Early ("Research") Datakit

Hardware

- ATDM network, based on 180 bit packets @ 7.5Mbps:
  - Each packet has 18 "words" of 10 bits each:
    - Word 1: 9 bit link (host) address + parity
    - Word 2: 9 bit channel number + parity
    - Word 3-18: 9 bit envelope + parity
  - Each envelope contains a data or a control byte:
    - 0x1nn: data byte 0xnn
    - 0x0nn: control byte 0xnn (0x01-0x3f are system controls, 0x40-0x7f are user protocol controls)
    - 0x000: empty envelope
- As compared to Spider, the "loop" has effectively shrunken down to the switch backplane and each board on the backplane is a terminal interface unit:
  - One type of interface unit connected to 4 ascii RS232 lines @ 9600 bps (for use with TTY's).
  - Another type of interface connected to hosts using a many wire interface, possibly assisted by a front-end processor.
  - Maximum 512 logical interfaces on a single switch.
  - Star topology from switch to hosts / terminals.
- Each host interface has a fixed link number and can use 512 channels
  - Channel 0 dedicated to maintenance.
  - Channel 1 dedicated to signaling (= communication with switch controller)
  - Other 510 channels available for data communication.
- Packet guarantees:
  - packets will be delivered in order, without duplicates; packets could be dropped however (parity errors, buffer overflow)
  - Within one switch packet boundaries are respected, inter-switch it is possible packet contents are split/merged – without changing order ("it is a byte stream, not a packet stream").
- One "Common Control" (CC) per switch:
  - One interface board in each switch is special and controls the link address translation hardware (i.e. the switch proper).
  - This board is connected to a small computer (normally a PDP-11/23) that runs the CC software.
    - Automatically connected to channel 1 of each host interface in the switch
  - Two CC software generations: CMC by Chesson, later replaced by TDK by McMahon et al.

Link protocol

- Evolved over time. First version in 1979 by Chesson. Via "NK" and "FAU" arrives at "URP" around 1982, by Marshall et al.
- URP ("Universal Receiver Protocol") is best described in US patent 4,852,127, including pseudo-code for sender and receiver.
- URP has one algorithm on the receiver side and five sender algorithms (with/without flow control, with/without error correction, character or block oriented):
  - Flow control based on a window, typically 3 x 64 bytes in size; sender requests positive acknowledgements at certain intervals in order to checkpoint receiver status
  - Error correction based on restarting from point of error, resending everything after that point; receiver detected errors cause a negative acknowledgment to be sent
  - No facilities for virtual circuit build up and tear down – this happens at a higher level (see below)
  - In 8th Edition practice everything uses "protocol 5" (= block oriented with flow & error control)
- Focus on hardware implementation of URP, software implementation is seen as a fallback.

Virtual circuit build up and tear down (TDK)

- Design appears modelled on approach used for touch-tone phones. On/off hook type messages are sent on channel 1 in binary, destination ("dial") information is sent on the data channel in ascii. The switch adds source identification ("caller ID").
- Odd channels reserved for switch, even channels reserved for client/host

<u>Connecting a client host to a server host</u>

- Client selects a free channel N
- Client send connect message to CC on channel 1:
  - Binary "connect" message identifying channel N, and the traffic class (fast/slow)
- Client listens for dial tone on channel N:
  - Dial tone is switch sending a byte with ascii '0'
- Client sends dial string on channel N:
  - Two-line ascii message, maximum total length is 128 bytes
  - First line has destination name and up to 3 parameters, separated by dots; first parameter is service name (by convention)
  - Second line is the user name (normally added by device driver, not the user process)
- CC replies on channel N with confirmation/error message:
  - Binary message of fixed length
- Channel N now connected and ready for use

Announcing a server:

- Server selects a free channel N
- Server sends "announce" message to CC on channel 1:
  - Binary "announce" message identifying channel N and the traffic class (fast/slow)
- Server listens for dial tone on channel N:
  - Dial tone is switch sending a byte with ascii '0'
- Server sends announce string on channel N:
  - Two-line ascii message, maximum total length is 128 bytes
  - First line has announced name
  - Second line is the user id (normally added by device driver, not the user process)
- CC replies on channel N with confirmation/error message:
  - Binary message of fixed length
  - CC enters announced name in the Datakit namespace (see below)
- Server then listens on channel N for new connection requests

Server accepting a new client

- Server reads a 3-line text message from channel N:
  - First line has a channel no. (M), timestamp and traffic class, each as an octal ascii number
  - Second line has server name and first parameter (=requested service)
  - Third line has source name, user name and two parameters; only user name is not optional
  - All fields within a line separated by dots
- Server accepts or rejects the new connection:
  - Server sends binary "accept" or "reject" message on channel 1
  - When accepting, server opens channel M as the circuit to the new client
  - Using channel M is an implicit "accept" as well
- Server handles new client, possibly forking out a child for this task

Closing a channel

- Either a server or a client can close a connection by:
  - Sending a binary "close" message on channel 1, identifying the channel to be closed
  - Other side receives a binary "close" message on channel 1 and acknowledges this
  - Channel is fully closed after receipt of acknowledgement
- Once either side initiates a close, the channel enters a zombie state and no further data is passed

Heartbeat

- CC sends a regular keep-alive message on channel 1 of each link (=host)
- After missing 3 heartbeats a link is considered dead and no further virtual circuits are accepted

Datakit name space (TDK)

- Modelled after US phone system numbering (area-trunk-subscriber):
  - Three level name separated by slashes: area/switch/server
  - Optionally, area/ or area/switch/ can be omitted, local name is then implied
- The "server" part is the announced name:
  - Could be more than one per host, but convention is just one per host
  - Dynamic: if host not currently announced, connection requests fail
- All routing done inside the Datakit network:
  - The area and switch names are configured in the switch software
  - Hosts have no knowledge of how routing takes place; in this sense a Datakit switch has a similar role as an Arpanet IMP

Unix interface (8th edition - 1984)

- Low level interface:
  - Datakit channels exposed as (up to) 512 character (minor-) devices (8th Edition source sets up 95); a "free" channel is found by linear search through minor devices.
  - URP line protocol pushed on each connection stream
  - Connection protocol as per above steps encapsulated in two procedures: *dkdial* (client) and *dkmgr* (server)
  - No facilities for (simulated) datagrams
  - No facilities for urgent data (even though switch and URP protocol support it)
- High level interface:
  - One central Datakit server process per host
  - Accepts connections from the network, for any requested service (= first parameter)
  - Handles simple services in line, complex services handed to external server (much like *inetd* on BSD)