Frames: Morse, Teletype, Ethernet, X.25 and ATM

Morse in 1840 used long and short tones, Teletype in 1920 used high and low tones with start/stop signals between characters.
IEEE Ethernet packets launched at Xerox in 1973. Since 1983 they use a 56 alternating "1" & "0" bits timing preamble and a 48-bit global MAC address.
ITU's X.25 launched originally in 1976 with "01111110" delimiters → (evolving into) Frame Relay's use with a 10-bit address.
ITU's ATM launched originally in 1988 with a fixed 5-byte header and 48-byte body → IETF's MPLS 20-bit packet routing.

In 2026, Ethernet dominates all networking via the Internet, perhaps 80% to 90% or more.
Click to see MPLS in Australia in 2026. Click to see its evolution into SD-WAN.

Back to Teletype.

Teletype 5-bit frames started in Australia in the 1920s. When PMG launched Datel in 1969, these were 7-bit frames (one byte) ASCII correct?

Yes, your timeline regarding the transition from 5-bit to 7-bit ASCII systems is largely correct, but there is a crucial technical nuance to clarify regarding "one byte."

Here is how the history and technology break down:

1. The 1920s: 5-Bit Teletype Era

You are spot on here. In the 1920s, Australia's early teletype and telegraph systems relied on **5-bit character encodings**, specifically the **Baudot code** (which later evolved into the internationally standardized International Telegraph Alphabet No. 2, or **ITA2**).

- Because 5 bits only allows for 32 unique combinations (2^5), the system used a mechanical "Letters/Figures" shift key to swap between letters and numbers/punctuation—much like the Shift key on a modern keyboard.

2. 1969: PMG Launches Datel

When the Postmaster-General's Department (PMG) launched the **Datel service in 1969**, it was Australia's first official "common user data network". However, Datel itself wasn't a specific code or frame structure; it was a **data transmission service** that provided users with a PMG-supplied modem to send raw binary data over either ordinary switched telephone lines or leased lines.

Because Datel just provided the "pipe," it could technically carry whatever code a company's computers or terminals used. However, the dominant standard for the remote terminals utilizing Datel at that time was indeed **7-bit ASCII**.

3. The Nuance: 7 Bits vs. One Byte

While ASCII is strictly a **7-bit code** (allowing for 128 characters), it was rarely sent as a 7-bit frame over a telecommunications line:

- **The Parity Bit:** In standard asynchronous data communications of that era (such as on the ubiquitous Teletype Model 33), the 7 ASCII data bits were almost always accompanied by an **8th bit used for error checking (parity)**.
- **The "Byte" Alignment:** By 1969, the computer industry (heavily influenced by the 1964 release of the IBM System/360) was standardizing the definition of a "byte" as exactly **8 bits**.

Therefore, while the character encoding used by Datel terminals was **7-bit ASCII**, the actual physical transmission frame over the wire usually consisted of an **8-bit byte** (7 bits of ASCII data + 1 parity bit), bracketed by 1 start bit and 1 or 2 stop bits.

By "frame" I'm referring to start and stop bits, a standard part of Teletype

Got it—you are referring specifically to the **asynchronous serial frame structure**, which wraps the data bits with start and stop bits to synchronize the mechanical or electronic receiver.

With that definition of a "frame" in mind, **no, the Datel-era frames were not 7 bits total, nor were they exactly "one byte" in size.**

Here is the exact breakdown of how those 1969-era Datel frames were structured compared to the 1920s:

1. The 1920s Teletype Frame (5-bit data)

In the 1920s, a standard Baudot/ITA2 teletype frame was typically **7.5 bits long** (sometimes rounded up to 8 bits depending on the mechanical setup). It consisted of:

- **1 Start Bit** (Space)
- **5 Data Bits** (Baudot code)
- **1.5 Stop Bits** (Mark)—the fractional stop bit was a mechanical requirement to give the clunky teletype gears enough time to reset before the next character arrived.

2. The 1969 Datel/ASCII Frame (7-bit data)

When PMG's Datel service launched in 1969, a standard asynchronous ASCII frame (such as those used by the incredibly popular Teletype Model 33) was **11 bits long**.

Even though ASCII itself is a 7-bit code, the physical frame transmitted over the line looked like this:

- **1 Start Bit**
- **7 Data Bits** (The ASCII character)
- **1 Parity Bit** (For error checking)
- **2 Stop Bits** (Mechanical/electronic cushion)
- **Total = 11 bits per character frame.**

Why it doesn't align with "one byte"

Because these frames required 11 bits to transmit a single 7-bit ASCII character, they didn't cleanly map to "one byte" (8 bits).

If you were typing at a standard 110-baud Datel terminal in 1969, the system was moving at 110 bits per second. Because each frame was 11 bits long, you were transmitting exactly **10 characters per second**.

At each client, each frame was still character by character at a physical "packet" level. Ethernet and ATM packets physically didn't exist, and Hawaii didn't get its radio signal operating till 1971.

You have hit the nail on the head. In 1969, the entire paradigm of data transmission was fundamentally different from the networked world we know today.

Character-as-a-Packet

You are absolutely correct that each individual character frame acted as its own independent "packet" at the physical level. There was no concept of wrapping a block of text into a larger payload with a header and a trailer.

- **State of the Art:** Every time a user hit a key on a terminal, that single 11-bit asynchronous frame (Start + 7 Data + Parity + 2 Stop) was serialized and blasted down the PMG line on its own.
- **No Buffering:** The client terminal didn't buffer your sentence, wait for a carriage return, and send it as a block. It was pure, raw, character-by-character streaming. If you stopped typing, the line simply went idle (holding a continuous "Mark" or binary 1 state).

The Timeline Milestone

Your historical landmarks are exactly right:

- **ALOHANET (1971):** You are referring to Norman Abramson’s ALOHANET at the University of Hawaii, which became operational in 1971. It pioneered the random-access channel sharing technology (packet broadcasting over radio) that proved data *could* be chopped into packets and sent over a shared medium.
- **Ethernet (1973):** Drawing directly from ALOHANET's math, Bob Metcalfe and his team at Xerox PARC didn't invent Ethernet until 1973 (and it wasn't commercialised until much later).
- **ATM (1980s):** Asynchronous Transfer Mode (ATM) with its fixed 53-byte cells wouldn't emerge until the mid-to-late 1980s as broadband ISDN began to take shape.

So in 1969, when an Australian business hooked up to PMG Datel, they were operating in a world of dedicated point-to-point circuits (or circuit-switched telephone lines) where the physical "packet" size was exactly one character wide.

In the US, when did Ethernet packets and ATM packets (at a physical framing level) take off.

In the United States, the physical framing shift from character-by-character serial data to structured **Ethernet frames** and **ATM cells** happened in two completely different waves, serving distinct parts of the tech infrastructure.

Here is exactly when they took off and how their physical framing mechanics fundamentally redefined the wire.

1. Ethernet Frames: The Local Area Network (LAN) Boom

When it took off: The **mid-to-late 1980s** for corporate environments, and the **early 1990s** for mass commercial ubiquity.

- **The Evolution:** Though invented at Xerox PARC in 1973 and standardized by the IEEE in 1983/1985 (as IEEE 802.3), Ethernet was originally expensive and required stiff, hard-to-manage coaxial cables ("Thicknet" and "Thinnet"). It truly erupted in **1990–1991** with the introduction of **10BASE-T**, which allowed Ethernet to run over cheap, flexible unshielded twisted-pair (UTP) copper phone wires using RJ-45 clips.
- **The Physical Framing Level:** Instead of streaming 11-bit character frames as you typed, Ethernet introduced a massive hardware variable-length envelope (64 to 1518 bytes).
- **How it worked on the wire:** The interface card buffered data into a single physical block. On the wire, it began with a **Preamble** (7 bytes of alternating `10101010`) and a **Start Frame Delimiter** (1 byte: `10101011`). This hardware-level rhythmic "drumbeat" told the receiving network card to wake up and sync its clock because a large frame—containing source and destination MAC addresses, a payload of data, and a 4-byte cyclical redundancy check (CRC) error code—was arriving all at once.

2. ATM Cells: The Backbone and Carrier Revolution

When it took off: The **mid-to-late 1990s**.

- **The Evolution:** Developed in the late 1980s, Asynchronous Transfer Mode (ATM) was championed by massive telecommunications giants (like AT&T) and the Department of Defense. Telcos believed variable-length Ethernet packets were messy for mixing voice and data. ATM promised a unified world network. By the mid-to-late 90s, ATM heavily dominated regional carrier backbones and became the foundational Layer 2 framing mechanism for **ADSL consumer internet**.
- **The Physical Framing Level:** ATM famously rejected "packets" or "frames" in favour of fixed-size **"Cells."** Every single ATM cell on the wire was rigidly fixed at exactly **53 bytes** (5 bytes of header + 48 bytes of payload).
- **How it worked on the wire:** Telcos used a technique called cell-switching. Because a phone call cannot tolerate jitter (choppy delays), forcing data into small, identically sized 53-byte cells meant hardware switches could route them at blinding, predictable speeds without waiting for a massive 1,500-byte file transfer to clear the line. If you sent an IP packet over ATM, hardware chopped that packet into multiple 48-byte segments, adding a 5-byte header to every single piece.

Direct Comparison: The Framing Clash

By the late 1990s, these two technologies clashed in the marketplace.

Attribute	Ethernet Frames (LAN King)	ATM Cells (Carrier King)
Peak US Era	1990s onward (ubiquitous today)	Mid-1990s to early 2000s (now legacy)
Physical Size	Variable (64 to 1518 bytes)	**Fixed rigidly at 53 bytes**
Hardware Sync	Uses a physical 8-byte Preamble/SFD per frame	Continuous synchronous framing (usually mapped onto SONET/SDH fibre)
The "Tax"	Minimal overhead (typically 18–26 bytes per large packet)	High overhead ("Cell Tax")—splitting a packet across multiple 5-byte headers wasted bandwidth

Why ATM Died and Ethernet Won

ATM was elegant on paper, but it was complex and expensive. By the late 1990s, chip technology allowed Ethernet to jump from 10 Mbps to 100 Mbps ("Fast Ethernet"), and by 1998, to **Gigabit Ethernet** (1,000 Mbps).

At Gigabit speeds, bandwidth was so abundant that the predictability of ATM's 53-byte cells no longer mattered. It became far cheaper to blast massive, variable-length Ethernet frames down a fiber line than to use expensive specialized hardware to slice and dice data into tiny ATM cells. By the mid-2000s, Metro Ethernet entirely replaced ATM in carrier backbones, leaving ATM relegated to legacy copper DSL loops.

In Australia, my understanding is ATM over phone lines was launched alongside HFC Ethernet by Telstra in 1997, two new physical framing standards for packets.

You have the absolute correct technological blueprint, but **Telstra actually rolled these out in two distinct phases between 1996 and 2000, rather than a simultaneous 1997 consumer launch.**

While Telstra's nationwide commercial **ATM core network did officially launch in mid-1997**, it was exclusively for corporate data, government infrastructure, and backhauling traffic. The two physical framing standards you mentioned hit Australian consumers as a tale of two different infrastructures:

1. HFC Ethernet (Launched 1996–1997)

Telstra rolled out its Hybrid Fibre-Coaxial (HFC) cable network primarily to compete with Optus Vision for Pay-TV (Foxtel). Recognizing they could use the same wire for data, Telstra launched **BigPond Cable Internet** to the public in late 1996 (limited trials) and went nationwide in **May 1997**.

- **The Framing:** This introduced true packet framing to Australian homes. At the time, it didn't use DOCSIS yet; it used **Motorola proprietary cable modems**.
- **The Client Level:** The Motorola cable modem sat on the user's desk and bridged the proprietary HFC RF signals directly into a standard **10BASE-T Ethernet frame** via an RJ-45

LAN card in the user's PC. For the first time, an Australian consumer's computer wasn't sending characters—it was buffering and firing variable-length Ethernet packets.

2. ATM over Phone Lines (Launched 2000)

While Telstra was using ATM in its backbone in 1997 to link capital cities and handle heavy enterprise Frame Relay traffic, **ATM over consumer phone lines (ADSL) did not launch until August 2000.**

When Telstra BigPond finally deployed its first DSLAMs (Digital Subscriber Line Access Multiplexers) into telephone exchanges, they faced a massive technical challenge: the existing copper telephone network was built for steady, synchronous voice, not bursts of messy internet traffic.

- **The Solution:** They ran **ATM over the copper.**
- **The Framing:** Your PC would generate an Ethernet packet. The Alcatel or Efficient Networks ADSL modem on your desk would take that packet and chop it up using **PPPoA (Point-to-Point Protocol over ATM)** or **PPPoE**. It wrapped those pieces tightly into rigid **53-byte ATM cells** (48 bytes data + 5 bytes header) and shot them down the copper line.
- **The Destination:** At the local Telstra exchange, the DSLAM aggregated those ATM cells and passed them deep into Telstra's national ATM backbone switches (the ones they had been building since 1997).

Summary of the 90s Shift

By the turn of the millennium, Telstra had fundamentally broken the 1960s character-by-character Datel legacy. If you were a premium user on **BigPond Cable**, your physical framing standard was **Ethernet** from end-to-end. If you were on **BigPond ADSL**, your physical framing standard over the phone line was **ATM**, acting as a highly structured transit system to carry your data safely to the exchange.

Ok, ADSL wasn't till 2000, but ATM packets with 53 bytes was launched at a government / corporate level, supplanting teletype, in 1997

Yes, that is exactly correct. In **1997, Telstra officially launched its commercial ATM network** (marketed as *Telstra ATM Service*) aimed squarely at corporate headquarters, financial institutions, and government departments.

This rollout marked the final nail in the coffin for the remaining legacy of character-based teletype engineering at the enterprise level, fundamentally shifting Australia's business infrastructure to **53-byte cell switching**.

Here is how that 1997 corporate ATM deployment reshaped Australian data and officially supplanted the older systems:

1. The Death of the Dedicated Serial Circuit

Before 1997, large corporations (like the big banks) and government agencies connected their branches using older technologies like point-to-point leased lines, **X.25 packet switching**, or **Frame Relay**.

- Many of these systems still relied on serial protocols derived directly from early teletype concepts.
- They treated the connection as a dedicated, synchronous, or character-by-character pipe. If a branch office wasn't transmitting data, that expensive copper or satellite bandwidth was completely wasted.

2. The 1997 ATM Solution: "Slicing" the Enterprise

When Telstra lit up its national ATM backbone in 1997, it gave corporations a way to run voice, video, and data over a single physical pipe. Because everything was chopped into rigid **53-byte cells** at the hardware level, Telstra could offer enterprises something teletype and early packet systems never could: **Quality of Service (QoS)**.

- **Voice & Video:** Placed into ATM cells tagged with "Constant Bit Rate" (CBR). The hardware switches guaranteed these cells bypassed all other traffic so corporate phone calls wouldn't drop or jitter.
- **Standard Internet/Data:** Placed into ATM cells tagged with "Available Bit Rate" (ABR) or "Unspecified Bit Rate" (UBR), filling in the remaining gaps.

3. The Physical Reality at the Office

For a corporate IT department in Sydney or Melbourne in 1997, adopting ATM meant replacing old serial multiplexers with massive **ATM Edge Switches** or high-end routers equipped with ATM interface cards.

- At the office boundary, local variable-length Ethernet packets were ingested, stripped, and segmented by hardware into continuous streams of 53-byte ATM cells.
- These cells were then blasted across Telstra's high-speed **SONET/SDH fibre optic rings** connecting the capital cities.

By implementing this at the enterprise level in 1997, Telstra built the heavy-duty data engine that allowed Australian business infrastructure to digitise. It proved the reliability of 53-byte cell framing on a massive scale, setting the stage for Telstra to safely deploy that same ATM architecture to consumer phone lines via ADSL three years later.

And with HFC in 1997, its Ethernet packets were visualized more as a radio signal over coax, correct?

Yes, absolutely. Visualizing 1997-era HFC internet as a radio signal broadcast over a copper pipe is the perfect mental model for how the technology actually functioned.

When Telstra lit up BigPond Cable, your PC was dealing with standard, digital Ethernet packets, but the moment those packets hit the cable modem, they were fundamentally transformed into **Radio Frequency (RF) signals**.

Here is exactly how that "radio over coax" system worked on a physical framing level:

1. The Cable Modem was an Actual Radio

The word "modem" stands for **Mo**dulator/**Dem**odulator. In the context of HFC, your 1997 Motorola CyberSURFR cable modem was quite literally a highly sophisticated, two-way digital radio transmitter and receiver wrapped in a plastic box.

- **The Downstream (Listening):** Telstra assigned data to a specific, unused 6 MHz or 8 MHz analog television channel slot on the cable. Your cable modem tuned its radio receiver to that exact frequency (usually somewhere above 450 MHz) to "listen" for your incoming data.
- **The Upstream (Broadcasting):** When you clicked a link, your modem became a radio station, modulating your outbound data and broadcasting it backward up the wire on a low-frequency return band (typically between 5 MHz and 42 MHz).

2. Slicing Packets into Waves (Modulation)

Copper coaxial cable cannot inherently understand a raw binary `1` or `0` square-wave pulse over long distances without immense degradation. To bypass this, the modem took the digital Ethernet packet and used a technique called **QAM (Quadrature Amplitude Modulation)** to convert the binary bits into complex analogue radio waves.

- By shifting the phase (timing) and amplitude (strength) of the radio wave, a single wave cycle could represent a specific pattern of multiple bits at once.

3. The Physical Journey: Light to Airwaves

The physical framing transition across the Hybrid Fibre-Coaxial network was a brilliant exercise in media translation:

- **At Your PC:** Raw electrical data bits aligned in standard **Ethernet packets** left your network card.
- **At Your Modem:** The modem encoded those packets, converted them into **RF radio signals**, and injected them into the copper coax.
- **Down the Street:** The radio signals travelled through the copper shield of the coax, passing through analog RF amplifiers along the power poles to keep the signal strong.
- **At the Optical Node:** The neighbourhood node took that electrical radio signal and used it to amplitude-modulate a laser. The laser translated the exact frequency, shape, and rhythm of the radio waves into **pulses of light** to flash down Telstra's fibre backbone to the headend exchange.

4. It Was a Shared Radio Medium

Because it was fundamentally a radio signal propagating through a branching network of copper pipes, **the medium was entirely shared**. Everyone on your local street segment (often up to 500 or 1,000 homes) was tuned into the exact same physical radio broadcast coming down the line from Telstra.

Your modem only paid attention to the packets within that radio stream that had its specific hardware address stamped on them, ignoring the rest of the neighbourhood’s data cascading down the wire.

Are you interested in how this early proprietary **"radio over coax" framing** eventually became globally standardized under the **DOCSIS** specification that still runs cable networks today?

Ok, thinking back to 1971 and Aloha net, and 1980 and Ethernet, and then Wi-Fi in the 1990s, are there similarities, visually, between those packets?

Yes, absolutely. If you were to lay out the physical data streams of a 1971 ALOHANET packet, a 1980 DIX Ethernet frame, and a late-1990s Wi-Fi (802.11b) frame side-by-side, **their visual structures are strikingly similar.**

Bob Metcalfe (the inventor of Ethernet) and Vic Hayes (the "father of Wi-Fi") both explicitly based their technologies on Norman Abramson's mathematical breakthroughs with ALOHANET. Because they all solved the same problem—how to broadcast data across a single, chaotic, shared medium—their physical packet architecture followed the exact same visual anatomy.

If you visualize these packets as physical objects traveling through space or down a wire, they all share four distinct, sequential zones:

1. The "Wake Up" Zone (The Physical Preamble)

Before any real data can be read, the receiver's hardware clock must lock onto the incoming transmission rhythm.

- **ALOHANET:** Began with a raw bit-sync pattern.
- **1980 Ethernet:** Used a 7-byte **Preamble** of alternating ones and zeros (`10101010...`) followed by a 1-byte **Start Frame Delimiter** (`10101011`). It looked like a digital drumbeat screaming, *"Wake up, a packet is starting right now!"*
- **1990s Wi-Fi:** Borrowed this exactly. Early 802.11b frames used a massive **128-bit Preamble** of pure scrambled ones and zeros to give the radio receiver time to adjust its signal gain before the real data hit.

2. The "Who Is This For?" Zone (Addressing Headers)

Because all three technologies used a shared medium where *everyone* could hear *everything*, the physical front of the packet had to clearly state the sender and receiver.

- **ALOHANET:** Included a terminal ID.
- **Ethernet:** Introduced the **Destination MAC Address** and **Source MAC Address**.
- **Wi-Fi:** Expanded on Ethernet's look. Because wireless signals bounce off walls or hop through access points, Wi-Fi frames visually look like an extended Ethernet header, often carrying up to **four MAC addresses** (Source, Destination, Receiving Access Point, and Transmitting Access Point) to coordinate the radio traffic.

3. The Cargo Zone (The Payload)

The actual user data sits in the middle.

- **ALOHANET:** Carried a fixed, tiny **40-byte or 80-byte data block** containing a line of text typed from a remote terminal.
- **Ethernet & Wi-Fi:** Blew this wide open, wrapping a variable payload up to **1,500 bytes** long right in the center of the frame.

4. The "Did it Get Corrupted?" Zone (The Trailer)

Because radio waves fade and electrical currents on a shared wire suffer from interference, all three structures dedicated their final bits to error checking.

- **ALOHANET:** Concluded with a **16-bit Cyclic Redundancy Check (CRC)**. If the math didn't match at the central computer, it assumed a collision occurred and threw the packet away, triggering a random-delay retransmission.
- **Ethernet & Wi-Fi:** Upgraded this to a **32-bit Frame Check Sequence (FCS)** pasted to the very end of the packet.

The Evolution of the Packet Shape

1971 ALOHANET Packet	1980 Ethernet Frame	1999 Wi-Fi Frame
Bit Sync Preamble	8-byte Preamble / SFD	128-bit Radio Preamble
Terminal ID (Header)	12-byte MAC Addresses	24-byte Extended MAC Headers
40 or 80-Byte Payload	**46 to 1500-Byte Payload**	**0 to 2312-Byte Payload**
16-bit CRC Checksum	32-bit FCS Checksum	32-bit FCS Checksum

The Core Difference: How they "Listened"

While they looked almost identical on the wire/in the air, their physical behaviour varied slightly:

- **ALOHANET (Pure Randomness):** A node didn't listen to the radio channel before talking. It just blasted its packet. If it hit another packet, they collided, and both nodes tried again later.
- **Ethernet (Listen then Talk):** Because it ran on a copper wire, devices could safely "listen" first. If the wire was quiet, they fired. If two devices fired at the exact same millisecond, they could instantly detect the voltage spike (Collision Detection) and stop transmitting immediately to save bandwidth.
- **Wi-Fi (Listen, Predict, then Talk):** Radios cannot listen while they are transmitting on the same frequency (they blind themselves). So Wi-Fi frames added a physical **Duration field** into the header. This field told every other nearby laptop, *"Hey, I am about to transmit a packet that will take exactly 500 microseconds—set a timer and don't talk until I am done."*

** End of Page