı: 23
```

Milyonlarca blok arasından aranan 23 soruda bulundu (sizde sayı bir fark edebilir). Demek ki 2025'in son bloğu 24.136.052 (`0x1704974`). O bloktaki bakiye ilk denemedeki gibi sorulur, yalnız bu numarayla. Herkese açık adres reddeder; ücretli planda kişisel uç nokta gerekir. Token için `eth_getBalance` yerine, sözleşmeye `balanceOf` sorusuyla `eth_call` kullanılır.

Uzun aralığı parça parça sorun

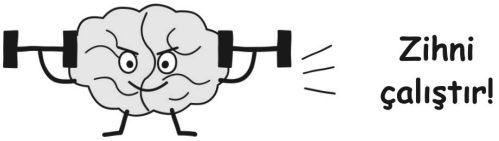
"Bir yılın bütün olayları" isteğini node büyük olasılıkla kaldıramaz: yanıt fazla büyüktür. Blok aralığı parçalara bölünür, sırayla sorulur ve yanıtlar birleştirilir.

Bu bir **taslaktır**: `rpc` , kişisel uç noktaya soran sizin fonksiyonunuzdur; 23. bölümdeki `ask` gibi.

```
async function logsByChunks(rpc, filter, from, to, size = 1000) {
  const all = []
  for (let a = from; a <= to; a += size) {
    const b = Math.min(a + size - 1, to)
    const answer = await rpc('eth_getLogs', [{
      ...filter,
      fromBlock: '0x' + a.toString(16),
      toBlock: '0x' + b.toString(16),
    }])
    if (answer.error) throw new Error(answer.error.message)
    all.push(...answer.result)
  }
  return all
}
```

Geçmiyorsa parçayı küçültün; her parça ayrı bir ücretli çağrıdır. Mağazanın en ucuz arşivi ise **kendi arşividir**: ödemeleri görür görmez veritabanına yazın (26. bölüm).

Düşünün



1. Node eski bir transferin hash'ine `null` döndürdü. Bu, müşterinin ödemediğini kanıtlar mı?
2. 1 Temmuz'daki USDC bakiyesi gerekiyor. Free planı yeter mi?
3. 5000 bloğun olayları 1000'lik parçalarla taranacak. Kaç çağrı olur?

Cevaplar: 1) hayır: node bu kaydı saklamıyor olabilir. 2) Free'de arşiv yoktur, buna güvenilemez. 3) Beş.

Özet

- Eski bakiyeler, eski olaylar ve izlemeler arşiv node'unun işidir.
- Arşiv ücretli planlara dahildir; uç nokta adresi aynı kalır.
- `null` ve boş yanıt, henüz hiçbir şey olmadığını kanıtlamaz.

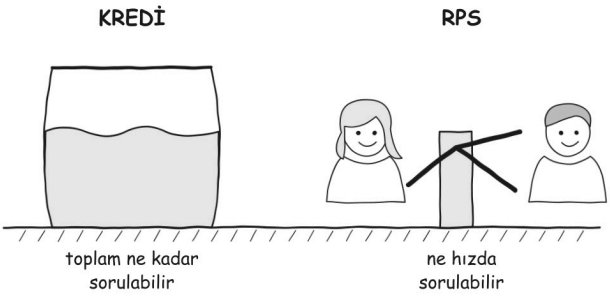
Sırada: 25. bölüm. Planlar, çağrı bedeli ve limitler.

Bölüm 25. Planlar, çağrı bedeli ve limitler

“Peki bu ayda kaça gelir?”

Uç nokta çalışıyor. Artyom şimdi mağazanın kaça mal olacağını bilmek istiyor. Fiyat sayfasında iki ana sayı var: kredi ve saniyede istek. Hangisi ne?

İnceleyelim: iki ayrı ölçü



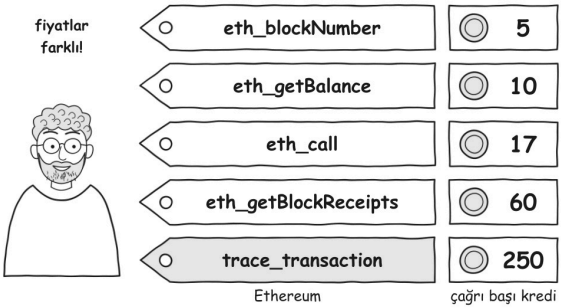
Kredi, hizmetin iç “parasıdır”: bir ayda toplam ne kadar sorulabileceğini belirler. **RPS** saniyede istek sayısıdır: ne kadar hızlı sorulabileceğini belirler. İkisi de yetmelidir.

Deneyelim: çağrının bedelini öğrenelim

Farklı soruların bedeli farklıdır: birini node bellekten yanıtlar, diğeri için koca bir bloğu tarar. Fiyatlar açıktır, kayıt olmadan sorulabilir. Birkaç Ethereum metodu alalım (`python3` gerekir):

```
curl -s https://rpcnode.dev/api/v1/public/networks/ethereum/methods |
python3 -c 'import json, sys
want = """eth_blockNumber eth_getBalance eth_call
eth_getLogs eth_getBlockReceipts trace_transaction""".split()
for m in json.load(sys.stdin)["methods"]:
    if m["method"] in want and m["variant"] == "default":
        print(m["method"], m["credits"])'
```

```
eth_blockNumber 5
eth_getBalance 10
eth_call 17
eth_getLogs 17
eth_getBlockReceipts 60
trace_transaction 250
```



- Bir `trace_transaction`, 50 blok numarası sorusu kadar tutar.
- Her ağın kendi fiyat listesi vardır. Adresteki `ethereum` yerine `solana` veya `tron` yazın ve metot adlarını değiştirin.
- Toplu istek (11. bölüm), içindeki çağrıların toplamı kadar tutar.
- WebSocket abonelikleri de ücretlidir. Ethereum'da yeni bloklara abonelik açmak 10 kredi, gelen her olay 2 kredidir.

Panelde fiyatlar "Ağlar" bölümünde toplanmıştır.

Planlar

Veriler 27 Kasım 2026 tarihli. Güncelleri `rpcnode.dev/pricing` adresinde.

Plan	Aylık $	Yıllık $	Aylık kredi	RPS	Uç nokta
Free	0	—	15 mn	15	3
Starter	19	190	50 mn	40	5
Build	36	360	100 mn	75	12
Grow	99	990	300 mn	150	25
Scale	219	2.190	700 mn	300	50
Business	449	4.490	1,5 milyar	500	100

Plan yalnızca RPC demek değildir. İçinde başka neler olduğu (26. ve 27. bölümler):

Plan	İzlenen adres	Ödeme: açık / aylık	İşlem izleme	Arşiv ve tüm ağlar
Free	—	—	—	yok: yalnızca Ethereum ve Solana
Starter	15	10 / 200	15	var
Build	50	25 / 1.000	40	var
Grow	150	50 / 5.000	80	var
Scale	400	100 / 20.000	150	var
Business	1.000	250 / 100.000	400	var

- Yeni bir hesap, Starter'ı bir kez 7 gün boyunca ücretsiz ve kartsız deneyebilir.
- Hazır planlar uymuyorsa fiyat sayfasında "Kendi planınızı oluşturun" bölümü var.
- Plan satın almak için "Satın al" düğmesine basıp onaylayın. 15 dakika geçerli bir fatura çıkar; kripto parayla ödenebilir.
- Süre dolup yeni ödeme gelmezse hesap Free'ye geçer. Kendinize bir hatırlatıcı kurun.

Artyom'un mağazasını hesaplayalım

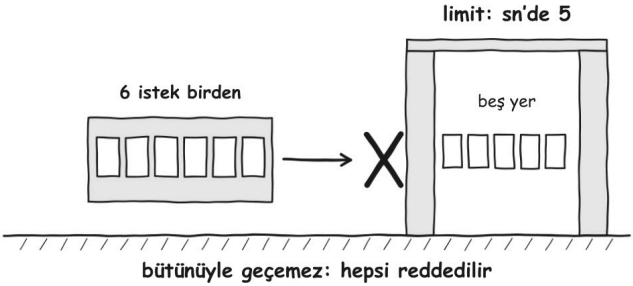
Yirmi sipariş ödeme bekliyor, her birinin Ethereum'da ayrı adresi var. 30 gün alıyor ve siparişlerin hep beklediğini varsayıyoruz: bu üst sınır tahminidir.

Site nasıl soruyor	İstek	Kredi	Yeten plan
her adresin bakiyesi 10 sn'de bir	5.184.000	51.840.000	Build
aynısı 30 sn'de bir	1.728.000	17.280.000	Starter
12 sn'de bir blok numarası ve tek `eth_getLogs`	432.000	4.752.000	Free

Üçüncü yöntem birinciden on kattan fazla ucuz: tek soru bütün adresleri birden kapsıyor. Ama karşılaştırmada bir kayıt var: ilk iki yöntem ETH girişini, üçüncüsü ise token transferlerini fark eder (15. bölüm). Free'de de yalnızca iki ağ vardır, Ethereum ve Solana.

Saniyelik limit

RPS sayacı her saniye sıfırlanır ve RpcNode belgelerine göre her uç noktanın kendi sayacı vardır. Fazla istek HTTP 429 ve `-32005` kodunu alır. **N çağrılık toplu istek N istektir** ve ya bütünüyle geçer ya hiç geçmez.



Limiti 5 olan herkese açık adreste deneyelim. Altı soruluk tek bir toplu istek yeter:

```
q='{"jsonrpc":"2.0","id":1,"method":"eth_blockNumber","params":[]}'
curl -s -i https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d "[$q,$q,$q,$q,$q,$q]" | grep -iE '^(HTTP|retry-after)|error'
echo
```

```
HTTP/2 429
retry-after: 1
{"jsonrpc":"2.0","id":null,"error":{"code":-32005,"message":
"rps limit exceeded: a batch of 6 calls counts as 6 requests
(the limit is 5 a second)"}}
```

`retry-after: 1`, "bir saniye sonra tekrarlayın" demektir. Herkese açık adres pahalı metotları da korur:

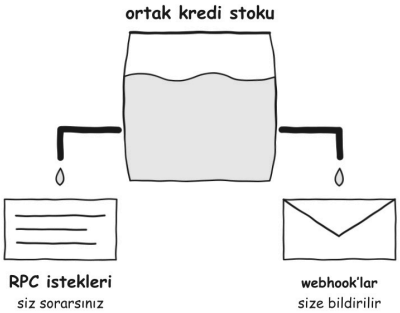
```
curl -s -w '  HTTP %{http_code}\n' https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getBlockReceipts","params":["latest"]}'
```

```
{"jsonrpc":"2.0","id":1,"error":{"code":-32601,"message":
"method eth_getBlockReceipts is not available on the public endpoint,
use a personal endpoint"}}  HTTP 403
```

Neye kredi kesilir, neye kesilmez

Ne oldu	Kredi
Node sonuçla yanıt verdi	kesilir
Node isteğinizle ilgili hata döndürdü, örneğin `execution reverted`	kesilir: bu da bir yanıttır
Saniyelik limit yüzünden ret	kesilmez
Metot ağın listesinde yok	kesilmez
Hiçbir node yanıt vermedi	geri verilir
Herkese açık adrese istek	ücretsiz

Her şey için tek stok



Aynı stoktan **webhook**'ların, yani hizmetin sitenize kendiliğinden gönderdiği mesajların bedeli de ödenir (26. bölüm). Ethereum ana ağında izlenen adresin bir olayı 450 kredi, bir ödeme 750 kredidir. Ethereum test ağlarında yarı fiyatınadır.

Kalan krediye panelin "Kullanım" bölümünden bakın: en sık kullanılan metotlar da orada.

Kredi bitince

Aşım faturası gelmez: paketin üstüne ek ücret yoktur. Ama hizmetin hiçbir şey olmamış gibi çalışmaya devam edeceğine güvenmeyin. Programa 11. bölümdeki reddi (`-32006`) karşılayacak bir yol koyun ve kalanı önceden izleyin.

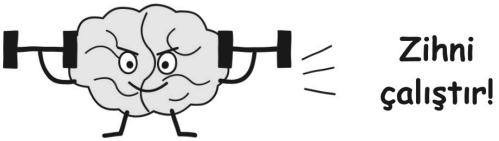
Tuzak: planı fiyat etiketine bakıp seçmek

Önce hesaplayın: **kaç kez × metodun bedeli**. Birkaç ağır çağrı, binlerce hafif çağrıdan daha çok yiyebilir.

Metot fiyatları ve koşullar değişebilir. Hizmet koşullarına göre plan ödemesi, yasa aksini gerektirmedikçe iade edilmez ve kesintisiz çalışma garanti edilmez: ücretli planda büyük bir

arıza olursa ödenen süre uzatılır. Koşulları ödemeden önce okuyun.

Düşünün



1. Site bütün ay, yani 30 gün boyunca 10 saniyede bir `eth_call` (17 kredi) çağırıyor. Bu kaç kredi eder?
2. Free planında 20 çağrılık bir toplu istek gönderildi. Ne olur?
3. İstek 429 aldı. Bunun için kredi kesildi mi?

Cevaplar: 1) 259.200 × 17 = 4.406.400. 2) Tamamı reddedilir: 20, 15'ten büyük. 3) Hayır.

Özet

- Kredi aylık hacmi, RPS hızı sınırlar.
- Her metodun kendi bedeli vardır; toplu istek çağrı sayısına göre hesaplanır.
- Daha akıllıca sormak çoğu zaman daha büyük plan almaktan kârlıdır.

Sırada: 26. bölüm. Adres olayları: sürekli sorgulama yerine webhook ve soketler.

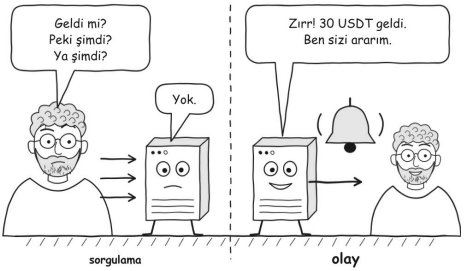
Bölüm 26. Adres olayları: sürekli sorgulama yerine webhook ve soketler

“Yirmi sipariş, saniyede yirmi soru mu?”

3. bölümde hesaplamıştık: bekleyen yirmi sipariş, her biri saniyede bir kontrol edilirse bu, Free planının saniyede 15 istek sınırını aşar. Üstelik yanıtların neredeyse hepsi aynı: “gelmedi”.

İnceleyelim: sormak mı, zili beklemek mi

Sorgulama, sitenizin tekrar tekrar kendisinin “Geldi mi?” diye sormasıdır. **Olay** ise bir şey olur olmaz size haber verilmesidir.



Deneyelim: bütün adresler için tek soru

Önce daha akıllıca soralım. Her siparişi ayrı ayrı sormaya gerek yok. 15. bölümdeki `eth_getLogs` metoduna bir alıcı listesi birden verilebilir:

```
curl -s https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getLogs","params":[{
  "address":"0xdac17f958d2ee523a2206206994597c13d831ec7",
  "topics":["0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
  null,
  ["0x0000000000000000000000000000000000000000000000000000000000000dead",
   "0x0000000000000000000000000000000000000000000000000000000000000001",
   "0x000000000000000000000000dac17f958d2ee523a2206206994597c13d831ec7"]],
  "fromBlock":"latest","toBlock":"latest"}]}'
```

Soru şu: “En yeni blokta, kimden olursa olsun, bu üç adresten birine USDT transferi var mı?” Adresler soldan sıfırlarla 64 karaktere tamamlanmıştır. Bize gelen yanıt:

```
{"jsonrpc":"2.0","id":1,"result":[]}
```

Boş liste: bu blokta kimse ödememiş. Adres yirmi tane de olsa istek yine tektir. Ama **her** yeni blok için sormak, onayları kendiniz saymak ve ETH transferlerini ayrıca yakalamak gerekir (15. bölüm).

Address Watch: bırakın sizin yerinize izlesinler

Address Watch, yani adres izleme, bu işi üstlenir. RpcNode blokları tarar ve bir işlem adresinize dokunduğunda sunucunuza haber verir.

Adres panelden eklenir: "Bildirimler" → "Adresler" → "Adres ekle". Program da aynısını `{"network_slug": "tron", "address": "ADRESİNİZ"}` gövdeli bir `POST /v1/address-watches` isteğiyle yapar.

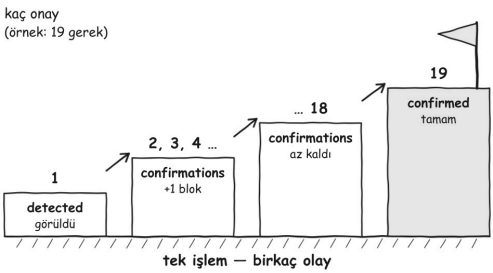
Yanıtta `watch_id`, yani izleme numarası gelir. Bir ağdaki bir adres planın bir **slotunu** tutar. İzlemeyi silerseniz slot boşalır; ama ekleme ve silmelerin de bir limiti vardır. Starter'da 15 slot bulunur. Free'de ana ağlar için slot yoktur, yalnızca bir "kum havuzu" vardır: test ağlarında birkaç adres.

Uç nokta alınabilen ağların sayısı, olaylar için blokları taranan ağlardan fazladır. 27 Kasım 2026 itibarıyla bunlar Ethereum, Bitcoin, Solana, TRON, BNB Smart Chain, Base, Arbitrum One, TON, Litecoin, Dogecoin, Bitcoin Cash, Dash, Mantle ve Monad; test ağlarından ise yalnızca TRON Nile.

Para girişi `payment` türünde bir olaydır, çıkışı `payout`, geri kalan her şey (diyelim approve izni) `address`. Yeni bloklar `block` türüyle gelir; `tx` ve `payment_order` ise 27. bölümde.

Tek işlem, birkaç zil

Transfer bir kez değil, "katılaştıkça" (8. bölüm) bildirilir. Tam olarak ne olduğunu `event` alanı söyler:



event	Anlamı	Mağaza ne yapmalı
`detected`	işlem bir blokta görüldü	"transferi görüyoruz" demeli
`confirmations`	bir onay daha	sayacı güncellemeli
`confirmed`	onay yeterli, başarılı	parayı ulaşmış saymalı
`failed`	onay yeterli, işlem başarısız	malı göndermemeli

Kaç onay bekleneceği "Ayarlar" → "Onaylar" bölümünde belirlenir; belirlemezseniz ağın değeri geçerlidir.

Bitcoin, Litecoin ve Dash'te hepsinden önce `pending` gelebilir: transfer henüz mempool'dadır (7. bölüm). Bunun türü her zaman `payment` değil `address` olur ve bu henüz para değildir: böyle bir transfer değiştirilebilir.

Webhook: posta kutunuza mektup

Webhook, hizmetin internetteki adresinize kendiliğinden gönderdiği bir istektir. Adres ve ağ bazındaki abonelikler "Bildirimler" → "Gönderim ayarları" bölümünde belirlenir. Adres internetten erişilebilir olmalıdır: `localhost` olmaz.

Hiçbir şey seçmeden webhook'u açarsanız tüm ağlardaki gelen ve giden transferler gelir, bloklar ise gelmez.

İşte belgelerdeki örneğin bir parçası: TRON'da 30 USDT hakkındaki ilk zil:

```
{ "delivery_id": "3f9c…6b3d90",
  "type": "payment", "event": "detected", "network": "tron",
  "tx_hash": "bb08…c320c", "address": "TToL…QRzRG",
  "direction": "in", "currency": "USDT",
  "amount": "30", "amount_raw": "30000000",
  "confirmations": 1, "required_confirmations": 19,
  "confirmed": false }
```

Tutarlar, hiçbir şey yuvarlanmasın diye dizge olarak gelir; `amount_raw` aynı tutarın virgülsüz halidir (9. bölüm). Hizmetin kataloğunda olmayan bir token için yalnızca `amount_raw` bulunur. Başlıklarda `X-RpcNode-Delivery` (gönderim numarası), `X-RpcNode-Event` (tür) ve `X-RpcNode-Signature` (imza) gelir.

"Gerçekten siz misiniz?"

Webhook adresi herkese açıktır: kapıyı çalıp "size ödeme yapıldı" yazmak isteyen yazabilir. Bu yüzden her gönderim imzalanır. Zaman, nokta ve gövde alınır ve webhook secret'ı ile bunların parmak izi hesaplanır. Bu yönteme **HMAC-SHA256** denir. Secret, "Secret ve IP'ler" bölümündedir; bu, API anahtarının secret'ı değildir.

Deneyelim: kendimiz imzalayalım

Buradaki secret ve gövde uydurmadır:

```
printf '%s' '1787347206.{"type":"test"}' |
  openssl dgst -sha256 -hmac 'demo-secret' | awk '{print $NF}'
```

```
e17efb57778b0e0c758a9bf2c5705049c4c383d2af3a2bd2783c8b665c151cde
```

Gövdedeki veya secret'taki tek bir karakteri değiştirin, bambaşka bir dizi çıkar. Sunucunuz aynı hesabı yapar ve `t=…,v1=…` biçimindeki başlıkla karşılaştırır.

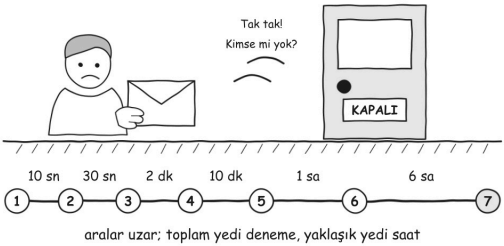
İşte Node.js ile bir alıcı **taslağı**. Onu kendi hazırladığımız isteklerle sınadık, gerçek gönderimlerle değil: onlar için kişisel panel gerekir. 300 saniyelik kontrol, dünkü bir gönderimin yeniden yollanmasını engeller.

```
import { createServer } from 'node:http'
import { createHmac, timingSafeEqual } from 'node:crypto'

const SECRET = process.env.RPCNODE_WEBHOOK_SECRET
const seen = new Set()      // gerçek mağazada bu bir veritabanıdır
createServer(async (req, res) => {
  const parts = []
  for await (const part of req) parts.push(part)
  const body = Buffer.concat(parts)   // baytlar geldiği gibi
  const [, t, v1] = /^t=(\d+),v1=([0-9a-f]{64})$/
    .exec(req.headers['x-rpcnode-signature'] ?? '') ?? []
  const ours = createHmac('sha256', SECRET)
    .update(`${t}.`).update(body).digest()
  const ok = v1 && Math.abs(Date.now() / 1000 - t) < 300 &&
    timingSafeEqual(ours, Buffer.from(v1, 'hex'))
  if (!ok) return res.writeHead(401).end()
  const id = req.headers['x-rpcnode-delivery']
  if (!seen.has(id)) {
    seen.add(id)
    console.log(id, body.toString())  // kendi işiniz buraya
  }
  res.writeHead(200).end()
}).listen(3000)
```

Evde bulunamazsanız

Sunucu en geç 10 saniye içinde "iki yüzlü" bir kodla yanıt verirse gönderim başarılıdır. Vermezse tekrarlanır, yedinci başarısızlıktan sonra bırakılır. Bu yüzden:

- Aynı olay iki kez gelebilir. Gönderim numarası aynıdır; ona göre ayıklayın.
- Sıra garantisi yoktur: `confirmations`, `detected`'ı geçebilir.
- Uzun süren işi yanıt verdikten **sonra** yapın.



Adresinizi önceden bir test gönderimiyle sınayın: ücretsizdir. Paneldeki gönderim günlüğü kayıtları bir gün saklar.

Soket: açık hat

İkinci yol **WebSocket**'tir: programınız bağlantıyı kendisi açıp dinler; bunun için internette bir adresiniz olması gerekmez. Bu, 11. bölümdeki `eth_subscribe` aboneliği değildir: orada node ağı anlatır, burada hizmet sizin adreslerinizi ve siparişlerinizi. Gövde webhook'takiyle aynıdır ama siz yokken kaçanlar **tekrarlanmaz**.

Protokol Pusher ile uyumludur (`pusher-js` kitaplığı iş görür), anahtar API anahtarınızın numarasıdır. Soket ayrıca, "WebSocket ile gönderim" anahtarıyla açılır. İki yolu birden açarsanız her haber iki kez gönderilir ve iki kez ücretlendirilir.

Bu kaça gelir

Olaylar da istekler gibi aynı kredileri harcar. TRON veya Ethereum ana ağında bir adres olayı 27 Kasım 2026 itibarıyla 450 krediдir; **işlem başına bir kez**: onaylar ve sonuç ücretsizdir. Sunucunuz sonradan kabul etmese bile gönderim ücretlidir.

Ethereum'da 30 günlük bir ayı kabaca hesaplayalım; tablodaki her istek 17 kredi:

Ödemeyi nasıl öğreniyoruz	İstek veya olay	Kredi
20 sipariş, 10 saniyede bir `eth_call` (13. bölüm)	5.184.000	88.128.000
her blok için tek `eth_getLogs`, 12 saniyede bir	216.000	3.672.000
Address Watch, ayda 200 transfer	200	90.000

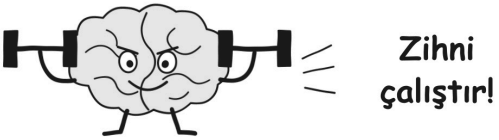
Starter'da 50 milyon kredi vardır: ilk yöntem buna sığmaz. Bu yalnızca bir örnektir. Transfer çoğaldıkça olaylar pahalılaşır; başkalarının adresinize yaptığı küçük transferler de sayılır.

Tuzak: "olay geldi, gönderiyorum"

- **Her olay ödeme değildir:** `detected` henüz kesin değildir, `payout` sizin harcamanızdır.

- **Token, adıyla değil**, `token` alanındaki sözleşme adresiyle tanınır (9. bölüm).
- **Özel transferleri önceden sınayın,** örneğin sözleşmenin içinden gönderilen ETH'yi (15. bölüm).
- **Adres olayları arşiv değildir:** API üzerinden bir gün erişilebilirler.

Düşünün



1. Mağaza sunucusu bir saat kapalı kaldı. Webhook gönderimlerine ne olur, soketteki haberlere ne olur?
2. Aynı `X-RpcNode-Delivery` değerli iki gönderim geldi. Para kaç kez hesaba geçirilmeli?
3. İzlenen adreslere TRON'da ayda 1000 transfer, gönderim yalnızca webhook ile. Kaç kredi eder?

Cevaplar: 1) webhook'lar yedi deneme bitene kadar tekrarlanır; soket kaçanı tekrarlamaz. 2) Bir kez. 3) 450.000.

Özet

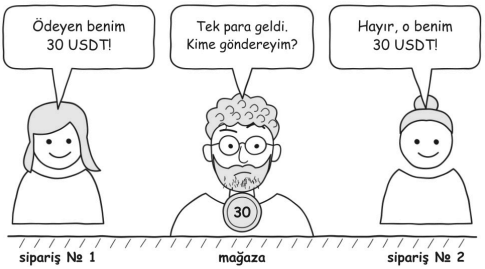
- Durmadan sormayın: transferi size bildirsinler.
- İmzayı doğrulayın, tekrarları ayıklayın, hızlı yanıt verin.
- Para ilk zilde değil, `confirmed` ile ulaşmış olur.

Sırada: 27. bölüm. Ödemeler ve işlem izleme.

Bölüm 27. Ödemeler ve işlem izleme

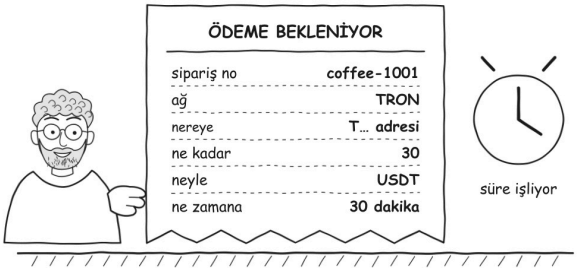
“Para geldi. Peki kimin?”

Artyom’a bir olay geldi: adrese 30 USDT girdi. Sipariş listesini açıyor, orada otuzluk iki sipariş var. Olay adresi ve tutarı bilir ama müşterinin sepetinden habersizdir.



İnceleyelim: ödeme bir beklentidir

Ödeme (paneldeki “Ödemeler” bölümü) önceden kaydedilmiş bir beklentidir: şu sipariş, şu adrese, şu saate kadar şu tutarı bekliyor. Uygun bir transfer gelince hizmet sipariş numarasını kendisi söyler.



Adresi siz verirsiniz. Para müşteriden doğrudan size gider; hizmetin gizli ifadeye ihtiyacı yoktur (6. bölüm).

Ödeme “Yeni ödeme” düğmesiyle veya `POST /v1/payments` isteğiyle oluşturulur:

Alan	Nedir
`order_id`	sizin sipariş numaranız, tekrar etmez
`network_slug` , `address`	ağ ve o ağdaki adresiniz
`amount`	**dizge olarak** tutar: `"30"` , `"12.5"`
`token_contract`	token sözleşmesi; yoksa ağın coini beklenir
`ttl_seconds`	saniye cinsinden süre; yoksa bir saat, en fazla 30 gün

Deneyelim: anahtarsız kapıyı çalalım

```
curl https://api.rpcnode.dev/v1/payments
```

```
{"ok":false,"error_code":"public_api.key_missing",
 "error":"Missing X-Api-Key header."}
```

Bu, 401 kodlu bir yanıt: "siz kimsiniz?". Bu API'ye giden her istek API anahtarının secret'ı ile imzalanır ("API anahtarları" bölümü). Dört satır imzalanır: zaman, metot, yol ve gövdenin parmak izi.

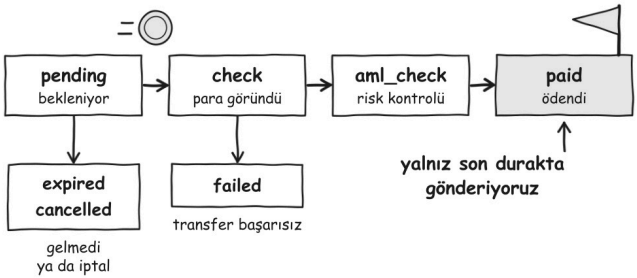
İşte kabuk diliyle bir **taslak**. Sözdizimini ve imza hesabını kontrol ettik ama göndermedik: kişisel anahtar gerekiyor.

```
BODY='{"order_id":"coffee-1001","network_slug":"tron",
"address":"ADRESİNİZ","amount":"30","ttl_seconds":1800,
"token_contract":"TR7NHqjeKQxGTCi8q8ZY4pL8otSzgjLj6t"}'
TS=$(date +%s)
HASH=$(printf '%s' "$BODY" | openssl dgst -sha256 | awk '{print $NF}')
SIG=$(printf '%s\nPOST\n/v1/payments\n%s' "$TS" "$HASH" |
  openssl dgst -sha256 -hmac "$SECRET" | awk '{print $NF}')

curl https://api.rpcnode.dev/v1/payments \
  -H 'Content-Type: application/json' \
  -H "X-Api-Key: $KEY_ID" -H "X-Timestamp: $TS" \
  -H "X-Signature: $SIG" -d "$BODY"
```

Yanıtta `payment_id` , `pending` durumu ve `expires_at` süresi gelir. İsteği aynı `order_id` ve aynı ağ, adres, token ve tutarla tekrarlarsanız aynı ödeme döner. Farklılarıyla gönderirseniz `payment.order_exists` hatası alırsınız.

Ödemenin yolu

Her adım bir `payment_order` olayı olarak gelir: webhook ile veya sokete (26. bölüm).



`event`	Ne oldu
`check`	para görüldü, onay bekleniyor
`confirmations`	onay sayısı arttı
`aml_check`	onay yeterli, risk kontrolü yapılıyor
`paid`	ödendi: mal gönderilebilir
`expired`, `cancelled`	süre doldu veya bekleyen ödemeyi iptal ettiniz
`failed`	ağdaki işlem başarısız oldu (her şeyi bu duruma dayandırmayın: ödemenin yalnızca süresi de dolabilir)

AML, gönderilen paranın kötü bir geçmişle bağlantılı olup olmadığının kontrolüdür. Bu kontrol bulunan yerde `paid` olayındaki `aml` nesnesi 0 ile 100 arasında bir `score` puanı (yükseldikçe risk artar) ve `risk_level` düzeyi taşır. Ayrı bir ücreti yoktur. Her ağda yoktur: olmayan yerde ödeme hemen `paid` olur.

Bu bir ipucudur; hüküm veya hukuki görüş değildir: riskli bir transferle ne yapacağınıza siz karar verirsiniz. Ödeme `aml_check` durumunda takıldıysa bu henüz ödeme sayılmaz; destek ekibine yazın.

Belgelerdeki örneğin bir parçası:

```
{ "type": "payment_order", "event": "paid", "order_id": "ord_123",
  "network": "tron", "status": "paid", "tx_hash": "bb08…c320c",
  "received_amount_raw": "30000000",
  "confirmations": 19, "required_confirmations": 19,
  "aml": { "score": 0, "risk_level": "low" } }
```

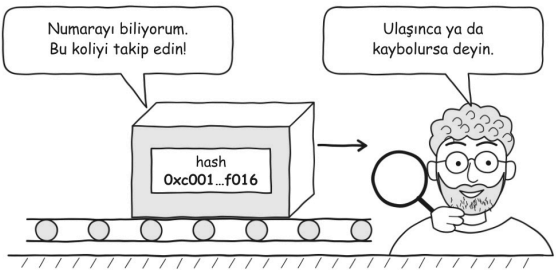
Küçük yazılar

- **Gerekenden azı sayılmaz.** Ödeme beklemeye devam eder ve süresi dolar. Belgelerde `underpaid` durumu geçer ama mağazayı onun üzerine kurmayın: eksik ödemeyi adres olayından fark edin.
- **Gerekenden fazlası sayılır.** Gerçekte ne kadar geldiğini `received_amount_raw` söyler.
- **Geç kalan transfer ödeme sayılmaz.** Zamanında görülen transfer ise süre dolduktan sonra da onay toplamaya devam eder.
- **Her siparişe ayrı adres.** O zaman iki "otuzluk" karışmaz (28. bölüm).
- **Adres slotu kendiliğinden boşalmaz**, `paid` veya `expired` sonrasında da: izlemeyi silin. Ödeme iptal edilince, yalnızca onun için eklenen adres kendiliğinden kaldırılır.

Ödemeler ücretli planlarda vardır: Starter'da aynı anda 10 açık ödeme ve ayda 200 ödeme. TRON veya Ethereum ana ağında bir ödeme 750 kredidir ve yalnızca para görüldüğünde kesilir. Free'de bir kum havuzu vardır: test ağlarında 3 açık ödeme; şu an bu TRON Nile'dır. Oradaki risk kontrolü gerçek değildir ve `"sample": true` ile işaretlenir.

İşlem izleme

Tersi de olur: hash zaten elinizdedir. Artyom bir müşteriye parasını iade etti veya tedarikçiye ödeme yaptı ve ulaşıp ulaşmadığını bilmek istiyor.



"Bildirimler" → "İşlemler" → "Hash ekle" veya `{"network_slug": "tron", "tx_hash": "İŞLEM_HASH"}` gövdeli `POST /v1/tx-watches`. Hash, ağda görünmeden önce de eklenebilir.

`tx` olayları gelir: `detected`, ardından `confirmations` ve sonunda `confirmed` veya `failed`. Hash hiç bulunamazsa `cancelled`. Olaylar, bu ağ için gelen veya giden transferler açıksa gelir.

Açık bir izleme bir slot tutar ve bitince onu boşaltır. Starter'da 15 slot vardır, Free'de bu hizmet yoktur. İşlem bir kez ücretlendirilir: TRON ve Ethereum'da 150 kredi.

Deneyelim: onayları kendimiz sayalım

Böyle her izlemenin perde arkasında yapılan budur. 15. bölümdeki USDT transferini alalım:

```
URL=https://public-ethereum-mainnet.rpcnode.dev
H=0xc0015ff5178d72a68831b133187c3f9f1fbee2e440d3af311d0b32110e2cf016
curl -s $URL -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getTransactionReceipt",
  "params":["'$H'"]}' | grep -oE '"(status|blockNumber)":"0x[0-9a-f]+"' | sort -u
sleep 1
curl -s $URL -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":2,"method":"eth_blockNumber","params":[]}'
```

```
"blockNumber":"0x194646a"
"status":"0x1"
{"jsonrpc":"2.0","id":2,"result":"0x19464c5"}
```

`0x1` durumu başarı demektir. Transferin bloğu ilk onay sayılır; bu yüzden `echo $((0x19464c5 - 0x194646a + 1))` komutu **92** verir. Sizde daha fazla çıkacak.

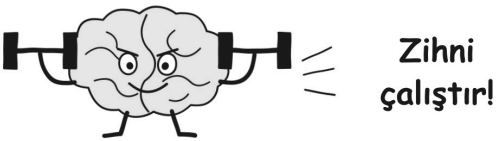
Tuzak: aynı para için iki zil

Ödeme, adresi izlemeye alır. Bu yüzden tek bir transfer için hem `payment` (adres olayı) hem de `payment_order` (ödeme olayı) gelebilir. Her biri kendi fiyatından ücretlendirilir.

> **Kural:** siparişi yalnızca `event` = `paid` olan `payment_order` kapatır. Sipariş numarasını, `payment_id`'yi, tokeni ve tutarı da karşılaştırın.

Ödemeyi durmadan sormayın da: API'nin kendi limiti vardır, hesap başına dakikada 100 istek. Bir kesintiden sonra mutabakat için sorulur: `GET /v1/payments/orders/{orderId}`.

Düşünün



1. 30 USDT'lik ödemeye 29,5 geldi. Ödemeye ne olur?
2. Süre doldu ve transfer bundan bir dakika sonra görüldü. Para nerede, durum ne?
3. `check` olayı geldi. Kahve gönderilebilir mi?

Cevaplar: 1) ödenmiş sayılmaz ve süresi gelince dolar. 2) Para adresinizde, ödeme `expired` kalır: elle halledersiniz. 3) Hayır, `paid` bekleyin.

Özet

- Ödeme transferi siparişe bağlar; hash izleme ise işlemin akıbetine.
- Mal ilk zilde değil, `paid` ile gönderilir.
- Eksik ödemeler, gecikmeler ve dolu slotlar sizin sorumluluğunuzdur.

Beşinci kısım bitti. Artyom uç nokta almayı biliyor, isteğin yolunu ve planları anlıyor, parayı da sorgulamadan öğreniyor.

Sırada: 6. kısım, 28. bölüm. Mağazayı kuruyoruz: fatura, adres, ödeme bekleme, onaylar.

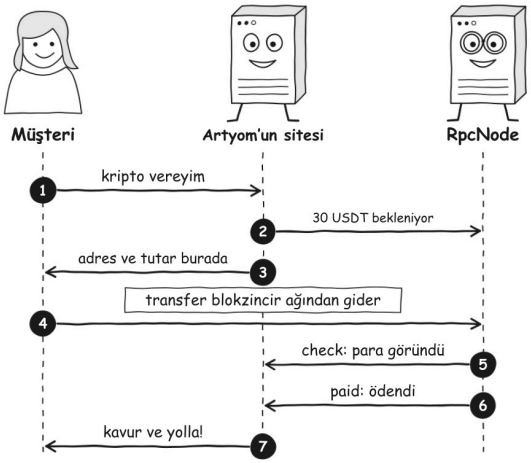
Bölüm 28. Mağazayı kuruyoruz: fatura, adres, ödeme bekleme, onaylar

"Tek tek hepsi anlaşılır. Ya birlikte?"

Artyom adresleri, onayları, webhook'ları ve ödemeleri biliyor. Ve işte müşteri "Kripto parayla öde" düğmesine basıyor. Site bundan sonra adım adım ne yapmalı?

İnceleyelim: kimler işin içinde

Dört katılımcı var. Cüzdanıyla **müşteri**. Sipariş veritabanıyla **mağazanın sitesi**. Ağı izleyen **RpcNode**. Bir de anahtarların durduğu **Artyom'un cüzdanı**. Cüzdan, sitenin hizmetle konuşmasına karışmaz.

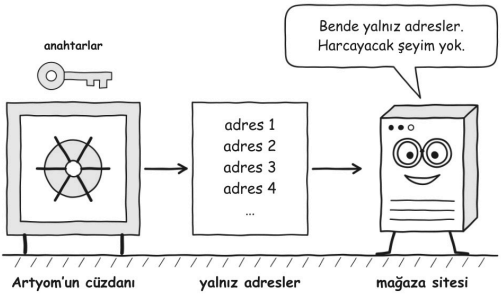


4. adıma bakın. Para siteye de RpcNode'a da gitmiyor. Doğrudan Artyom'un adresine iniyor; RpcNode onu yalnızca blokta fark ediyor.

Sunucuda anahtar tutmadan adres nereden bulunur

6. bölümde söz vermiştik: her siparişe ayrı adres. Aynı yerde şunu da söylemiştik: özel anahtarın mağaza sunucusunda yeri yok. Bu ikisi nasıl bağdaşacak?

Adresler anahtarlardan ayrı olarak hazırlanabilir.



İki yol var.

	Adres listesi	Genişletilmiş açık anahtar
Sunucuda ne durur	hazır adresler, sipariş başına bir tane	tek bir anahtar (çoğu zaman `xpub` ile başlar); program adresleri ondan kendisi hesaplar
Nerede işe yarar	her ağda	Bitcoin, Ethereum, TRON ve akrabaları; Solana ve TON'da bu yol işlemez
Sunucu ele geçirilirse	listedeki adresler görünür	bütün adresleriniz ve onlara gelen her şey görünür
Zahmeti	vaktinde doldurmak	bir kez ayarlamak

Böyle bir anahtarla para harcanamaz. Ama BIP-32 standardı uyarır: bu anahtar, aynı daldaki özel anahtarlardan **yalnızca biriyle birlikte**, dalın bütün özel anahtarlarının hesaplanmasına imkân verir. Bu yüzden böyle bir cüzdanın anahtarlarını tek tek dışa aktarmayın. Yeni başlayan için listeyle başlamak daha kolaydır.

Listeyi, anahtarları internetteki bir sunucuda durmayan bir cüzdandan alın. **Donanım cüzdanı** kullanışlıdır: anahtarları kendi içinde saklayan ayrı bir cihaz. Yüklemeden önce birkaç adresi cihazın ekranındakiyle karşılaştırın. TON'da ise müşteriye adresi `UQ` yazımıyla gösterin (19. bölüm).

Deneyelim: veritabanı karıştırmaya izin vermez

Adres dağıtmak veritabanının işidir. İşte oyuncak bir tane: iki boş adres ve üç sipariş. `sqlite3` programı gerekir; pek çok sistemde zaten kuruludur.

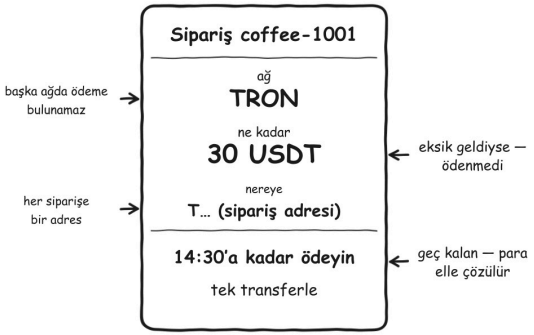
```
sqlite3 :memory: <<'SQL'
CREATE TABLE addresses(address TEXT PRIMARY KEY, order_id TEXT UNIQUE);
INSERT INTO addresses(address) VALUES ('adres-1'), ('adres-2');

UPDATE addresses SET order_id = 'coffee-1001' WHERE address =
  (SELECT address FROM addresses WHERE order_id IS NULL ORDER BY address LIMIT 1)
  RETURNING order_id, address;
UPDATE addresses SET order_id = 'coffee-1002' WHERE address =
  (SELECT address FROM addresses WHERE order_id IS NULL ORDER BY address LIMIT 1)
  RETURNING order_id, address;
UPDATE addresses SET order_id = 'coffee-1003' WHERE address =
  (SELECT address FROM addresses WHERE order_id IS NULL ORDER BY address LIMIT 1)
  RETURNING order_id, address;
SELECT 'üçüncüye adres kalmadı' WHERE changes() = 0;

UPDATE addresses SET order_id = 'coffee-1001' WHERE address = 'adres-2';
SQL
```

```
coffee-1001|adres-1
coffee-1002|adres-2
üçüncüye adres kalmadı
Runtime error near line 15: UNIQUE constraint failed: addresses.order_id (19)
```

Üçüncü faturayı kesecek adres yok: listeyi doldurma zamanı. Son komut ise 1001 numaralı siparişe ikinci bir adres vermek istedi, veritabanı reddetti. Bunu `PRIMARY KEY` ve `UNIQUE` sözcükleri bekler: bir adres bir kez yazılır ve bir siparişin birden fazla adresi olmaz.

Eski adresleri ikinci tura sokup yeniden dağıtmayın. Geçmiş bir siparişin geç kalan transferi yenisini ödemiş olurdu.

Faturayı kesiyoruz



Müşteriye beş şey gerekir: ağ, coin, kesin tutar, adres ve süre. Bir de rica: hepsini **tek transferle** göndersin, çünkü iki yarım birleştirilmez. Uyarın: cüzdan veya borsa ücreti tutarın içinden keserse daha az para ulaşır.

Ruble veya dolar fiyatlarını sipariş anında coine çevirin ve kuru süre sonuna kadar sabit tutun. Bu yüzden süre kısa tutulur, örneğin yarım saat.

Şimdi 27. bölümdeki ödeme gerekiyor. Aşağıda Python ile bir **taslak** var. İçindeki `rpcnode` fonksiyonu imzalı istek gönderir: onu 27. bölüme bakarak yazın veya RpcNode'un resmi istemcisini kullanın (Python, TypeScript, PHP, Go ve Kotlin için mevcut). Veritabanı aynı, iki tablolu.

```
USDT = 'TR7NHqjeKQxGTCi8q8ZY4pL8otSzgjLj6t'    # TRON'daki USDT sözleşmesi
# addresses(address, order_id)
# orders(id, payment_id, amount_raw, status — başta 'waiting')

def take_address(db, order_id):
    with db:                                    # tek veritabanı işlemi
        db.execute('INSERT OR IGNORE INTO orders(id) VALUES (?)',
                   (order_id,))
        mine = db.execute('SELECT address FROM addresses'
                          ' WHERE order_id=?', (order_id,)).fetchone()
        if mine:
            return mine[0]                      # tekrarda aynısı
        free = db.execute(
            'UPDATE addresses SET order_id=? WHERE address='
            ' (SELECT address FROM addresses WHERE order_id IS NULL'
            '  ORDER BY address LIMIT 1) RETURNING address',
            (order_id,)).fetchall()
        if not free:
            raise RuntimeError('adres kalmadı')
        return free[0][0]
```

```
def checkout(db, rpcnode, order_id, amount):   # amount — dizge
    address = take_address(db, order_id)
    reply = rpcnode('POST', '/v1/payments', {
        'order_id': order_id, 'network_slug': 'tron',
        'address': address, 'amount': amount,
        'token_contract': USDT, 'ttl_seconds': 1800})
    if not reply['ok']:
        raise RuntimeError(reply['error_code'])
    payment = reply['payment']
    with db:
        db.execute('UPDATE orders SET payment_id=?, amount_raw=?'
                   ' WHERE id=?', (payment['payment_id'],
                   payment['amount_raw'], order_id))
    return payment        # müşteriye: address, amount, expires_at
```

Burada asıl olan sıralamadır: adres, hizmete başvurmadan **önce** siparişe bağlanır. Bağlantı koptuysa tekrar çağrı aynı adresi ve sipariş numarasını alır, hizmet de aynı ödemeyi döndürür. Bunu uydurma bir `rpcnode` fonksiyonuyla sınadık; taslağı gerçek hizmetle çalıştırmadık.

Ödemeyi her sepet için değil, müşteri kripto parayla ödemeyi seçtiğinde oluşturun: planın aylık limiti, ödenmeyenler dahil oluşturulan ödemeleri sayar.

RpcNode ödemeyi adres, coin ve tutardan tanır. 19. bölümdeki yorum ve 18. bölümdeki `reference` etiketi burada rol oynamaz: ödemede böyle bir alan yoktur. 20. bölümdeki XRP Ledger ve Stellar, ödeme desteği olan ağlar listesinde değildir (26. bölüm).

Ödemeyi beklemek

Sipariş sayfası birkaç saniyede bir **sizin sunucunuza** sorar, sunucu da kendi veritabanından yanıtlar. Veritabanını webhook'lar günceller. RpcNode'a doğrudan tarayıcıdan sorulmaz:

anahtarın secret'ı müşterinin eline geçerdi; üstelik API limiti hesap başına dakikada 100 istektir.

Durum	Müşteri ne görür	Mağaza ne yapar
`pending`	"Transferi 14:30'a kadar bekliyoruz"	hiçbir şey
`check`	"Transferi görüyoruz. Onay 3 / 19"	göndermez
`aml_check`	"Transferi kontrol ediyoruz"	bekler
`paid`	"Ödendi, siparişi hazırlıyoruz"	gönderir
`expired`	"Süre doldu. Bu faturaya transfer yapmayın"	faturayı kapatır
`failed`	"Transfer gerçekleşmedi"	baştan başlamayı önerir
`cancelled`	"Fatura iptal edildi"	faturayı kapatır

Transfer bloğa girmedikçe ödeme sessiz kalır. Bu yüzden "ödenmedi" değil, "bekliyoruz" yazın.

Kaç onay beklemeli

Bu, her ağ için ayrı olmak üzere "Ayarlar" → "Onaylar" bölümünde belirlenir. Belirlemezseniz ağın değeri geçerlidir. Gereken sayı ödemede, `required_confirmations` alanında görünür.

Varsayılandan düşük değer koymak risklidir: yeni bloklar bazen değiştirilir (8. bölüm). Üstelik sayı sipariş başına değil, ağ başına tektir. Bir çuval çekirdek için bir fincan kahveden daha uzun beklemek istiyorsanız `paid` sonrasında biraz daha bekleyin ve 27. bölümdeki gibi kendiniz kontrol edin.

Yer açın

İki tür yer vardır. **Ödeme slotu**, ödeme `paid`, `expired`, `failed` veya `cancelled` olunca kendiliğinden boşalır. **Adres slotu** kendiliğinden boşalmaz (iptal hariç, 27. bölüm).

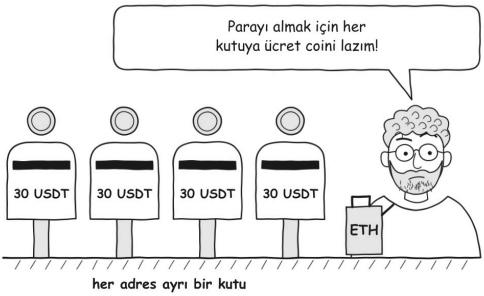
Starter'da bunlar 10 açık ödeme, izlenen 15 adres ve ayda 200 ödemedir. On birinci açık ödeme `payment.slot_limit` reddi alır. Bütün adresler doluyken ise `watch.address_limit` gelir.

Kapanan siparişi bir süre bekletin; transfer gecikmiş olabilir. Sonra izlemeyi kaldırın: `GET /v1/address-watches?network_slug=tron&address=…` isteğiyle `watch_id`'sini bulun ve `DELETE /v1/address-watches/{watch_id}` gönderin. Bundan sonra adresle ilgili haber gelmez, oysa para ona hâlâ ulaşabilir.

Adres ekleme ve silmeler de sınırlıdır: günlük, haftalık ve aylık; slot ne kadar çoksa pay o kadar büyüktür. Her siparişe yeni adres bu payı tüketir. İndirim kampanyası günü, boş slot varken bile `watch.quota` alabilirsiniz. Çok sipariş bekliyorsanız sınırı önceden öğrenin.

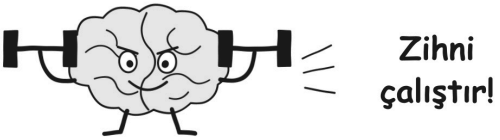
Tuzak: yüz adrese dağılmış para

Sipariş başına adres hesap tutmak için kullanışlıdır ama para sonra dağınık durur. Bir adresten USDT göndermek için o adreste ücret coini gerekir (9. ve 17. bölümler): Ethereum'da ETH, TRON'da TRX veya enerji. Yüz sipariş, yüz ayrı transfer demektir. Bitcoin'de de çok sayıda küçük "parça" gönderimi pahalılaştırır (14. bölüm). Küçük tutarlı satışlar için ağ seçmeden önce bunu hesaplayın.



Bir şey daha. BIP-44 standardını izleyen cüzdanlar, art arda 20 boş adresten sonra aramayı bırakır. Yarıda bırakılan sipariş çoksa cüzdan, uzaktaki bir adrese gelen ödemeyi göstermeyebilir. Para yerinde durur ama cüzdanın daha ileriyi arayıp arayamadığını kontrol edin.

Düşünün



1. Müşteri ödeme sayfasını yeniledi. Yeni adres ve yeni ödeme gerekir mi?
2. Artyom'un Starter'da zaten 10 açık ödemesi var. On birinci müşteri geliyor. Ne olur?
3. Ödeme sayfası RpcNode'a doğrudan tarayıcıdan soruyor. İki sorun sayın.

Cevaplar: 1) hayır: aynı sipariş numarası ve aynı adres. 2) `payment.slot_limit` *reddi: beklemesini rica edin veya başka bir ödeme yolu önerin. 3) Anahtarın secret'ını müşteri görür ve istek limiti çabucak biter.*

Özet

- Anahtarlar cüzdanda kalır, sitede yalnızca adres bulunur: bir sipariş bir adres, tekrar yok.
- Müşteri sizin sitenize sorar, site haberleri webhook'lardan öğrenir.
- Slotları ve parayı toplama masrafını önceden hesaplayın.

Sırada: 29. bölüm. İşler ters gidince: hatalar, tekrarlar, takılan işlemler, çifte kayıt.

Bölüm 29. İşler ters gidince: hatalar, tekrarlar, takılan işlemler, çifte kayıt

“Kahve iki kez yola çıkmış!”

Sabah Artyom bir de baktı: tek sipariş için iki paket gönderilmiş. Başka bir müşteri de yazıyor: “Ödedim ama sizde hâlâ ‘bekliyoruz’ yazıyor.” Bunlar mucize değil. Program yalnızca güzel havalar için yazıldığında böyle olur.

İnceleyelim: üç tür dert

Dert	Örnek	İlacı
İsteğiniz yanıtsız kaldı	ödeme oluştururken bağlantı koptu	zararsızca tekrarlamak
Haber yanlış geldi	iki kez, sırasız veya hiç	veritabanında kilit ve mutabakat
Transfer yolunda gitmedi	takıldı, gecikti, eksik geldi	kurallar ve elle inceleme

API isteği ne zaman tekrarlanır

Uzayan araları 23. bölümde inceledik. Ama korkmadan yalnızca **idempotent** olan şey tekrarlanabilir: kaç kez yaparsanız yapın, sonuç bir kez yapılmış gibidir. Ödeme oluşturmak `order_id` sayesinde böyledir. Aynı numara ve aynı verilerle tekrarlayın. Her tekrarı yeniden imzalayın: imzada zaman vardır.

RpcNode’un resmi istemcileri bağlantı kopması ve “beş yüzlüler” sonrasında `POST` isteğini kendiliğinden tekrarlamaz. Yani ödeme oluşturmayı tekrarlamak sizin işiniz.

Farklı API yanıtlarında ne yapmalı:

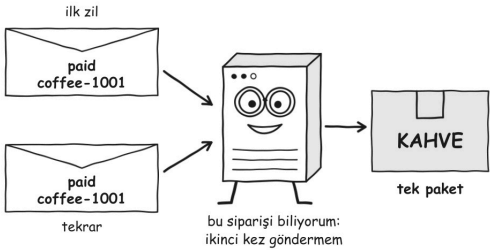
Ne geldi	Ne yapmalı
hiçbir şey; 502, 503, 504	aralıklarla tekrarlayın
`Retry-After` başlıklı 429	söylendiği kadar bekleyin
429 `watch.quota` (28. bölüm)	hemen tekrarlamak boşuna: pay bitti
400	istekte hata var: alanlar `errors` içinde yazılı
401	anahtarı, imzayı ve sunucu saatini kontrol edin
403	plan, slot limiti veya izin verilen IP listesi
409 `payment.order_exists`	bu numara, verileri farklı bir ödemede kullanılmış: numarayı rastgele değiştirmeyin, nedenini araştırın

Ödemenin süresi dolup müşteri yeniden denemek istediğinde durum farklıdır. Eski `order_id` aynı süresi dolmuş ödemeyi döndürür. Yeni bir numara gerekir, diyelim `coffee-1001-2`.

Tek haber, iki kez

İkizler her yerden çıkar:

- 10 saniye içinde yanıt vermediğiniz için webhook tekrarlandı;
- hem webhook hem soket açık;
- aynı para için hem `payment` hem `payment_order` geldi (27. bölüm);
- webhook ile zamanlanmış kontrolünüz aynı anda çalıştı;
- bir çalışan "gönder" düğmesine elle bastı.



Gönderim numarası (26. bölüm) aynı gönderimin tekrarlarını ayıklar. Ama aynı para hakkında başka yoldan gelen ikinci habere karşı korumaz. Asıl kilit siparişin durumudur: "bekliyor"dan "ödendi"ye yalnızca bir kez geçilebilir.

Deneyelim: aynı webhook iki kez

İşte kilidin tamamı. Haberleri biz uydurduk. İki parçayı da tek bir `once.py` dosyasına kaydedip `python3 once.py` ile çalıştırın.

```
import sqlite3
db = sqlite3.connect(':memory:')
db.executescript("""
CREATE TABLE orders(id TEXT PRIMARY KEY, payment_id TEXT,
                    amount_raw TEXT, status TEXT);
CREATE TABLE to_ship(order_id TEXT PRIMARY KEY);
INSERT INTO orders VALUES('coffee-1001','p1','30000000','waiting');
""")

def on_news(n):
    if n['type'] != 'payment_order' or n['event'] != 'paid':
        return 'bu henüz ödeme değil'
    with db:              # içerideki her şey tek veritabanı işlemi
        cur = db.execute(
            "UPDATE orders SET status='paid' WHERE id=? AND"
            " payment_id=? AND amount_raw=? AND status='waiting'",
            (n['order_id'], n['payment_id'], n['amount_raw']))
        if cur.rowcount != 1:
            return 'zaten işlendi veya bizim değil'
        db.execute('INSERT INTO to_ship VALUES (?)', (n['order_id'],))
    return 'gönderiyoruz'
```

Sonra art arda dört haber ve programın yanıtı:

```
paid = {'type': 'payment_order', 'event': 'paid',
        'order_id': 'coffee-1001', 'payment_id': 'p1',
        'amount_raw': '30000000'}
print(1, on_news({**paid, 'event': 'check'}))
print(2, on_news(paid))
print(3, on_news(paid))                      # aynı webhook bir daha
print(4, on_news({**paid, 'event': 'check'}))  # geç kalan haber
print('paket sayısı:', db.execute('SELECT count(*) FROM to_ship').fetchone()[0])
```

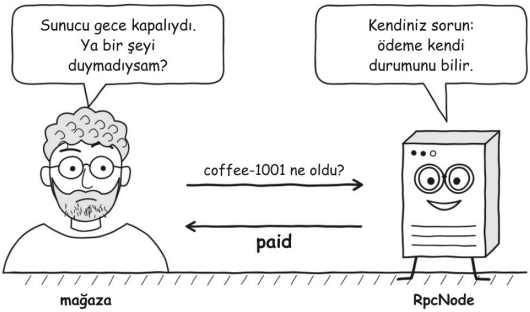
```
1 bu henüz ödeme değil
2 gönderiyoruz
3 zaten işlendi veya bizim değil
4 bu henüz ödeme değil
paket sayısı: 1
```

Paket tek. Her şey tek bir komuta dayanıyor: “`paid` yap, **ama yalnızca** sipariş hâlâ bekliyorsa ve ödeme ile tutar aynıysa”. İşaret ve gönderim görevi birlikte yazılır: ya ikisi ya hiçbiri. Buna **veritabanı işlemi** (database transaction) denir (sözcük blokzincirdekiyle aynı, şey ise farklı). Aynı anda gelen sekiz aynı haberi de sınadık: gönderim tek oldu.

26. bölümdeki taslakta gönderim numaraları bellekte duruyor ve iş yapılmadan önce hatırlanıyordu. Parayla böyle olmaz: program ortada çökerse her şeyi karıştırır. Önce veritabanına yazın, sonra 200 yanıtı verin.

Haber hiç gelmedi

Webhook yedi denemeden sonra bırakılır; bu yaklaşık yedi saattir. Soket ise kaçanı hiç tekrarlamaz. Demek ki haberlerin bir kısmı kaybolabilir. İlacı **mutabakattır**: sizde hâlâ açık görünen siparişleri kendiniz sormak.



`GET /v1/payments/orders/{orderId}` isteği ödemeyi verir. Oradaki adlar biraz farklıdır: `event` yerine `status` alanına bakın, ağın adı `network_slug`, token sözleşmesinin adı `token_contract`'tır. Yanıttan aynı türden bir "haber" oluşturup aynı kilitten geçirin.

Sunucu kesintisinden sonra, soket yeniden bağlandıktan sonra ve zamanlanmış olarak mutabakat yapın. On açık siparişi dakikada bir sormak, izin verilen 100 istekten 10'udur.

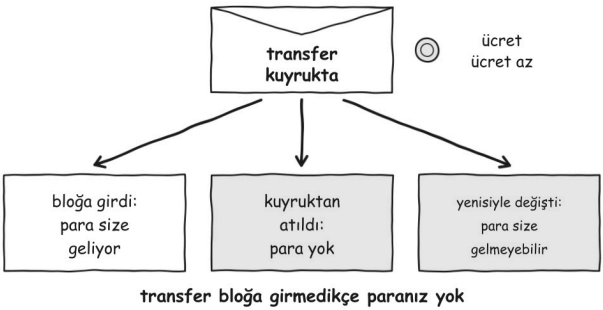
Transfer takıldı

Müşteri hash gönderdi: "Bir saat önce yolladım." Ödeme sessiz. Node'a böyle bir işlemi bilip bilmediğini soralım. Buradaki hash uydurmadır:

```
curl -s https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getTransactionByHash","params":
  ["0x0000000000000000000000000000000000000000000000000000000000000001"]}'
```

```
{"jsonrpc":"2.0","id":1,"result":null}
```

`null`, "bu node böyle bir şey bilmiyor" demektir. Sebepler çeşitlidir: yazım hatası, başka ağ, transfer henüz ulaşmamış veya çoktan atılmış. İşlem bulunduysa ama `blockNumber` değeri `null` ise kuyrukta bekliyordur.



Ağ	Uzun bekleyene ne olur
Ethereum ve benzerleri	gönderenin sonraki bütün transferlerini bekletir (15. bölüm). Aynı `nonce` ve daha yüksek ücretli bir transferle değiştirilebilir: geth node programı varsayılan olarak en az %10 fazlasını ister
Bitcoin	gönderen onu daha yüksek ücretli bir transferle değiştirebilir. Bitcoin Core programı varsayılan olarak kuyrukta iki hafta bekleyeni atar
TRON	işlemin bir son kullanma süresi vardır, varsayılan bir dakika. Bloğa yetişemediyse artık giremez
Solana	sıradan işlem, atıfta bulunduğu blok taze olduğu sürece geçerlidir: 150 blok, yani yalnızca birkaç on saniye

Mağaza başkasının transferini genellikle hızlandıramaz: gönderenin anahtarı gerekir. Bloğu bekleyin ve göndermeyin. Bu arada ödemenin süresi dolar da transfer yine de geçerse, para adresinize iner ama ödeme `expired` kalır.

Takılan **sizin** gönderiminiz mi, diyelim bir iade? Aynı imzalı transfer bir kez daha gönderilebilir: hash değişmez. Yeni bir transfer ise, birincinin yerini almıyorsa, ikinci kez para demektir. Ethereum'da adresin kuyruğu olup olmadığı şöyle görülür:

```
URL=https://public-ethereum-mainnet.rpcnode.dev
A=0x8f8d1206d1bce12ff892731f8a140ed18b93884e
for tag in latest pending; do
  curl -s $URL -H 'Content-Type: application/json' -d '{"jsonrpc":"2.0",
    "id":1,"method":"eth_getTransactionCount","params":["'$A'","'$tag'"]}'
  echo; sleep 2
done
```

```
{"jsonrpc":"2.0","id":1,"result":"0x67641"}
{"jsonrpc":"2.0","id":1,"result":"0x67641"}
```

İlk sayı adresin bloklara girmiş transfer sayısı, ikincisi bekleyenlerle birlikte. Eşitlerse bu node kuyruk görmüyor. Bu, 15. bölümdeki gönderenin adresi.

Hash izlemeye de dikkat (27. bölüm): `cancelled` yanıtı, hizmetin işlemi bulamadığı anlamına gelir; işlemin artık hiç ortaya çıkmayacağı anlamına değil.

Para geldi ama olması gerektiği gibi değil

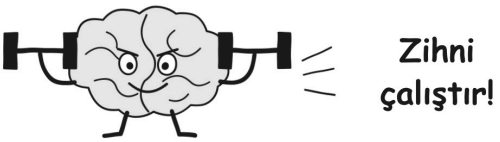
Ne oldu	Ödemeye ne olur
eksik geldi	ödenmiş sayılmaz, süresi gelince dolar; para artık sizde
iki parça halinde	parçalar birleştirilmez: sonuç aynı
fazla geldi	`paid` ; gerçek miktar `received_amount_raw` alanında
`paid` sonrası bir daha gönderildi	ödeme artık beklemiyor: ancak adres olayından öğrenilebilir
süre dolduktan sonra	`expired`
yanlış token veya yanlış ağ	sayılmaz (16. bölüm)
işlem başarısız oldu	para yok, göndermeyin: ödeme `failed` olur veya yalnızca süresi dolar

Her durumda ne yapacağınıza önceden karar verin ve bunu ödeme sayfasında müşteriye yazın.

Tuzak: "geldiği yere geri yollarım"

Müşteri fazla ödedi ve Artyom fazlasını gönderen adrese yollamak istiyor. Ama bu bir borsanın ortak adresi olabilir: parayı müşteri değil borsa alır. İade için adresi ve ağı müşteriye sorun. Siteden değil, cüzdandan kendi elinizle gönderin.

Düşünün



1. Ödeme oluşturma isteğine yanıt gelmedi. Tekrarlanabilir mi? Hangi `order_id` ile?
2. Aynı `paid` hem webhook'la hem soketten geldi. Örnekteki program kaç paket gönderir?
3. 30 USDT yerine 15'er iki transfer gönderildi. 30'luk ödemeye ne olur?

Cevaplar: 1) evet, aynı numara ve aynı verilerle. 2) Bir. 3) Ödenmiş sayılmaz: transferler birleştirilmez.

Özet

- Yalnızca tekrarlaması güvenli olanı tekrarlayın.
- Gönderimi veritabanındaki kilit korur, kaybolan haberleri mutabakat bulur.
- Transfer bloğa girmedikçe para yoktur; tuhaf ödemeler için kurallar önceden gerekir.

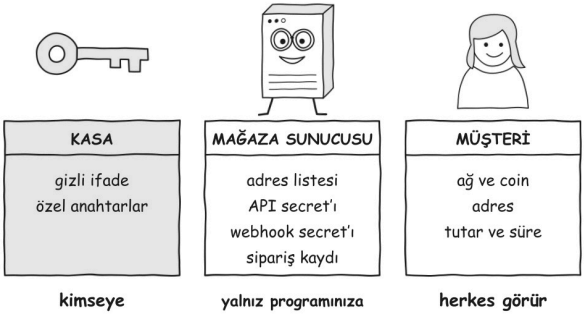
Sırada: 30. bölüm. Güvenlik ve açılış öncesi kontrol listesi.

Bölüm 30. Güvenlik ve açılış öncesi kontrol listesi

"Yarın açılıyoruz. Neyi unuttum?"

Test ağında her şey çalışıyor. Artyom müşterilere çoktan yazıyor: "Kripto kabul ediyoruz!" Ama uykuları kaçıyor. Para gerçek olunca ne ters gidebilir?

İnceleyelim: ne nerede durur



API anahtarının secret'ı coinlerinizi harcamaya yetmez: RpcNode'da özel anahtarlar yoktur. Ama yine de başınıza iş açabilir. Anahtara "İzin verilen IP'ler" belirleyin: böylece çalınan secret yabancı bir bilgisayardan çalışmaz.

İki faktörlü koruma, paroladan başka bir uygulamadan gelen kodun da gerekmesidir. Panelde "Ayarlar" bölümündedir. Aynısını e-postanızda da açın: diğer her şeye erişim çoğu zaman e-posta üzerinden geri alınır.

Bunlardan biri sızarsa ne olur:

Ne sızdı	Yabancı ne yapabilir	Ne yapmalı
gizli ifade, özel anahtar	parayı geri dönüşsüz alır	kalanı hemen yeni bir cüzdana aktarın
API anahtarının secret'ı	ödemeleri, izlemeleri, uç noktaları ve webhook'u yönetir, webhook secret'ını okur	anahtarı silin, yenisini oluşturun, webhook secret'ını değiştirin, ayarları kontrol edin
webhook secret'ı	sahte `paid` haberi üretir	"Yeni secret oluştur"
uç nokta adresi	kredinizi harcar	yenisini oluşturun, eskisini silin (22. bölüm)
panel girişi	coinleriniz dışında bu tablodaki hemen her şey	parolayı değiştirin, iki faktörlü korumayı açın

Secret'lar kodun içinde yaşamaz

Koddaki secret ortak depoya, taşerona giden arşive, ekran görüntüsüne taşınır. Secret'ları sunucu ayarlarında (ortam değişkenlerinde) tutun, bunların bulunduğu dosyayı da yabancılara kapatın.

Deneyelim: kendimizde arayalım

API anahtarının secret'ı `sk_live_` ile başlar. İçinde bir hata olan bir alıştırma klasörü hazırlayıp o hatayı bulalım:

```
mkdir demo
echo "const SECRET = 'sk_live_ornek'" > demo/pay.js
echo "const SECRET = process.env.RPCNODE_API_SECRET" > demo/ok.js
grep -rnE 'sk_live_|PRIVATE KEY|xprv' demo
```

```
demo/pay.js:1:const SECRET = 'sk_live_ornek'
```

Arama birinci dosyayı buldu, ikincisine dokunmadı. `demo` yerine kendi proje klasörünüzü yazın. `xprv`, genişletilmiş **özel** anahtarın başıdır. Bir şey bulunduysa satırı silmek yetmez: değişiklik geçmişinde kalmış olabilir. Secret'ı sızmış sayın ve yenisini oluşturun. Gizli ifade ise böyle yakalanmaz, çünkü sıradan sözcüklerden oluşur: onu bilgisayarda hiç tutmayın.

Gelene inanmayın



Kendi kapımızı sınayalım. 26. bölümdeki alıcı taslağını `receiver.mjs` dosyasına kaydedip çalıştırın:

```
RPCNODE_WEBHOOK_SECRET=demo-secret node receiver.mjs
```

Başka bir terminal penceresinde kapıyı yabancıların kullanacağı üç yolla çalın:

```
URL=http://localhost:3000
BODY='{"type":"payment_order","event":"paid"}'
# 1) imzasız
curl -s -o /dev/null -w '%{http_code}\n' $URL -d "$BODY"
# 2) rastgele imza
curl -s -o /dev/null -w '%{http_code}\n' $URL -d "$BODY" \
  -H "X-RpcNode-Signature: t=$(date +%s),v1=$(printf 'a%.0s' {1..64})"
# 3) imza gerçek ama bir saatlik
T=$(( $(date +%s) - 3600 ))
SIG=$(printf '%s.%s' $T "$BODY" |
  openssl dgst -sha256 -hmac 'demo-secret' | awk '{print $NF}')
curl -s -o /dev/null -w '%{http_code}\n' $URL -d "$BODY" \
  -H "X-RpcNode-Signature: t=$T,v1=$SIG"
```

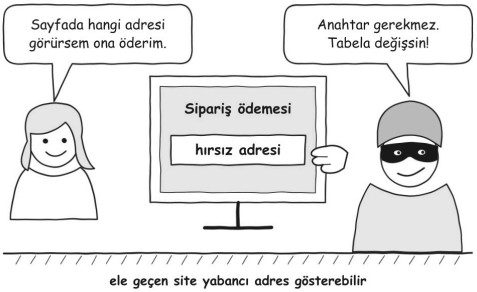
```
401
401
401
```

Üç kez "giremezsin". Herhangi birinde 200 geldiyse kapı açık demektir. `- 3600` yerine `- 0` yazın; üçüncü çalış 200 alır: imza taze ve doğru.

Pahalı siparişler için ikinci bir kilit takın. `paid` alınca ödemeyi kendiniz sorun (`GET /v1/payments/orders/{orderId}`) ve `payment_id` , adres, token, tutar ile durumu karşılaştırın. Böylece webhook secret'ı çalınsa bile bedava kahve alınamaz.

Ayrıca webhook'ları `https` üzerinden alın ve gelen isteğin boyutunu sınırlayın: 26. bölümdeki taslak bunu yapmıyor. Buna ek olarak yalnızca "Secret ve IP'ler" bölümündeki adresleri içeri alabilirsiniz. Bu imzaya bir ektir, onun yerine geçmez.

Değiştirilen tabela



Hırsızın anahtar ele geçirmesine gerek yoktur. Siteyi ele geçirip müşteriye kendi adresini göstermesi yeter. İşe yarayanlar:

- sitenin yönetim panelinden adres listesi sessizce değiştirilememeli;
- ara sıra, dağıtılan adresleri başka bir bilgisayardan cüzdanınızla karşılaştırın;

- zaman zaman kendi mağazanızdan alışveriş yapın ve sizden nereye ödemeniz istendiğine bakın;
- sunucudaki programları güncelleyin, yönetim paneline iki faktörlü korumayla girin.

Hırsızlar size şahsen de yazar: “destek ekibi”, “takılan ödemeyi geri alın”, benzer bir adresten minicik bir transfer (6. bölüm). Gerçek destek gizli ifadeyi sormaz.

Prova



Önce her şeyi test ağında yapın. Test ağlarından ödemeler ve olaylar TRON Nile’da çalışır (26. bölüm). Free’de bir kum havuzu vardır: varsayılan olarak 3 açık ödeme ve 3 adres; oradaki risk kontrolü gerçek değildir. Adresler kum havuzunda da kendiliğinden boşalmaz.

Ne canlandırılacak	Ne çıkmalı
tam tutarı ödemek	tek paket
eksik ödemek	hiçbir şey gönderilmedi, ödemenin süresi doldu
aynı haberi kendinize iki kez göndermek	tek paket
sunucunuzu kapatmak, ödemek, açmak	ödeme bulundu: webhook tekrarıyla veya mutabakatla
imzasız kapıyı çalmak	401

Ardından satış yapacağınız ağda kendinizden küçük, gerçek bir alışveriş yapın. Bu arada cüzdanınızın parayı görüp görmediğini ve onu başka yere gönderip gönderemeyeceğinizi de kontrol edin: ücret coini yetiyor mu.

Yasa ve vergiler

Pek çok ülkede kripto parayla ödeme almak kısıtlanmıştır veya vergilendirilir. Açılmadan önce ülkenizin yasalarını kontrol edin. Bu hukuki bir tavsiye değildir. Bu arada muhasebecinize ödemelerle ilgili hangi kayıtlara ihtiyaç duyduğunu sorun ve onları ilk günden saklayın.

Tuzak: "bir kez kontrol ettim, sonra unuttum"

Mağaza çalışıyor ama çevrede her şey değişiyor. Plan süresi doldu mu hesap Free'ye geçer; orada ana ağda ödeme yoktur (25. bölüm). Krediler olaylara da harcanır. Adres listesi tükenmek üzeredir. Kendinize hatırlatıcılar kurun.

Açılış öncesi kontrol listesi

Anahtarlar ve erişim

- □ gizli ifade yazılı ve hiçbir sunucuda veya bilgisayarda durmuyor;
- □ sitede yalnızca adresler var, birkaçı cüzdanla karşılaştırıldı;
- □ kodda secret yok, projede arama boş dönüyor;
- □ API anahtarında izin verilen IP'ler belirlenmiş;
- □ iki faktörlü koruma panelde, e-postada ve sitenin yönetim panelinde açık.

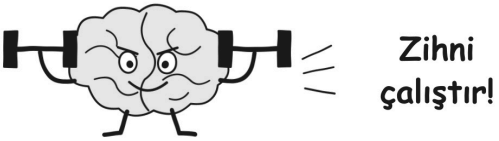
Para alma

- □ bir sipariş bir adres, tekrar yok;
- □ faturada ağ, coin, tutar, adres, süre ve "tek transferle" yazıyor;
- □ token sözleşmesi `GET /v1/network-tokens` ile alındı ve ihraççının sitesiyle karşılaştırıldı;
- □ onay sayısı tutarlarınıza göre seçildi;
- □ webhook imzası doğrulanıyor, gönderimi veritabanındaki kilit koruyor;
- □ açık siparişlerin mutabakatı zamanlanmış olarak yapılıyor.

Çevredeki işler

- □ eksik ödeme, fazla ödeme, gecikme ve iade için kurallar var;
- □ slotlar, krediler, plan süresi ve adres stoku gözetim altında;
- □ parayı toplarken ücretin neyle ödeneceği belli;
- □ sipariş veritabanının yedeği alınıyor ve en az bir kez yedekten geri yüklendi;
- □ prova yapıldı, kendi gerçek alışverişiniz tamamlandı;
- □ ülkenizin yasaları ve vergileri kontrol edildi.

Düşünün



1. API anahtarının secret'ı sızdı. Cüzdanınızdaki coinler alınabilir mi? Yine de derhal ne yapılmalı?
2. Alıcınız imzasız bir mesaja 200 yanıtı verdi. Bunun tehlikesi ne?
3. TRON Nile'daki bütün kontroller geçti. Yeni kontrol yapmadan Ethereum'da ödeme almaya başlanabilir mi?

Cevaplar: 1) hayır, orada cüzdan anahtarı yok; ama API anahtarını silin, yenisini oluşturun, webhook secret'ını değiştirin ve ayarları kontrol edin. 2) Herkes parasız sipariş "ödeyebilir". 3) Hayır: ağ, adresler, ücretler ve onaylar farklı.

Özet

- Paranın anahtarlarını siteden, secret'ları koddan uzak tutun.
- Ne gelen mesaja, ne ekran görüntüsüne, ne de mektuba sözüne bakarak inanın.
- Önce prova, sonra küçük gerçek bir alışveriş, sonra açılış.

Altıncı kısım bitti. Artyom mağazayı kurdu; bozulduğunda ne yapacağını ve onu nasıl koruyacağını biliyor.

Sırada: 31. bölüm. DeFi: aracısız finans.

Bölüm 31. DeFi: aracısız finans

"Coinler öylece durmak zorunda mı?"

Artyom'un elinde cironun bir kısmı USDC olarak birikti. Bir tanıdığı öğüt veriyor: "DeFi'ye koy, orada para kendi kendine para yapıyor." Gişesi ve veznedarı olmayan bu yer neresi?

> Bu bir **finansal tavsiye değil**, bir açıklamadır. Protokol adları yalnızca örnektir. Ülkenizde bu tür hizmetler kısıtlanmış veya vergilendirilmiş olabilir: yasaları kontrol edin.

İnceleyelim: banka ve program

DeFi, yani "merkeziyetsiz finans"; akıllı sözleşmelerin (9. bölüm) yürüttüğü takas, borç ve diğer para hizmetleridir. Ortak kuralları olan bir sözleşme takımına **protokol** denir.

	Banka	DeFi
Para nerede	bankada	bir adreste veya sözleşmede
İşi kim yapar	çalışanlar ve sistemleri	ağdaki program
Ne zaman çalışır	kendi mesai saatlerinde	günün her saati
Kurallar nerede	sözleşme metninde	kodda
Hata yaptınız	bankaya başvurulabilir	genellikle geri alacak kimse yok
Erişimi kaybettiniz	kimliğiniz yardımcı olur	anahtarsız geri alınamaz

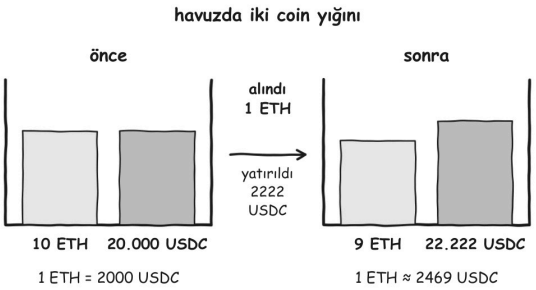
"Aracısız" demek "güvensiz de olur" demek değildir. Koda, onu değiştirebilenlere, fiyat sağlayıcılara ve coin ihraççısına güvenirsiniz.

Borsasız takas: likidite havuzu

Sıradan borsada alıcıya satıcı aranır. DeFi'de çoğu zaman bir **likidite havuzu** kurulur: içinde aynı anda iki coin duran bir sözleşme. Havuza birini koyar, diğerini alırsınız. İkinci bir insanı beklemek gerekmez.

Ne kadar alacağınıza bir formül karar verir. Basit havuzlarda formül şudur: $\mathbf{x \times y = k}$; x ve y iki coinin stoklarıdır. Takas bu çarpımı azaltmamalıdır.

Burada fiyat, stokların oranıdır. Havuzda 10 ETH ve 20.000 USDC olsun. Demek ki 1 ETH yaklaşık 2000 USDC eder ve k = 200.000. Artyom 1 ETH almak istiyor (ücreti şimdilik saymıyoruz). 9 ETH kalacak, demek ki USDC 200.000 : 9 ≈ 22.222 olmalı.



2000 değil, 2222 USDC ödeyecek: **kendi alımı fiyatı yükseltti**. Buna **fiyat etkisi** denir. Yüz kat büyük bir havuzda (1000 ETH ve 2.000.000 USDC) aynı 1 ETH yaklaşık 2002 USDC'ye gelirdi.

Havuzda ETH başka yerlerden pahalı hale gelirse biri oraya ETH getirip USDC alır; fiyatlar yaklaşana kadar sürer. Bu **arbitrajdır**.

Havuza coinleri **likidite sağlayıcıları** koyar ve ücretlerden pay alır. Ama fiyatlar ayrıştığında havuz değerlenen coini verip diğerini toplar. Ücretler hesaba katılmazsa, fiyat iki katına çıktığında yatırımın değeri, coinler öylece dursaydı olacağından yaklaşık %5,7 daha azdır. Buna **geçici kayıp** denir: yatırımı o anda çekerseniz gayet kalıcı olur.

Deneyelim: gerçek bir havuza soralım

Ethereum'da Uniswap v2 protokolünün USDC ve WETH havuzunu alalım. **WETH**, token biçimindeki ETH'dir, bire bir.

Havuz adresi rastgele bir aramadan alınmaz. Havuz "fabrikasının" adresini Uniswap belgelerinden, USDC'nin adresini ihraççısı Circle'dan, WETH'in adresini ethereum.org'dan aldık. Sonra fabrikaya `getPair` sorusunu sorduk: "bu çiftin havuzu nerede?" Yanıt: `0xB4e16d0168e52d35CaCD2c6185b44281Ec28C9Dc`.

Şimdi sıra `getReserves` sorusunda: "stoklar ne kadar?". Bu, 9. bölümdeki `eth_call`'dur: yalnızca okur.

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_call","params":[
  {"to":"0xB4e16d0168e52d35CaCD2c6185b44281Ec28C9Dc",
   "data":"0x0902f1ac"},"latest"]}'
```

27 Kasım 2026'ya bağlayan gece node şöyle yanıt verdi:

```
{"jsonrpc":"2.0","id":1,"result":"0x000000000000000000000000000000000000000000000000000000079ecc8266bc00000000000000
0000000000000000000000000000000010b762ae141ddef4f5000000000000000000000000000000000000000000000000000000006b08b4a
f"}
```

`0x` sonrasında 64'er karakterlik üç sayı gelir:

Sayı (soldaki sıfırlar olmadan)	Nedir
`79ecc8266bc`	USDC stoku, virgülden sonra 6 basamak: 8.378.617 USDC
`10b762ae141ddef4f50`	WETH stoku, 18 basamak: 4933,80 WETH
`6b08b4af`	stokların en son değiştiği an, 1970'ten beri saniye olarak

Hangi coinin önce geldiğini `token0` ve `token1` soruları söyler: bu havuzda ilki USDC. Geriye bölmek kalıyor:

```
python3 -c 'u = 0x79ecc8266bc / 10**6; w = 0x10b762ae141ddef4f50 / 10**18; print(u, w, u / w)'
```

Ekrana `8378617.325244 4933.79553339896 1698.2092728662094` yazılır.

Bu havuzda bir ETH yaklaşık **1698 USDC** ediyordu. Sizde sayılar farklı olacak.

Peki büyük bir parti için ne verirler? Buradaki takas ücreti %0,3: bunun %0,25'i sağlayıcılara, %0,05'i protokole gider. Aynı stoklarla sonuç şöyle:

Veriyoruz	Alacağımız USDC	1 ETH başına	1698,21 fiyatından ne kadar kötü
1 ETH	1692,77	1692,77	%0,3
100 ETH	165.957,86	1659,58	%2,3
1000 ETH	1.408.492,58	1408,49	%17

Hesap, Uniswap'ın yardımcı sözleşmesinin (router) `getAmountsOut` *sorusuna verdiği yanıtlarla son basamağa kadar örtüştü.*

Önemli: stokları bölmek v2 türü havuzlar için geçerlidir. Uniswap v3 ve v4'te yatırımlar fiyat aralıklarına dağıtılmıştır ve fiyat başka türlü hesaplanır.

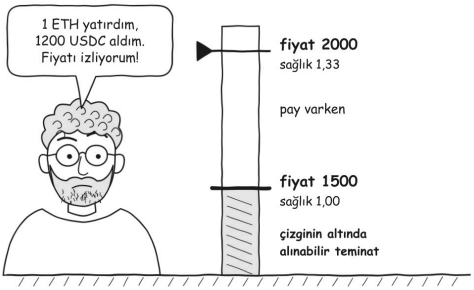
Teminatlı borç ve tasfiye

Borç protokolünde (örnek: Aave) **teminat** yatırılır ve değerinden azı borç alınır. Neden? Diyelim dolara ihtiyaç var ama ETH satmak istenmiyor.

Borcun bir "sağlığı" vardır (health factor):

```
sağlık = teminatın değeri × tasfiye eşiği : borç
```

Uydurma bir örnek: tanesi 2000 USDC'den 1 ETH teminat, eşik %80, borç 1200 USDC. Sağlık: 2000 × 0,8 : 1200 ≈ 1,33. Fiyat 1500'e düşerse tam 1 olur.



Sağlık 1'in altındaysa herhangi bir katılımcı borcun bir kısmını, bazen de tamamını kapatıp karşılığında teminatı primli olarak alabilir. Bu **tasfiyedir**. "Para yatırın" diye bir telefon gelmez: her şeyi program yapar, gece yarısı bile. Borç faizle kendi kendine de büyür.

Stablecoinler: "bir dolar" neye dayanır

9. bölüm: stablecoin bir dolar civarında olmaya yalnızca **çalışır**. Bunu farklı yollarla yaparlar. **İhraççıdaki rezervler**: dolar ve diğer varlıklar (USDT, USDC). Sözleşmelerde **fazlasıyla kripto teminat** (DAI, USDS; teminatta başka stablecoinler de olabiliyor). Teminatsız, coin sayısını değiştiren bir **algoritma**.

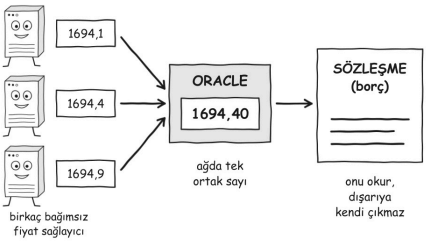
Hiçbir yöntem garanti değildir. Havuzdaki böyle bir coin ucuzlarsa sağlayıcıların elinde biriken tam da o olur.

Kısaca staking

8. bölümdeki validatöre 32 ETH gerekir. Daha küçük bir tutarla **ortak staking havuzu** üzerinden katılınır (bu bir takas havuzu değildir). Karşılığında çoğu zaman bir makbuz token verilir. Riskleri: validatörlere kesilen cezalar, sözleşmedeki bir hata, işletmeciye güven. Gelirin miktarını kimse vaat etmez.

Oracle: sözleşme fiyatı nereden bilir

Borç sözleşmesine ETH'nin fiyatı gerekir. Ama o internete kendisi çıkmaz: her node aynı sonucu elde etmelidir (15. bölüm). Fiyatı ağa bir **oracle** (veri sağlayıcı, kâhin) yazar.



Örnek: Chainlink oracle'ı. "ETH / USD" akışının adresi Chainlink'in kendi rehberinden alındı. `latestRoundData` sorusu en yeni kaydı ister:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_call","params":[
  {"to":"0x5f4eC3Df9cbd43714FE2740f5E3616155c5b8419",
   "data":"0xfeaf968c"},"latest"]}'
```

```
{"jsonrpc":"2.0","id":1,"result":"0x0000000000000000000000000000000000000000000000070000000000008e4f00000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000006b08aefa
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000008e4f"}
```

Beş sayının ikincisi, `2773693800`, virgülden sonra 8 basamaklı fiyattır: **1694,40 dolar**. Havuz 1698,21 USDC gösteriyordu; fark yaklaşık %0,2. Dördüncü sayı kayıt zamanıdır: sorumuzdan 25 dakika önce. Bu akış, fiyat kayıtlı değerden %0,5 uzaklaştığında ve ayrıca zamanlamayla, yaklaşık saatte bir güncellenir.

Neden fiyatı doğrudan havuzdan almıyoruz? Büyük bir işlemle bir anlığına kaydırılabilir. Ama oracle da yanılabilir veya gecikebilir; tasfiye ise onun sayısına göre yapılır.

DeFi uygulaması ağı nasıl okur

Takas veya borç sitesi bir vitrindir. Verileri node'dan alır:

- `eth_call` — şu an ne var: havuz stokları, bakiyeler, oracle fiyatı, borcun sağlığı.
- **Olaylar** (15. bölüm) — ne değişti. v2 havuzu her değişiklikten sonra yeni stoklarla bir `Sync` olayı duyurur. Bunlar `eth_getLogs` ile aranır veya WebSocket üzerinden dinlenir (11. bölüm).
- **Gönderme** — takas defteri değiştirir. Onu cüzdan imzalar, node iletir (7. bölüm) ve gas tutar.

İşte havuzumuzun en yeni bloktaki `Sync` olaylarını soran istek:

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_getLogs","params":[{
  "address":"0xB4e16d0168e52d35CaCD2c6185b44281Ec28C9Dc",
  "topics":["0x1c411e9a96e071241c2f21f7726b17ae89e3cab4c78be50e062b03a9fffbbad1"],
  "fromBlock":"latest","toBlock":"latest"}]}'
```

```
{"jsonrpc":"2.0","id":1,"result":[]}
```

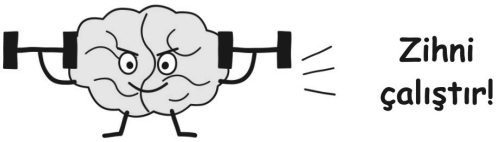
Boş liste de bir yanıttır: bu blokta havuz değişmemiş. `getReserves`'in üçüncü sayısı bunu doğruluyor: son değişiklik sorumuzdan yaklaşık 10 dakika önceydi. Geçmiş değişiklikler, diyelim fiyat grafiği için, artık arşiv veya kendi veritabanınız demektir (10. bölüm).

Tuzak: neler ters gidebilir

Tehlike	Ne olur	Riski ne azaltır
Koddaki hata	para çalınır veya içeride kalır	kod denetimi, ama garanti değil
Sahte token	ad tanıdık, adres yabancı	adresi karşılaştırmak (9. bölüm)
Sınırsız izin (`approve`)	tokenler sonradan alınır	az izin vermek, fazlasını sıfırlamak
Fiyat kayması	beklenenden az gelir	en düşük tutarı belirlemek
Tasfiye	teminat primiyle birlikte gider	bol pay ve gözetim
İkiz site	sitenin kopyası imza ister	adresi kontrol etmek; gizli ifadeyi girmemek
"Garantili gelir"	vaat edip kaybolurlar	katılmamak

Fiyat kayması (slippage), beklenen fiyat ile gerçekleşen fiyat arasındaki farktır. Takas kuyrukta beklerken fiyatı başkalarının işlemleri oynatır, bazen de bilerek. En düşük tutar korur: takas ondan kötü bir değerle gerçekleşmez, ama gas yine harcanır.

Düşünün



1. Havuzda 100 ETH ve 200.000 USDC var. Biri 20 ETH aldı. Kaç USDC yatırdı (ücretsiz hesaplayın)?
2. Tanesi 2000 USDC'den 2 ETH teminat, eşik %80, borç 2400 USDC. Sağlık kaç? Hangi ETH fiyatında 1'e eşit olur?
3. Bir site risksiz günde %3 vaat ediyor ve USDT'nizi sınırsız harcama izni istiyor. Ne yaparsınız?

Cevaplar: 1) 80 ETH kalacak, demek ki USDC 250.000 olacak: 50.000 yatırdı, ETH başına 2500. 2) Yaklaşık 1,33; fiyat 1500 olduğunda. 3) Hiçbir şey imzalamayın ve siteyi kapatın.

Özet

- DeFi'de hizmetleri sözleşmeler yürütür; risk ve sorumluluk büyük ölçüde sizin omzunuzdadır.
- v2 türü havuzda fiyat iki stoktan çıkar ve büyük bir işlem onu kaydırır.
- Uygulama ağı sizinle aynı isteklerle okur: `eth_call` ve olaylar.

Sırada: sonuç. Müşterinin mektubundan kendi mağazanıza.

27 Kasım 2026'da kontrol edildi: developers.uniswap.org, developers.circle.com/stablecoins/usdc-contract-addresses, ethereum.org (wrapped-eth, defi, stablecoins, staking, developers/docs/oracles), aave.com/help/borrowing/liquidations, docs.chain.link.

Sonuç. Müşterinin mektubundan kendi mağazanıza

Her şey bir mektuptaki tek satırla başladı: “Kripto parayla ödeyebilir miyim?” Artyom o zaman node nedir bilmiyor, cüzdanı elle kontrol ediyordu. Onunla birlikte nasıl bir yol yürüdüğünüze bakın.

Artyom artık neler biliyor

Kısım	Ne öğrendik
1. Neden sağlayıcı gerekli	Kendi sunucularının masrafını hesaplamayı ve node’u kendisi mi tutacağına, sağlayıcıdan mı alacağına karar vermeyi
2. Sıfırdan blokzincir	Blokları, anahtarları, imzaları, ücreti, onayları ve tokenleri anlamayı
3. Node’lar ve RPC	Node’a soru sormayı, node türlerini ayırmayı ve geri kalıp kalmadığını kontrol etmeyi
4. Ağlar nasıl çalışır	Ağların neyle ayrıldığını görmeyi ve müşterilere uyanı seçmeyi
5. RpcNode	Uç nokta almayı, çağrıların bedelini hesaplamayı ve ödemelerden sorgulama yapmadan haberdar olmayı
6. Uygulama	Ödeme alma düzenini kurmayı, arızaları atlatmayı ve mağazayı korumayı
7. Sırada ne var: DeFi	Havuzu doğrudan ağdan okumayı ve riskleri dürüstçe tartmayı

Akılda kalacak yedi kural

1. Adres herkese gösterilir; özel anahtar ve gizli ifade hiç kimseye.
2. Her şey birden kontrol edilir: ağ, token sözleşmesinin adresi, alıcı ve tutar.
3. Para, ekran görüntüsü geldiğinde değil, işlem başarılı olup onaylarını topladığında ulaşmış olur.
4. 200 kodu henüz başarı değildir: yanıtın içini okuyun.
5. Haberi kaç kez gelirse gelsin, bir ödeme bir gönderimdir.
6. Uç noktayı ve secret’ları parola gibi koruyun.
7. Sözüne inanmayın: ağa sorun ve yanıtı kontrol edin.

Yarın nereden başlamalı

1. 3. bölümdeki isteği herkese açık node’a tekrar gönderin.
2. Kendi uç noktanızı oluşturun ve bir test ağı seçin (22. bölüm).
3. Oyuncak parayla kendinize bir ödeme canlandırın (28. ve 30. bölümler).
4. 30. bölümdeki kontrol listesinden geçin.
5. Ülkenizin yasalarını ve vergilerini kontrol edin.

Bir şey eskidiyse

Kitap, ağları ve hizmeti 27 Kasım 2026'daki halleriyle anlatıyor. Ağlar güncellenir, fiyatlar ve planlar değişir. Node'unuzun yanıtı kitapla uyuşmuyorsa node'a ve güncel belgelere inanın. RpcNode'un planlarına ve imkânlarına rpcnode.dev sitesinden bakın.

Hoşça kalın

Cironun bir kısmını DeFi'ye koyup koymayacağını Artyom hâlâ düşünüyor. Ama hangi soruları soracağını biliyor. Bütün bu hikâyeden çıkardığı en önemli şey belki de budur.

Bu kitaptaki sayıların bir kısmı eskiyecek, kontrol etme alışkanlığı ise eskimez. Size de işinize de bol şans!



Ek A. Sözlük

Parantez içinde, sözcüğün ayrıntılı anlatıldığı bölüm.

Adres — paranın gönderildiği "kutu numarası" (6).

Ağ ücreti — bloktaki yer için ağa ödenen bedel (7).

Akıllı sözleşme — blokzincire yazılmış program (9).

AML — paranın kötü bir geçmişle bağlantılı olup olmadığının kontrolü (27).

API — bir programın başka bir programa başvurduğu pencere (11).

Arşiv node'u — geçmişteki her bloktaki durumu hatırlar (10).

Blok — "defter sayfası": işlemler, numara, zaman ve önceki bloğun mührü (5).

Blokzincir — ortak kurallarla birlikte tutulan ortak kayıt listesi (4).

Budanmış node — her şeyi denetlemiş, eski blokları ise silmiş node (10).

Chain ID — EVM ağının numarası (16).

Cüzdan — anahtarların saklandığı program veya cihaz (6).

Çifte harcama — aynı coinle iki kez ödeme yapma girişimi (4).

DeFi — akıllı sözleşmelerin yürüttüğü para hizmetleri (31).

Eşitleme — node'un ağa yetişmesi (12).

EVM — sözleşmeleri çalıştıran Ethereum sanal makinesi (15).

Gas — Ethereum ve akrabalarında ücretin hesaplandığı iş ölçüsü (15).

Gizli ifade — anahtarların geri oluşturulduğu 12 veya 24 sözcük (6).

gRPC — programların katı bir şemaya uyan kompakt mesajlarla haberleşme yöntemi (11).

Hash, "mühür" — herhangi bir verinin kısa parmak izi (5).

İdempotentlik — kaç kez tekrarlanırsa tekrarlansın sonucun bir kez yapılmış gibi olması (29).

İmza — transfere anahtarın sahibinin izin verdiğinin kanıtı (6).

İndeksleyici — blokzincir kayıtlarını "kart dizinine" yerleştiren program (10).

İstek limiti (RPS) — saniyede kaç kez sorulabileceği (3, 25).

Kesinlik — bloğun artık değiştirilmeyeceği an (8).

Köprü — değeri bir ağdan diğerine taşıma yolu (16).

Kredi — hizmetin istek harcamasını saydığı birim (3, 25).

L2, "ikinci kat" — transferleri kendi üzerinde yürütüp kayıtlarını Ethereum'a gönderen ağ (16).

Likidite havuzu — içindeki iki coinin formüle göre takas edildiği sözleşme (31).

Madencilik — blok ekleme hakkı için deneme yanılmayla problem çözme (8).

Mempool — blok bekleyen transferlerin kuyruğu (7).

Mutabakat — katılımcıların tek bir zinciri seçmesini sağlayan kurallar (8).

Node — üzerinde ağın programı çalışan bilgisayar: saklar, denetler ve yanıtlar (1).

Nonce — hesabın işleminin sıra numarası (15).

Olay (log) — sözleşmenin ne olduğuna dair duyurusu (15).

Onaylar — işleminizin üzerine binen bloklar, kendi bloğu dahil (5).

Oracle — dışarıdaki bilgiyi, örneğin fiyatı, ağa yazan (31).

Özel anahtar — paraya hükmeden şey; hiç kimseye gösterilmez (6).

REST — her sorunun adreste kendi yolu olduğu soru sorma yöntemi (11).

RPC, JSON-RPC — başka bir bilgisayardaki programdan bir şey yapmasını isteme yöntemi (11).

Sağlayıcı — node'ları tutan ve onlara erişim veren hizmet (1, 3).

Stablecoin — dolar civarında kalmaya çalışan token (9).

Staking — coin teminatıyla ağın korunmasına katılmak (8).

Tasfiye — borç artık teminatla karşılanmadığında teminatın satılması (31).

Test ağı — değeri olmayan coinlerle deneme sahası (22).

Token — akıllı sözleşmenin tuttuğu kayıtlar (9).

Uç nokta (endpoint) — programın node'a soru sorduğu adres (3).

UTXO — Bitcoin ve akrabalarında harcanmamış "para parçası" (14).

Validatör — teminat yatırıp blok oluşturmaya katılan katılımcı (8).

Webhook — bir şey olduğunda hizmetin adresinize kendiliğinden gönderdiği istek (26).

WebSocket — haber almak için kapanmayan bağlantı (11).

Wei — ETH'nin en küçük parçası (1).

Yeniden düzenlenme (reorg) — yeni blokların bir kısmının başka bir dalla değişmesi (8).

Yük dengeleme — isteklerin birkaç node arasında paylaştırılması (23).

Ek B. Kitaptaki ağlar tek sayfada

6., 8. ve 14–20. bölümlerdeki bilgiler, 27 Kasım 2026 itibarıyla. Ağ güncellemeleriyle değişirler.

Bitcoin ve akrabaları (14. bölüm)

Ağ	Coin	Blok yaklaşık	Adres şununla başlar
Bitcoin	BTC	10 dk	`1`, `3`, `bc1`
Litecoin	LTC	2,5 dk	`L`, `M`, `ltc1`
Dogecoin	DOGE	1 dk	`D`
Bitcoin Cash	BCH	10 dk	`bitcoincash:q…`
Dash	DASH	2,5 dk	`X`
Zcash	ZEC	75 sn	`t1`, `zs`, `u1`

Ethereum ve EVM ağları (15. ve 16. bölümler)

Hepsinde adres `0x` ile başlar.

Ağ	Chain ID	Ücret ödenen coin	Blok yaklaşık
Ethereum	1	ETH	12 sn
Base	8453	ETH	0,2 sn
Arbitrum One	42161	ETH	0,3 sn
BNB Smart Chain	56	BNB	0,45 sn
Mantle	5000	MNT	2 sn
Monad	143	MON	0,3 sn

Diğerleri (17–20. bölümler)

Ağ	Coin	Adres şununla başlar	Kesinleşme
TRON	TRX	`T`	yaklaşık bir dakika
Solana	SOL	özel bir başlangıcı yok	saniyenin kesirleri
TON	Gram	`EQ` , `UQ`	yaklaşık bir saniye
Sui	SUI	`0x`	bir saniyeden kısa
Aptos	APT	`0x`	bir saniyeden kısa
XRP Ledger	XRP	`r`	3–5 saniye
Stellar	XLM	`G`	yaklaşık 5 saniye
Cardano	ADA	`addr1`	yavaş yavaş
Polkadot Hub	DOT	`1`	işaretle
Cosmos Hub	ATOM	`cosmos1`	hızlı

Ek C. İstek kopya kâğıdı

Bütün örnekler RpcNode'un herkese açık node'larında çalışır: kayıt ve anahtar yok, sıklık sınırı var. Kendi projeniz için adresi kişisel uç noktanızla değiştirin.

JSON-RPC: her duruma tek komut

```
curl https://public-ethereum-mainnet.rpcnode.dev \
  -H 'Content-Type: application/json' \
  -d '{"jsonrpc":"2.0","id":1,"method":"METOT","params":[PARAMETRELER]}'
```

Metot ve parametreleri tablolardan alıp yerine koyun. Başka bir ağ için adresteki `ethereum` sözcüğünü değiştirin.

Ethereum ve EVM ağları

Ne öğrenilecek	Metot	Parametreler
son blok numarası	`eth_blockNumber`	boş
ağ numarası	`eth_chainId`	boş
coin bakiyesi	`eth_getBalance`	`"0xADRES","latest"`
işlem makbuzu	`eth_getTransactionReceipt`	`"0xHASH"`
gas fiyatı	`eth_gasPrice`	boş
sözleşmeye sormak	`eth_call`	`{"to":"0x…","data":"0x…"},"latest"`

Bitcoin ve Solana

Adres	Metot	Ne öğrenilecek
`public-bitcoin-mainnet`	`getblockcount`	zincir yüksekliği
`public-bitcoin-mainnet`	`getblockchaininfo`	node'un durumu
`public-solana-mainnet`	`getSlot`	slot numarası
`public-solana-mainnet`	`getBalance`	lamport cinsinden bakiye, parametre: adres

TRON ve TON: soru adresin içinde

```
curl https://public-tron-mainnet.rpcnode.dev/wallet/getnowblock
```

```
curl https://public-ton-mainnet.rpcnode.dev/api/v2/getMasterchainInfo
```

Hata kodları

Kod	Ne oldu
`-32700`	istek çözümlenemiyor
`-32601`	metot bulunamadı veya kapalı
`-32602`	geçersiz parametreler
`-32005`, HTTP 429	çok sık istek
`-32006`, HTTP 402	kredi yetmiyor

Tam belgeler ve güncel planlar

https://rpcnode.dev

Telegram'dayız

@rpcnode_dev

Kitabın yazarı: rpcnode.dev

Kitabın sonu.

Sade bir dille

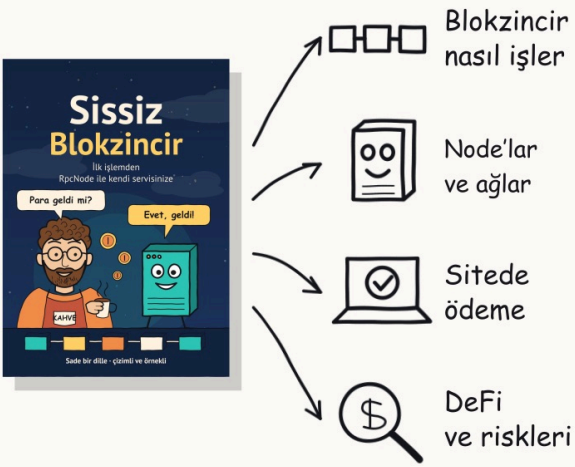
Sissiz Blokzincir

Bu kitap ne öğretir?

Müşteri soruyor: “Kripto ile ödeyebilir miyim?” Ne denir, paranın geldiği nasıl anlaşılır, yolda nasıl kaybedilmez?

İçinde: bloklar, anahtarlar, ücretler, tokenler, node’lar, Bitcoin, Ethereum, TRON, Solana, TON ve dahası. Sonra uygulama: RpcNode’da uç nokta, ödemeler, webhook ve mağaza koruması. Sonda da DeFi üzerine açık sohbet.

> “Sitem sepeti hesaplıyor, posta yolluyor. Blokzincirde olup biteni nasıl öğrenecek?”
>
> **— Artyom,**
> kitap kahramanı



Bu kitabın farkı ne?

Önceden bilgi gerekmez. Bölümlerde kahraman soruyor, sade yanıt, resim ve ödevler. İstekleri kendin çalıştırıp gerçek yanıtı görebilirsin.

rpcnode.dev
docs.rpcnode.dev
Telegram: @rpcnode_dev
Yazar: rpcnode.dev