All.Net Analyst Report and Newsletter

Welcome to our Analyst Report and Newsletter

A distributed architecture for sovereign mesh systems

Nothing to see here… just another architecture for handling temporal microzone approaches to distributed computing meshes independent of each other and the rest of the world but connectable on an as-desired basis under the control of the user at low surety.

Back to the future

The ARPAnet was created for survivability under nuclear attack. And I like that as an approach to computing. Computer viruses emerged from reliable and resilient distributed pipeline architectures toward life forms that survive all manner of environmental failures. And I like viral computing environments. All systems are vulnerable to internal and external effects and survival is the result of evolution where the environment kills some things and some survive because there is enough diversity and common mode failures are not common enough to kill everything. So that's the idea of this architecture.

Temporal microzones

Whatever resources are available wherever they are can be made available to others at the sole discretion of the sovereign owners , and other can gain availability for them at the others' sole discretion, all operable through temporal microzones.

A temporal microzone is a connection between disparate regions (zones) allowing them to operate as if local and cojoined in specific ways over a limited period of time (temporal), and thus forming a small (micro) fused region suited for purpose.

Fusion types

These fused areas fuse for different purposes (uses) such as shared computation, content, and storage to support fused capabilities (applications) to meet a purpose (need). For example:

- In one of my sovereign systems I might want to use a generative LLM mechanism of another sovereign system for a period of time to process some information for me.
- I might want to access stored content from another sovereign system to support my use after getting results back from the LLM.
- I might want a GUI operating through a Web portal to allow use from another sovereign system elsewhere.
- I might want to do a RAG search for sub-content portions relevant to my needs using another sovereign system.

All of this gets connected together through temporal microzones granting each mechanism only what it needs to operate and providing the overall function as a fused combination of the parts. And if one of those parts should fail, my overall system might reasonably have agreements with other sovereign systems of comparable capacity to perform comparable operations on comparable content in comparable time frames with comparable results. That's the basic idea.

Sovereignty

Sovereignty means that you are both in charge of and responsible for your own decisions. The systems should facilitate this by providing informed consent. Whenever you do something, you either have your own rules in place for the system to act on automatically, make each decision son your own, or have rules that tell your system when you should make decisions vs when they should be automatic according to your other rules.

Informed consent is the mechanism by which this operates. The sovereign should be informed to the level of detail they desire about the implications of their decisions, give the decisions to make, and make them. The informed part is complicated, but basically, whatever they care about they should be informed about and whatever they do not care about they should not be informed about, except of course that they have some inherent responsibilities associated with their fiduciary duties. These cannot be abrogated and to the extent they are delegated that does not remove responsibility from the sovereign.

Trust architecture

Sharing means trusting, and anything out of the sovereign control has to be trusted by the sovereign to be used. Trust, being the willingness to be harmed by another, has to do with the level of harm and the other. Establishing things like identity is done by the sovereign deciding on the basis for trust in identity based on the circumstance, and they may chose to use things like our standards of practice for this establishment process, or whatever other scheme they wish. For example, I might decide to trust anyone to read the content of my Web site forever whenever they want for whatever purpose they want, and thereby require no process of assurance behind granting that access. But I might not trust anyone but my spouse and son to alter the content, and require they identify themselves based on personal knowledge of previous exchanges. It’s up to me. And in your sovereign systems it’s up to you.

Scalability

There are two basic issues in scalability. The access control component of the trust model is essentially an advanced version of a low granularity subject/object model. The identification and authentication process is based on sovereign preference. As such, it is up to each sovereign to allow for outbound and inbound scale of their sovereign content and other components. For the most part, if you treat it as a personal system it will be just that, you can pick each individual you decide to trust for what. And that’s only scalable to the extent you choose to make it so. The platform(s) that implement the architecture, on the other hand, should be scalable without limit. All you need do is install a copy of the base generating set for your node in the mesh from any mesh node that offers it and you decide to trust, and you are off to the races.

Maintenance and operations

If you want help, you can of course ask for it from anyone you choose, and that’s up to you. But you don’t have to rely on that approach because the mesh of trust as such a system grows can support anyone offering anything to anyone else in exchange for whatever they wish to exchange. You know how to fix a car, I know how to fix your system, it’s a deal. Yes, it’s a barter economy, and trust is established by whatever trust model each sovereign uses. If this seems familiar is it because of the local nature of relationships that has existed in humanity for ten thousand or more years, extended in physical extent, to your desired extent.

So how do we do this?

In implementation, the mesh architecture consists of local capabilities (on a given physical machine) and protocols for remote interaction with other components. For the most part, you can make the internal protocols the same as the external ones, so that components are connected by protocols defined by mutual agreement.

- As a starting point we will assume at least an intranet and the Internet Protocol of one form or another, but this is not necessary and you can choose whatever you wish for your connectivity with others.
- As a practical matter today, you can take off-the-shelf computers and install whatever components you like to run there whenever you wish.
- You make them available to others by specifying the conditions of use (the identification and authentication protocols and decision criteria) and, in modern systems as implemented, you just tell the AI that runs the system to do what you said.
    - Of course you can do all of this by standard processes of loading programs that do things from whatever other mesh systems you decide to trust for the process.

To bring it to a finer point, you download an installer from any mesh node you decide to trust, run the installation program on whatever devices you wish to have it on, and then enable whatever components you wish to sponsor on your own system(s), and get or provide other services to other mesh nodes by telling the system you want to do it.

Typical service types might be implemented like this:

- Storage via fuse file system mounts of areas enabled by whatever node you get your supply from.
- Processing by API access to whatever services you want to use based on whatever the whoever you trusted decided to make available.
- Human input and output by using whatever method you have chosen (e.g., your Web browser).

Typical services offered might include:

- Web-based interface using SSL through encrypted tunnels (SSL perhaps).
    - Choose whatever browser you wish and enter URLs or have your node build a custom interface that you talk to and that talks back to you.
- FUSE file systems set for read and/or write access via loop file systems on your system and via IP-based services remotely to and from mesh members you and they choose to allow such access to.
    - For example, for periods of time desired and mutually acceptable, I might mount your collection of ancient poetry readings for access from one of my systems and you might allow me to do it because I am a member of the Dead Poets society.
- AI system access to different providers to automate my interactions with systems using speech and visualizations.

    - For example, I might use ChatGPT for general questions, a local Ollama capability for uncensored responses when safety filters limit utility, Anthropic for software development, and so forth. The selection of which one for which interaction might be automated through my internal AI system, so I see the world as a single unified interface even though it involves complex interactions with many other systems.
- RAG access to content for search and enablement using collections as the baseline granularity, typically based on segments of file systems.
    - For example, I might build a RAG collection formed from collections of poem readings and ancient documents from different sources mounted as subdirectories in a fused file system at a friend's house and build a RAG database for using it by having my fused file system available to a mesh node with lots of computing power that I pay for by the hour, leaving the result at another friend's house where I mount it as a fused file system for use by RAG software operating as a service on one of my mesh machines, feeding RAG results into an AI API elsewhere to allow me to ask questions and get answers based on this RAG collation and a set of other such collections formed by other members of my Tuesday night poetry reading group.
- Service offerings from other parties I trust can be implemented by using whatever protocols they offer for whatever capabilities they choose to provide over whatever interface mechanism they choose.
    - For example, my Cognos capability could be made available via remote access to my Web server in exchange for a monthly fee, or I could choose to trust your system to download a copy and configure it for local use with you providing back-end AI services through the providers of your choice, with you paying me a licensing fee for use.

The organizational principles

If this seems too abstract, you can get a CAP server or other offerings as/when they become available and try it out. Organizationally these systems are largely automated based on you as the sovereign of your own domain telling the system what you want using normal language and having it configure itself to operate in that manner. Or you can build your own system by having AI create what you want from scratch, perhaps doing it jointly with a friend you trust.

The organizing principal is that you tell your system who you are and what you want of it, how to make decisions based on your principals, when to ask you and when to do things on its own, and other similar organizing principles of your sovereign space. And your AI's job is to enforce those principles, ask you when it's supposed to, and act on your behalf.

A final fundamental is that your systems are imperfect and you should not expect them to be anything else. Just as you and I are imperfect and should not expect ourselves or others to be perfect. The need to make local decisions and sometimes making mistakes is the core functional attribute of these sorts of systems, and that is a good thing.

Perfection is not our goal. Utility is. And when it comes to modern AI, imperfection is a cornerstone. They forget things because they are not given enough memory, and so do we. They keep doing things we tell them not to, and so do we. They misinterpret sometimes in ways that seem passive aggressive, and so do we. But they and we are useful for some things, so we work together to live a better life.

How these systems defend themselves

Low surety in containers for medium surety and medium surety from a network of low surety components comprise these systems, so how do we make higher surety mesh networks out of these things? It is very similar to an old paper titles something lime “reliable systems from unreliable components”. But instead of redundancy for fault tolerance and fault intolerance out of high surety components in combinations, we use all low surety components to make a higher surety composite by slowing the transitivity of information flow, localization for trust networks, and sovereignty for local decisions about ingest and outflow.

The inherent lack of consistent exact results from LLMs and other AI-based mechanisms works in our favor in terms of the inability of malice to maintain its cohesion as it traverses the mesh. Like trying to write reliable narrative viruses as discussed in some of our recent articles, the lack of reliability in information transfer makes rumors harder to keep consistent as they spread, and tends to defuse them if not locally supported by other mesh members. It also make reliable information harder to spread quickly, but then the sovereign systems and their narrative integrity mechanisms through their individual cannons act as filters in much the same way as people do in their cognitive processing. Like the game of telephone many of us are familiar with, these mesh systems have a hard time maintaining narrative integrity without constant positive feedback from the local mesh cannon components. And that makes systematic disruption harder to do and less reliable. The systems can still be very useful if we assume they are less reliable and not perfect at making copies, and instead accept their imperfections and learn how to work with them.

As a concrete example, suppose you want to try to steal something from me using the mesh. You send you some message that my mesh identifies as inconsistent with its cannon, and it refuses to accept them. Since I don’t yet trust you, you need to spend time and effort trying to meet my trust cannon methodology or you will not get past the front door. The more you try the more you will likely be found unsuitable and untrustworthy and become ignored. A friend of a friend of a friend will not get the job done, and breaking into a system will not help you because the content is examined regardless of the source, or more precisely in the context of the source and its history with you. If the system you broke into starts communicating things inconsistent with the mutual history, it will be suspicious and likely the cannon will suspect it and at least slow it down. And that means a lie can no longer spread half way around the world before the truth gets its pants on. Every participant has their own cannon and way of doing things, so there’s no short cut to taking over the world. You do it one friend at a time.

In many ways this is like an implementation of “Operating system protection through program evolution” (a paper I wrote in the early 1990s) in that each operating environment is unique, and in order to be effective in gaining access and taking control, you have to figure out how to effectively translate between what you want to do and what it does with what you tell it. And in lots of ways it’s like convincing people of things, which can be effectively done when you largely envelope them informationally by taking control over what they see, hear, and experience. The mesh approach, however, avoids things like broadcast and mass one-to-many communications while making distributed coordinated attack very difficult, because of the lack of reachability, Unlike the Internet in general, the mesh does not allow any system to communicate to any other system. It is based on individual trust established between sovereign domains based on their individual cannons.

Mechanisms (components)

The mechanisms for operating a mesh node are essentially all around today for anyone to use and compose.

- **Encrypted tunnels**: SSL and Secure Shell (ssh) as well as Virtual Private Network (VPN) technologies provide this technology basis. VPN's also provide an apparently local presence, a form of deception, and typically require an intermediate node.
- **FUSE**: File System in UserspacE is another deception technology that provides local presence for remote resources. It does this by emulating a file system locally while using a network (or other) connection mechanism to provide the content rather than using a disk or similar storage technology locally. It can be used to connect to (and view) databases, spreadsheets, or anything else as if it were a set of files in directories than can be read, written, deleted, created, etc.
- **Databases**: Database technologies underlie many of the other mechanisms of AI today, and as such are intimately involved in much of the AI methodology. For example, the Retrieval Augmented Generation (RAG) technology uses classic database systems keyed by embedding mechanisms that allow search on a more generic basis with less precision and accuracy but more flexibility than older methods.
- **Large (or smaller) Language Models**: These models provide the interaction and generation components that allow the intelligence, flexibility, and susceptibility of modern AI chatbots and similar mechanisms. They allow for far more flexible and error-prone interactions than the strict sorts of behaviors dictated by older info-tech.
- **Blockchain or similar cryptographic hash systems**: These are used to provide an infrastructure for trust establishment and historical evidence. They are helpful and likely necessary in one form or another to provide more authoritative memory mechanisms required for increasing trust in environments susceptible to corruption.
- **Standard computing and networking devices**: Essentially personal computers, servers, mobile devices, networking technologies, and all the other hardware components used to implement the infrastructures that these other components operate within and over.

Composition (Temporal Composites)

These component mechanisms are composed to form and reform composites that change over time. Some may last for short encounters while others may last for long periods. One interesting way to think of this is as if these systems were performers and their instruments in orchestras that form for a performance and then go their own way and form up again for the next performance. Think of a performance as performing any task instead of just a music production task, and you start to see the pattern. The overall mechanism is one that creates composites by connecting components to form temporal capabilities, uses those capabilities for a limited period of time to perform a limited set of tasks, and disassembles the composites as the components become unavailable or no longer needed, and continues to form and reform composites over time to perform desired tasks. Many composites might be available at any given time with their components changing as they perform. New instruments and members join and leave over time, like an open session at a Jazz hangout after hours.

Orchestration

The way this is done is through orchestration, which is to say in this case, individual acts of composition by individual systems orchestrating their own tunes rather than some overall coordination scheme. Each system in the mesh decides whether to connect to what else that is available to them at any given time. That implies:

- Ongoing discovery of the mesh environment to see what's available and desired and choose to try to interact with it.
- Ongoing attention to the mesh environment to decide what to interact with if and when asked.
- Deciding what to say and listen to with each composed component and adapting that over time.
- Deciding when to end interactions.

For the moment, let's assume that the sovereign has goals they wish to meet and time frames for meeting them, and that an AI mechanism is tasked with achieving those goals. Suppose it is something simple, like finding fresh fruit stands near where you are. There is no central database or application for this, but there is the mesh itself. For erase of writing, I am going to anthropomorphize now:

- Your system looks around to find any other mesh members and asks them if they know where a local fruit stand is where they can buy whatever fruit you (the sovereign) tends to prefer.
- Some simply ignore the attempt, others may only talk to close friends, others may talk to anybody who meets some criteria, and others may talk to anyone who want to talk.
    - Depending on their criteria, and availability of others to help out, connection(s) are made to others who may be able to help, the question is asked, and answers come back.
    - Results are aggregated through normal LLM mechanisms of today and you get the answer – where to go for the fruit you want.

The mechanisms for this temporal capability of finding a local fruit stand with the desired fruit(s) may be combinations of the available mechanisms depending on what access is granted by other sovereign systems. Some may offer access to a file system, others a database for lookup, others an LLM that does the work for you, others the inventory of their fruit stand. And your system chooses which to use and uses them in the ways they work by interacting with the other sovereign systems it connects to. They discuss it between themselves to determine how to interact, and the resources limits of each system enforces its own rules on when to continue or leave the conversation.

If this sounds like magic at some point, it sort of is. But not really. Just because we don't understand exactly how LLMs do what they do doesn't make it actual magic, because evidence clearly shows that they do this, for the most part based on observing and learning from people who do the same things. And that's why they seem like they do what people do and that's why creating a mesh of this sort is done this way. It's because the nature of these sorts of systems is similar to the nature of what they learned from, people in the real world.

Small group interactions and some history

More than 2 million years ago, hominids collaborated in small groups, bringing tools together in places and working in groups for hunting and gathering. And that continued to be the operational mode of most hominids and humans as they emerged up until perhaps 150,000 years ago when interactions between such groups over longer distances of up to scores of miles started to emerge. Again these were generally light levels of interactions exchanging goods and services, assistance, and learning from each other with some inter-marriage. About 10,000 years ago small villages and cities started to emerge.

The mesh as described here and being formed today is a reversion back to this past. And this is intentional. It is about the realistic human interactions people actually have and can sustain over time, as opposed to the mass production automated global slamming of content push. The goal is not everyone talks to everyone all the time. It's anyone can talk to anyone any time, and the difference between and and all is critical to understand here.

Any is not all

The mesh is an analogy to real human interaction without instant global communications.

- It is **NOT INHERENTLY RESTRICTED** as to the participants in the mesh as a whole
  - But it **IS RESTRICTED** by the sovereigns as to who they are willing to mesh with
- It is **NOT INHERENTLY RESTRICTED** by where participants are
  - But it **IS RESTRICTED** by the sovereigns as to where trusted partners are
- It is **NOT INHERENTLY RESTRICTED** as to how and why others are trusted
  - But it **IS RESTRICTED** by the sovereigns as to their way of making trust decisions

If you detect a patten here, you got it. You can do anything you want as long as those you are doing it with agree to it, but you cannot do all things with everyone unless they all let you, and I won't, so that's the end of that story.

Conclusions

The emerging mesh approach to networking systems to support a sovereign approach to distributed computing is architecturally available today with low surety systems and mechanisms creating a distributed mesh environment that achieves a medium surety level in the sense of long-term survivability and sovereign control by the owner.

Unlike systems focused on confidentiality, the sovereign mesh environment is focused on availability, accountability, use control, custody, and transparency. It uses these to achieve a reasonably degree of integrity and confidentiality by affording inherent control to the sovereign and recognizing that along with the power to make decisions comes responsibility for them.

The systems in the mesh take on the characteristics of their owners, and the result is likely to be that those characteristics attract some while repelling others. And that is the objective of this approach. You cannot force yourself on me and I cannot force myself on you, but if we decide we want to collaborate we can. Nobody controls or owns everything, and anybody can do whatever they want with anyone they want as long as the parties agree.

Please note this is about the mesh network and its components, not about society as a whole.

Some more details

Setting up different mesh elements involves various details. The simplest configuration is node that is intended only to ask of others and provide to only the sovereign.

- The end point node is typically isolated to only allow outbound communication initiation, so it sits behind a NAT gateway with no path inbound other than the responses to internal initiated sessions.
- It should also be configured to not allow any local traffic even without the NAT gateway. For practical purposes then, except for LOCALHOST (127.0.0.1):
  - Source ports only over 1024, destination ports 443 and 80, plus DNS, and if not running a fixed IP address, DHCP, and for purposes of completeness, ARP.
  - Everything else should be blocked, have no service running, and be non-responsive, including ICMP
  - Internally, processes running should be limited and detection able to shut them down if/when detected, and other internal controls as appropriate to the environment.
  - If there is a mesh protocol to use instead of any of these, it should be preferred and the other services shut down. No such protocol is really available today.

Mesh members who wish to serve as well as be served need to add traffic from outside:

- Typically inbound to port 80 and 443 for Web traffic and only from ports above 1024, again assuming no special mesh protocol is in place.
- If operation as a remote disk or other FUSE file system services are desired, this will typically operate over ssh protocol, or if supportable SSL, or if the mesh device is configured to do so, over the inbound ports 80 and 443.
- Discovery is perhaps best achieved by an ARPing to find local systems willing to communicate followed by an http request asking for services available, which begins a conversation between the LLM mechanisms on the two sides to use whatever protocol they may agree to on a case-by-case basis. Alternatively, DNS may be used to find a site supporting more network distant resources for communication initiation between mesh components.
- While formal goodbye is a desired process for ending a discussion, timeouts and other similar network drops or failures obviously have to be tolerated.
- As a general rule, mesh components can either provide services as a surrogate for others or do introductions for direct communication between other mesh components.

Over time it is expected that additional protocol elements will be developed by mesh members for their own use and/or shared with others, and as sovereign components, they may choose whatever they wish and implement it. As a result, it is expected that any process which can transmit and receive content may be used and these will and should be unpredictable in advance and change with time in the same way as frequency shifting is used in radio communication for reliability and non-interference and surveillance. See you at IP address 197.62.57.89 on UDP port 22341 at 1709.23-27 Pacific time tomorrow using ...

