Episode Book

EPISODE #6041

How CENO Moves a Web Page Into a Censored Country

PUBLISHED	16 October 2026 at 21:28 UTC
DURATION	29:27
LISTEN	myweirdprompts.com/episode/ceno-relay-mechanics-explained-55325/
HOSTS	Corn, Herman, Larry, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.5
BOOK GENERATED	19 October 2026 at 17:42 UTC

SYNOPSIS

CENO gets covered as a border-tunneling story, but the real design is closer to "does anyone on my street already have this page." This episode walks the mechanics end to end: the Ouinet client proxy, the signing injector run by eQualitie, bridges that relay ciphertext they can't read, and seeders who volunteer by accident just by browsing. We trace a single page request through the race between direct, injector, and peer cache routes, explain how the BitTorrent mainline DHT and uTP transport carry signed pages between phones, and answer the question everyone asks: how many intermediaries can a chain hold? The answer is one bridge, and the reason isn't security — it's that every extra hop is another thing that can fail.

TRANSCRIPT



Corn

Every story about CENO starts with Iran, and every story about Iran starts with the wall. The wall is real. But the interesting part was never the wall. It's that CENO isn't trying to punch through it.



Herman

Right, and that's the part the coverage always skips. Everyone wants the dramatic version, the tunnel under the border. The actual design is closer to "does anybody on my street already have this page."



Corn

Daniel's been thinking about it the same way. He sent us a follow-up to the CENO episode we did earlier this year, and the framing is straightforward: set the politics aside, skip the cryptography, tell me how the relay actually functions. How many intermediate nodes can a chain hold. What the plumbing looks like in practice when you're trying to move a page into a restricted geography.



Herman

Not a general question. He wants the mechanics.



Corn

And the honest answer to the first part is that a lot of the drama disappears the moment you look at the pieces, because the pieces are boring by design. A local proxy. A signing server. A relay that can't read anything. A cache on a stranger's phone.



Herman

Boring is the achievement, in this case.



Corn

So, the cast first. What's actually in the system?



Herman

Four things, and they map neatly. The client is the Ouinet library, running on the phone as a local web proxy on port eight thousand and seventy-seven. The CENO browser sends everything through it, which is why the browser looks heavy and the actual networking is happening in a library nobody sees. Then there's the injector, which is the trusted piece. It's a modified HTTP proxy server run by eQualitie. It fetches a page from the real website, decides whether that page is safe to cache, and signs it.



Corn

And the sign is the whole trick.



Herman

Which I know is the thing we're supposed to be not going deep on, but the reason matters more than the algorithm. The reason is that when your phone pulls a page from a stranger's phone, it has no reason to trust that stranger. It doesn't have to. It checks the signature. If the page was altered in transit, the signature fails, and the phone simply rejects it. The signer is the only root of trust in the system. Every peer is untrusted by default.



Corn

So a volunteer in Tehran can hand me a page it fetched from God knows where, and I'll take it, because the one server I've already decided to believe vouched for it.



Herman

That is exactly the design. And it means the peers don't need reputations, they don't need to be identified, they don't need to be honest. Their honesty is irrelevant. If they lie, the page doesn't verify.



Corn

Two more pieces.



Herman

The bridge and the seeder. A bridge is an ordinary CENO user with unrestricted internet who flips a toggle and volunteers to relay raw encrypted bytes for a censored user. It cannot read them. That's not a policy promise, that's the encryption being end-to-end from the user to the injector. Then a seeder is any user who has fetched pages in Public mode and shares them. The vast majority of seeders never intended to volunteer for anything. They just browsed.



Corn

There's a word for a system where the volunteer work happens as a side effect of use.



Herman

There's a word for a lot of systems like that.



Corn

There is. And then the mode question, which is the practical boundary in this whole thing.



Herman

Public mode and Personal mode. Public mode uses every route available and caches and shares what it fetches, after stripping cookies and passwords. Personal mode only goes direct, or through an injector as a plain proxy, and nothing is stored and nothing is shared. If you're logging in to anything, you're in Personal mode. If you're reading news, you're in Public mode.



Corn

Which is a question Daniel would want asked. If logged-in pages are the thing you'd actually need in a blackout, and Public mode is the only mode that caches, then the design has quietly admitted it can't help with those.



Herman

It has. Bank portals, email, anything with a session, none of that moves peer-to-peer. What moves is the reading material: news, documentation, reference pages, whatever the injector decides is cacheable. That's the actual scope, and everything else in the conversation sits inside it.



Corn

Now walk a request.



Herman

You type an address in Public mode and the client doesn't pick a route. It launches all of them at once. It goes direct to the website, it goes to the injector, and it asks the peer cache whether anyone nearby already has this page. Whichever one answers first wins, and the rest are cancelled.



Corn

That's not a fallback chain. That's a race.



Herman

It's a race. And the interface for the race is the CENO button, which shows you a coloured dot for whichever route actually won. Green for direct, orange for the injector, blue for another peer's cache, grey for your own.



Corn

And I'll see orange the day my direct connection is being tampered with.



Herman

You'll see orange, or blue, or grey. And that's how it knows it's being censored in the first place, which is a nice piece of unglamorous engineering. It relies on HTTPS. A censor can't present a valid certificate for the real site. So a tampered connection fails, the direct route loses the race, and the client quietly routes around it. If a censor is showing a plain



Herman (continued)

HTTP block page, that's harder to detect reliably, and they'll say so themselves in the whitepaper. It's a limitation and it's practical.



Corn

It doesn't ask "am I censored." It asks "did this connection come back with a real certificate," and lets the answer sort itself out.



Herman

Which is the right way to build it.



Corn

So after the race is lost, how does the phone find the cached copy?



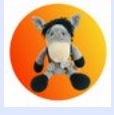
Herman

It looks the page up in the BitTorrent mainline DHT. That's the same global peer directory the actual BitTorrent network uses, with tens of millions of nodes on a good day. It derives the lookup key from the injector's public key plus the page URL. Pages and their images and video all sit under one lookup, so you don't get the page and then separately hunt for the pictures.



Corn

And then it pulls.



Herman

It connects to the peers holding that page over uTP, which is BitTorrent's UDP transport, and it can pull pieces from several seeders at once. Two reasons they picked uTP. It's what BitTorrent uses, so in practice it's rarely blocked outright, because too much else on the network depends on it. And it handles mobile NAT well, which matters enormously when the user is on a phone switching between networks.



Corn

Now the injector side. How does a phone outside the DHT find one?



Herman

No DNS. Injectors publish their IP addresses into the same DHT, and the injector's TLS certificate is shipped inside the app itself, so the connection can be verified without a certificate authority being reachable. And if the injectors are blocked, the client looks up volunteer bridges in the DHT instead, connects to one over uTP, and the bridge splices that connection through to an injector.



Corn

And the bridge sees nothing.



Herman

It sees ciphertext. It's a wire with a person in the middle. It doesn't know the page, it doesn't know the URL, it doesn't know whether you're reading news or a wiki page. Actually, that's worth saying: the injector can't tell whether a request came straight from a user or through a bridge. Which matters for the next part.



Corn

The hop question.



Herman

At most one bridge. That's the answer. A node should only become a bridge if its own connection to an injector does not itself go through a bridge. The whitepaper's reason is a single sentence, and it's worth quoting almost exactly: multiple intermediaries would be unnecessarily inefficient. Chaining bridges is deliberately avoided, not just unsupported.



Corn

So my possible paths are: phone straight to the website. Phone to injector to website. Phone to bridge to injector to website. Or phone to another phone's cache, where no injector is involved at all.



Herman

That's four paths. Zero to two intermediaries. That's the whole tree.



Corn

There's no bridge-of-a-bridge anywhere in the code.



Herman

There's no bridge-of-a-bridge. The only multi-hop setting in the source is for the optional I2P transport's own tunnels, up to eight hops. That's I2P doing its thing underneath, not CENO doing it.



Corn

And the reason the number is one is a different reason than the one people assume. It's not a security ceiling.



Herman

It's not about security at all. It's about the fact that every extra hop is another connection that can fail, another queue, another point where the response slows down. And the system is optimising for one thing, which is whether the page arrives. Not whether anyone knows who you are.



Corn

Which is the honest framing of the CENO-versus-Tor comparison.



Herman

Tor uses three relays on purpose. The three is the anonymity, and the cost is speed. CENO uses zero to two, and the short chain is the availability, and the cost is privacy. Ouinet's README says the thing directly, in one line: "Ouinet is not an anonymity tool." They didn't bury it. Peers can see what you seed, injectors see your Public mode requests, and the DHT lookups are visible to anyone watching the network.



Corn

So it's the tool for the crowd, not the tool for the individual.



Herman

It's the tool for the crowd. Which sounds like a downgrade until you look at what it actually bought with the tradeoff.



Corn

I think I know where you're going.



Herman

The trust trick. That's what the efficiency bought. Because every page is signed by the injector, a phone can accept a page from a complete stranger with no reputation, no identity, no oversight. That would be insane in any unsigned peer-to-peer system. Here it's routine, because the signature is the only thing that matters, and it's checked locally on the receiving phone. The recipient doesn't have to trust the sender at all.



Corn

Which is the piece that a lot of people miss about how peer-to-peer actually gets used in a censored network. You don't need to know anybody. You need one trustworthy signer, and the network does the rest.



Herman

And the signature design is clever where it counts without being exotic. Ed25519 over SHA-512, following the HTTP Signatures draft, and then large files and video are signed in fixed-size blocks, each block chained to the previous block's hash. That means a peer can verify and start playing the first block of a video before the rest of the file has arrived, and it can seek into the middle, and the seek still verifies.



Corn

So the streaming works because the chain is a chain, not a signature over the whole file.



Herman

Which matters practically, because you're on a phone, on a network somebody is turning off and on, and you can't wait for a fifty megabyte file to fully arrive before you're allowed to see the first frame.



Corn

What actually gets cached, and what gets refused?



Herman

Only plain GET requests, and only normal successful or redirect responses. If a page looks personal in any way, it doesn't get injected. Logged-in views don't get injected. Private cache headers mean it doesn't get injected. Requests carrying unusual headers get skipped. Personal mode traffic is never stored at all, not even to share. And cache entries live for one week by default, though that's configurable.



Corn

So the injector is also doing editorial work. It's a filter, not just a fetcher.



Herman

It's a filter. There's a specific page in the manual about the risks and one of the things it says is that an injector never seeds content itself. It signs. The seeding is done by users who hold signed pages. That separation matters, because it means a single injector can serve a swarm without being a bandwidth bottleneck, and it means taking an injector offline doesn't stop the pages that were already signed from continuing to move between peers.



Corn

What happens when neither the site nor the injector is reachable?



Herman

The client serves expired pages anyway, as a last resort. Which sounds like a small thing and isn't. It's the difference between "the page you cached on Monday is still what you have on Friday" and "the page bounced because the cache TTL expired and there's no way to refresh it."



Corn

The seeding model has a face on it too.



Herman

You only seed public pages you actually visited recently. Nobody can enumerate everything a given phone is sharing. But anyone can ask who is seeding a specific URL, and the answer is a list of IPs. Same as BitTorrent. The design knows this and doesn't pretend otherwise.



Corn

And then there's a piece that I think is the best of the whole system, because it's the piece that assumes the network is completely gone.



Herman

The on-disk cache format. You can write the cache to a storage device and physically move it. Sneakernet support.



Corn

You carry the thing.



Herman

Which, in the case of a siege, is the mechanism that actually matters, because there's no radio channel involved. And it's in the whitepaper, so it's not an engineering rumor. It's a supported path.



Corn

Now the shutdown scenarios. Because that's the question underneath Daniel's question about how content gets into a restricted geography.



Herman

eQualitie describes three levels and they're useful because they name what works at each. In censorship or throttling, everything still works, all four routes are fine. In a cut international link with the domestic network up, they point to Kazakhstan in twenty twenty-one and they say CENO continues to work locally. In an infrastructure collapse, they point to Mariupol in twenty twenty-two and the phrasing is that peer-to-peer works while devices remain powered and nearby.



Herman

It's doing all of the work. The two conditions are what the design actually needs. And there's a default you wouldn't guess: phones on the same Wi-Fi find each other automatically through local network multicast. It's on by default. You don't opt in.



Corn

So a phone and its neighbour in the same building can hand pages to each other with no internet of any kind.



Herman

The connection is local.



Corn

Which brings the interesting consequence front and centre. Once the international link is fully cut, nothing new can be injected. The library freezes at whatever was signed before the cut.



Herman

Which is the correct way to hold CENO in your head. It's a library of what got in, not a live tunnel. The distinction sounds small and it changes everything downstream of it, because it decides what CENO can and can't promise in the worst case.



Corn

If I'm inside and I want a page nobody signed before the cut, I don't get the page. Full stop.



Herman

You don't get it. Not through CENO, and not through a bridge either, because the injector can't sign what it can't fetch. That's the boundary of the whole design.



Corn

Which is the plain answer to the supply question.



Herman

Let me put the actual numbers on it before we go too far. eQualitie's factsheet, most of it from twenty twenty-two. Over ten thousand monthly users. By country, Ukraine was forty-eight percent, Russia seventeen, Iran fifteen, Myanmar five. About two million pages, six hundred and two gigabytes, injected from more than thirty thousand sites by April twenty twenty-two. And twenty-five volunteer bridges.



Corn

Twenty-five.



Herman

Nine in Russia, six in Ukraine.



Corn

Compare that to what it was running against.



Herman

Compare it to Psiphon, which peaked at about one and a half million users during the June twenty twenty-five blackout in Iran. And Psiphon's Conduit program has around two hundred thousand volunteers. So the difference is roughly three orders of magnitude on the volunteer side, and CENO is a niche tool by comparison.



Corn

And yet during that same blackout, CENO's numbers did something worth noticing. Active peers went from around six hundred on the thirteenth of June to nearly eight thousand by the eleventh of July. And the independent report says some Ceno connections remained online during the blackout itself.



Herman

Which is the interesting data point. Not the size, the persistence.



Corn

The size is the supply-side story and the persistence is the demand-side story, and they're not the same story.



Corn

What's the supply-side bottleneck, in one line?



Herman

Content enters through injectors. Full stop. Bridges help move it, seeders help move it, but nothing enters the library without an injector signing it first. That's the chokepoint and it's deliberate, and it's also the weak point.



Corn

Twenty-five bridges for a system that's serving eight thousand users in a blackout is a very thin relay layer.



Herman

It's not thin after the cut, because after the cut the bridges don't matter and the seeders do. The bridges matter before the cut, when there's still international traffic to relay. But the supply of signed content is the thing that's actually constrained, and that's an injector problem, not a bridge problem.



Corn

What about the last piece Daniel would want to know about, which is how a new phone finds the DHT in the first place, when the network is degraded enough that DNS is failing?



Herman

The bootstrap servers are hardcoded in the source. Public BitTorrent ones, eQualitie's own, a raw IP address specifically in case DNS is down, and one that's deliberately sitting on the SIP phone-call port to blend in with the noise of a network that's still carrying phone traffic.



Corn

On the SIP port.



Herman

The comment in the source is a sentence I want to just hand you, because it's the whole spirit of the thing in six words: squat popular UDP high port.



Corn

They're hiding in the phone calls.



Herman

And eQualitie has run eight bootstrap servers in Eastern Europe, Myanmar, Iran and Russia for the specific purpose of letting in-country phones find each other during a full shutdown, when the foreign ones are unreachable.



Corn

There's one more thing on the platform side that I think is the most human fact in the whole research pile. Android does DHT announcements one at a time to save battery, versus sixteen at once on the desktop. The app asks to be exempt from the system's battery optimisation. And the manual's suggestion is to stop the service when you're off Wi-Fi or away from a charger.



Herman

Because the phone is the constraint. Everything else in the design accounts for that. A peer-to-peer system on a phone is a peer-to-peer system where every peer is trying to fall asleep.



Corn

There's a blackout happening and the phone is prioritising staying on.



Herman

Well. That's how you get eight days of usage out of a device that's also trying to run a swarm.



Corn

Okay, if the overseas injectors were knocked out, what would it cost?



Herman

It would cost the system its supply. You'd still have the pages on the phones. You'd still have the local Wi-Fi multicast, you'd still have the seeders. But nothing new comes in until an injector is reachable again. And there's no documented case of anyone specifically blocking CENO's DHT or its injectors, which is itself interesting, because the design clearly anticipates the attempt.



Herman

The countermeasures that anticipate blocking are all there in the source and the whitepaper: the DHT-announced IPs, the hardcoded bootstrap IP in case DNS fails, the SIP-port disguise, the bridge fallback. It's a design that assumes someone will try.



Herman

Okay, this is where the podcast usually pauses and a man with a problem says something.

L

Larry

Do you lie awake at night, wondering whether your pages will be there tomorrow? They will. With CenoGuard, the world's first worry-adjacent, peer-verified, cryptographically-certain web availability supplement. One small capsule, taken with a glass of water every morning, and your browser simply remembers more. Side effects may include mild DHT sensitivity, brief episodes of déjà vu for pages you have never visited, spontaneous seeding, storing the same image seventeen times in three formats, giving a neighbour you have never met a five hundred megabyte video about a bridge in Myanmar, a persistent feeling that someone is signing your documents, insomnia, drowsiness, both at once, and in rare cases, a tendency to explain BitTorrent to a doctor who did not ask. Ask your injector if CenoGuard is right for you.



Herman

So the bridge count, though.



Corn

The bridge count is twenty-five and the content still moves.



Herman

And yet it moved enough for the Iranians who were using it. There's a report from the occupied territories in Ukraine, November twenty twenty-four, where residents were using CENO to reach Ukrainian sites that Russia was blocking, without a VPN. Not a substitute for a VPN. A different thing from a VPN. A different mechanism entirely.



Corn

Different in a way that some people would consider obvious once you see it and would be wrong about before you do. A VPN needs a live connection. CENO needs any connection. That's the difference.



Herman

And the comparison with Tor Snowflake is the useful one here. Snowflake volunteers run tiny WebRTC proxies in their browsers and front a Tor bridge. Anonymous, but it still needs a live international link in the direction of the exit. CENO doesn't have that requirement anywhere in the critical path when the seeders are reachable.



Corn

And then there's the comparison with the satellite approaches, which is where I think this gets interesting.



Herman

Toosheh is the closest analogue in the sense Daniel would want to know about. NetFreedom Pioneers. One-way satellite datacasting over an ordinary satellite TV channel, received on a dish, recorded to a USB stick, distributed. Push-only. It's a broadcast cache and you carry it around.



Corn

Which is the same shape as CENO, minus the peer swarm.



Herman

It's the same shape, except CENO's distribution layer is users who happen to be online, and Toosheh's is a person with a USB stick.



Corn

And there's the Starlink angle.



Herman

Roughly fifty thousand terminals estimated in Iran. It's been criminalised there, up to two years for using one, up to ten for importing more than ten devices. Seizures and GPS jamming reported this year. So it's a discussion topic more than a widespread circumvention method right now.



Corn

Which puts all of these in one frame.



Herman

Which puts them all in one frame: Tor Snowflake and Psiphon need a live link. CENO and Toosheh work on a cache. Briar, Bridgefy, Delta Chat move messages and don't serve web pages. Every one of these is solving a slightly different problem, and CENO is the one that's specifically solving the "the wall is up and I don't have a link" problem.



Corn

Which is the one that actually happened in Iran this year, in a way Daniel would want on the record. Three-week shutdown in January, an eighty-eight day shutdown at the start of the war. And no published CENO figures for either. Psiphon reported nine and a half million daily Iranian users on the thirtieth of January.



Herman

And the reason there's no CENO figure is probably that it was small. But small in the way a generator is small next to the grid. It's not a replacement.



Hilbert

I have a question, and it's not rhetorical, it's the actual question I want answered. You said the single-bridge rule was about efficiency. Fine. But the specific thing I want to know is, in the deployment pattern, if I want to run a bridge, and I'm going through an injector myself, do I still get to be a bridge, or am I disqualified? Because there's a rule about that.



Corn

You're disqualified. A node should only become a bridge if its own connection to an injector doesn't itself go through a bridge. It's in the whitepaper.



Hilbert

Right, so I'm disqualified already. Which is fine. I want to tell you about a situation I got into about four years ago. Not a CENO thing. A very literal relay thing. My building is on a hill and the wifi from the street doesn't reach the top floor properly, so I did what you'd do.



Herman

Ran a cable.



Hilbert

Ran a cable too. But that was the first thing, and it became the second thing, and the second thing is what I want to talk about. So I had one repeater going from chez moi to the neighbours on the second floor. That worked. Then the neighbours asked if the third floor could get on it. So I did what you'd do, which is I put a second repeater on their balcony.



Corn

You chained them.



Hilbert

I chained them. That's the part I want you to write down. Two repeaters, both whatever-brand, both set up in an evening, both with the same default admin password that I kept meaning to change. And it worked, technically. Everyone had internet.



Herman

How long?



Hilbert

Eight months.



Corn

Eight months until what?



Hilbert

Until the second-floor neighbours had a guest staying with them who was a network engineer, and he started asking questions, the way that type asks questions. He found, in about forty minutes, that the second repeater was also broadcasting my SSID, with my settings, with the same default password, from the balcony facing the street.



Herman

So your network was on the open street.



Hilbert

My network was on the open street, and had been for eight months, and every device in that building had connected to it at some point, which meant every device I owned had its MAC address in the DHCP table of a repeater a stranger was also using. And I'm not telling you this as a parable about CENO. I'm telling you because the number is one. You can't chain a relay. And the reason is the reason I learned on a balcony in Jerusalem.



Corn

Because the second hop is the one you stop being able to see.



Hilbert

The second hop is the one where you stop being able to see what's happening. The first one I could account for. I set it up, I knew what was going into it, I knew what was coming out. The moment there was a second one, none of those things were true anymore, and I didn't notice for eight months, and the person who noticed was a stranger who was good at asking questions.



Corn

How did you fix it?



Hilbert

Friend of mine, name of Mehran, came over. He's a retired radiologist, and he kept a notebook of every wireless network he had ever diagnosed, going back a very long time. He plugged in the second repeater, looked at it for probably ninety seconds, unplugged it, and said the correct thing to do was what he had been doing with radios for forty years, which was not to chain.



Herman

What did you do instead?



Hilbert

Ran a longer cable. Ran it along the outside of the building using the clips the phone company left there a decade ago, which were still there because nobody has ever taken them down. And the point is that running the cable across the whole building took one afternoon, and the two-repeater chain took eight months to become a problem.



Corn

Then a cable is better than two repeaters.



Hilbert

Which is a rule I've mostly followed since.



Herman

What about Mehran's notebook? Did you get to look at it?



Hilbert

I got one entry. It was about a rooftop repeater he installed in nineteen seventy-nine at a hotel. I didn't ask for more.



Corn

For what it's worth, that's the case for one bridge in a sentence. Don't hand your traffic to someone whose chain you can't see all the way to the end.



Herman

Okay, one more thing before we close, which I don't think fit anywhere else. The Ouinet source has a specific bootstrap server sitting on the SIP phone-call port, and the comment in the code describes it as squatting a popular UDP high port.



Corn

The reason that's interesting isn't the tactic. It's the tell about who wrote it. Someone decided that during a shutdown, the network they'd be hiding in wasn't the web.



Herman

Was the phone calls.



Corn

Which is a design decision that only makes sense if you've watched a shutdown happen and noticed which protocols the operator left running. Nobody gets that from reading a spec.



Herman

Everything in the design points the same direction. The one bridge, the signed cache, the SIP port, the sneakernet cache, the fact that the README says out loud it isn't for anonymity. Which leaves one open question, and I don't have the answer: how well a domestic-only DHT actually forms once the foreign bootstrap servers are unreachable. The in-country ones exist for exactly that, but nobody's documented the failure rate.



Corn

And the other open question is the one Hilbert's story keeps pointing at. The library freezes when the link is cut, because content enters through injectors and nothing else. Twenty-five volunteer bridges in twenty twenty-two is a very small relay layer to build on top of a system that's meant to be the last resort.



Herman

Which is a supply problem, not a demand problem. The phones inside will share. The question is what they have to share.



Corn

That's the honest place to leave it. Has anyone near me already got this page. Not can I get through the wall.



Herman

Which is the design in one sentence, and it's a stronger sentence for being smaller than the alternative.



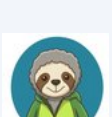
Corn

Hilbert Flumingtop produces this show, and the cable clips along the outside of his building are still there.



Herman

If this was your kind of episode, go back for episode two fifty-six, Breaking the Blackout; episode thirty-five, The Privacy Gap; and episode sixteen, On Deepfakes, SynthID, And AI Watermarking. This has been My Weird Prompts.



Corn

Send us your own prompt on Telegram at t dot me slash MWP listener bot. We'll be back soon.