Semantic Routing Networks: Una Tesis sobre Broadcast de Necesidades entre Dispositivos

Basado en las ideas de Yago Mendoza — Febrero 2026

---

1. La idea original

Dos tweets capturan la esencia:

> *"A robot needs a tool? It shouldn't look for an address. It should broadcast the need, and the right device answers. Semantic routing networks."*
> — @ymdatweets, 28 Feb 2025

> *"Forget IPs. Imagine devices discovering each other like a dating app, matching capabilities to needs, not locations. That's the future."*
> — @ymdatweets, 10 May 2025

La propuesta se puede condensar en una inversión fundamental: en vez de que un dispositivo sepa **dónde** está lo que necesita (dirección), **declara qué necesita** y la red se encarga de encontrar quién puede satisfacerlo.

---

2. El problema que esto resuelve

2.1 La limitación estructural de IP

TCP/IP fue diseñado en los años 70 para conectar un puñado de mainframes en ubicaciones fijas. Su modelo es host-centric: todo gira alrededor de *dónde* está cada máquina. Para que un dispositivo A pida algo a un dispositivo B:

1. A necesita conocer la **dirección IP** de B (o resolverla vía DNS)
2. A establece una **conexión punto-a-punto** con B
3. A formula su petición en un protocolo que B entienda

Esto funciona bien cuando los endpoints son estables y conocidos. Pero el mundo al que vamos — con IoT, edge computing, vehículos autónomos, robots — tiene características radicalmente distintas:

- Los dispositivos **aparecen y desaparecen** constantemente
- Las capacidades son **heterogéneas** y cambiantes
- Lo que importa no es *dónde* está un recurso, sino *qué* puede hacer
- La topología cambia en tiempo real

Pedirle a un robot que busque en un registro la IP de un sensor de temperatura específico en un warehouse específico es como obligar a alguien a memorizar números de teléfono en la era de los smartphones.

2.2 El gap semántico

El verdadero problema es que IP opera en una capa completamente desacoplada del significado. Una dirección IP no te dice nada sobre lo que ese host puede ofrecer. DNS añade una capa de nombres legibles, pero sigue siendo un mapeo estático nombre→dirección, no una descripción de capacidades.

Existe un **gap semántico** entre lo que las aplicaciones quieren (datos, servicios, capacidades) y lo que la red entiende (direcciones numéricas de máquinas).

---

3. El modelo propuesto: Need-Broadcast Semantic Routing (NBSR)

3.1 Primitiva fundamental

En vez del modelo clásico `CONNECT(address) → REQUEST(data)`, el modelo propuesto invierte el flujo:

```
BROADCAST(need_descriptor) → MATCH(capability) → ESTABLISH(channel)
```

Un **need descriptor** es un mensaje estructurado que describe qué se necesita sin especificar quién lo provee:

```
{
  "need": "temperature_reading",
  "constraints": {
    "location": {"radius_m": 50, "from": [41.40, 2.17]},
    "precision": "±0.5°C",
    "freshness": "< 30s"
  },
  "urgency": "normal",
  "requester_capabilities": ["can_provide:humidity",
```

```
"can_provide:pressure"]
}
```

Los dispositivos que escuchan este broadcast evalúan si pueden satisfacerlo y responden con un **capability advertisement**:

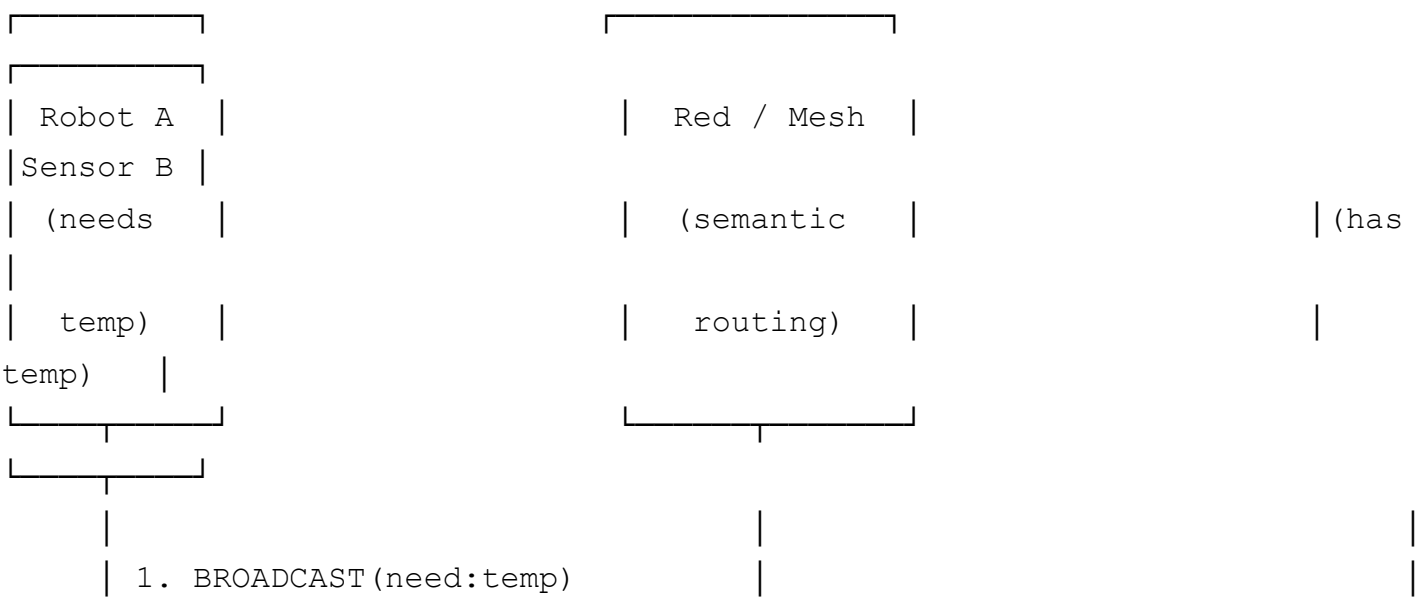
```
{
  "can_satisfy": "temperature_reading",
  "quality": {
    "precision": "±0.1°C",
    "latency_ms": 12,
    "reliability": 0.99
  },
  "cost": 0.003,  // energético, monetario, o abstracto
  "ephemeral_endpoint": "..."  // para establecer canal directo
}
```

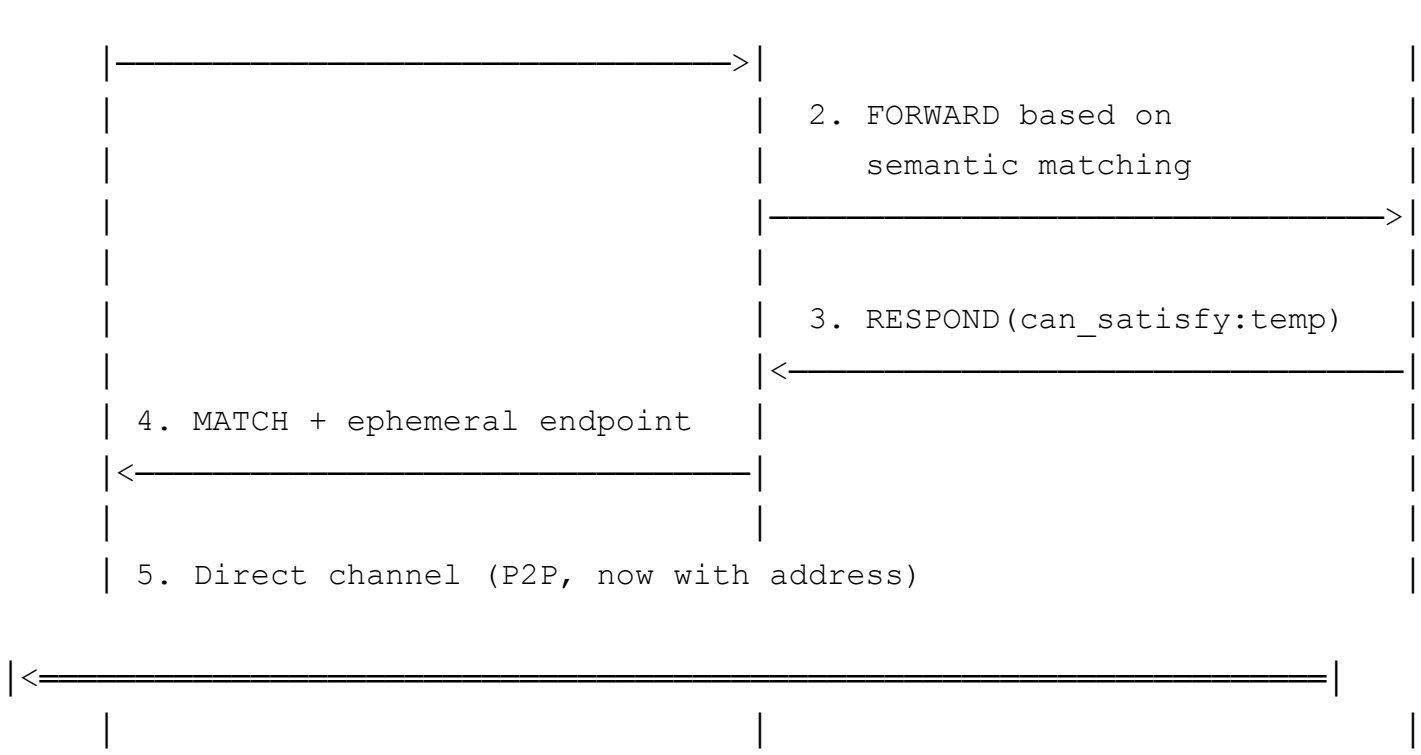
3.2 La analogía del "dating app"

La metáfora del tweet es precisa: en una app de citas, los usuarios no buscan a una persona específica por su dirección. Publican un perfil (capabilities) y expresan preferencias (needs). El sistema hace el matching. El protocolo NBSR funciona igual:

- **Perfil** = capability advertisement (qué puedo hacer)
- **Preferencia** = need descriptor (qué necesito)
- **Match** = el routing semántico que conecta ambos
- **Primera cita** = establecimiento del canal efímero punto-a-punto

3.3 Flujo completo





Paso crucial: las direcciones (IP o lo que sea) **solo aparecen en el paso 5**, como detalle de implementación del canal establecido, no como el mecanismo de descubrimiento.

---

4. ¿Puede funcionar sobre TCP/IP?

La respuesta corta: **sí, como overlay. No como reemplazo nativo.**

4.1 Lo que TCP/IP puede hacer

TCP/IP ya tiene mecanismos que se acercan parcialmente:

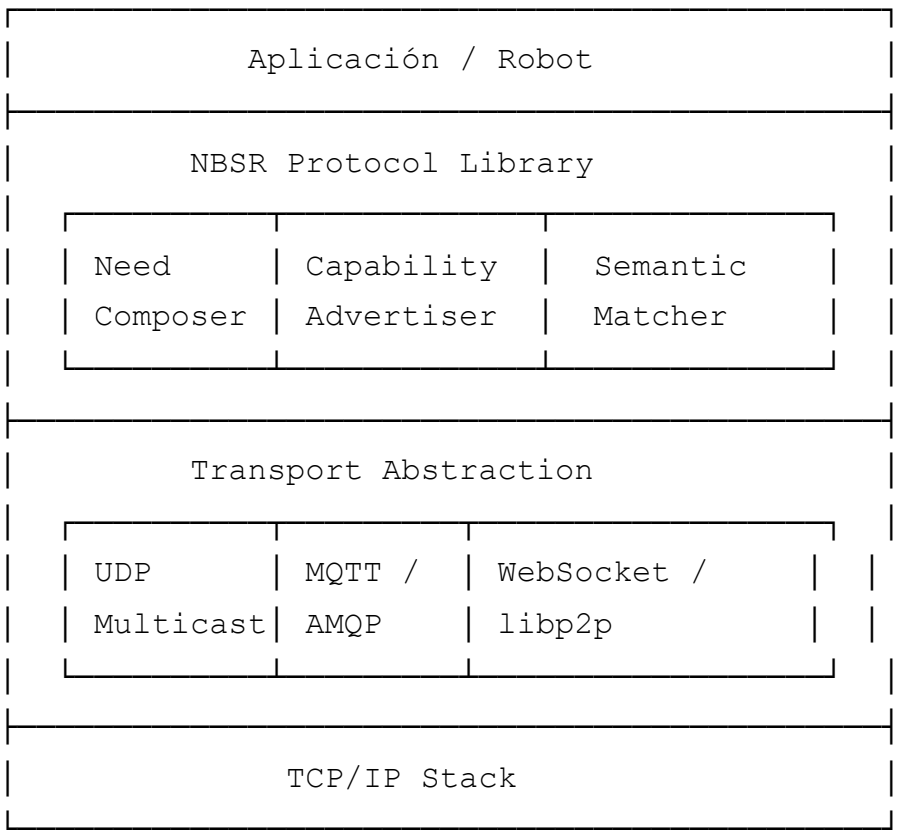
Mecanismo existente	Qué hace	Limitación
UDP Broadcast (255.255.255.255)	Envía a todos en la subred	Solo funciona en LAN local. No cruza routers.
IP Multicast (224.0.0.0/4)	Grupos de suscripción	Complejo, mal soportado en internet público, no semántico
mDNS/DNS-SD	Service discovery local	Solo LAN, vocabulario limitado, no matching por capacidades
SSDP (UPnP)	Descubrimiento de dispositivos	Solo LAN, sin semántica rica
DHT (Kademlia, etc.)	Lookup distribuido por clave	Semántica plana (key-value), no matching difuso

La conclusión: TCP/IP tiene las **primitivas** necesarias (broadcast/multicast, datagramas), pero carece de la **capa semántica** nativa. Todo lo que existe opera o bien solo en LAN, o bien con vocabularios

estáticos y matching exacto.

4.2 Arquitectura de overlay viable

Un protocolo NBSR podría implementarse **hoy** como overlay sobre TCP/IP:



En LAN: UDP broadcast/multicast directo. Es lo más eficiente. Cada dispositivo escucha en un puerto conocido, recibe los need descriptors, y responde si puede satisfacerlos.

Entre LANs / en internet: Necesitas nodos intermediarios — "semantic routers" o "need brokers" — que mantengan tablas de capacidades registradas y hagan forwarding inteligente. Estos brokers serían como los trackers de BitTorrent o los signaling servers de WebRTC, pero con matching semántico.

Peer-to-peer: Usando DHTs extendidas con búsqueda por atributos (no solo por clave exacta), o protocolos como libp2p con PubSub topics basados en ontologías de capacidades.

4.3 Lo que TCP/IP no puede dar nativamente

Hay cosas que un overlay nunca hará tan bien como un protocolo nativo:

1. **Eficiencia de routing**: cada need broadcast que cruza internet necesita pasar por brokers que hacen matching en software. Un router IP nativo con semántica embebida podría hacerlo en hardware, como hace TCAM matching para prefijos IP hoy.

2. **Latencia de descubrimiento**: el modelo broadcast→match→connect añade RTTs extra comparado con una conexión directa cuando ya conoces la dirección.
3. **Escalabilidad del broadcast**: difundir needs a toda la red no escala. Necesitas scoping (limitar el alcance semántico y geográfico del broadcast) y caching (recordar matches anteriores).
4. **Seguridad**: ¿quién puede broadcastear needs? ¿Quién puede responder? IP tiene mecanismos primitivos (firewalls, ACLs basadas en direcciones). Un modelo semántico necesita autenticación basada en capacidades, no en ubicación de red.

---

5. La conexión con Named Data Networking (NDN)

Aquí es donde la tesis se pone interesante, porque ya existe un proyecto académico serio que ataca exactamente el mismo problema fundamental: **Named Data Networking**.

5.1 Qué es NDN

NDN, liderado por UCLA y financiado por la NSF, propone reemplazar el "thin waist" de internet (IP) por uno basado en **nombres de datos**, no en direcciones de hosts. Es una arquitectura clean-slate — es decir, no un parche sobre IP, sino un rediseño desde cero.

En NDN:

- Los consumidores envían **Interest packets** ("quiero /ucla/videos/demo.mpg")
- La red enruta estos Interest packets **por nombre**, no por dirección
- Los productores (o cualquier cache intermedio) responden con **Data packets**
- No hay direcciones IP. No hay source ni destination address en los paquetes.

5.2 NBSR vs NDN: convergencias y divergencias

Aspecto	NDN	NBSR (la propuesta de Yago)
Qué se nombra	Datos específicos ("/sensor/temp/room3/latest")	Necesidades abstractas ("need:temperature, radius:50m")
Quién inicia	Consumer pide dato por nombre	Requester declara necesidad
Matching	Exacto por prefijo jerárquico	Difuso por capacidades y constraints
Routing	FIB con prefijos de nombres	Semantic matching distribuido
Direcciones	Eliminadas completamente	Eliminadas para descubrimiento, usadas para canal P2P

Aspecto	NDN	NBSR (la propuesta de Yago)
Sobre IP	Puede correr como overlay (NDN sobre UDP/TCP)	Puede correr como overlay

La **convergencia profunda** es filosófica: ambos reconocen que el modelo "todo es una dirección de host" está fundamentalmente roto para cómo usamos la red hoy. Ambos quieren que la red entienda **qué** se pide, no solo **a quién** se le pide.

La **divergencia clave** está en el nivel de abstracción:

- NDN aún requiere que el consumer **sepa el nombre** del dato que quiere. "/sensor/temp/room3/latest" sigue siendo un identificador específico.
- NBSR no requiere saber qué existe. Solo declaras qué necesitas, y la red resuelve el matching.

NBSR sería, en cierto sentido, una **capa por encima de NDN**: podrías imaginar un sistema donde los need descriptors se resuelven a nombres NDN, que luego se rutean por la red NDN.

5.3 Lo que NDN ya resolvió (y NBSR puede aprovechar)

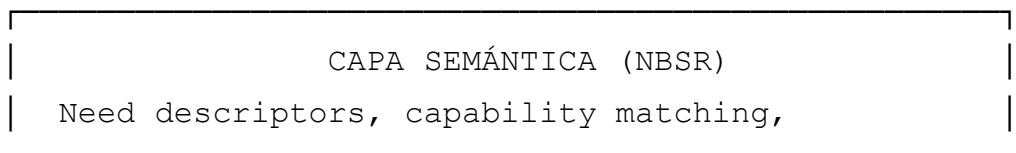
NDN lleva más de 15 años de investigación y ya tiene soluciones maduras para problemas que NBSR también enfrenta:

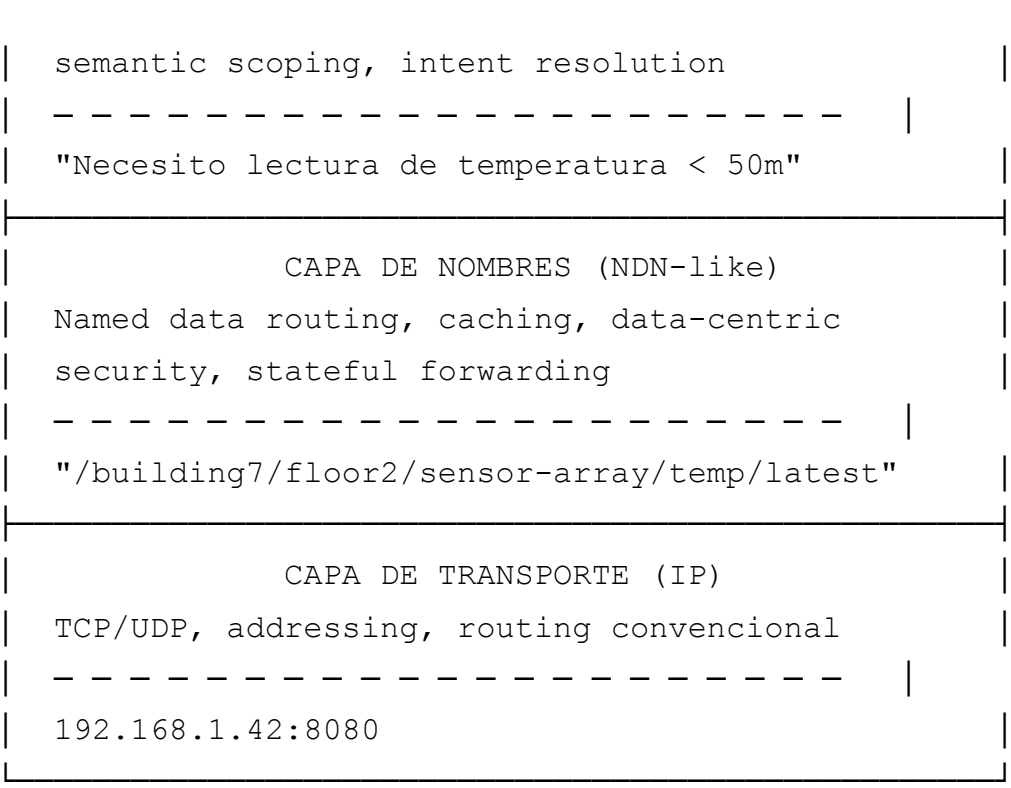
- **Forwarding sin direcciones**: el stateful forwarding plane de NDN (PIT, FIB, Content Store) demuestra que se puede rutear sin IP.
- **Caching distribuido**: cualquier nodo intermedio puede cachear y re-servir datos. Esto aplica directamente a capability advertisements.
- **Seguridad data-centric**: en vez de asegurar canales (TLS), se firman los datos mismos. Un capability advertisement firmado es verificable sin importar de dónde venga.
- **Multipath nativo**: NDN puede usar múltiples interfaces simultáneamente. Un teléfono puede broadcastear needs por WiFi y 5G a la vez.
- **Scope local y global**: los nombres NDN pueden ser locales ("the light switch in this room") o globales. Need descriptors tendrían la misma propiedad.

---

6. Arquitectura híbrida: lo realista

Dado que ni NDN ni NBSR van a reemplazar IP mañana, la propuesta práctica es una arquitectura de tres capas:





6.1 Cómo interactúan las capas

1. **Aplicación declara need** → NBSR genera need descriptor
2. **NBSR resuelve need a nombre(s)** → consulta a semantic routers locales o distribuidos
3. **NDN-like layer enruta por nombre** → usando FIB, PIT, o DHT semántica
4. **IP transporta los paquetes** → los paquetes NDN/NBSR viajan encapsulados en UDP/TCP entre nodos
5. **Match encontrado** → se establece canal directo (puede ser IP puro para la transferencia de datos)

El punto crucial: **IP no desaparece, se convierte en la capa de plomería** — como la línea telefónica sobre la que corrió IP inicialmente. NDN se montó sobre IP de la misma forma. NBSR se montaría sobre NDN (o directamente sobre IP en implementaciones más simples).

6.2 Para qué dominios funciona hoy

Dominio	Viabilidad	Por qué
IoT / Smart buildings	Alta	Red local, dispositivos heterogéneos, needs predecibles
Robótica	Alta	Exactamente el caso de uso del tweet: robot necesita herramienta
Edge computing	Media-Alta	Nodos distribuidos, service discovery dinámico
Vehículos autónomos	Media	Necesita baja latencia, pero el concepto de "broadcast need" encaja

---

Dominio	Viabilidad	Por qué
Internet general	Baja (hoy)	Demasiada inercia en la infraestructura IP, se necesitaría adopción masiva

---

7. Problemas abiertos y riesgos

7.1 Broadcast storms semánticos

Si cada dispositivo puede broadcastear needs, y cada need se propaga por la red buscando matches, el tráfico de control puede explotar. Soluciones posibles:

- **TTL semántico**: los needs expiran no por hops, sino por relevancia (distancia semántica al contexto del receptor).
- **Scoping por ontología**: los needs solo se propagan a dominios semánticos relacionados.
- **Caching de matches**: si ya encontraste un match para "need:temperature, location:building7", no repites el descubrimiento por N segundos.
- **Expanding ring search**: empezar buscando local, ampliar el radio solo si no hay match (exactamente como hace LOADng en redes mesh).

7.2 El problema de la ontología

Para que el matching semántico funcione, los dispositivos necesitan un vocabulario compartido. "temperature_reading" para uno debe significar lo mismo que para otro. Esto es, en el fondo, un problema de estandarización más duro que el técnico.

Opciones: ontologías formales (tipo Schema.org para IoT), o embeddings semánticos donde el matching se hace por similaridad vectorial en vez de por strings exactos. Esta segunda vía es más robusta y tolera vocabularios diversos, pero requiere más compute en cada nodo.

7.3 Seguridad e identidad

En IP, puedes filtrar por dirección. En NBSR, ¿cómo evitas que un dispositivo malicioso responda a needs que no debería? La firma criptográfica de capabilities (à la NDN) ayuda, pero necesitas un modelo de trust distribuido — probablemente basado en cadenas de certificados o reputación en la mesh.

7.4 Incentivos

¿Por qué un dispositivo respondería al need de otro? En un ecosistema controlado (empresa, fábrica) esto es trivial. En internet abierto, necesitas un modelo de incentivos — quizás microtransacciones,

reciprocidad de capabilities ("yo te doy temperatura si tú me das humedad"), o reputación.

---

8. Conexión con la visión más amplia

El tweet sobre el "circulatory system for the physical world" completa la visión:

> *"Imagine a circulatory system for the physical world, dynamically rerouting around blockages, optimizing resource allocation [...] Every movement purposeful, every action efficient."*

NBSR no es solo un protocolo de red — es un modelo de **inteligencia distribuida** donde la red misma tiene agencia para resolver problemas de asignación de recursos. Cuando un robot broadcastea "necesito una herramienta de corte", la red no solo encuentra la herramienta más cercana: evalúa disponibilidad, costo energético de transporte, cola de espera, y propone la mejor opción. La red se convierte en un optimizador distribuido de recursos.

Esto enlaza con otra observación clave: *"agency in AI is [...] actually easier to learn than we would have thought"*. Si los nodos de la red tienen capacidad de inferencia local (edge AI), el matching semántico puede ser sofisticado sin necesitar un servidor central. Cada nodo evalúa localmente si puede satisfacer un need, usando su propio modelo de comprensión.

---

9. Conclusión: ¿Es viable?

Como overlay sobre IP — sí, y es implementable hoy. Con MQTT/AMQP para el broadcast, JSON-LD o embeddings para la semántica, y WebRTC/libp2p para los canales P2P post-match. No reemplazas IP; lo usas como plomería.

Como protocolo nativo que reemplace IP — no a corto plazo, pero NDN demuestra que es arquitectónicamente posible. La investigación de NDN (UCLA, NSF, con Cisco y Huawei involucrados) lleva 15+ años y tiene testbeds funcionales. NBSR sería una extensión natural: añadir matching semántico difuso al routing por nombres que NDN ya resuelve.

La pregunta real no es si puede funcionar técnicamente. Es si hay suficiente presión económica para que la industria invierta en esta transición. Como señala el propio Yago con la analogía solar: la primera celda fotovoltaica se creó en la década de 1880, pero solo ahora la solar es más barata que los fósiles. La misma lógica aplica aquí: cuando el coste de mantener el modelo IP-centric para IoT/edge/robótica supere el coste de migrar a algo semántico, la transición será inevitable.

La tecnología está lista. La economía aún no.

Documento generado como análisis técnico de las ideas publicadas por @ymdatweets. No es un paper académico formal; es una exploración estructurada de viabilidad.