Introduction to Flexilink

John Grant, Nine Tiles

NGP#9, BSI (Chiswick, London), 30 November 2017

Introduction to Flexilink

Evolution of digital platforms

Forwarding plane services for the 21st century

Flexilink history & frame formats

Control plane: addressing and routing

Security

Transport layer options

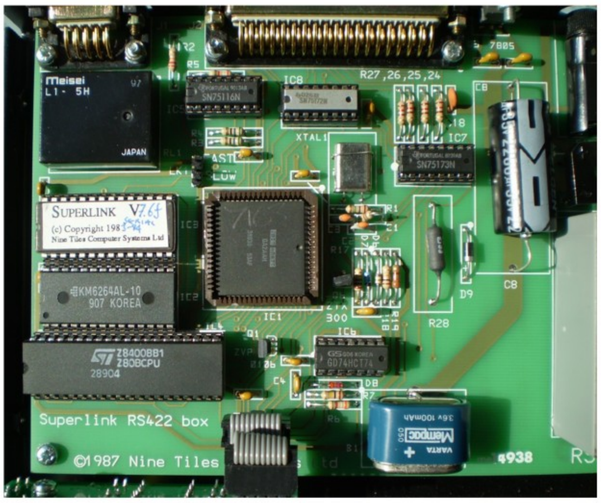
Migration

Topics where details need further study

- 1970s
  - Arpanet IMP
  - almost everything in software
    - hardware calculation of checksum
  - limited memory (64KB address space)
    - connectionless routing
      - to avoid keeping state
      - everything needed for routing must be in the header
  - line speed 0.056 Mb/s
  - 0.6 Mb/s (per pin) memory interface

The image shows one of the original Interface Message Processors, which were used for routing packets in the Arpanet.

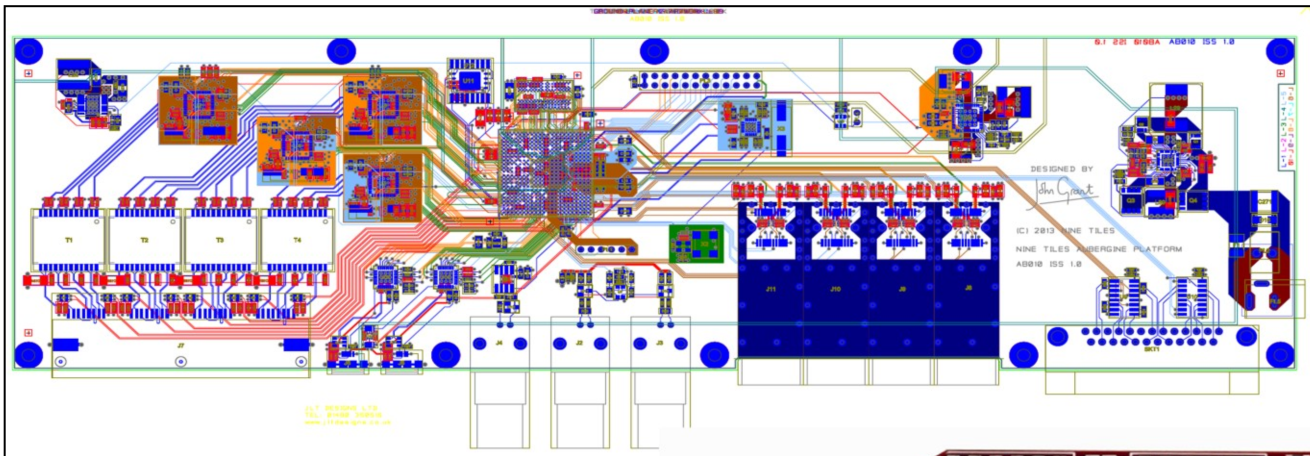
- 1980s
  - Nine Tiles Superlink
  - almost everything in software
    - ASIC: few hundred gates
  - limited memory (64KB address space)
    - connection-oriented routing
      - to reduce per-packet processing
      - can connect by name or location
  - line speed 1.5 Mb/s
  - 2 Mb/s memory interface

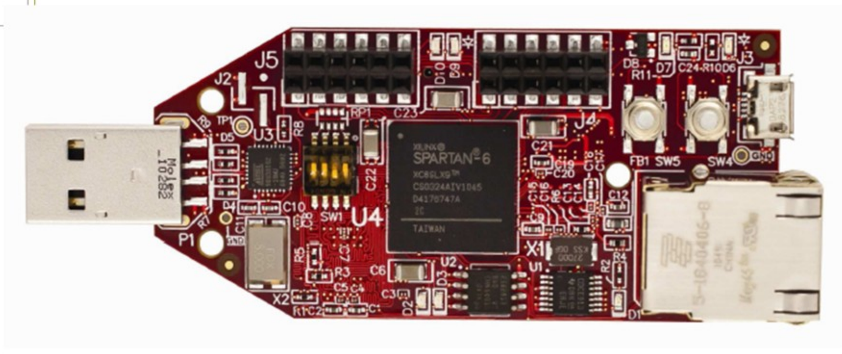


The image shows the interior of a Nine Tiles Superlink node which connected an RS422 interface into a Superlink ring network. (there was also an RS232 version, and plug-in cards for various I/O bus standards). The chipset consists of a Z80 CPU, UV-EPROM, sRAM, and an ASIC which performed a few simple functions such as converting the data between serial and parallel, and writing incoming packets directly into the sRAM.

Its connection-oriented routing allowed users to specify the destination of a flow in various ways, including by name and by position in the ring. Two rings could be connected by a “gateway”, and flows could then be routed through the gateway to a destination on the other ring. Information in packet headers to identify the destination was local to each ring or gateway.

Signalling was in-band, so when a connection was made to a gateway the protocol to make the connection on the second ring was carried as ordinary data on the first ring. Thus flows were source routed, reducing the amount of information each node needed to keep. When addressing content on a file server, the names of the gateway(s), host, and service simply looked like extra components in the file path.



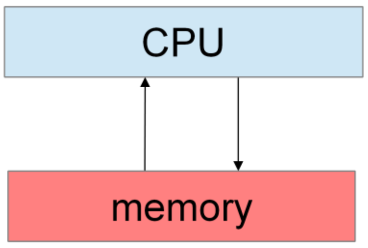


- 2010s
  - can do much more in logic
  - multiple Gb/s line speed
  - few Gb/s memory interface

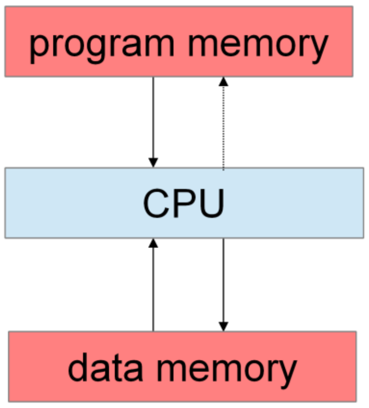
The upper image is of the circuitry in the platform on which Flexilink has been implemented; it contains a Xilinx Spartan 6 SLX45T FPGA, serial flash, dRAM, and physical layer interfaces. There is no hard-wired CPU, but the logic in the FPGA includes a “soft” processor which runs code compiled from a high-level programming language tailored to the needs of protocol processing. This CPU implements various control functions but is not involved in the forwarding of Flexilink packets.

The lower image is an Avnet Microboard, with a Spartan 6 SLX9, on which the soft processor was developed..

- Computing systems are sequential
  - with everything passing over the memory interface
    - incoming data must be buffered until the CPU is ready to look at it
  - and memory speeds haven't increased as much as other parameters



von Neumann architecture

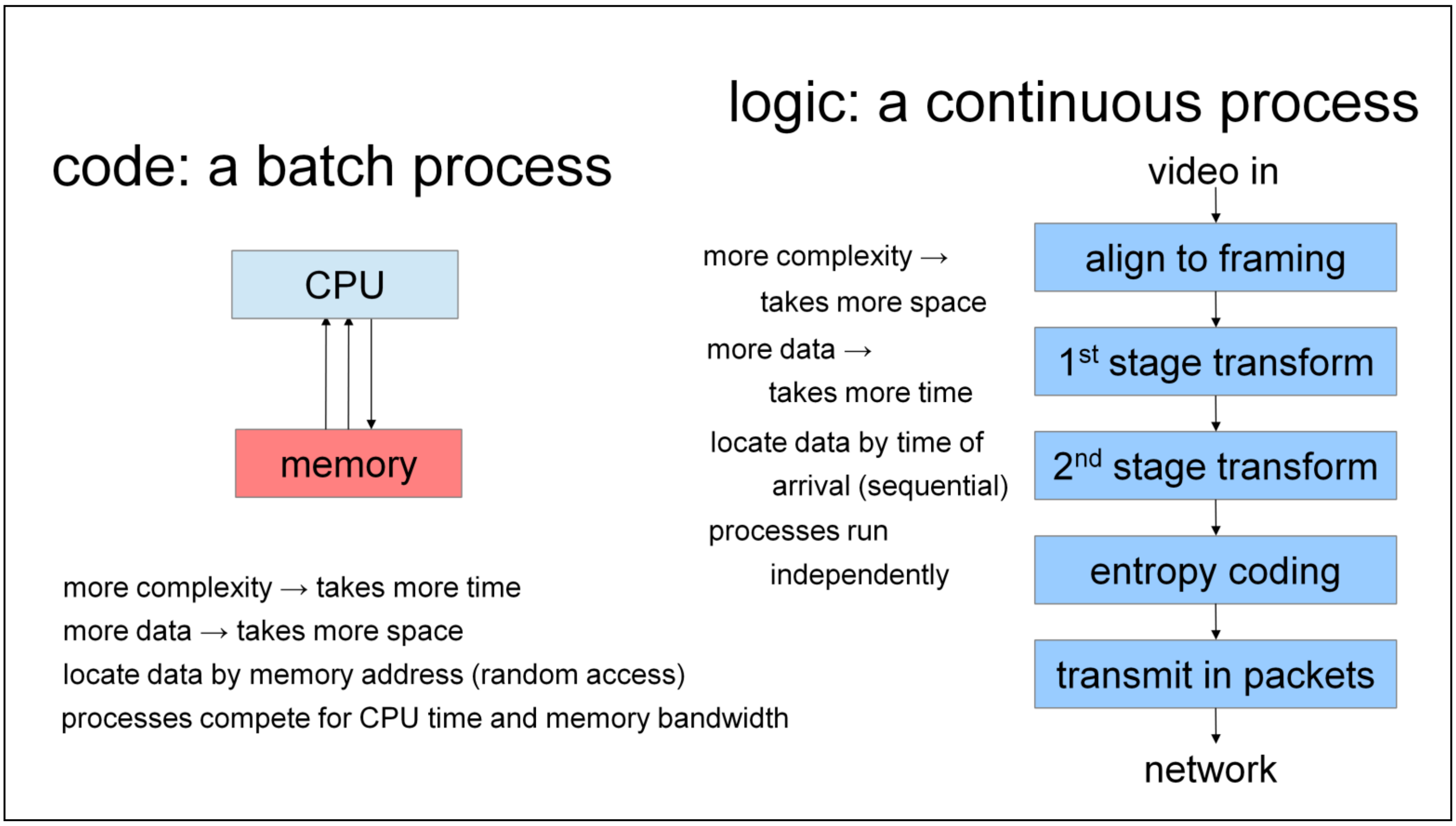


Harvard architecture

CPU speeds have gone up by a factor of about a thousand, from a few MHz to a few GHz, while line speeds have gone up by about a hundred thousand, e.g. from 9.6 kb/s to 1 Gb/s at the edge or from 1 Mb/s to 100 Gb/s in the core. Thus inspecting every packet by software is now less viable.

dRAM is an analogue technology, and speeds haven’t increased as fast as those of digital technologies; access times have gone down by a factor of about 10, though caches and bigger column sizes have provided some mitigation.

dRAM sizes, however, have gone up by a factor of about a million, from a few Kb to a few Gb. Thus it is possible to keep much more state now.



code: a batch process

more complexity → takes more time

more data → takes more space

locate data by memory address (random access)

processes compete for CPU time and memory bandwidth

logic: a continuous process

more complexity → takes more space

more data → takes more time

locate data by time of arrival (sequential)

processes run independently

Logic and software really are very different, with things that occupy time in one occupying space in the other.

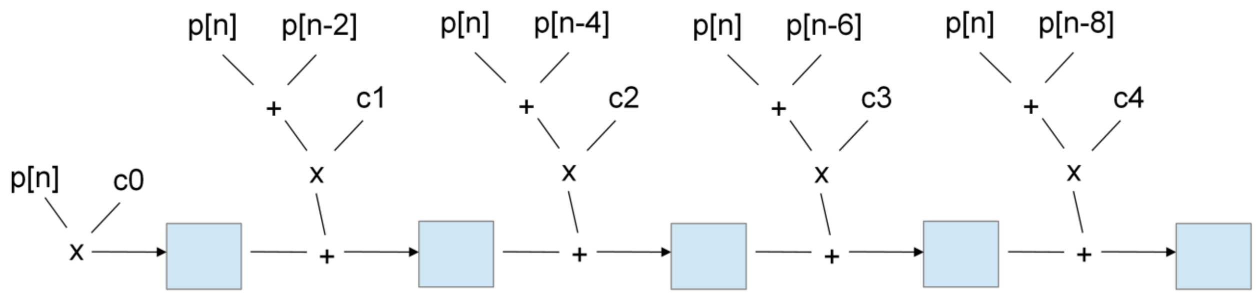
Programs operate on variables in memory; variables are updated one at a time. Logic operates on signals; all the signals change whenever the clock ticks.

Languages for describing logic are much less well-developed than programming languages; VHDL and Verilog are at a similar level to assembly-code, while “high level synthesis” is restricted to describing functions as if they are to be executed by a CPU.

The example on the right-hand side shows logic that transmits a video signal on the network. Each stage takes its input from the previous stage and they all run in parallel, unaffected by anything going on elsewhere in the system, whereas in software the different processes run one at a time, competing with each other and with other processes for CPU time and memory bandwidth. Thus the CPU must be clocked at a much higher rate than the pixel clock, and it needs to decode instructions in addition to performing the calculations, so it can be expected to consume much more power than in implementation in logic.

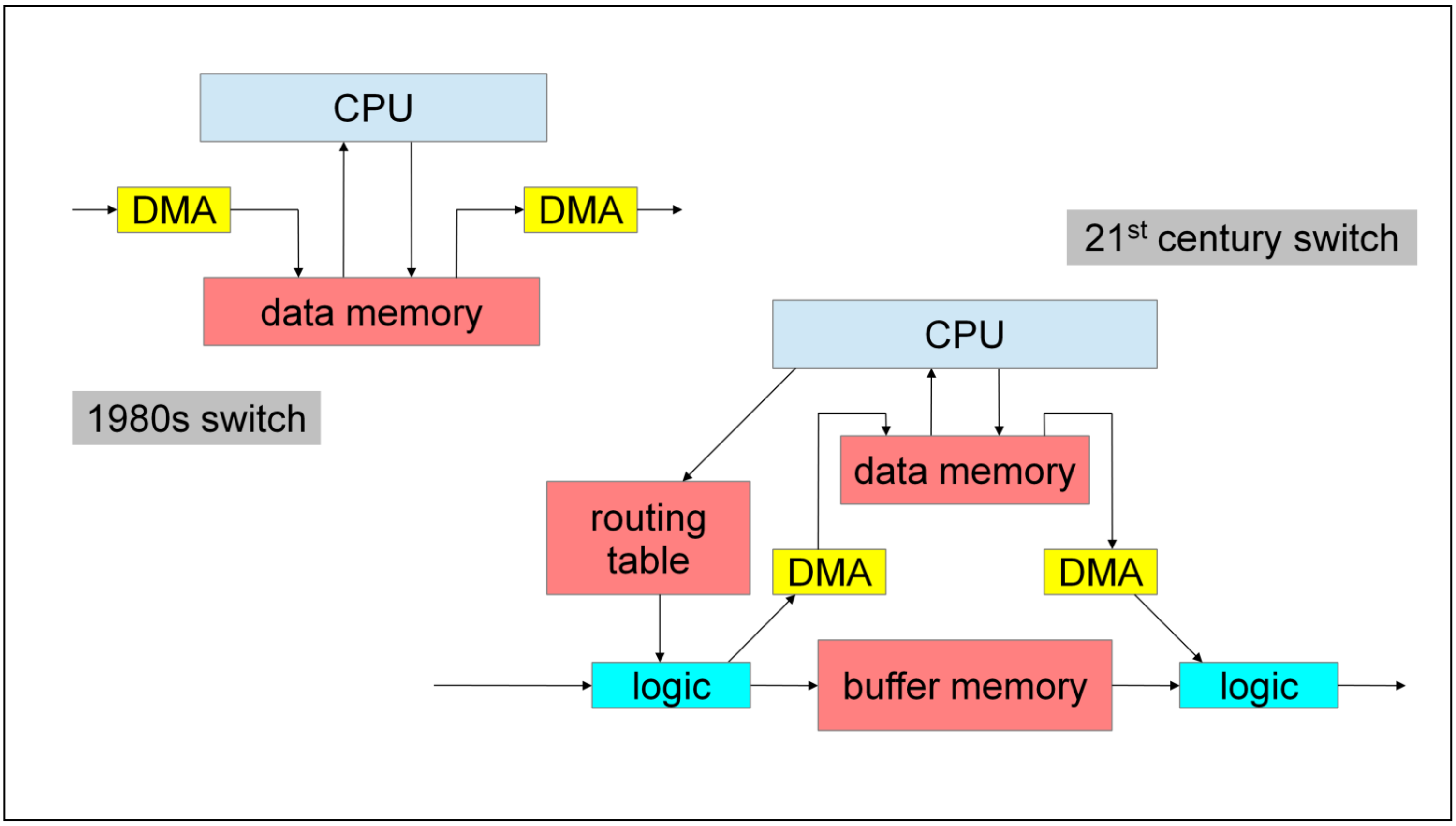
Example: wavelet transform

- 9 pixel values and 5 coefficients per pixel (for each component)
  - c0p[n] + c1(p[n-1] + p[n+1]) + c2(p[n-2] + p[n+2]) + … + c4(p[n-4] + p[n+4])
- – in logic a 5-stage multiply-and-accumulate at pixel clock rate
  - 5-clock pipeline: pixel values used by each block in turn



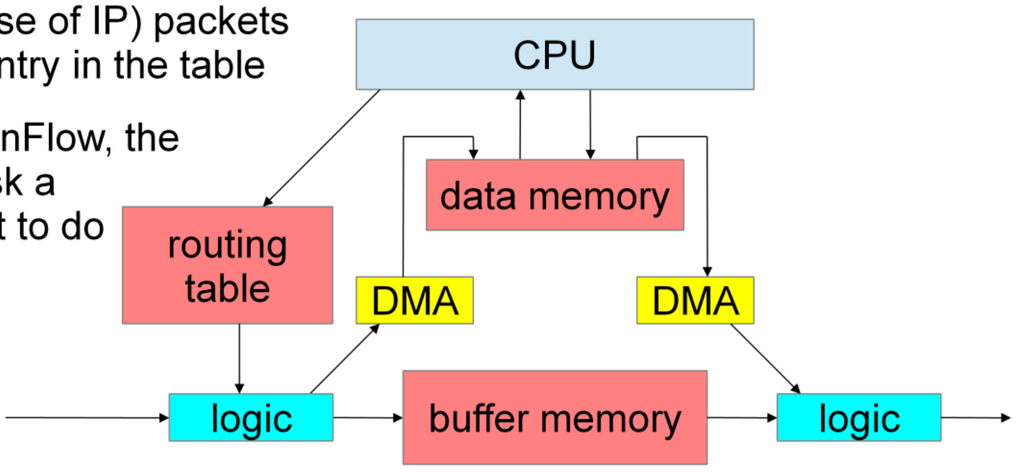
Each of the pale blue squares is a register holding the result of a calculation; n is the current pixel number, which is incremented at each clock. For example, the calculation for pixel 10 needs the values for pixels 6 to 14. It begins when n=10, storing c0p10 in the first register. In the next clock, when n=11, c0p10 + c1(p11 + p9) is stored in the second register. Meanwhile, the first register is updated to hold c0p11 for the next calculation, the third is set to c0p9 + c1(p10 + p8) + c2(p11 + p7), the fourth to c0p8 + c1(p9 + p7) + c2(p10 + p6) + c3(p11 + p5), and the fifth to the final result for pixel 7.

A naive calculation in software would read nine pixel values and five coefficients and perform eight add operations and five multiply operations for each pixel. Even with optimisations such as keeping pixel values in registers and using the multiply-and-accumulate instructions that are available in some processors, and assuming the code is in a cache rather than needing to be read from memory each time, this still requires the CPU to cycle many times faster than the pixel clock.

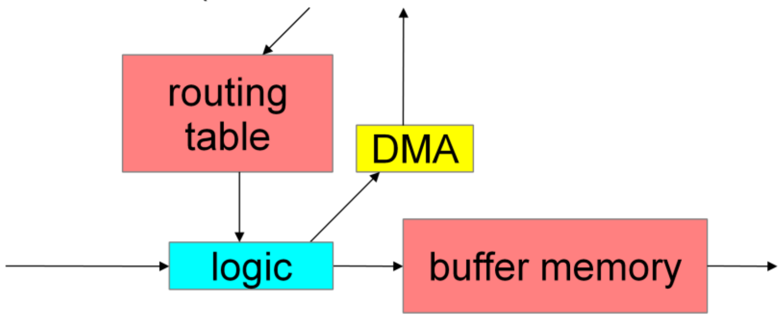


We no longer have to do everything in software.

- Most packets don't go through the CPU's memory
- Entry in the routing table shows how to route each “flow”
- Control and management packets routed to the CPU
  - also (in the case of IP) packets for which no entry in the table
  - with SDN/OpenFlow, the CPU has to ask a controller what to do

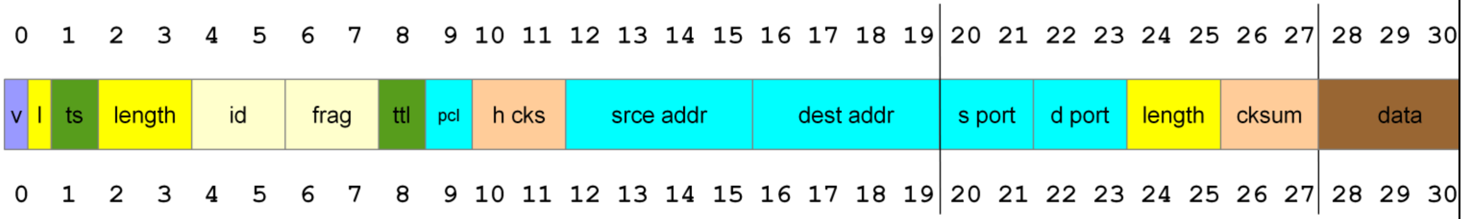


- IPv4 flow is identified by 104 bits in 5 different fields
  - makes the logic complex
  - requires a table search or content-addressable memory
- Better to have a “label” that can be used directly as table address
  - much smaller packet header, simpler processing
  - used in ATM, also MPLS (but that carries the other headers too)



IP and UDP headers

- in memory: numbers are byte offset from start of buffer
  - random access: can access fields in any order



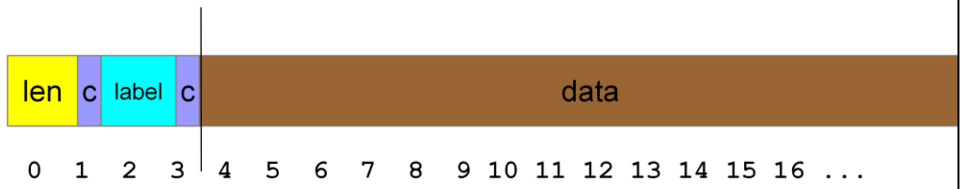
- on interface to PHY: numbers are clocks from start of datagram
  - sequential access: must buffer earlier fields if needed later

The byte numbers represent memory address offsets when the packet is stored in memory. In that case, the fields can be accessed in any order.

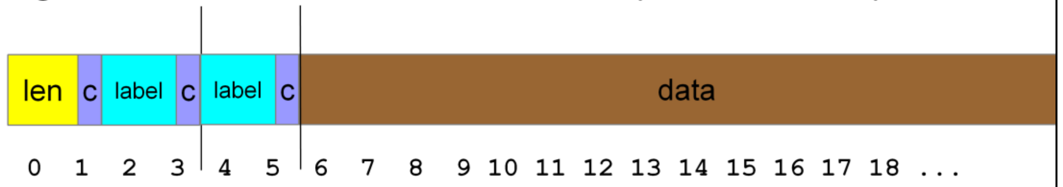
They represent the time of arrival if the packet is being processed as it arrives from the PHY chip (e.g. over the Ethernet Media-Independent Interface). Each byte must either be processed as it arrives or stored until it is needed. Similarly, on transmission if, say, the logic is to calculate the IP checksum it can’t transmit bytes 10 and 11 until it has seen bytes 12 to 19, so a delay must be inserted into the path. Similarly if it needs to calculate the IP datagram length from the UDP length.

Flexilink IT packets: typical header

“c” = 3-bit CRC



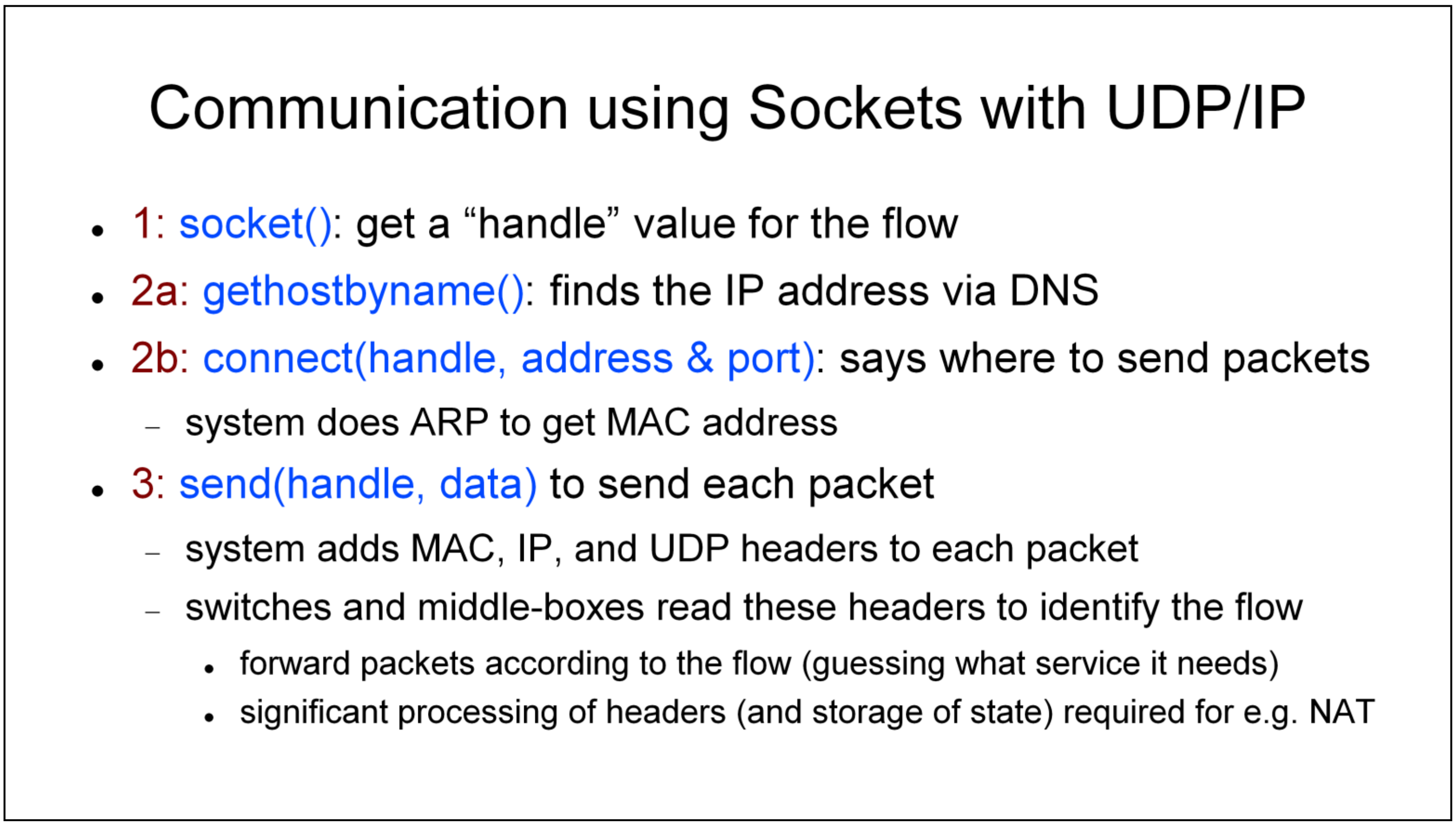
- Format is link-specific
  - more flows → bigger routing table → bigger “label” field
- Can do flow aggregation with an additional label (as in MPLS)



The format shown is used in the current implementation and allows for up to 8192 IT flows on each link. In a system where more flows are required per link, a different header format with a larger “label” field could be used; there are several ways in which use of this different format could be negotiated.

An alternative is to push another label on the front of the packet, adjusting the “length” value accordingly. Then the first label identifies a “bundle” of flows that are all routed together, and the second identifies an individual flow within the bundle. When the “bundle” flow is set up it is identified as a bundle, and at the end of its path the label is popped and the packet routed according to the next label. This is similar to MPLS “push” and “pop”, except that there is always at least one label. It is also similar to ATM VPI and VCI fields, except that there may be more than two labels and they are not constrained to be the fixed size specified for an ATM cell header. Indeed, switches through which the bundle passes see the other labels as part of the payload (bytes 4 onwards in the diagram) so do not need to understand the format of the other labels.

The label value on each link is chosen by the destination. If, say, its routing table only supports 512 flows (with label values 0 to 511), it can simply avoid using labels 512 to 8191; it doesn’t need to tell the link partner that it doesn’t support them.



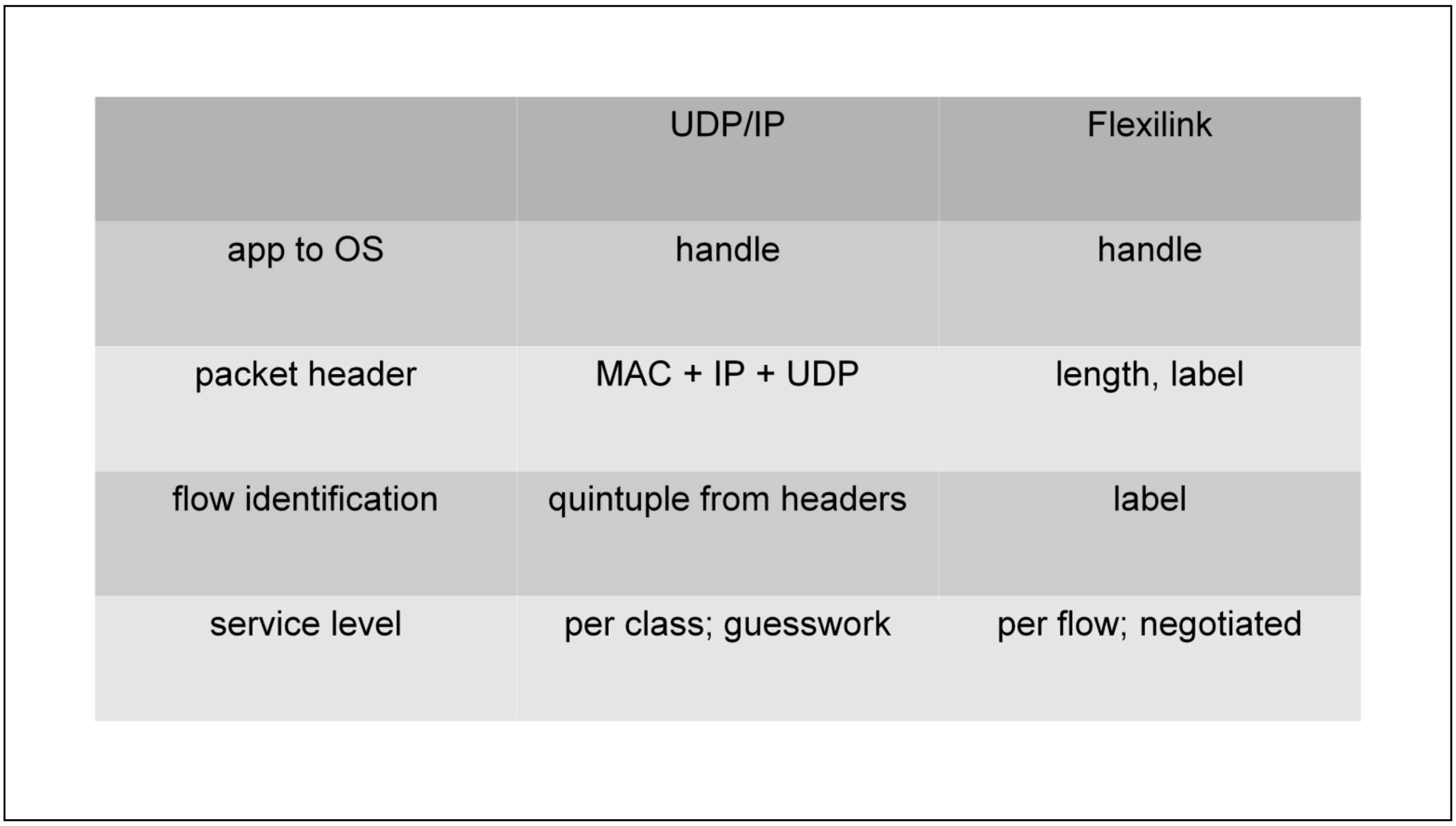
Communication using Sockets with UDP/IP

- 1: socket(): get a “handle” value for the flow
- 2a: gethostbyname(): finds the IP address via DNS
- 2b: connect(handle, address & port): says where to send packets
  - system does ARP to get MAC address
- 3: send(handle, data) to send each packet
  - system adds MAC, IP, and UDP headers to each packet
  - switches and middle-boxes read these headers to identify the flow
    - forward packets according to the flow (guessing what service it needs)
    - significant processing of headers (and storage of state) required for e.g. NAT

This demonstrates that when sending data the application uses the “handle” value to identify the remote application. The handle translates into multiple fields in the packet headers, and switches then have to use this information (typically a total of 104 bits, shown in cyan on slide 12) to identify the flow.

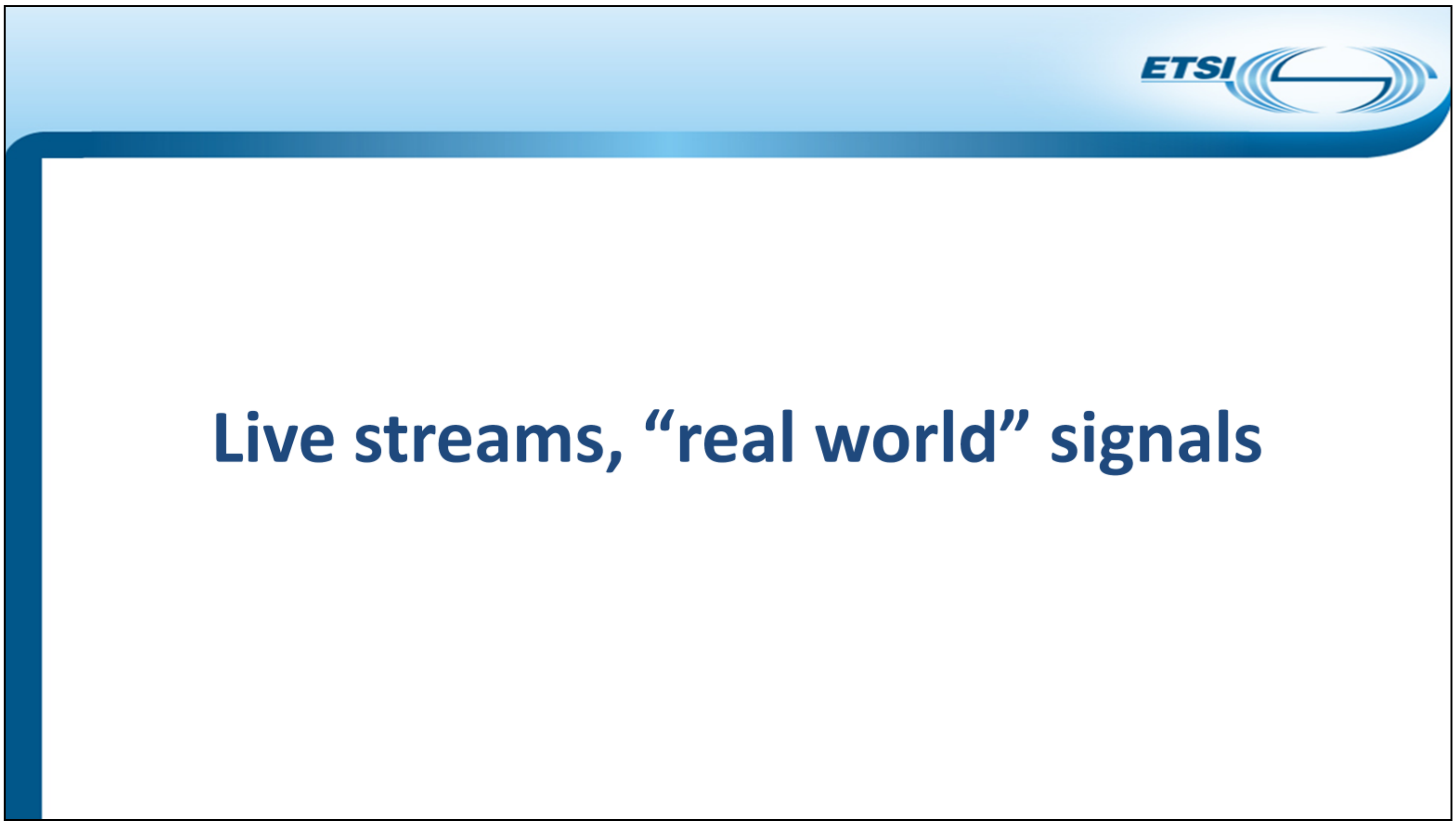
Communication using Flexilink IT service

- 1: generate a 128-bit flow id: this is the control plane handle
- 2: send FindRoute control plane request
  - reply includes data plane handle (same size as a routing table address)
- 3: send packets with data plane handle (label) in header
  - switches use the label directly to identify the flow
    - single-cycle lookup in table
    - forward packets according to the table entry for the flow
    - service required is signalled explicitly in FindRoute message
    - only processing required is replacing label with value from table



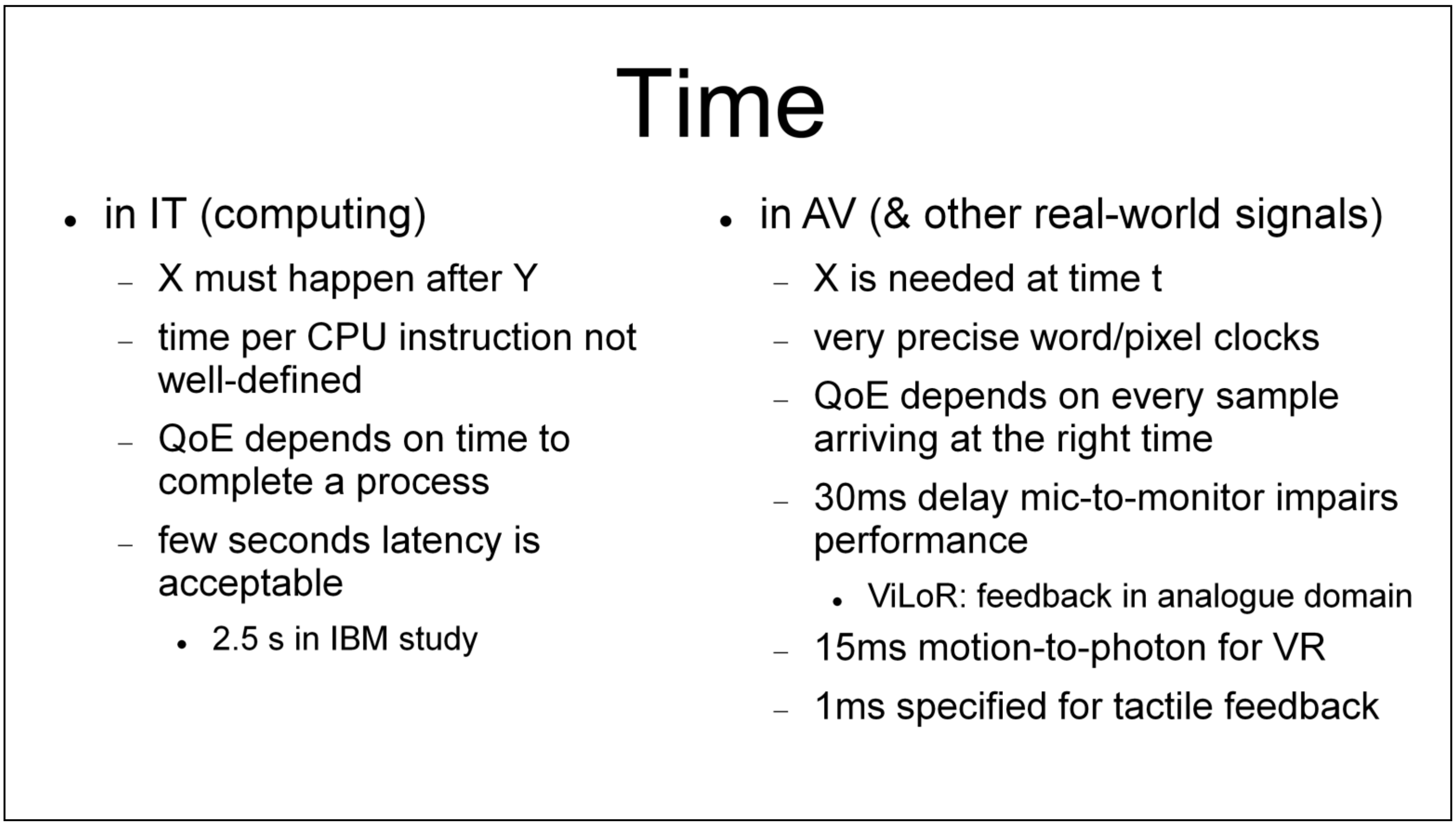
	UDP/IP	Flexilink
app to OS	handle	handle
packet header	MAC + IP + UDP	length, label
flow identification	quintuple from headers	label
service level	per class; guesswork	per flow; negotiated

Steam age vs 21st century?



Live streams, “real world” signals

18



Time

- in IT (computing)
  - X must happen after Y
  - time per CPU instruction not well-defined
  - QoE depends on time to complete a process
  - few seconds latency is acceptable
    - 2.5 s in IBM study
- in AV (& other real-world signals)
  - X is needed at time t
  - very precise word/pixel clocks
  - QoE depends on every sample arriving at the right time
  - 30ms delay mic-to-monitor impairs performance
    - ViLoR: feedback in analogue domain
  - 15ms motion-to-photon for VR
  - 1ms specified for tactile feedback

In the early 1970s, researchers at IBM found that where a task required several steps, each consisting of typing a command and waiting for the result, if the result took more than about 2.5 seconds to appear the user would lose their place in the sequence and have to think again what the next step should be. At the time, IBM's on-line system always took at least 10 seconds to produce a result.

In a meeting to discuss sending audio and video over packet networks, the delegate from a large cable operator said "media networks have requirements that IT folks don't understand". With audio, in particular, any irregularity in playing out the signal is very noticeable. The packets that carry audio are sent at very regular, predictable, intervals, and the network needs to be able to use that information to give them the service they require; this is not a facility that is provided by networks that are designed for IT functions such as copying files or downloading web pages.

A performer hearing their own voice through headphones will notice a delay of about 10 milliseconds, and find it difficult to work with a delay of a few tens milliseconds, probably because the brain interprets the delayed sound as someone trying to interrupt. So if the sound goes over the network to a mixing desk and then back over the network again to the performer, the delay should be less than 5 ms for each pass across the network. Contrast that with the 100 ms delay specified for QoS class 0 (voice over IP, high interaction) in ITU-T Rec Y.1541.

The BBC's Virtualisation of Local Radio (ViLoR) project, which connects local radio studios to central servers over an IP network, is careful not to feed people's voices back to them via the servers, instead mixing the microphone into the programme locally.

Packet networking as a best-effort service

- Competition for outgoing link in switches
  - can't predict offered traffic; requires queuing
  - longer queue → longer delay before forwarding
  - unbounded queue size + fixed memory size → packets can be dropped
- Good for unpredictable “IT traffic” (e.g. web surfing, file transfer)
  - using acknowledgement & retransmission as in TCP
- Not good for live continuous media where latency matters
  - including some of the new services proposed for 5G

Traditionally, packet networking has been seen as a “best effort” service where the offered traffic may exceed the capacity. When it does, queues form, just as on the roads.

Three levels of determinacy

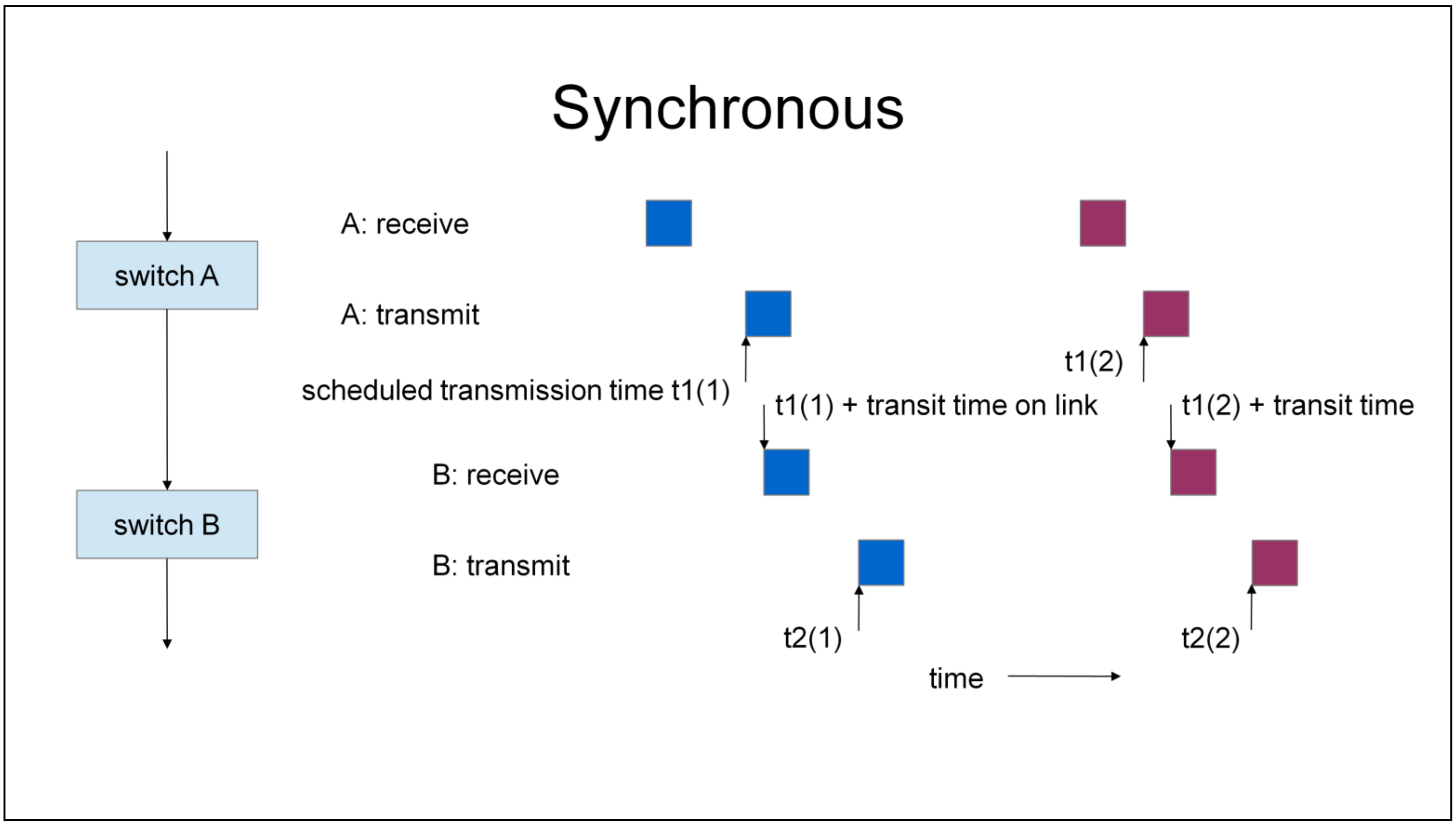
- also identified by IETF DetNet group

- Synchronous
  - scheduled transmission on each link
  - fixed latency, no dropped packets
- Asynchronous
  - reserved capacity on each link
  - bounded latency, no dropped packets
- Best-effort
  - no guarantees

The IETF group studying Deterministic Networking has identified two ways in which the kind of service that is required for live media can be provided. In each case, they say that the service can't be guaranteed unless resources are reserved along the path the packets are to follow before data transfer begins.

With the synchronous service, packets are transmitted according to a schedule, rather like a train timetable. (For those who are cynical about timekeeping on some countries' railways, I have used a picture of Swiss trains.) The end-to-end journey time is well-defined; the main difference is that the pattern repeats every millisecond (in the case of the current implementation of Flexilink) instead of every hour.

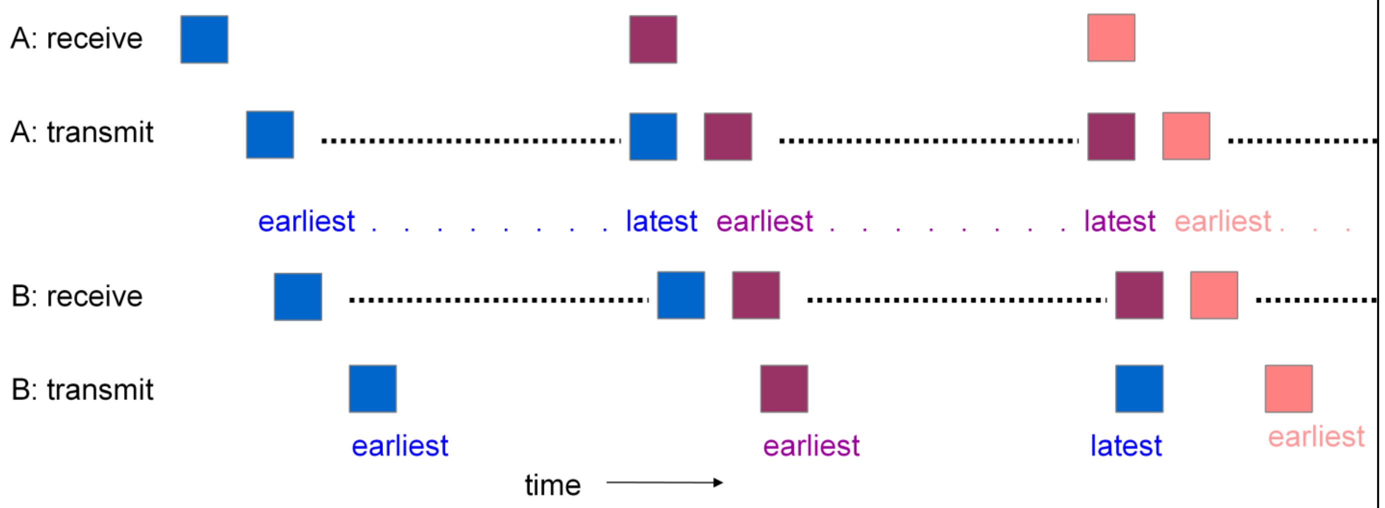
With the asynchronous service, there is simply a guarantee (for each link in the path) that the packet will be able to be transmitted within a specified time. This is like having a reserved lane on a highway and ensuring the amount of traffic entitled to use it does not exceed capacity, without setting up a specific schedule.



Synchronous

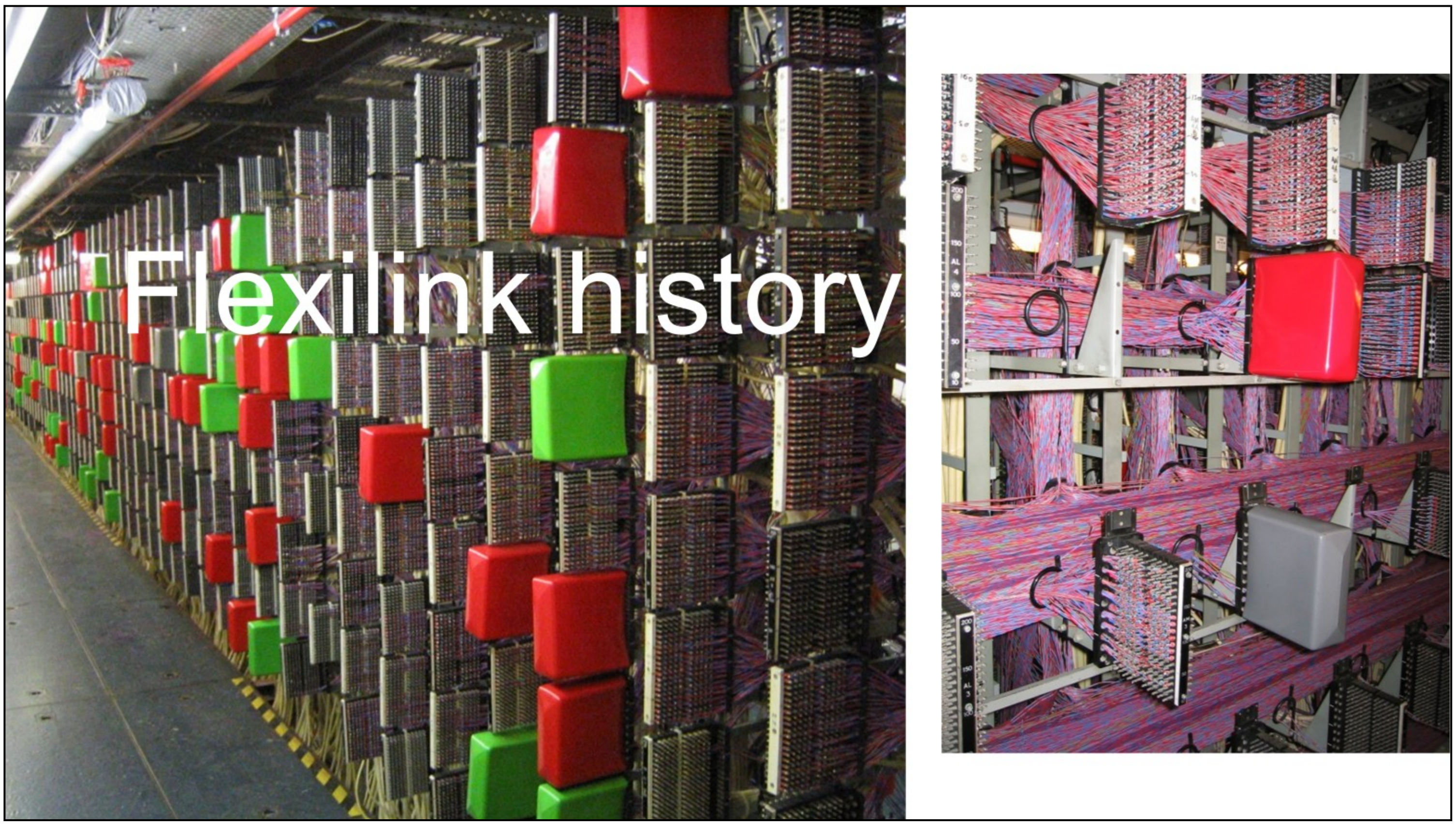
This is an illustration of a synchronous service; the blue rectangles represent one packet in the flow, and the purple ones are the next packet.

Asynchronous



This illustrates a typical asynchronous case, where the guarantee is that a packet will be able to be forwarded before the time for the next one in the flow to arrive. This was the definition of the ATM CBR service.

The blue packet will not overtake the purple one, but its time of arrival at the destination can vary by more than the interval between packets.



Flexilink history

These pictures are of the Main Distribution Frame in BBC Broadcasting House in London before the redevelopment in which it was replaced by an all-digital system (mainly MADI for the radio studios, and IP for other traffic).

The cables carried analogue audio, digital audio, and other signals. Note that they are all unshielded.

- Flexilink history
  - BBC project to implement Radio 4 broadcast chain over ATM
  - included developing RouteLive ATM switch
    - switched circuits and PNNI implemented
    - added features for broadcasters
    - two of the four ATM services implemented
      - UBR (unspecified bit rate, the best-effort service) for data services
        - e.g. device management, network management, Internet access
      - CBR (constant bit rate, the asynchronous service) for live audio streams
        - AES47 standard for carrying audio directly over ATM (no RTP/UDP/IP/AAL5 layers)
      - VBR (variable bit rate) would not have any advantage over CBR
        - still needs a reservation for the peak cell rate
      - ABR (available bit rate) would add significant complexity for little benefit
    - option to use Ethernet physical layer (AES51 standard)
      - use commodity Ethernet PHYs to carry byte stream; less delay in FIFOs than ATM PHYs

The ATM switch they had planned to use went out of production, so Nine Tiles was commissioned to produce an affordable switch targeted at live audio.

One added feature was for the LEDs on a network connection to indicate whether it is carrying an on-air signal.

Audio Engineering Society standard AES47 was also published as IEC 62365. It specifies how audio samples are packed directly into ATM cells, including appropriate data protection and timing information, and a CBR flow set up. This contrasts with the traditional method using RTP, which has many more layers and does not provide throughput or latency guarantees.

Other aspects of the system were standardised as IEC 62379, originally for audio over ATM but extended to cover video as well as audio, and other packet networking technologies as well as ATM.

ATM PHY chips all have cell FIFOs, and won’t begin to transmit a cell until the whole cell is in the FIFO. This introduces a longer delay than the small FIFOs used in Ethernet PHYs to cross between clock domains.

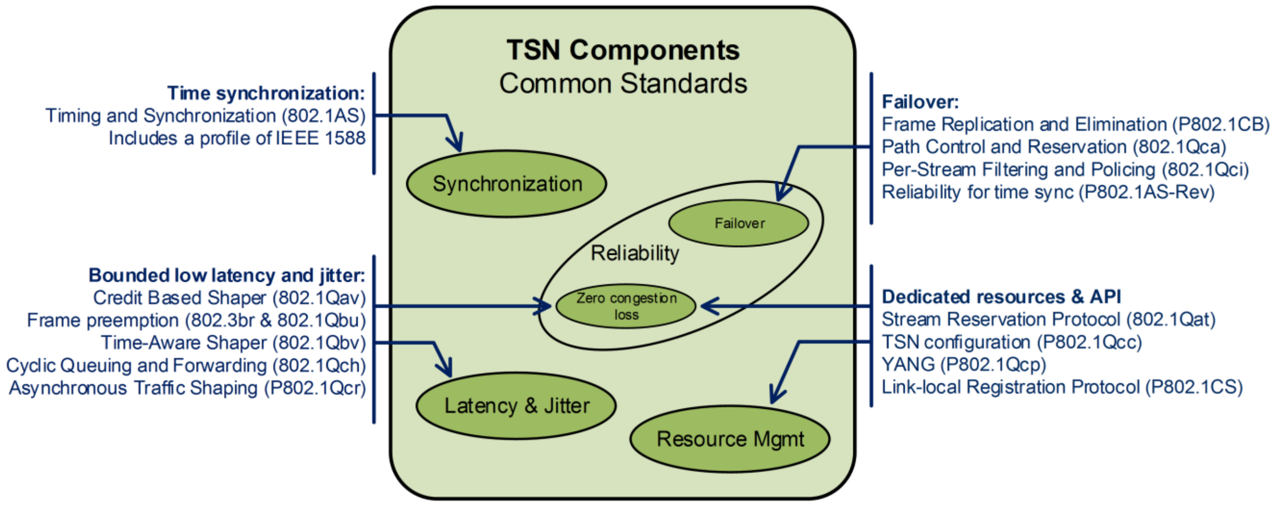
- Flexilink history (2)
  - CBR service was implemented by schedules
    - 4-cell circular buffer for each flow's incoming cells
    - single queue per output for UBR cells
    - each outgoing cell slot allocated to a specific CBR flow (or none)
    - send a UBR cell if no CBR cell available
    - CBR flow can be copied to ("taken" by) more than one output
  - easier than shaping, policing, and multiple queues
    - see next slide
  - links not synchronised
    - latency bounded but not fixed
      - drifts as phase drifts
      - not affected by other traffic

For the "traditional" ATM CBR service, higher-bandwidth flows need higher priority. For example suppose the link can carry 40 cells in a millisecond and there is one flow with 10 cells/ms and five flows with 1 cell/ms. The 10 cell/ms flow needs to be able to transmit within 4 cell times, so needs a higher priority in case cells arrive on all five of the other flows at the same time. Add more flows at different rates and it becomes apparent that multiple queues are needed.

More complexity is added by the requirement for "traffic shaping" to ensure cells are forwarded at roughly even intervals, and "policing" to make sure a source does not send more cells than it has reserved capacity for.

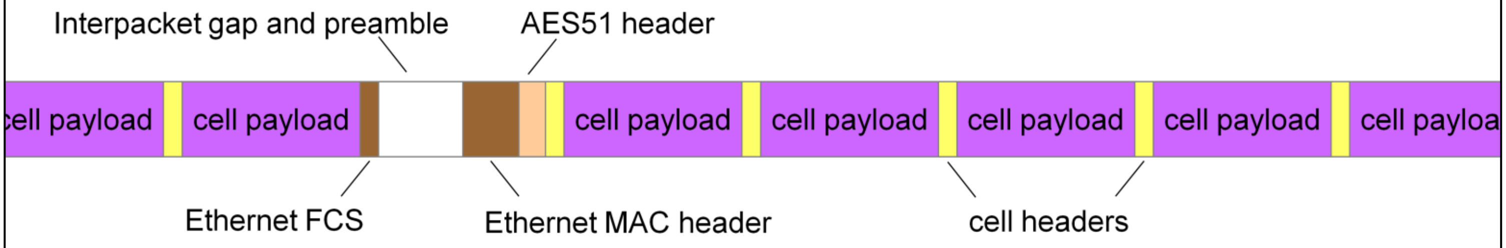
The CBR service was implemented as a schedule; this avoided all the above issues. However, the links were not synchronised so the latency was still variable. The picture shows the same scene as the "synchronous" case in the earlier slide, but it is as if the two railway lines use different clocks: there is an additional (and indeterminate, though never longer than the interval between trains) wait for the train for the onward journey.

Retrofitting timeliness to IT standards


TSN Components
Common Standards
Time synchronization:
Timing and Synchronization (802.1AS)
Includes a profile of IEEE 1588
Synchronization
Failover:
Frame Replication and Elimination (P802.1CB)
Path Control and Reservation (802.1Qca)
Per-Stream Filtering and Policing (802.1Qci)
Reliability for time sync (P802.1AS-Rev)
Failover
Reliability
Bounded low latency and jitter:
Credit Based Shaper (802.1Qav)
Frame preemption (802.3br & 802.1Qbu)
Time-Aware Shaper (802.1Qbv)
Cyclic Queuing and Forwarding (802.1Qch)
Asynchronous Traffic Shaping (P802.1Qcr)
Zero congestion loss
Dedicated resources & API
Stream Reservation Protocol (802.1Qat)
TSN configuration (P802.1Qcc)
YANG (P802.1Qcp)
Link-local Registration Protocol (P802.1CS)
Latency & Jitter
Resource Mgmt

This diagram (figure 1 from the current Work Item 8 draft) shows the complexity of adding determinism to packet network standards that were not designed for it.

- Flexilink history (3)
  - AES51 standard for ATM cells in Ethernet packets
  - negotiate to send a continuous stream of Ethernet frames
    - each containing a fixed number of cell slots
    - VPI reduced from 12 bits to 4 so cell header is 4 bytes
    - repeating pattern of slot allocations to CBR flows
    - each slot contains a whole CBR cell or a whole UBR cell or an idle cell



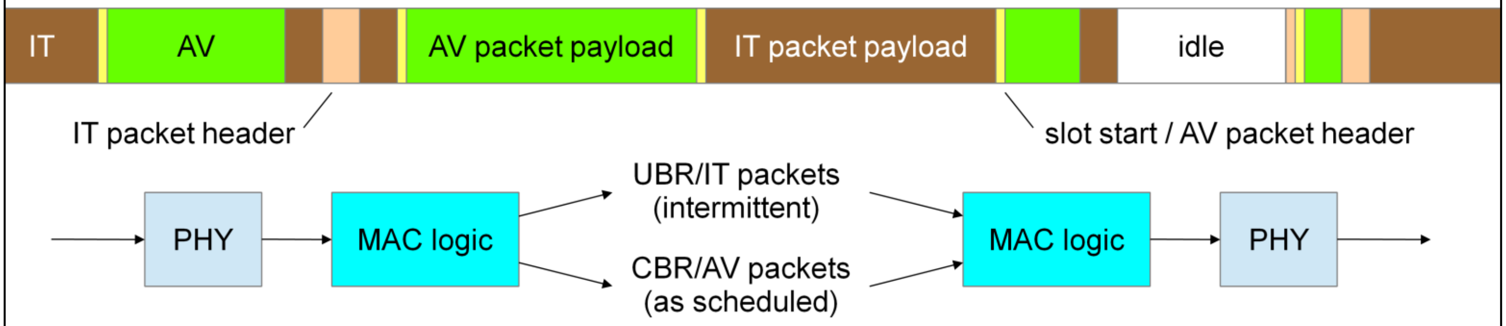
AES51 specifies how to send ATM cells over Ethernet links. (Why did the ATM Forum never produce such a standard? Good question. There was an initiative called “cells in frames” at Cornell University but it was too late and didn’t have industry support.)

The diagram shows some detail of the AES51 format. The Ethertype value is 0x88DD indicating an AES standard and the first six bytes of MAC client data are the AES51 header: the first byte is 02 which identifies it as AES51, the next byte is the AES51 packet type, which is 00 to indicate a packet containing ATM cells, and the other four contain timing information. The remainder of the MAC client data consists of ATM cells.

The Frame Check Sequence is only used as a check on the integrity of the link; cells are routed as they arrive, without waiting to check the FCS. This reduces latency; also note that cell headers include a CRC, as do AAL5 PDUs and audio samples in AES47.

When the link comes up, negotiation packets (which have packet types 0x80 to 0x82) are exchanged, after which the link partners switch to sending packets containing cells.

- Flexilink history (4)
  - evolved to avoid fixed cell size
    - can use a more sophisticated format with today's electronics
    - position CBR / AV packets where required ("foreground layer")
    - UBR / IT packets use remaining bytes ("background layer")
      - IT packets can be longer than a slot


IT
AV
AV packet payload
IT packet payload
idle
IT packet header
slot start / AV packet header
PHY
MAC logic
UBR/IT packets (intermittent)
CBR/AV packets (as scheduled)
MAC logic
PHY

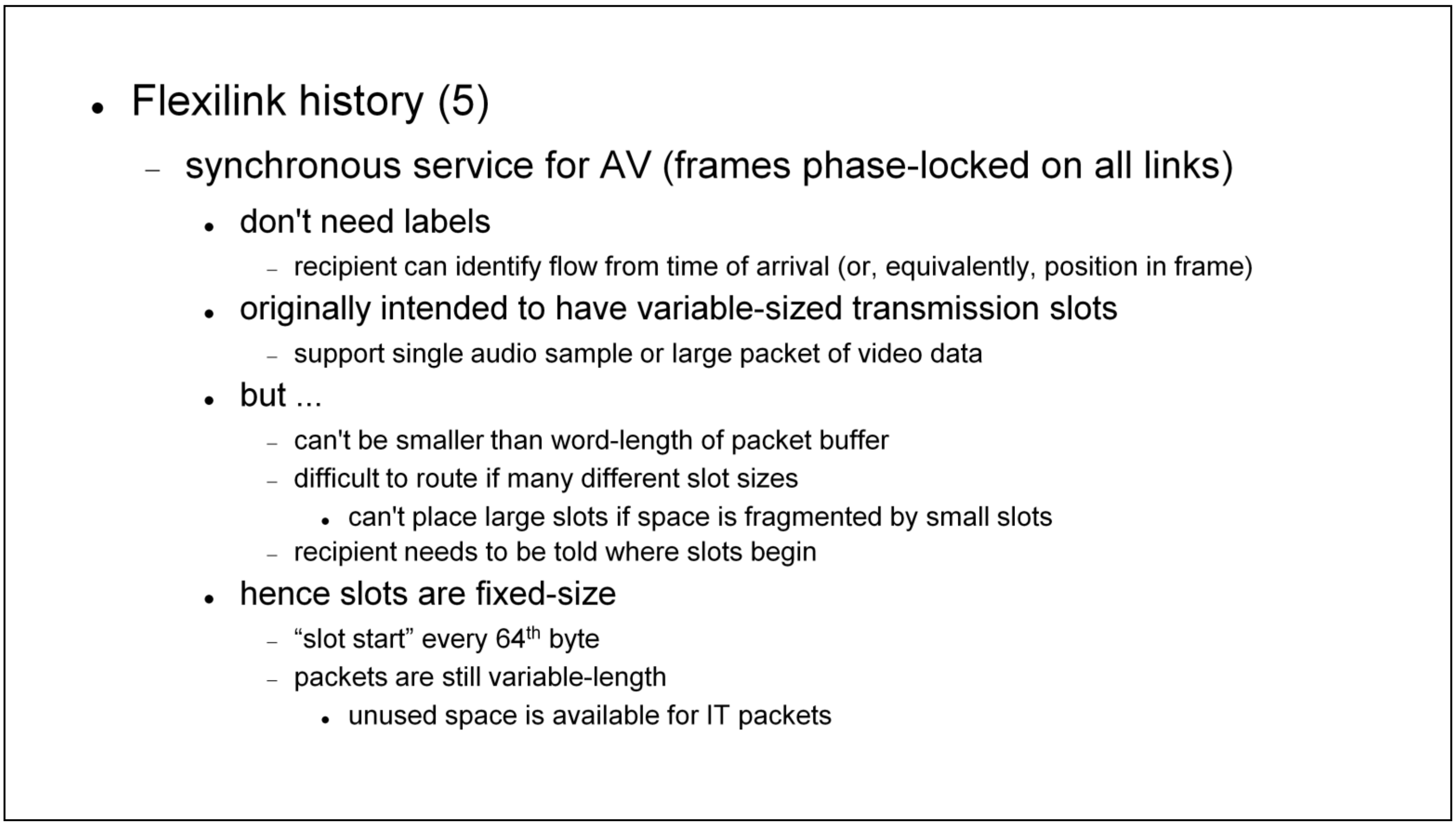
Flexilink is an evolution from AES51 which does not require the fixed cell size that ATM uses. When ATM was defined in 1989 the fixed cell size made many aspects of the forwarding process easier to implement, but now we can do more in logic so Flexilink uses a small increment in the complexity of the format on the cable to bring big benefits in usability.

In Flexilink, the services are referred to as AV (audiovisual) and IT rather than CBR and UBR.

The diagram shows some detail of the format used in the current implementation. Every 64th byte is a "slot start" (shown in yellow, format in a later slide) which shows the length of the AV packet (shown in green) in each slot. The second complete slot shown contains a 63-byte packet, the third doesn't contain a packet, the others contain packets shorter than 63 bytes.

The rest of the bytes that are transmitted carry IT packets, shown with the header (4 bytes, format in an earlier slide) in orange and the payload in brown. "Idle" bytes (which can't be the first byte of an IT packet) are sent when there is no IT packet to be transmitted; they are shown in white. In the diagram, another packet arrives in time to send one byte before a slot start; the second byte is sent after the AV packet.

On the receiving side, the MAC logic sends the whole stream to the AV routing buffer, and the IT packets and idle bytes to the IT routing process. On the transmitting side it takes AV packets from the AV routing buffer according to the schedule, and takes bytes from the IT routing process to fill the remaining space. More detail in later slides.

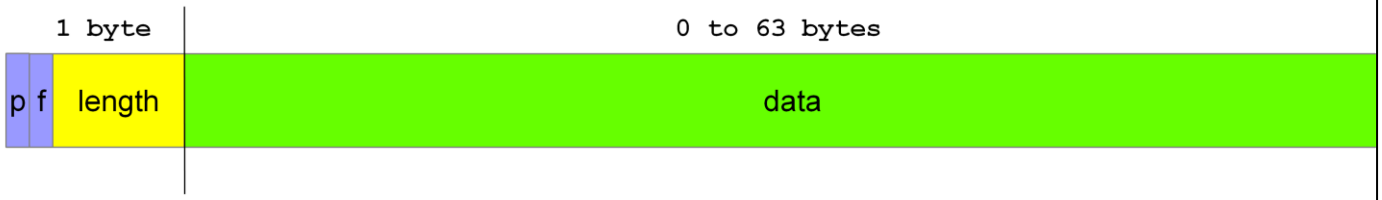


- Flexilink history (5)
  - synchronous service for AV (frames phase-locked on all links)
    - don't need labels
      - recipient can identify flow from time of arrival (or, equivalently, position in frame)
    - originally intended to have variable-sized transmission slots
      - support single audio sample or large packet of video data
    - but ...
      - can't be smaller than word-length of packet buffer
      - difficult to route if many different slot sizes
        - can't place large slots if space is fragmented by small slots
      - recipient needs to be told where slots begin
    - hence slots are fixed-size
      - "slot start" every 64th byte
      - packets are still variable-length
        - unused space is available for IT packets

A very simple process to phase-lock the frame structure on all links has been found to be effective. This allows AV packets to be routed without examining their contents at all.

It is interesting that slots with a fixed size, about the same size as an ATM cell, have been found to be the best way to carry the time-critical traffic. Clearly that was something ATM got right.

Flexilink AV packets



p = parity; f = flag which can be used to show end-of-message

- Format is global
  - no label because flow is identified by time of arrival
  - no change to packet header when forwarded
    - may be useful for forwarding in the optical domain

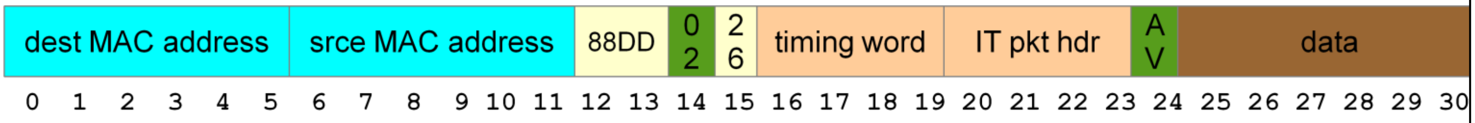
The flag f is intended to be used as a “not end of packet” mark when packets are fragmented to fit in 63-byte slots, in a similar way to AAL5 in ATM. The only assumption the network makes is that it should be set to 1 (and the length to zero) in an unused slot.

Unlike IT packets, there is no processing of the header when a packet is forwarded; the format should therefore ideally be the same throughout the network.

Routing of AV packets in all-optical networks is expected to be made easier by avoiding the need for any part of the packet to be read or modified.

Flexilink frame formats

- AV packets over Ethernet MAC (“virtual” link)



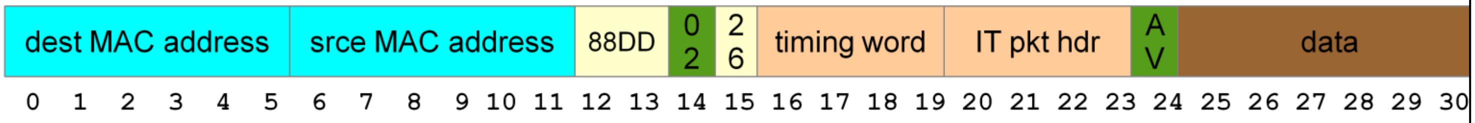
  – encapsulated in IT flow
    - asynchronous service: can’t route on time of arrival
    - IT packet can carry several AV packets, possibly from different AV flows
    - written to de-jitter buffers, from which copied to AV routing buffer
  – same format used for other IT flows
    - IT packet payload begins at byte 24
- Similar format over UDP

This shows the encapsulation in Ethernet; 0x88DD is the Ethertype value for AES standards, and 02 in the following byte identifies it as AES51. The next byte is the AES51 packet type, and 0x26 identifies it as a Flexilink IT packet.

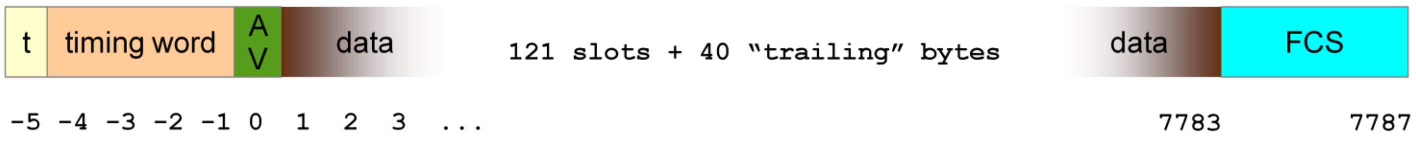
The IT packet simply contains one of more AV packets; there is a de-jitter buffer for each AV flow, and the buffer to which each AV packet shall be written is identified when the IT flow is set up. The process that copies the AV packets out of the de-jitter buffers has the same status within the switch as a physical link; to avoid adding complexity to Flexilink switches it could be implemented (along with the de-jitter buffers) as a middlebox.

Flexilink frame formats

- AV packets over Ethernet MAC (“virtual” link)

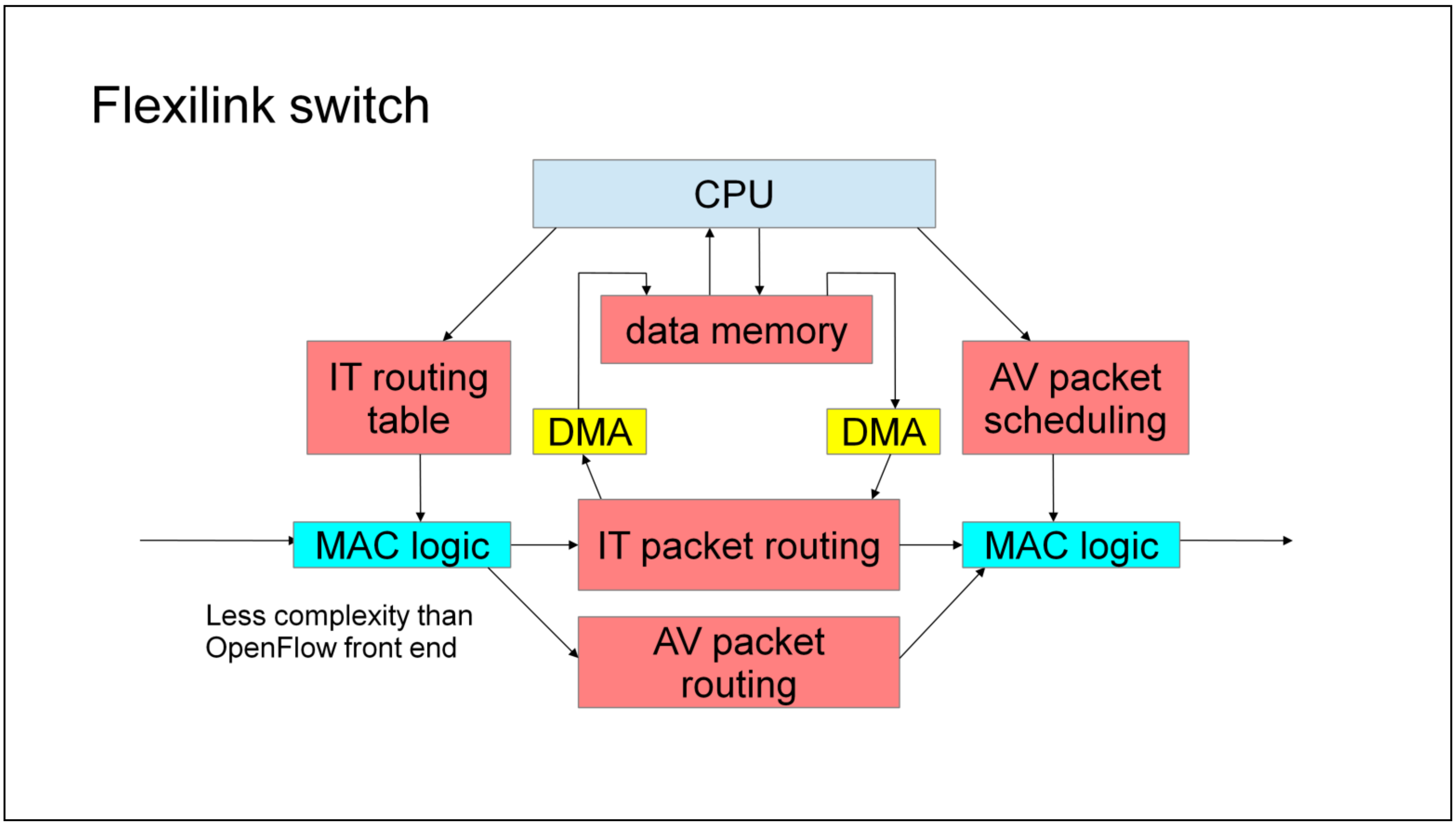


- Frame format over gigabit Ethernet PHY (“physical” link)



  - slots divided into 4 “subframes”
    - rolling 1 subframe of data in forwarding buffer at any time
      - FCS is only used as a check on the link integrity (as in SDH)

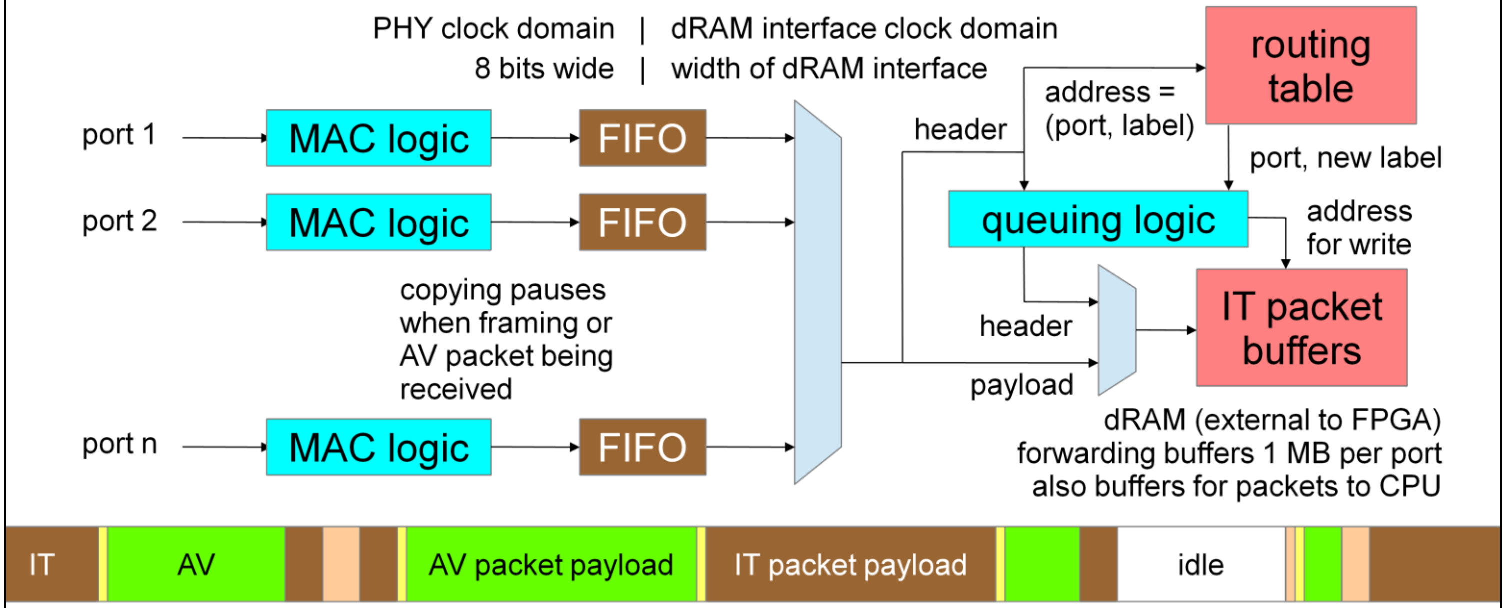
The second diagram shows the frame format used on links that implement the synchronised Flexilink service. There is an inter-frame gap of 12 to 16 bytes (12 bytes is the minimum specified for Ethernet), 2 bytes preamble, and 1 byte SFD, for a nominal frame size of 7810 bytes. That equates to 16000 frames per second plus a small margin to allow for tolerance in clock frequencies.



Flexilink switch

The following slides show the detail of the two kinds of packet routing.

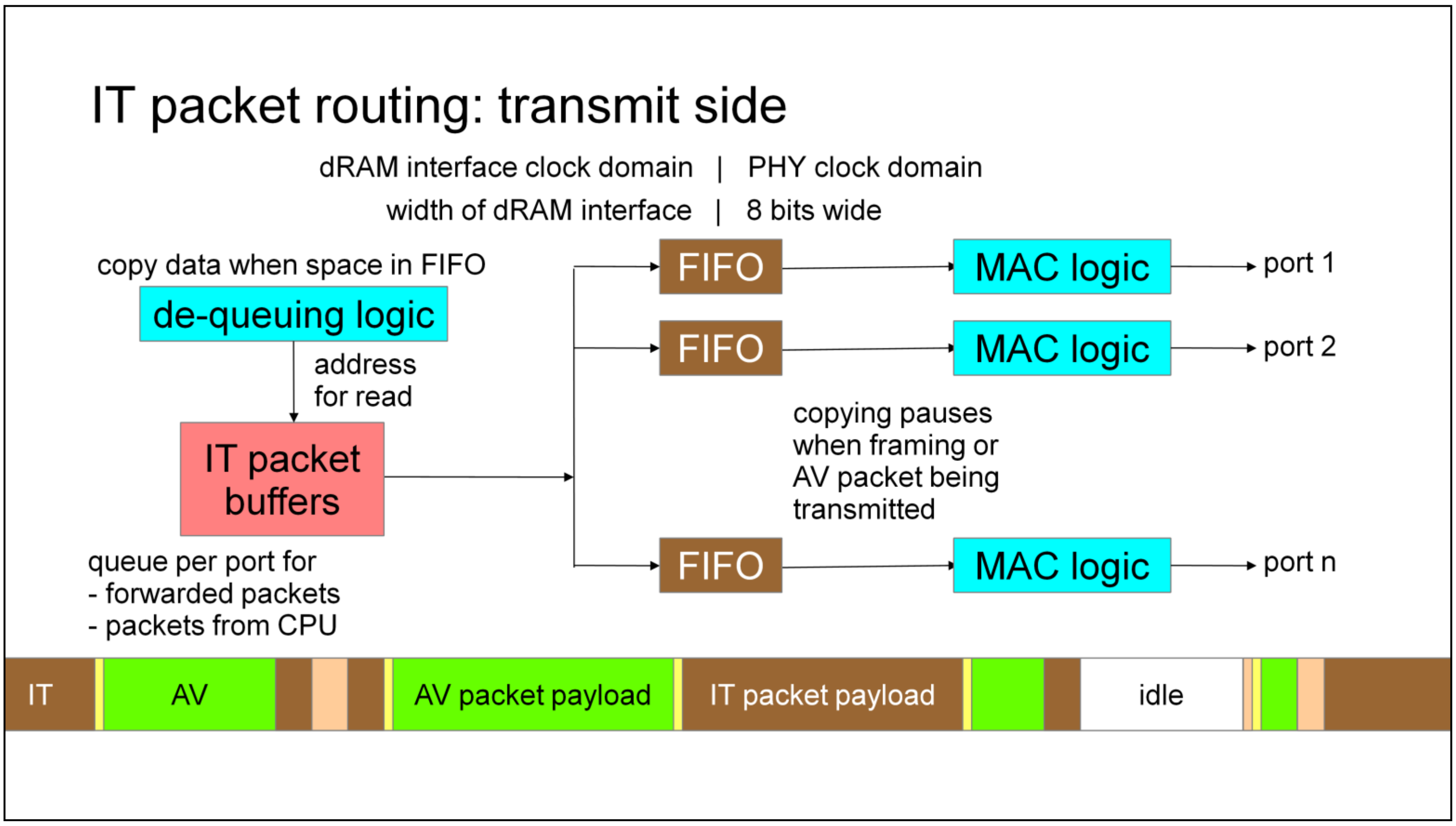
IT packet routing: receive side


PHY clock domain | dRAM interface clock domain
8 bits wide | width of dRAM interface
port 1
port 2
port n
MAC logic
FIFO
copying pauses when framing or AV packet being received
header
address = (port, label)
routing table
port, new label
queuing logic
address for write
header
payload
IT packet buffers
dRAM (external to FPGA) forwarding buffers 1 MB per port also buffers for packets to CPU
IT
AV
AV packet payload
IT packet payload
idle

This slide and the next three show detail of the routing process.

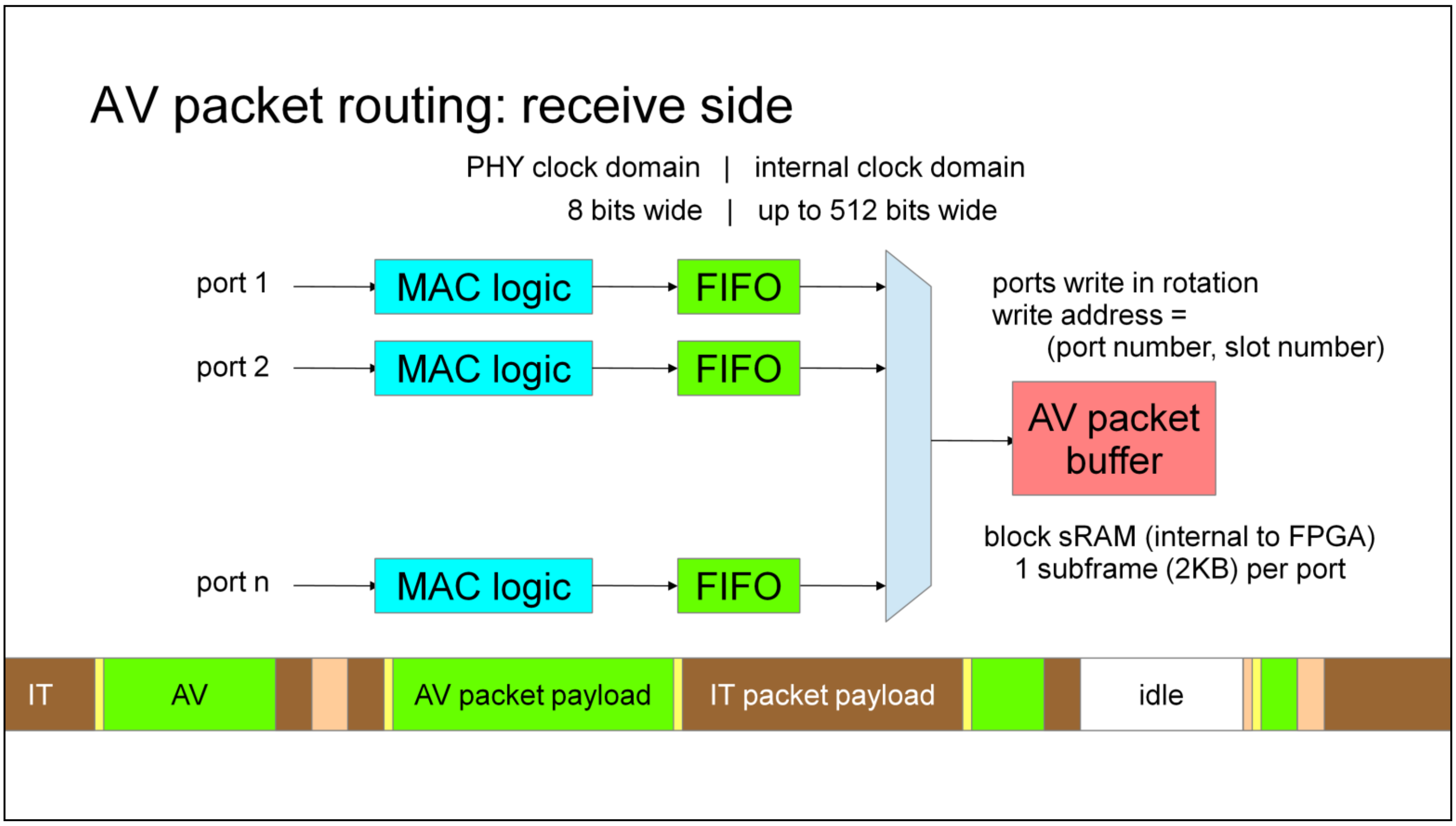
IT packets are written to the FIFO as they arrive; when there is a complete packet in the FIFO it is processed according to the entry in the routing table corresponding to its label. Normally this processing consists of replacing the label and adding it to one of the queues.

Note that the rate at which a packet is written to the FIFO depends on how much of the link is taken up with AV traffic. The FIFOs are serviced in rotation; the dRAM interface is wider than the MAC logic and its clock is faster.



IT packet routing: transmit side

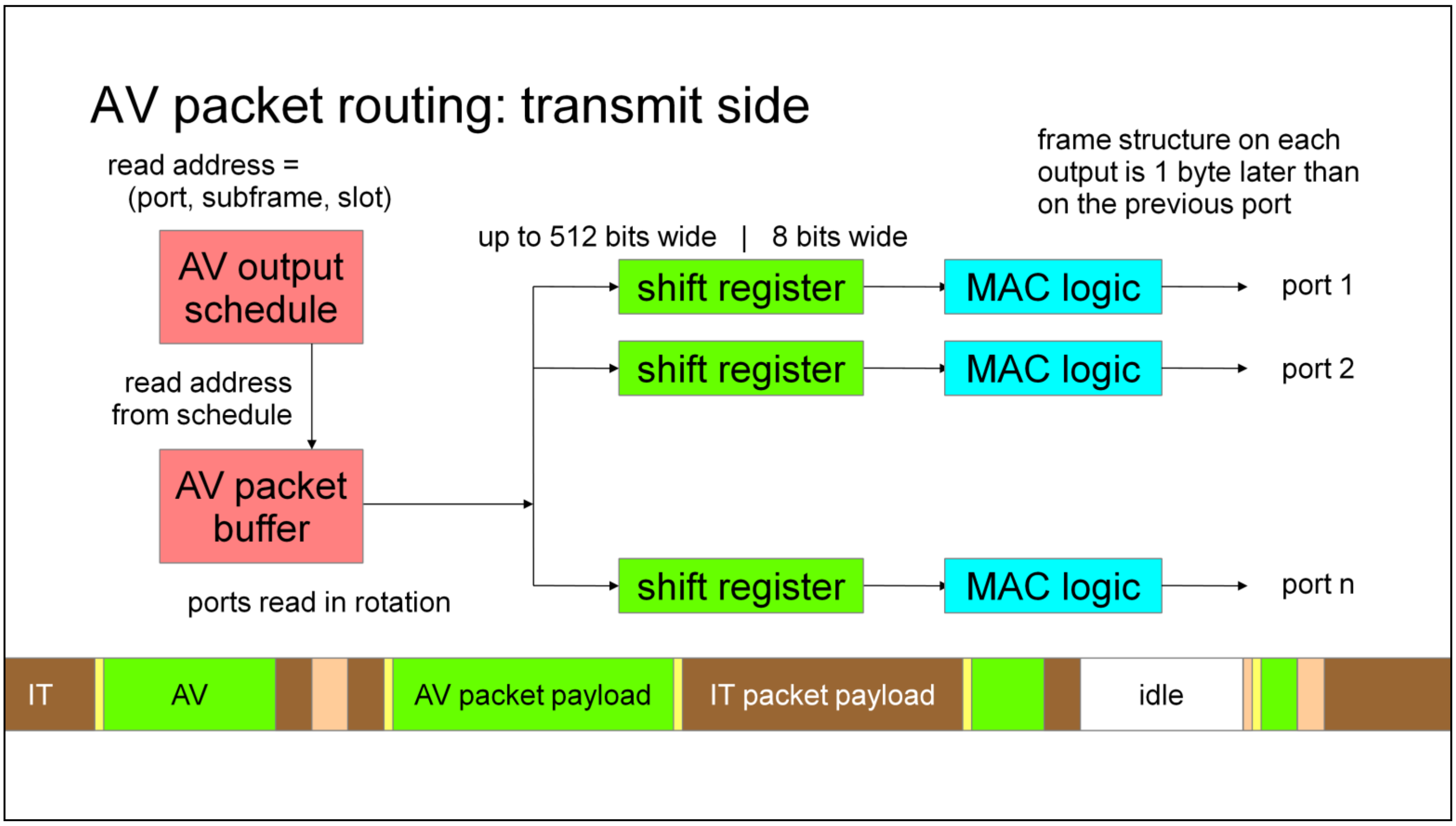
The process that copies packets into the FIFOs services each FIFO in turn. A byte is copied out of the FIFO for each byte on the cable that is not part of the framing or of an AV packet. If the FIFO is empty, and idle byte is written.



AV packet routing: receive side

PHY clock domain | internal clock domain
8 bits wide | up to 512 bits wide
port 1
port 2
port n
MAC logic
FIFO
ports write in rotation
write address =
(port number, slot number)
AV packet buffer
block sRAM (internal to FPGA)
1 subframe (2KB) per port
IT
AV
AV packet payload
IT packet payload
idle

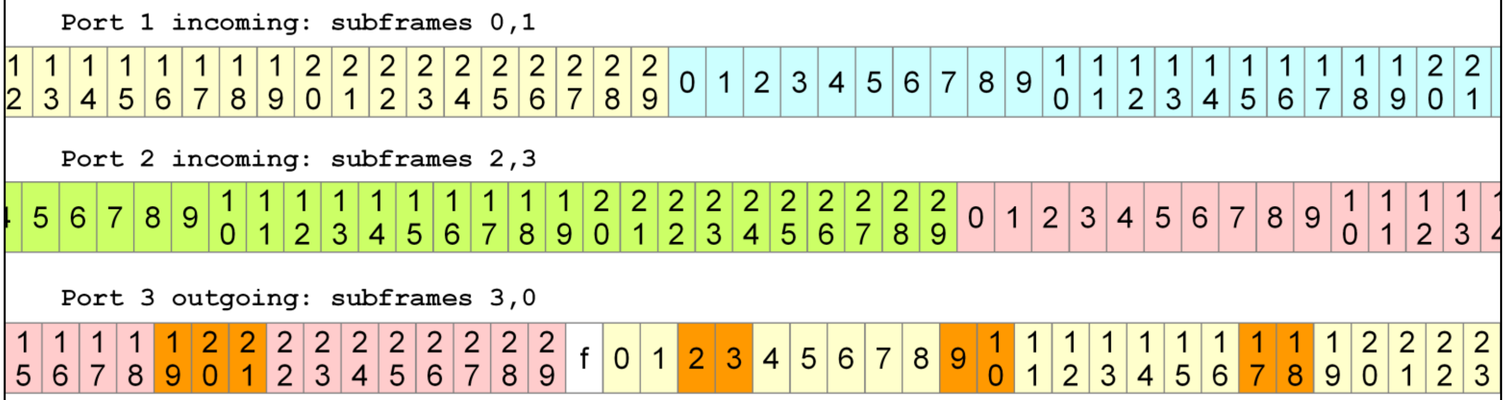
These FIFOs are quite small; they need to hold just over twice the width of the packet buffer (e.g. in the current implementation the packet buffer is 8 bytes wide so they need to be more than 16 bytes).



AV packet routing: transmit side

The timing of the outputs is arranged such that the access to the buffer can be choreographed appropriately.

AV packet forwarding

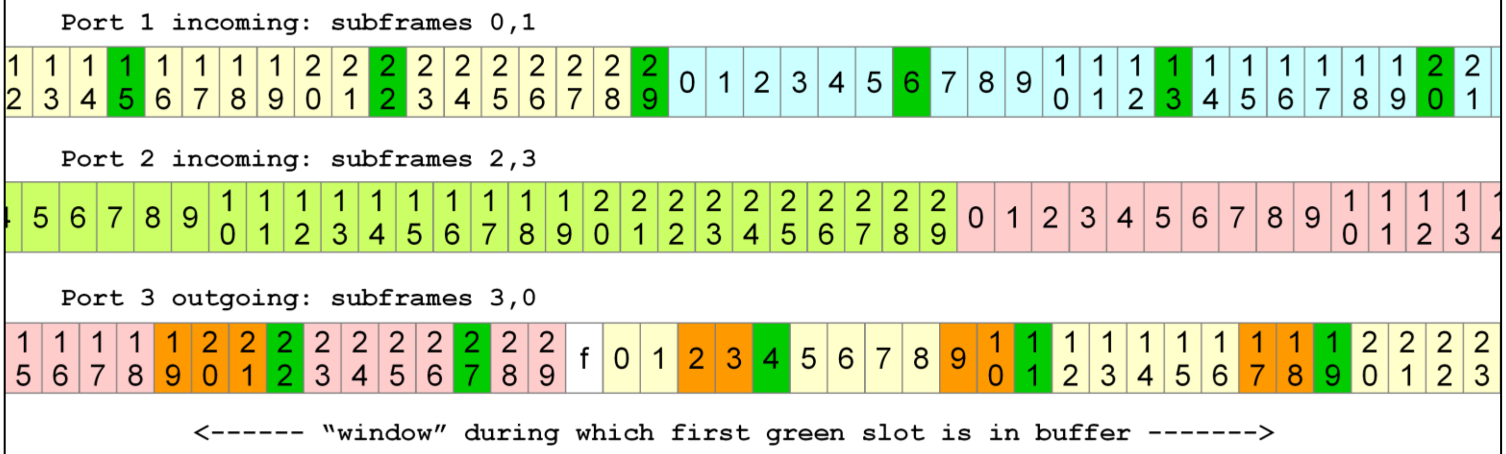


This slide and the next four show how the AV packet routing is done. It shows the incoming signals on ports 1 and 2 and the outgoing signal on port 3.

The exact time a frame begins in the outgoing signals is taken from one of the incoming streams (except in the unit that acts as master, where it is free running). If, say, it is taken from port 2, then the frame starts when a particular byte in slot 20 of subframe 2 is received on that port. This will have been the point at which the output frames were starting when port 2 was selected as the “upstream” port, from which the frame timing is taken.

The orange slots have been allocated to traffic from ports other than 1 and 2.

AV packet forwarding

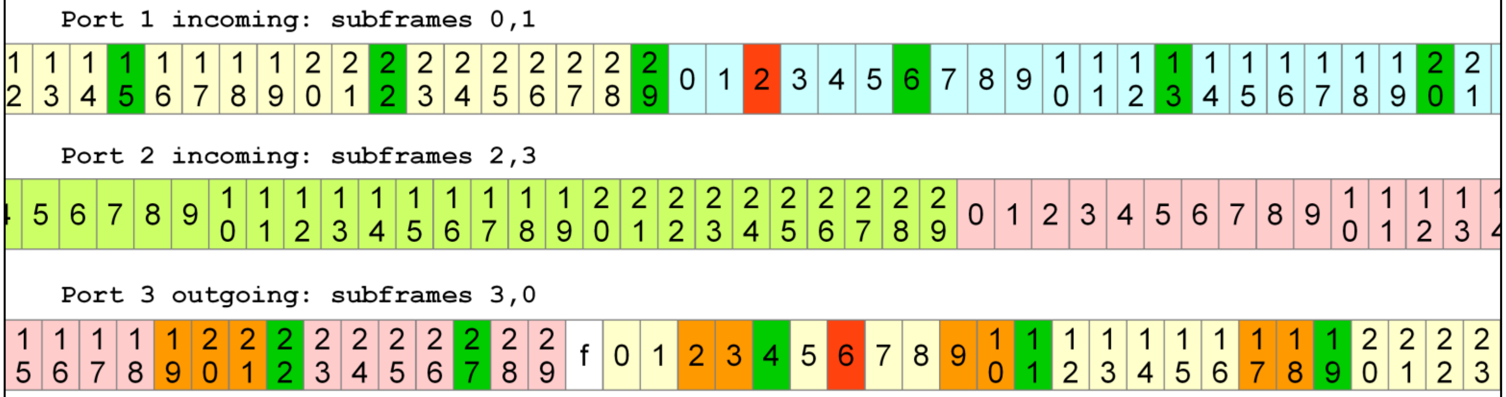

Port 1 incoming: subframes 0,1
Port 2 incoming: subframes 2,3
Port 3 outgoing: subframes 3,0
<------ "window" during which first green slot is in buffer ------->
f

A flow that occupies the green slots on port 1 has been routed to port 3. The packet in slot 15 of subframe 0 will be in the forwarding buffer until it is overwritten by the packet in slot 15 of subframe 1; during that time it is available to be forwarded.

The control plane protocols include a measure of how much uncertainty there is in the phase relationship between the incoming stream on port 1 and the outgoing streams (which we've suggested might be aligned with port 2); the window is made smaller if that uncertainty is greater.

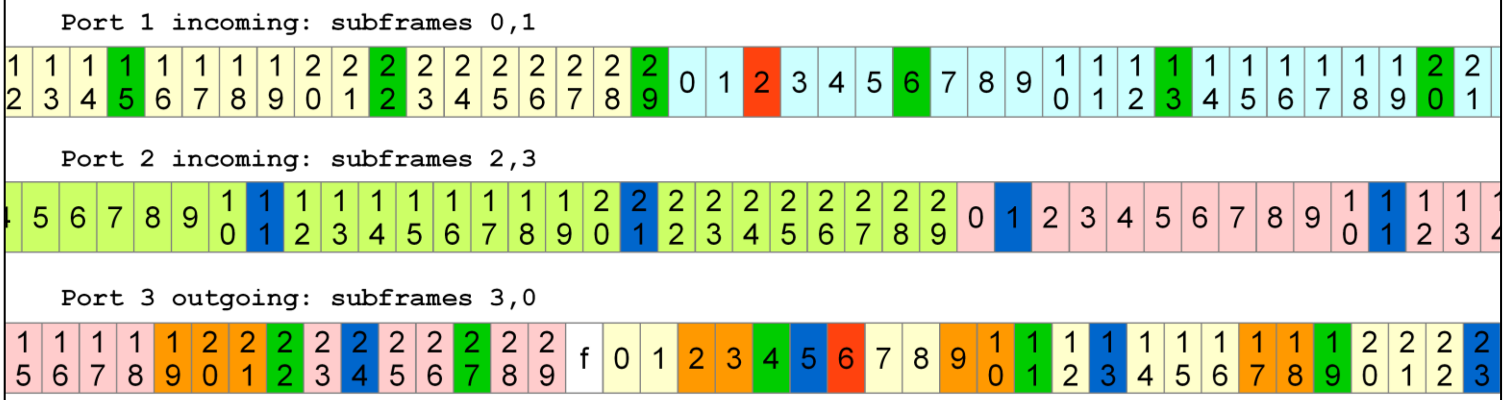
In the current implementation, each slot is simply chosen as the earliest available slot in its window. A more sophisticated algorithm could be used, for instance to distribute occupied slots more evenly and thus make it less likely that, when routing a future flow, all slots in the window would be occupied.

AV packet forwarding


Port 1 incoming: subframes 0,1
Port 2 incoming: subframes 2,3
Port 3 outgoing: subframes 3,0

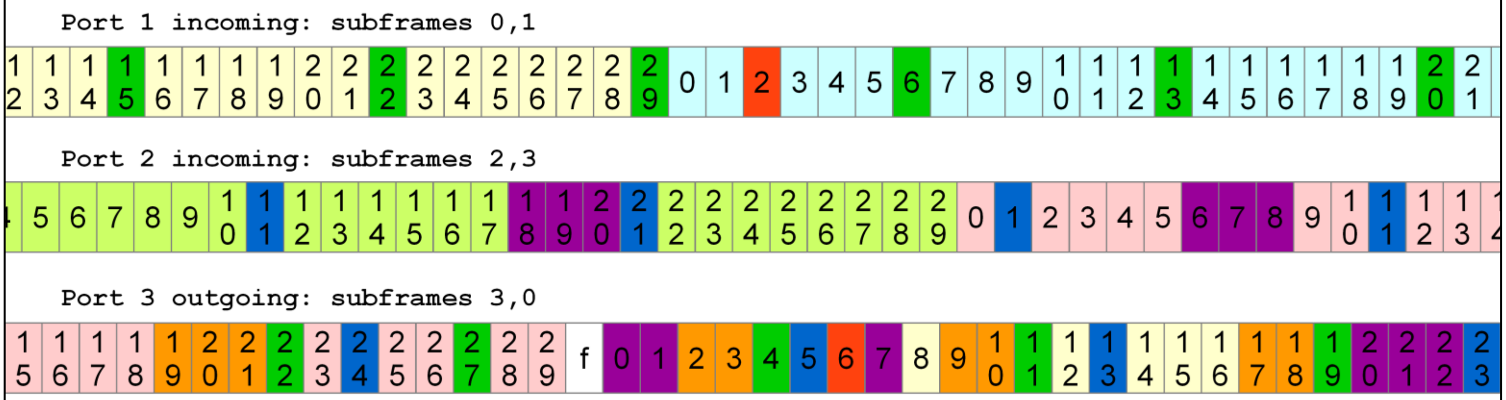
The red slot is another flow which has now been routed.

AV packet forwarding


Port 1 incoming: subframes 0,1
Port 2 incoming: subframes 2,3
Port 3 outgoing: subframes 3,0

Now the flow with the blue slots has been routed.

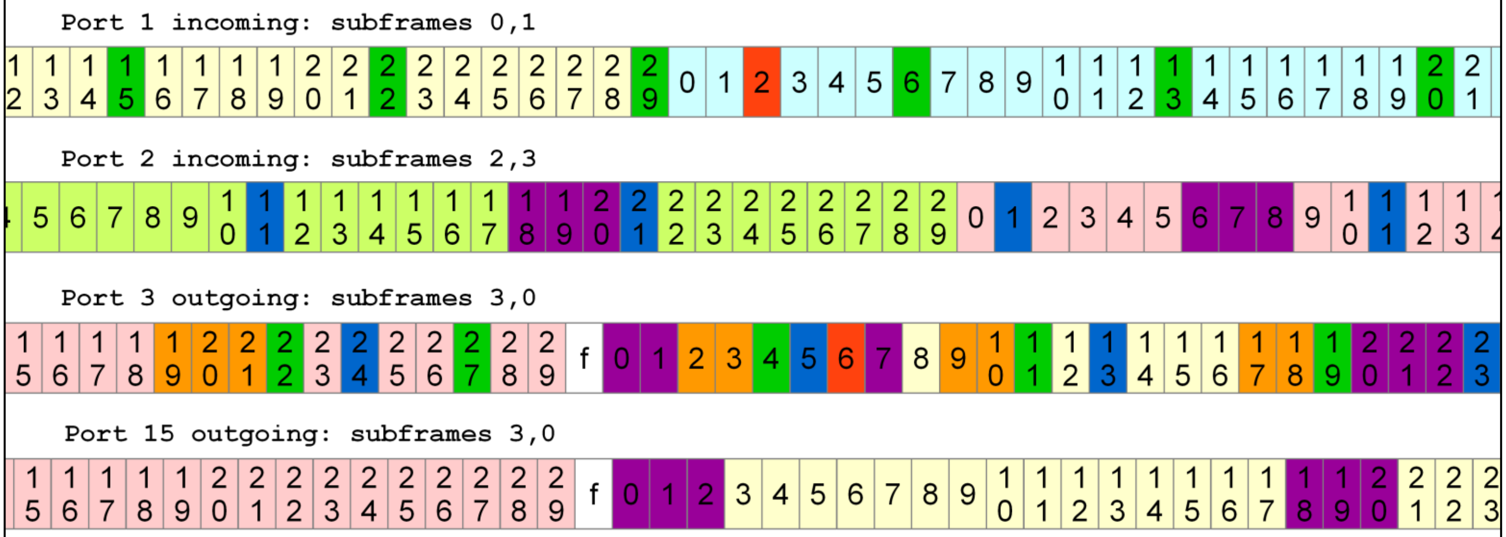
AV packet forwarding


Port 1 incoming: subframes 0,1
Port 2 incoming: subframes 2,3
Port 3 outgoing: subframes 3,0

Here the purple flow has been routed. Note that the slots are bunched together; maybe this is routing a flow that carries MPEG2 Transport Stream packets (which take up 3 slots each). One of the bunches has become separated in the port 3 stream; this illustrates the reason for using small slots rather than requiring larger packets to occupy a contiguous piece of space.

A packet arriving in slot 21 of subframe 2 will overtake one arriving in slot 20; this only happens because they are different flows and were routed at different times.

AV packet forwarding

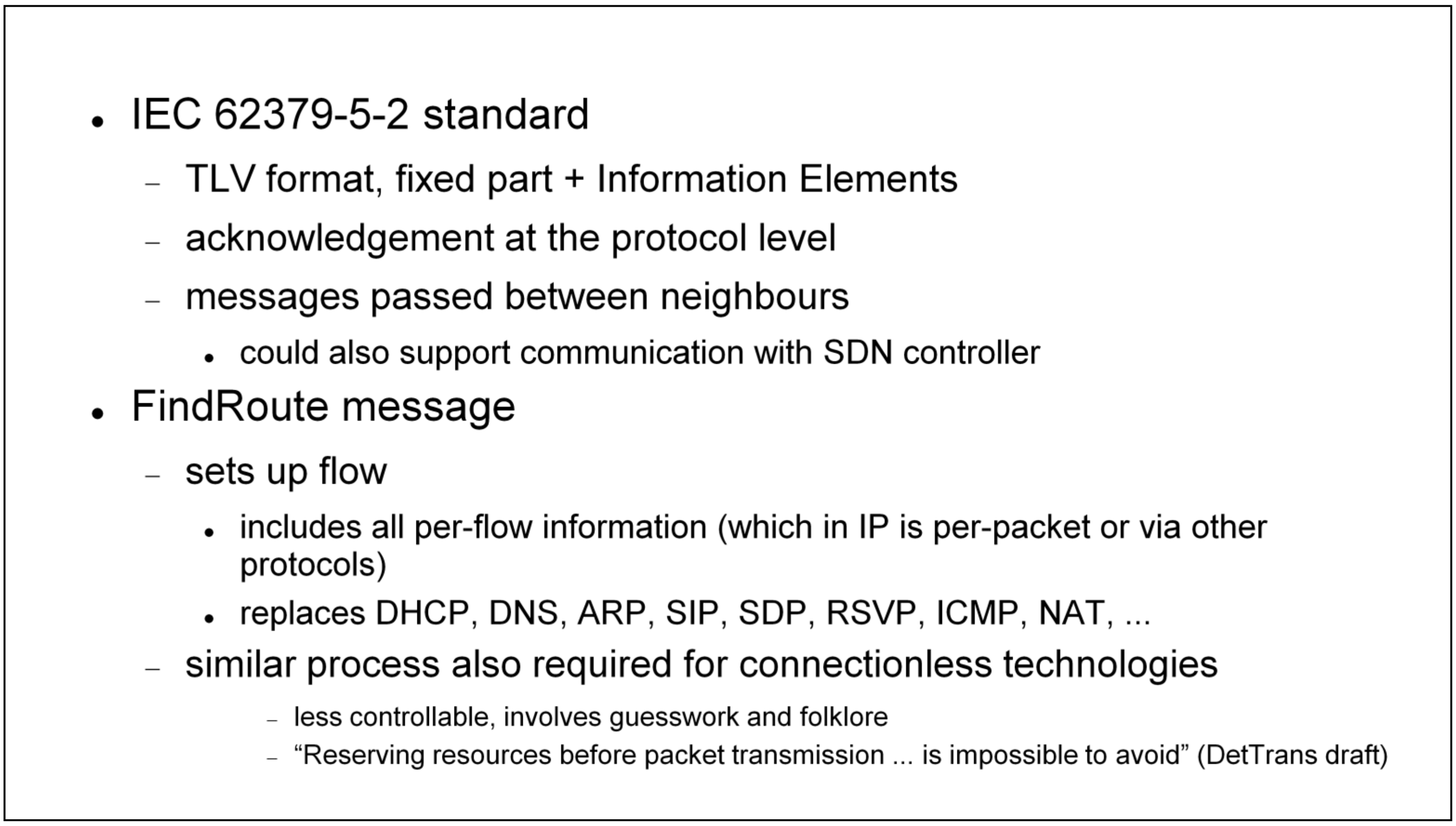

Port 1 incoming: subframes 0,1
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Port 2 incoming: subframes 2,3
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 0 1 2 3 4 5 6 7 8 9 10 11 12 13
Port 3 outgoing: subframes 3,0
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 f 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Port 15 outgoing: subframes 3,0
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 f 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

Now the purple flow has also been routed to port 15, which didn’t have any flows routed previously so each slot occupies the first position in its window. This shows that multicasting is achieved by simply reading the packets out for more than one port.

Port 15’s frames are a few bytes later than port 3’s, so that they read the AV packet buffer at a different time.

Control plane signalling

45

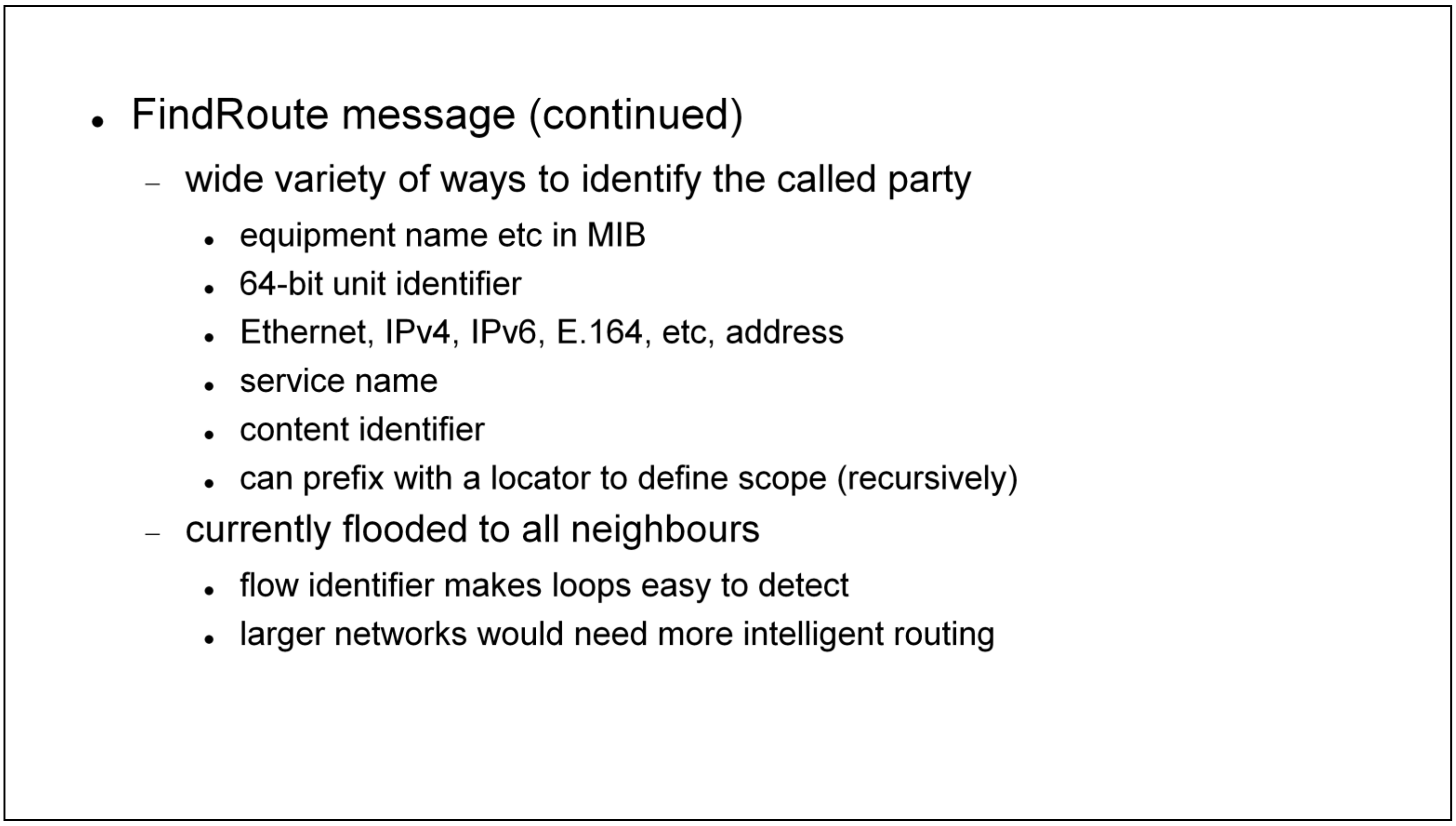


- IEC 62379-5-2 standard
  - TLV format, fixed part + Information Elements
  - acknowledgement at the protocol level
  - messages passed between neighbours
    - could also support communication with SDN controller
- FindRoute message
  - sets up flow
    - includes all per-flow information (which in IP is per-packet or via other protocols)
    - replaces DHCP, DNS, ARP, SIP, SDP, RSVP, ICMP, NAT, ...
  - similar process also required for connectionless technologies
      - less controllable, involves guesswork and folklore
      - "Reserving resources before packet transmission ... is impossible to avoid" (DetTrans draft)

The TLV format (unlike text-based formats) is economical in both the size of messages and the processing power required to parse them, while being flexible and extensible. It is also easy to display in a human-readable form in applications such as WireShark.

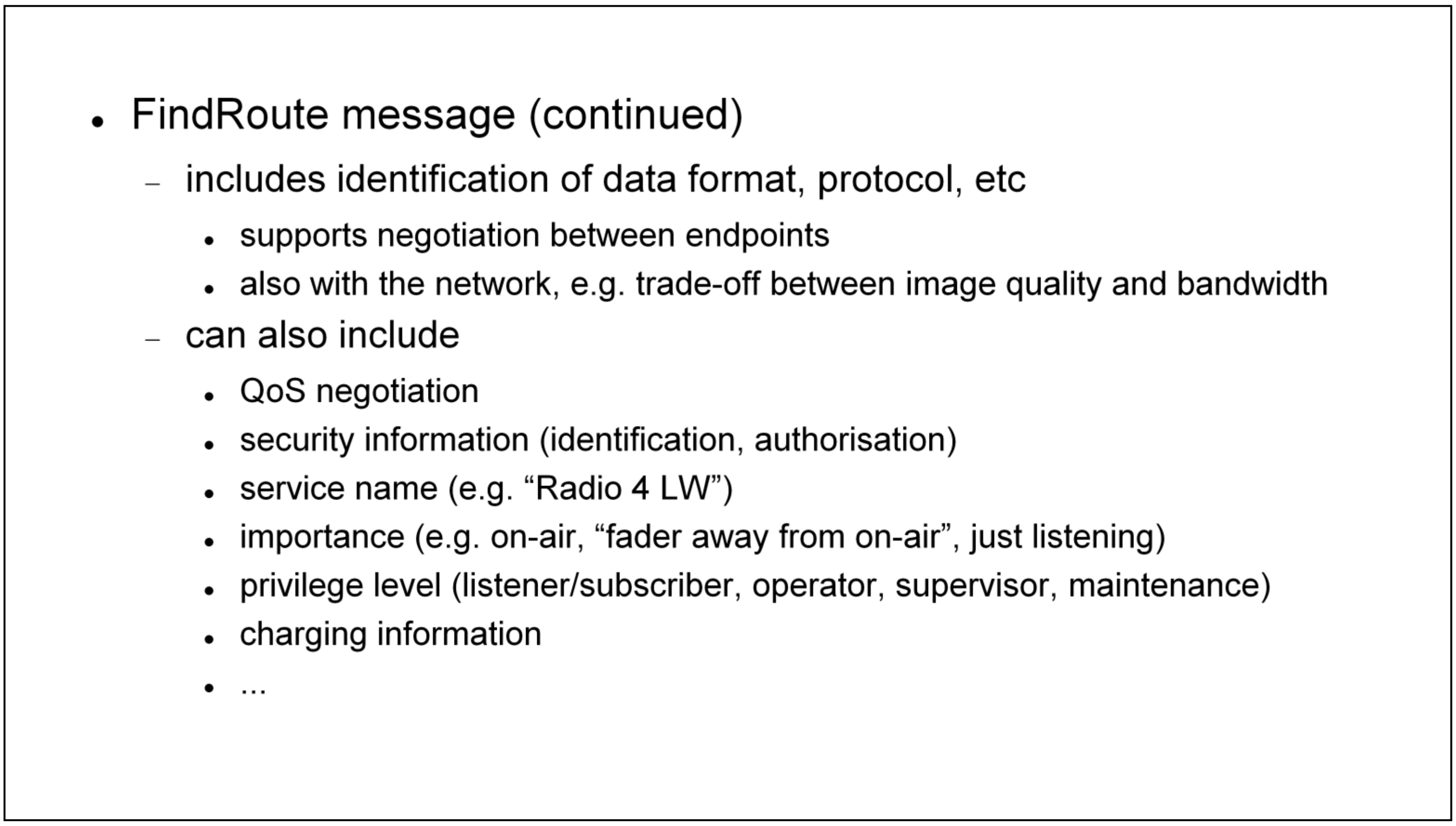
For every message there is an acknowledgement (either an explicit acknowledgement or a further message which is also a reply) and unacknowledged messages are repeated. Thus, the only transport layer functionality required is data protection.

In the current implementation, each incoming message has been either originated or processed by the neighbouring equipment. Therefore, messages can be trusted to the extent that the neighbouring equipment can be trusted. In an SDN-like configuration, additional measures would be required to protect communication with the controller.



- FindRoute message (continued)
  - wide variety of ways to identify the called party
    - equipment name etc in MIB
    - 64-bit unit identifier
    - Ethernet, IPv4, IPv6, E.164, etc, address
    - service name
    - content identifier
    - can prefix with a locator to define scope (recursively)
  - currently flooded to all neighbours
    - flow identifier makes loops easy to detect
    - larger networks would need more intelligent routing

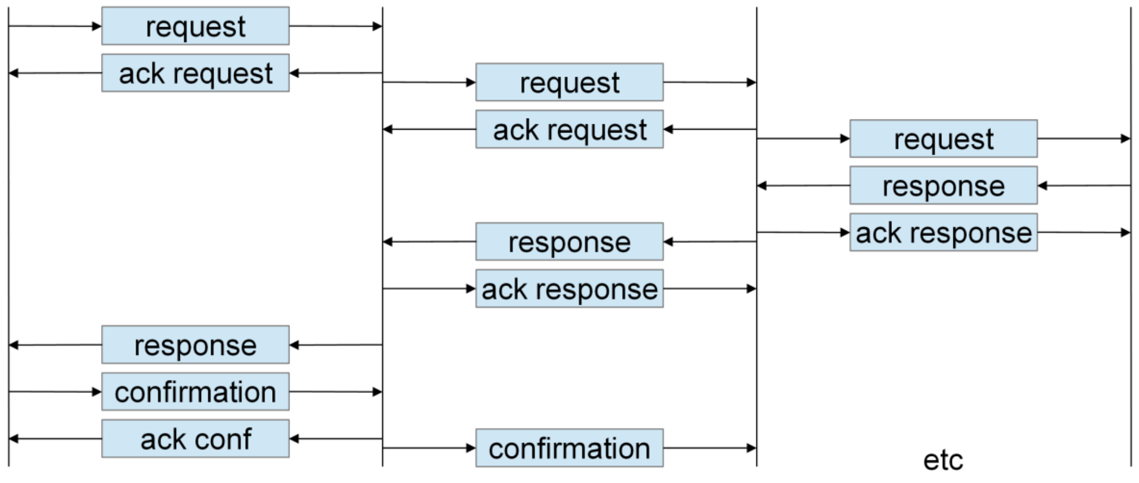
Whereas on an IP network a gethostbyname() call, resulting in a DNS transaction, is made to find the IP address corresponding to a domain name, and that IP address is then used for routing the packets, in Flexilink the domain name can be used directly to identify the remote entity. The access network might route all such calls to the nearest PGW and do the DNS transactions there, reducing traffic on the radio link.



- FindRoute message (continued)
  - includes identification of data format, protocol, etc
    - supports negotiation between endpoints
    - also with the network, e.g. trade-off between image quality and bandwidth
  - can also include
    - QoS negotiation
    - security information (identification, authorisation)
    - service name (e.g. “Radio 4 LW”)
    - importance (e.g. on-air, “fader away from on-air”, just listening)
    - privilege level (listener/subscriber, operator, supervisor, maintenance)
    - charging information
    - ...

Some of the facilities were included specifically for broadcasters, but could have wider application.

- Flow set-up
  - up to 4 messages (plus acknowledgement for each)
    - request, response, confirmation, completion



The acknowledgement is optional if the reply message (e.g. response in reply to a request, confirmation in reply to a response) can be sent immediately. A reply received while waiting for an acknowledgement is treated as an acknowledgement.

There is a timeout after which a message is retransmitted if no acknowledgement has been received.

A ClearDown message for a flow that is in the process of being set up cancels any timeouts for that flow. If a link on which an acknowledgement is awaited goes down it is treated as if a ClearDown had been received.

- Flow set-up
  - up to 4 messages (plus acknowledgement for each)
    - request, response, confirmation, completion
  - current implementation just uses the first two
    - AV flow: call is to “take” from source
      - slots allocated and routing tables written as response message is passed on
    - IT flow: call is bidirectional (sets up two flows)
      - label allocated and routing tables written as message in opposite direction is passed on
  - use more if required to delay setting up flow in forwarding plane
    - authentication
    - negotiating charge for service
    - checking capacity on whole route, or that backup route follows a different path
      - though may be more efficient to set up and then backtrack, because writing tables is easy
    - ...

The route for an AV flow is set up in the same direction as the data flow so that the positions of the slots on the incoming link are decided first. This is necessary to support multicasting, and also to allow simple edge devices to transmit in a fixed set of slots.

The route for an IT flow is set up in the opposite direction. Thus, the whole route has been set up by the time the sender discovers its label value, so the first packet can be sent immediately. Also, on each link the label is chosen by the unit that will use it as a routing table address.

- Flow identifier (128 bits, globally unique)
  - 64 bits globally unique identifier of the “owner” (caller or sender)
  - 32 bits call reference
  - 7 bits route reference
    - allows multiple copies of the data to be sent for resilience
  - 1 bit direction (towards or away from the owner)
  - 24 bits flow reference
    - call may be a bundle of several flows, e.g. audio, video, subtitles, …
- Option to connect route as “cold standby”
  - AddFlow message makes forwarding plane connection

The way flows are grouped was originally influenced by requirements in broadcasting, but should be relevant to other application areas too.

A “call” might link a studio with a control room, and carry several flows including programme audio and video, monitoring, talkback, and control signals for “on air” lights and other studio equipment.

A live TV programme such as a sporting event might be distributed with one flow carrying a low-resolution image suitable for small screens and another carrying additional information to construct a higher-resolution image; different audio flows could carry stereo or surround sound, or commentary in different languages, and further flows could carry subtitles, audio description, and sign language. A user could choose to take only the flows that were needed.

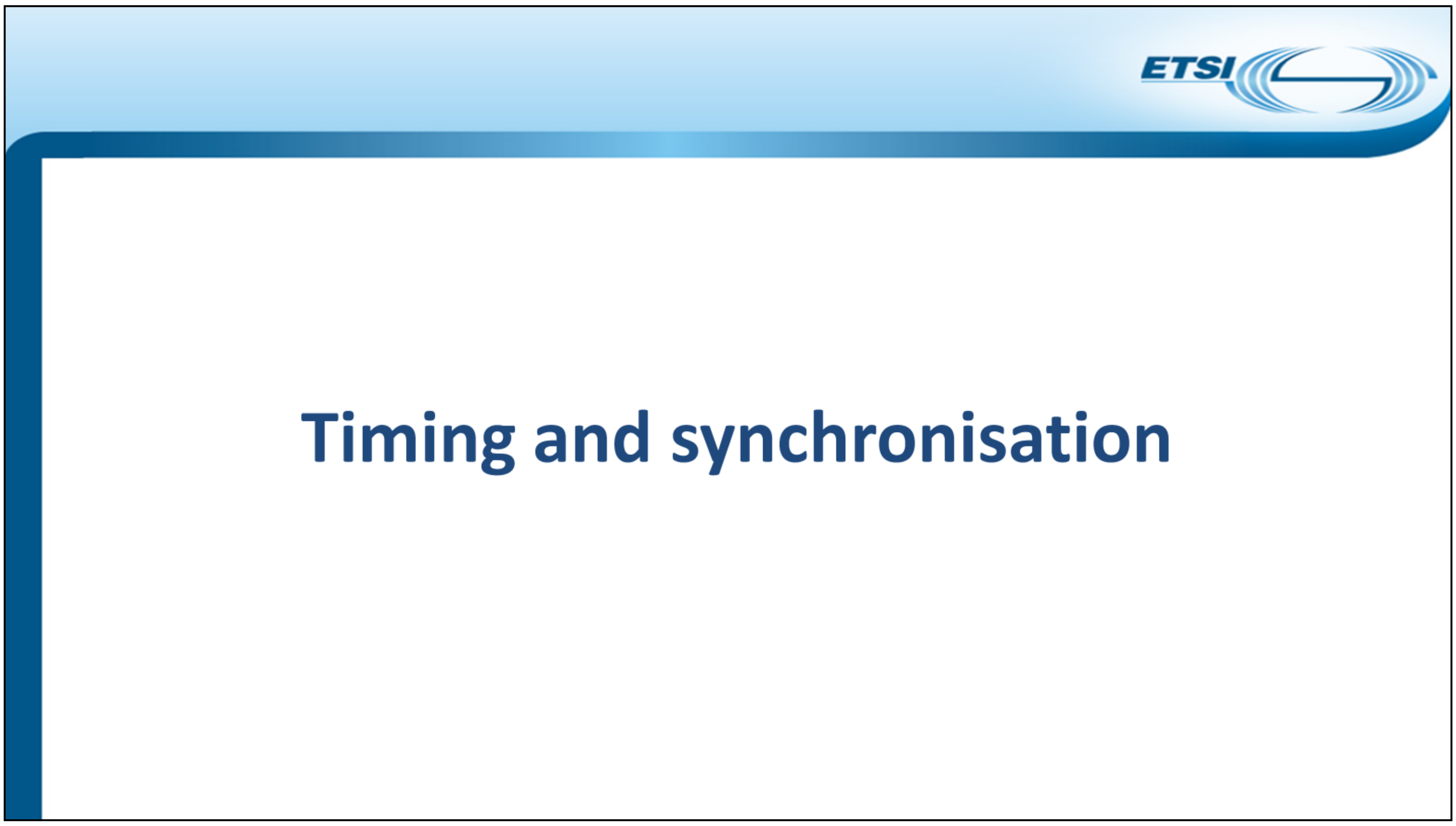
Whereas SDI digital video carries audio, subtitles, etc, in the blanking intervals (which are only there for compatibility with analogue signals intended for display on cathode ray tubes), SMPTE standards for video over packet networks separate these different components out into different flows.

Some of the fields in the flow identifier are probably bigger than they need to be, but it is only used in control plane messages, so size is less important. It is the same size as an IPv6 address, of which two are sent in every packet.

- Quasi-connectionless option
  - many-to-one IT flows (similar to MPLS Forwarding Equivalence Class)
  - appropriate for tunnelling IP
  - potentially more efficient routing for popular destinations
- Other signalling messages
  - ClearDown
    - single message can clear multiple flows, e.g. when link disconnected
  - AsyncSetup (for the quasi-connectionless service)
  - NetworkData, EndToEndData
    - out-of-band messages; currently used to update Importance
  - SyncInfo

NetworkData and EndToEndData can be used to send messages that are not part of a sequence without setting up a forwarding plane flow.

SyncInfo is used to exchange information on synchronisation of frame structures and distribution of network time (see following slides).



Timing and synchronisation

53

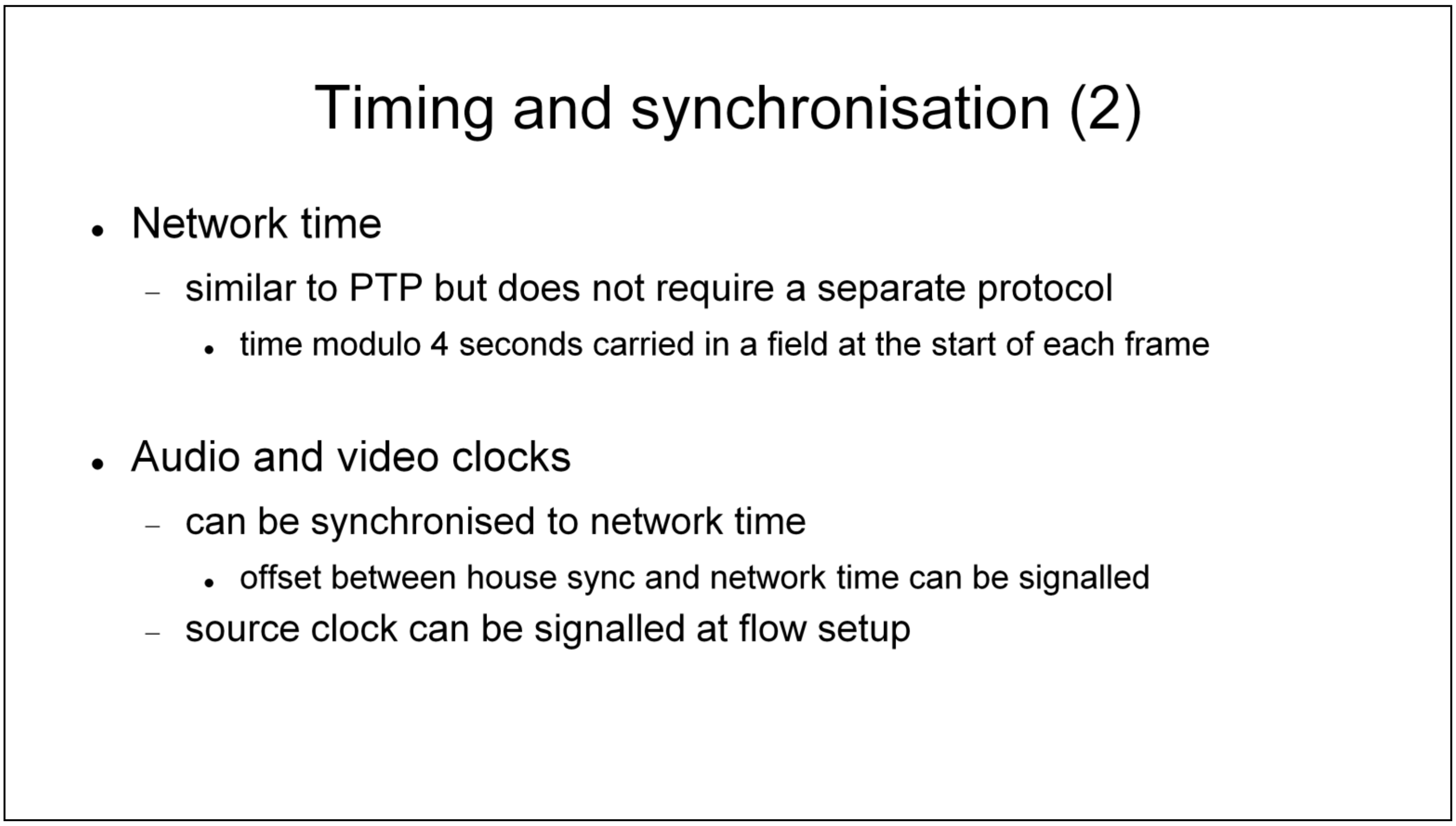
Flexilink islands

- Links between network elements are “physical” or “virtual”
  - physical link frames are phase-locked, for lowest-latency AV flows
  - virtual links are carried over other protocols such as UDP
    - ideally with guaranteed bandwidth and latency
    - de-jitter buffers add to latency
- An “island” consists of units connected by physical links
- Virtual links can be used to connect islands together
  - frames on different islands are frequency-locked
- Islands can also interwork with other technologies via gateways

Timing and synchronisation (1)

- Framing on physical links
  - within an island, all links are phase-locked
    - each unit takes its timing from its “upstream” neighbour
    - outgoing framing maintains constant phase
      - relative to incoming frames on the upstream link
    - simple protocol ensures lock to within about 10ns per hop
      - lock acquired in a few ms
    - no explicit “spanning tree” or “grandmaster selection” needed
  - IT packets still can be forwarded when not phase-locked

Although there is no explicit spanning tree protocol (each unit simply takes its timing from the most appropriate neighbour), the upstream links do form a spanning tree. Each unit keeps track of the frame rate, and if the upstream link is lost it continues to generate frames at the same rate using its local clock as a reference. This ensures that there is not too much drift during the second or two that it takes to form a new spanning tree.



Timing and synchronisation (2)

- Network time
  - similar to PTP but does not require a separate protocol
    - time modulo 4 seconds carried in a field at the start of each frame
- Audio and video clocks
  - can be synchronised to network time
    - offset between house sync and network time can be signalled
  - source clock can be signalled at flow setup

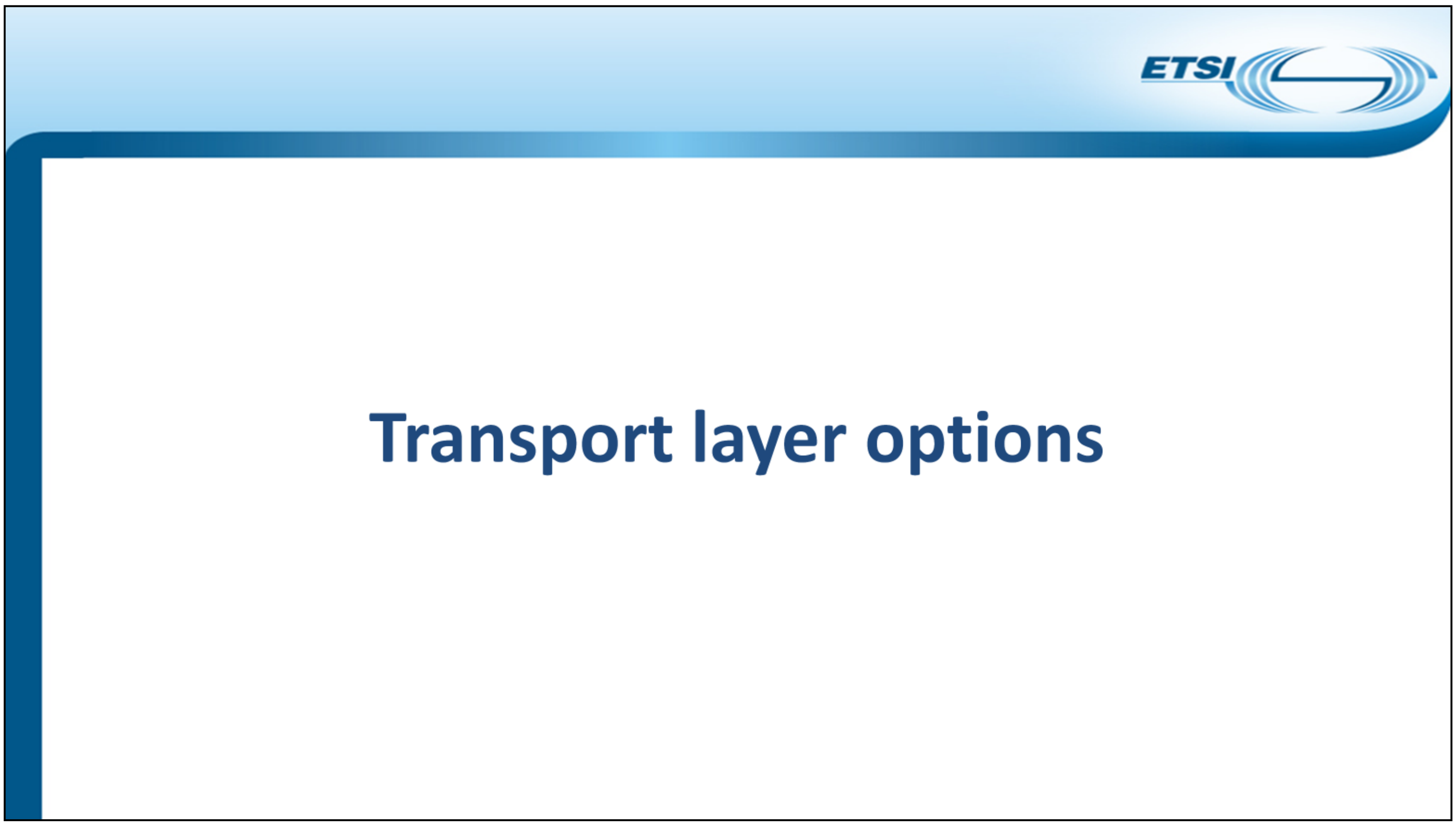
AES51 defines a 32-bit field to carry timing information. Flexilink uses a different format, for compatibility with PTP, consisting of the ls 2 bits of the number of seconds and 30 bits containing the number of nanoseconds. It carries the sender's value for network time at the beginning of the packet. The rest of the number of seconds is sent in a SyncInfo packet, which also reports the source of the sender's network time. Each unit chooses which, of the various sources that are available, to use for its own network time. As with the frame structure, it continues at the same frequency if the reference is lost.

Note that although the frame structure and network time use similar mechanisms they are not related to each other. In particular, there is a measure of "quality" for network time, as there is in PTP, whereas for frames there is no requirement to synchronise with anything in the real world and in practice the timing is taken from the local oscillator in the unit with the numerically largest 64-bit identifier.

Individual audio and video flows are not expected to be synchronised to network time or to each other, and certainly not to the frame rate. Various methods of indicating the relationship between a media clock and network time are possible. In the absence of any other information, playout clocks are adjusted to match the rate at which the stream arrives over the network.

Security

- Routing only uses information from neighbour
  - signalling messages forwarded by neighbour after its own processing
    - if using SDN, area covered is a walled garden
  - can check before setting up flow in forwarding plane
    - more controllable than firewalls based on IP address and port number
    - message can include information on trustworthiness of upstream systems
    - possibility to notify “fingerprint” of DDoS attack to edge
  - forwarding plane only routes flows set up by control plane
    - IT flow label from neighbour’s routing table
    - AV slot positions assigned by neighbour
- Signalling messages get priority over other IT flows
  - can’t be impeded by excess of user traffic

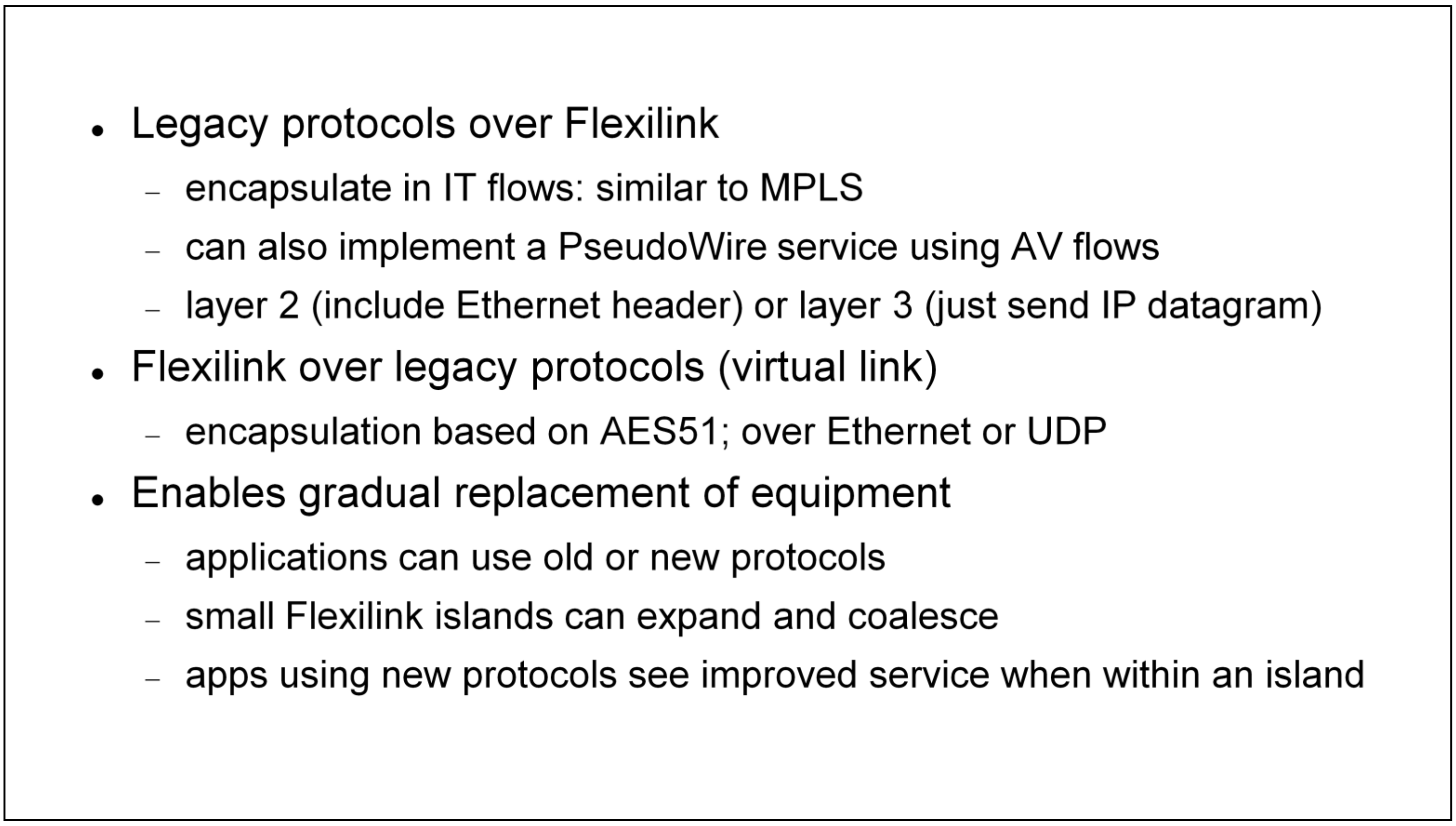


Transport layer options

- TCP
  - adjusts throughput to match network
  - retransmits dropped packets
  - provides an end-to-end data integrity check
- All still needed if used over the IT service
- For transport over the AV service
  - throughput can be negotiated between endpoints and network
  - no dropped packets
  - just need end-to-end data integrity check
  - confirmation of correct receipt can be combined with flow clear-down

Migration

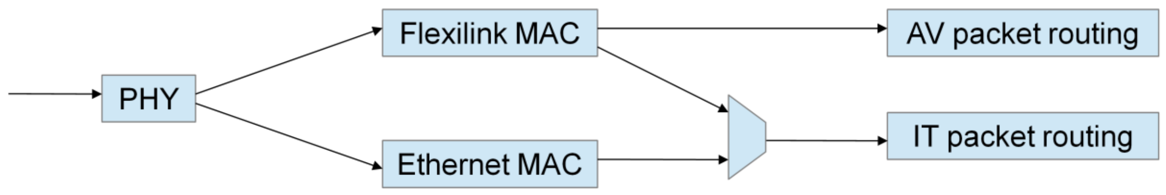
61



- Legacy protocols over Flexilink
  - encapsulate in IT flows: similar to MPLS
  - can also implement a PseudoWire service using AV flows
  - layer 2 (include Ethernet header) or layer 3 (just send IP datagram)
- Flexilink over legacy protocols (virtual link)
  - encapsulation based on AES51; over Ethernet or UDP
- Enables gradual replacement of equipment
  - applications can use old or new protocols
  - small Flexilink islands can expand and coalesce
  - apps using new protocols see improved service when within an island

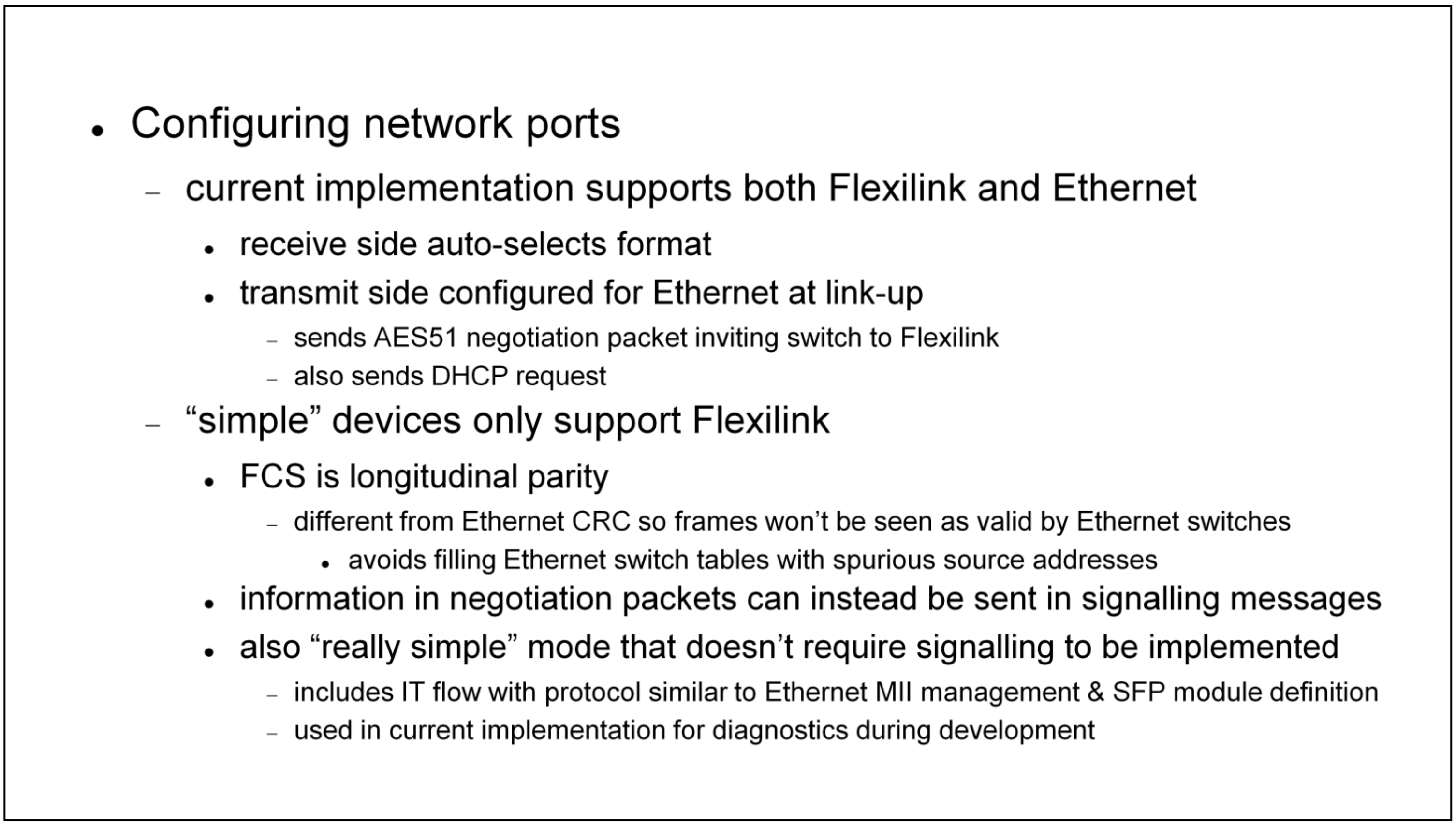
Flexilink over UDP is similar to over Ethernet (see earlier slide) but with addition of IP and UDP headers. Port number is tbd; the current implementation uses the same 16-bit value as the AES Ethertype.

- Configuring network ports
  - current implementation supports both Flexilink and Ethernet
    - receive side auto-selects format



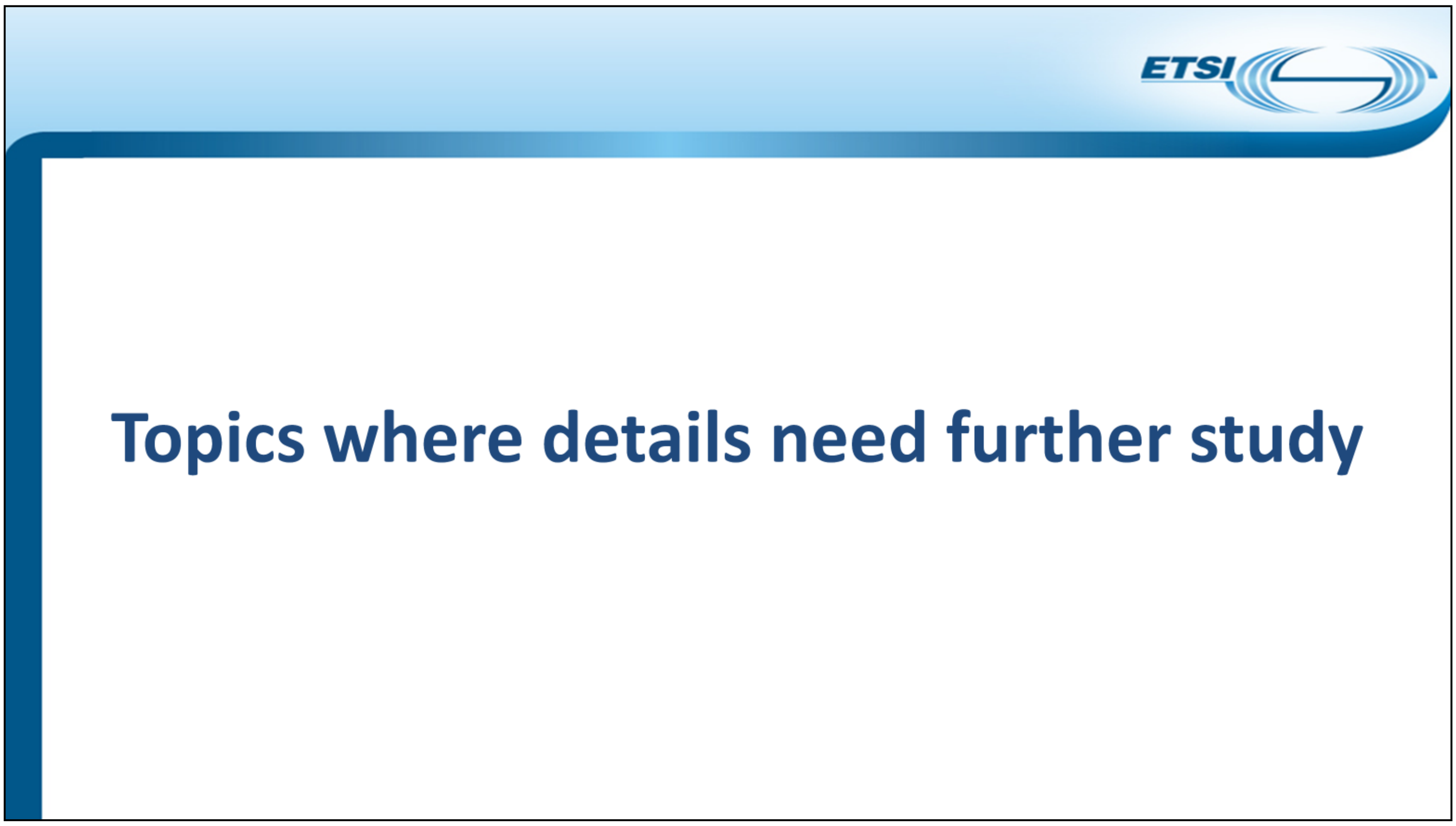
  - Flexilink MAC is selected if it recognises correct frames
  - Ethernet MAC is selected otherwise

Both MAC logics are active except that whenever the Flexilink MAC is receiving a valid signal the Ethernet MAC can be powered down. The Flexilink MAC is quite simple, so keeping it active while receiving Ethernet packets does not consume much power, but it could be kept powered down until the Ethernet MAC detects a jumbo frame. Note that there is no performance penalty in keeping them both active; this would not be the case if they were implemented in software, sharing a CPU.



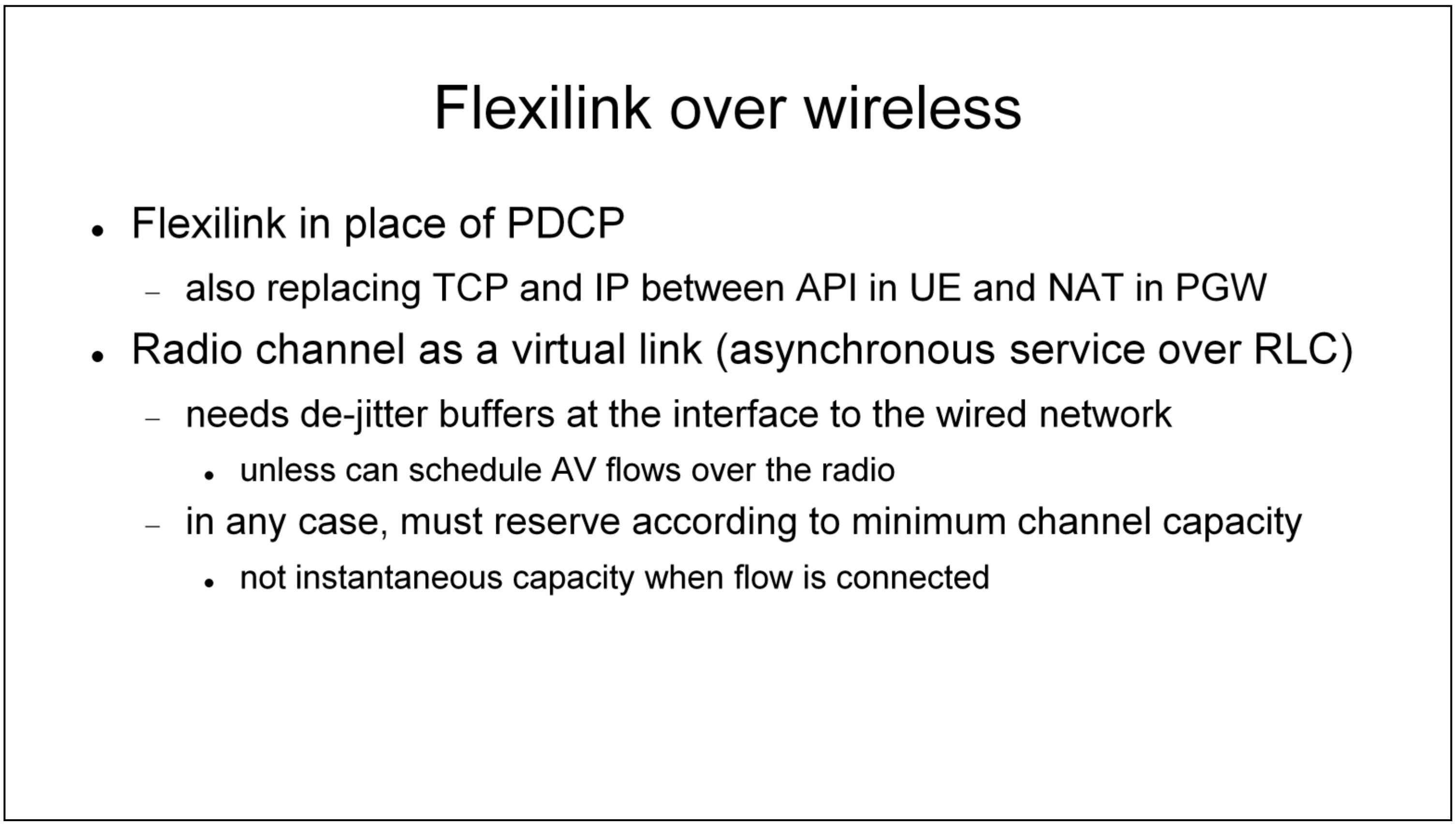
- Configuring network ports
  - current implementation supports both Flexilink and Ethernet
    - receive side auto-selects format
    - transmit side configured for Ethernet at link-up
      - sends AES51 negotiation packet inviting switch to Flexilink
      - also sends DHCP request
  - “simple” devices only support Flexilink
    - FCS is longitudinal parity
      - different from Ethernet CRC so frames won’t be seen as valid by Ethernet switches
        - avoids filling Ethernet switch tables with spurious source addresses
    - information in negotiation packets can instead be sent in signalling messages
    - also “really simple” mode that doesn’t require signalling to be implemented
      - includes IT flow with protocol similar to Ethernet MII management & SFP module definition
      - used in current implementation for diagnostics during development

The transmit side is switched to the format for a Flexilink physical link in response to an AES51 negotiation packet, or when Flexilink frames are detected on the receive side.



Topics where details need further study

65



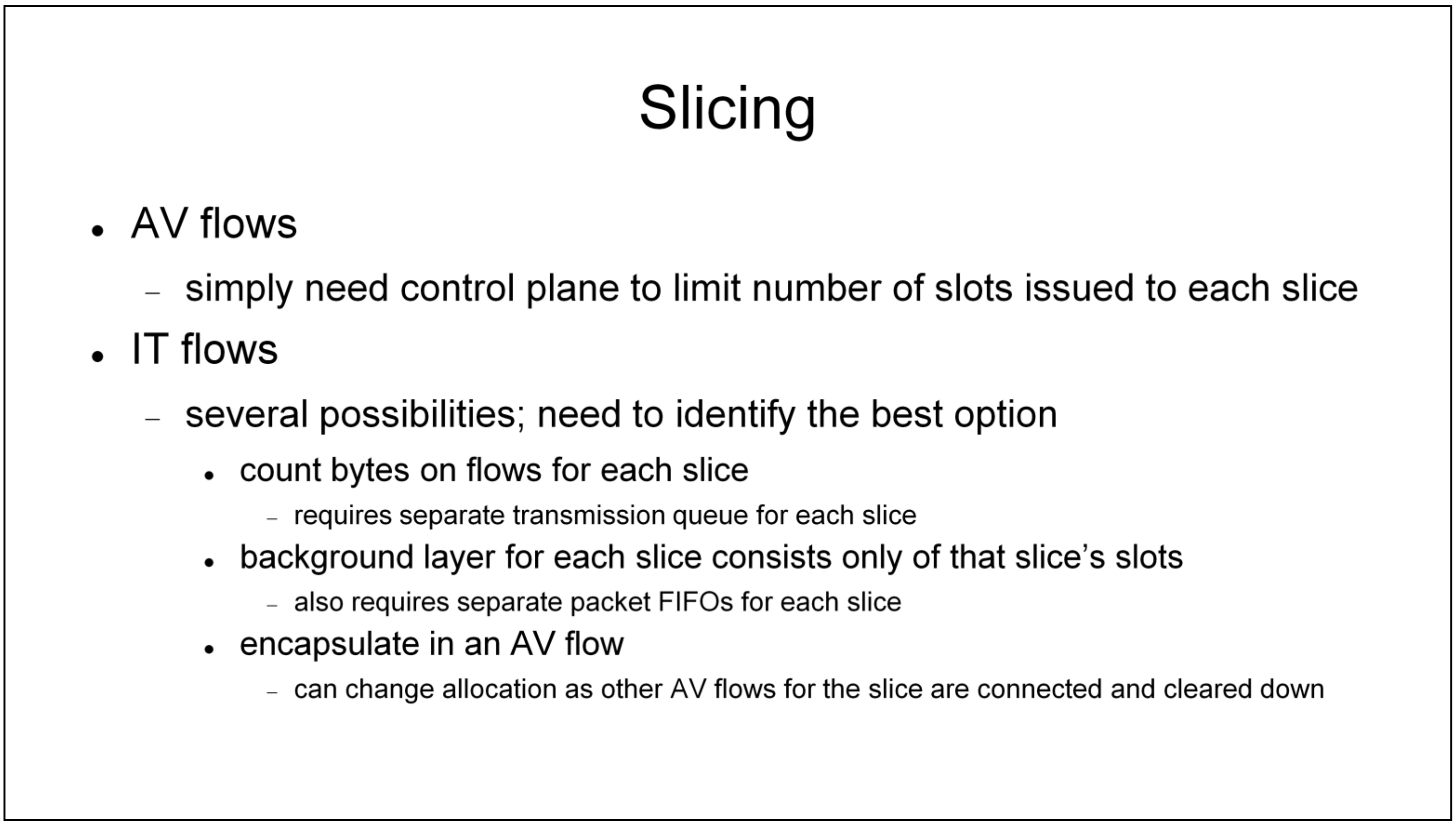
Flexilink over wireless

- Flexilink in place of PDCP
  - also replacing TCP and IP between API in UE and NAT in PGW
- Radio channel as a virtual link (asynchronous service over RLC)
  - needs de-jitter buffers at the interface to the wired network
    - unless can schedule AV flows over the radio
  - in any case, must reserve according to minimum channel capacity
    - not instantaneous capacity when flow is connected

We need to review the structure of the core and access networks, to identify which of the entities that were present in the previous generation are no longer needed, or would not be needed in a Flexilink based system.

Assuming retransmission is needed when packets are lost due to interference or fading, the time of arrival of an AV packet cannot be accurately predicted, and variation in the time may be more than the size of a subframe (which defines the size of the "window" in the AV forwarding buffer, see earlier slides). However, there may be other ways to identify the flow to which it belongs than adding a label. Also, for some real-time traffics retransmission may not be appropriate, for instance repeating a previous value may be less useful than sending a new, up-to-date, value.

Algorithms that reserve resources for AV flows need to allow for fluctuations that may occur in the data rate achievable in the radio channel.

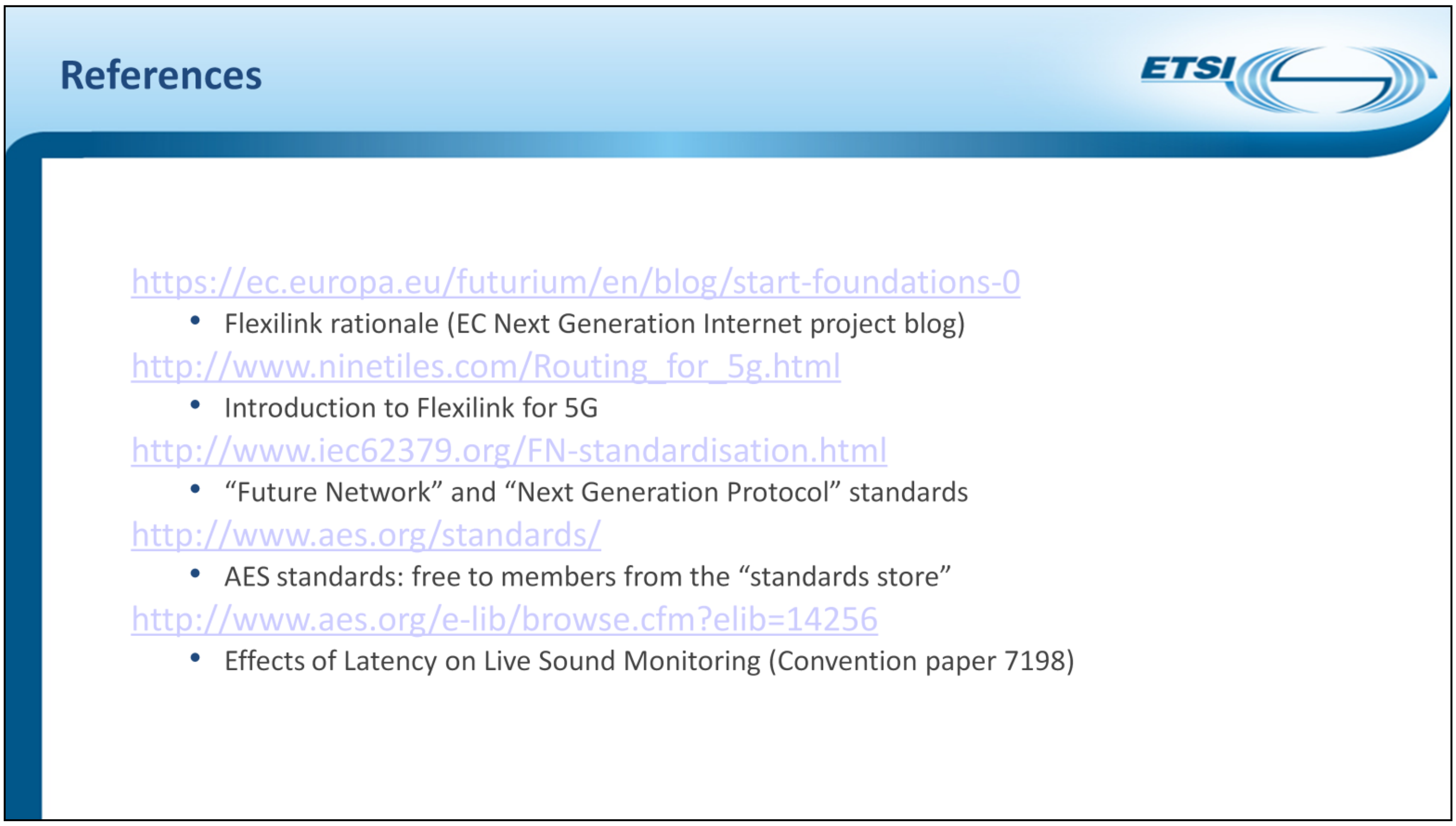


Slicing

- AV flows
  - simply need control plane to limit number of slots issued to each slice
- IT flows
  - several possibilities; need to identify the best option
    - count bytes on flows for each slice
      - requires separate transmission queue for each slice
    - background layer for each slice consists only of that slice’s slots
      - also requires separate packet FIFOs for each slice
    - encapsulate in an AV flow
      - can change allocation as other AV flows for the slice are connected and cleared down

Slicing requires reservation of resources, which is already a feature of the AV service.

For the IT service, reservation might be achieved by rationing the total number of bytes that can be sent for each slice (stopping transmission of IT packets when the allocation is exceeded), or by a hard partitioning of the frame, or by leveraging the facilities of the AV service. The chosen solution would need to be compatible with any IP-based slicing defined for Release 16.



References

https://ec.europa.eu/futurium/en/blog/start-foundations-0
- Flexilink rationale (EC Next Generation Internet project blog)

http://www.ninetiles.com/Routing_for_5g.html
- Introduction to Flexilink for 5G

http://www.iec62379.org/FN-standardisation.html
- “Future Network” and “Next Generation Protocol” standards

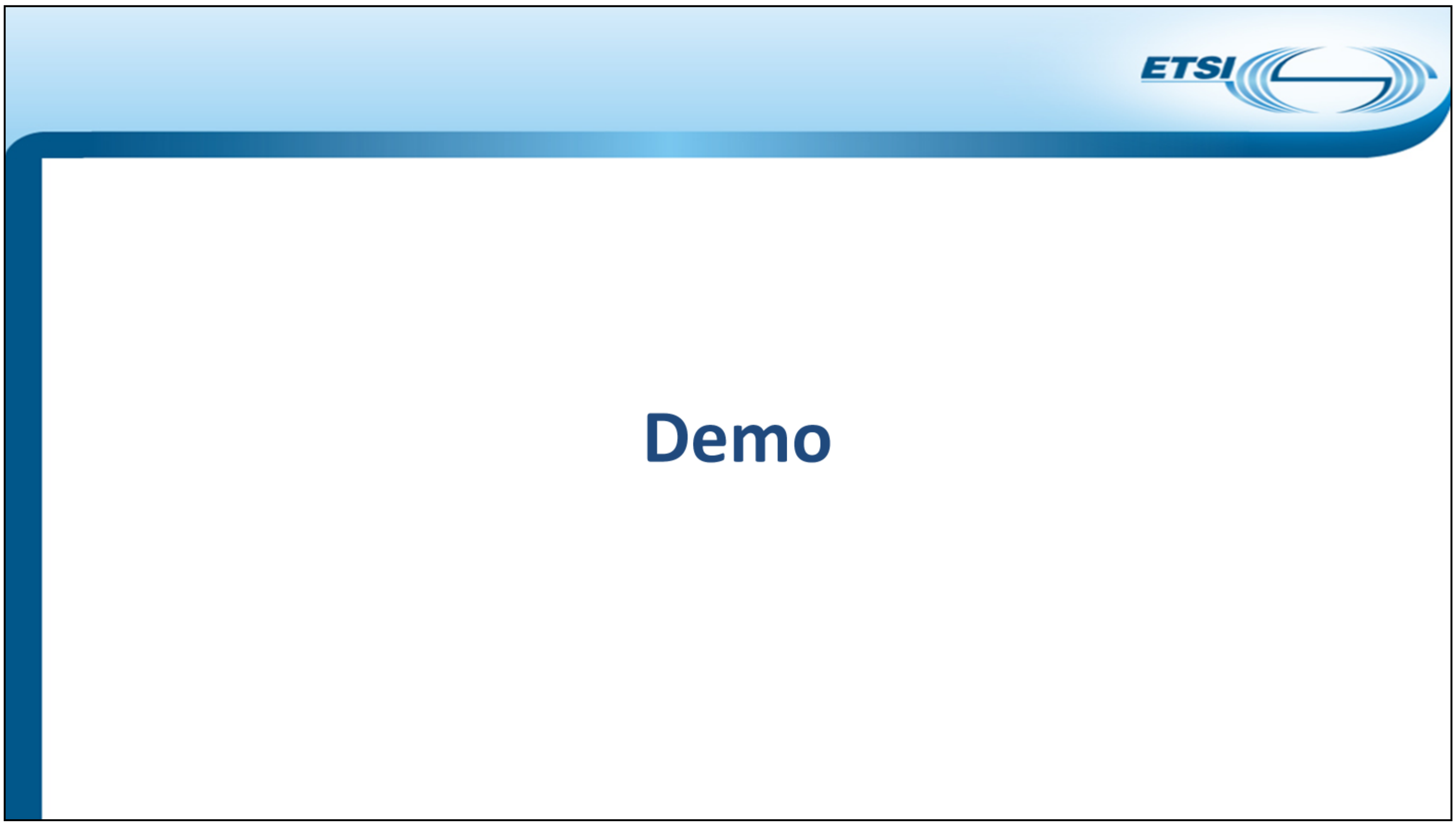
http://www.aes.org/standards/
- AES standards: free to members from the “standards store”

http://www.aes.org/e-lib/browse.cfm?elib=14256
- Effects of Latency on Live Sound Monitoring (Convention paper 7198)

Hyperlinks (which don’t seem to be live in the slide)

https://ec.europa.eu/futurium/en/blog/start-foundations-0
http://www.ninetiles.com/Routing_for_5g.html
http://www.iec62379.org/FN-standardisation.html
http://www.aes.org/standards/
http://www.aes.org/e-lib/browse.cfm?elib=14256

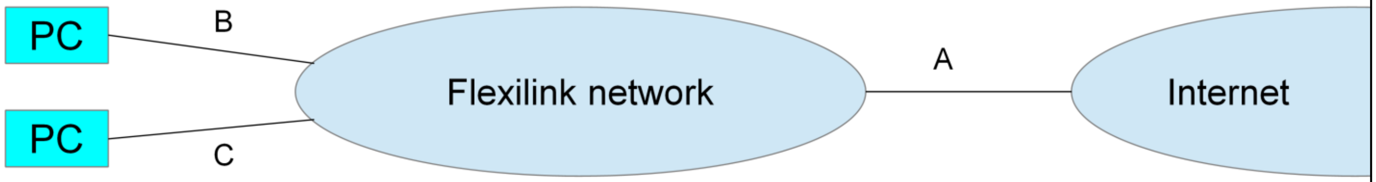
68



Demo

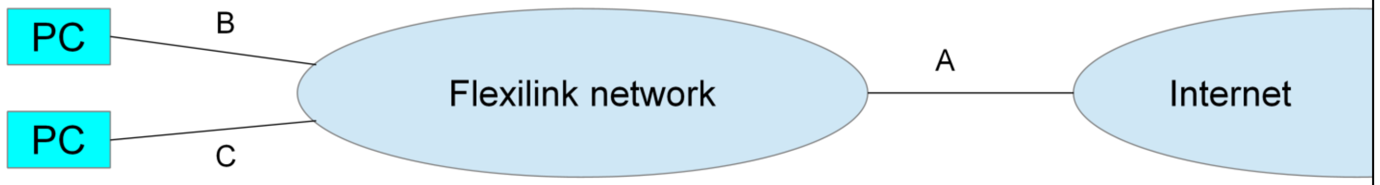
The following slides explain some aspects of the demonstration.

Access to Internet



- Link A comes up: Flexilink port acquires an IP address
- Link B comes up: Flexilink port detects link partner is a single device
  - requests a layer 2 “tunnel” connection for link partner’s MAC address
    - request can be accepted by any port with a connection to an IP network
  - packets arriving on A for that address are sent down the tunnel
- Link C comes up: similar tunnel connected to A
  - multicast packets arriving at A are sent down both tunnels

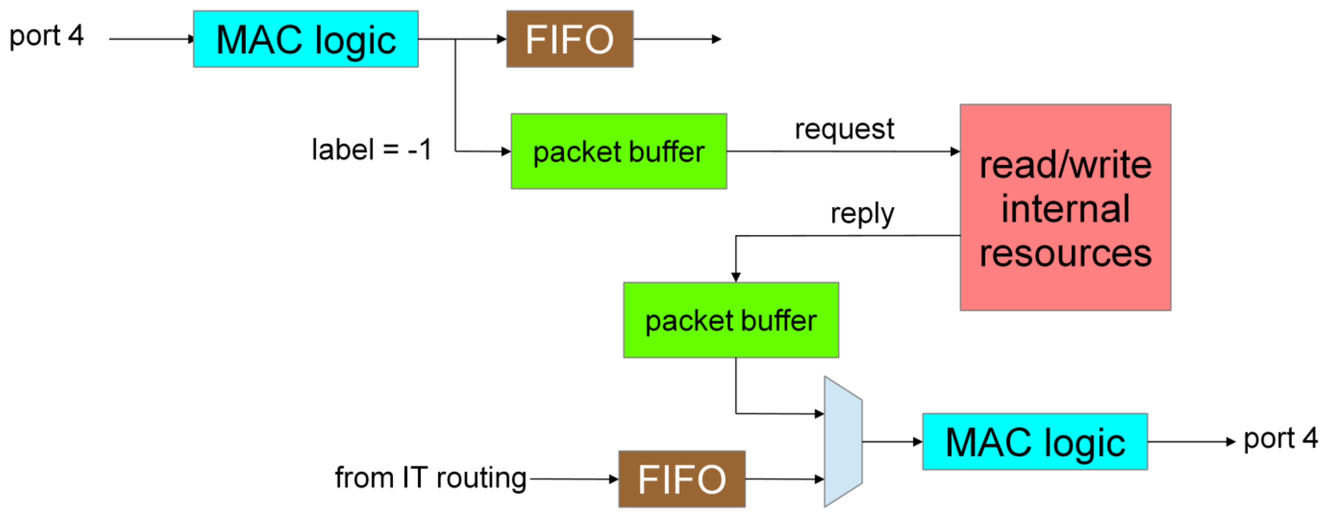
Access to Flexilink network

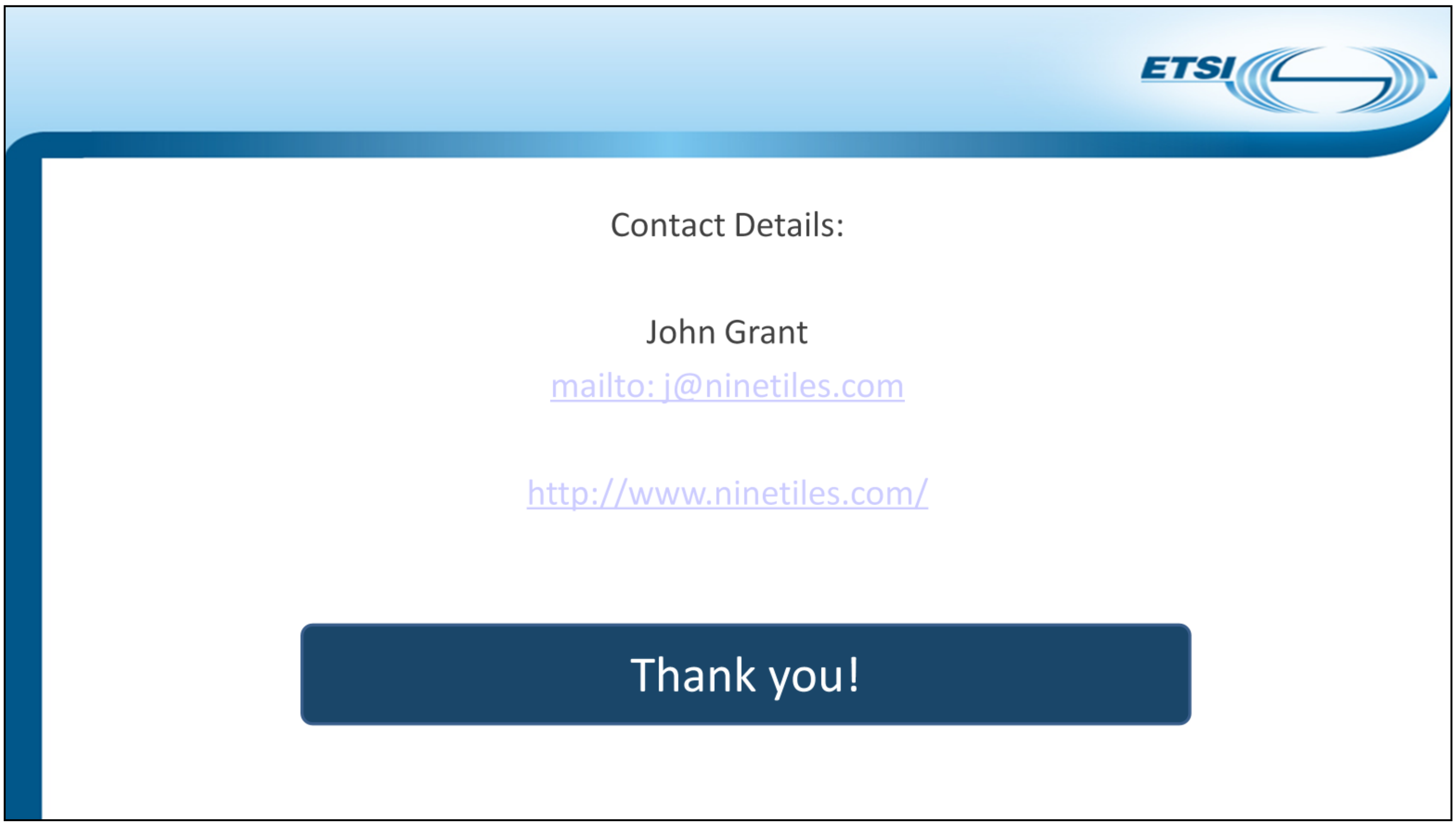


- Multicast packets arriving at B are processed by the Flexilink node
- Program on PC broadcasts a request to connect a virtual link
  - Flexilink node accepts, telling PC its Ethernet and IP addresses
    - "filtering engine" in MAC logic routes packets to those addresses to the CPU
  - PC program sends FindRoute for a management connection
    - messages use AES51 encapsulation: "filtering engine" strips encapsulation, routes them directly

dest MAC address	srce MAC address	88D D	0 2	2 6	timing word	IT pkt hdr	data

Simple Control Protocol





Contact Details:

John Grant

mailto: j@ninetiles.com

http://www.ninetiles.com/

Thank you!

Hyperlinks (which don't seem to be live in the slide)

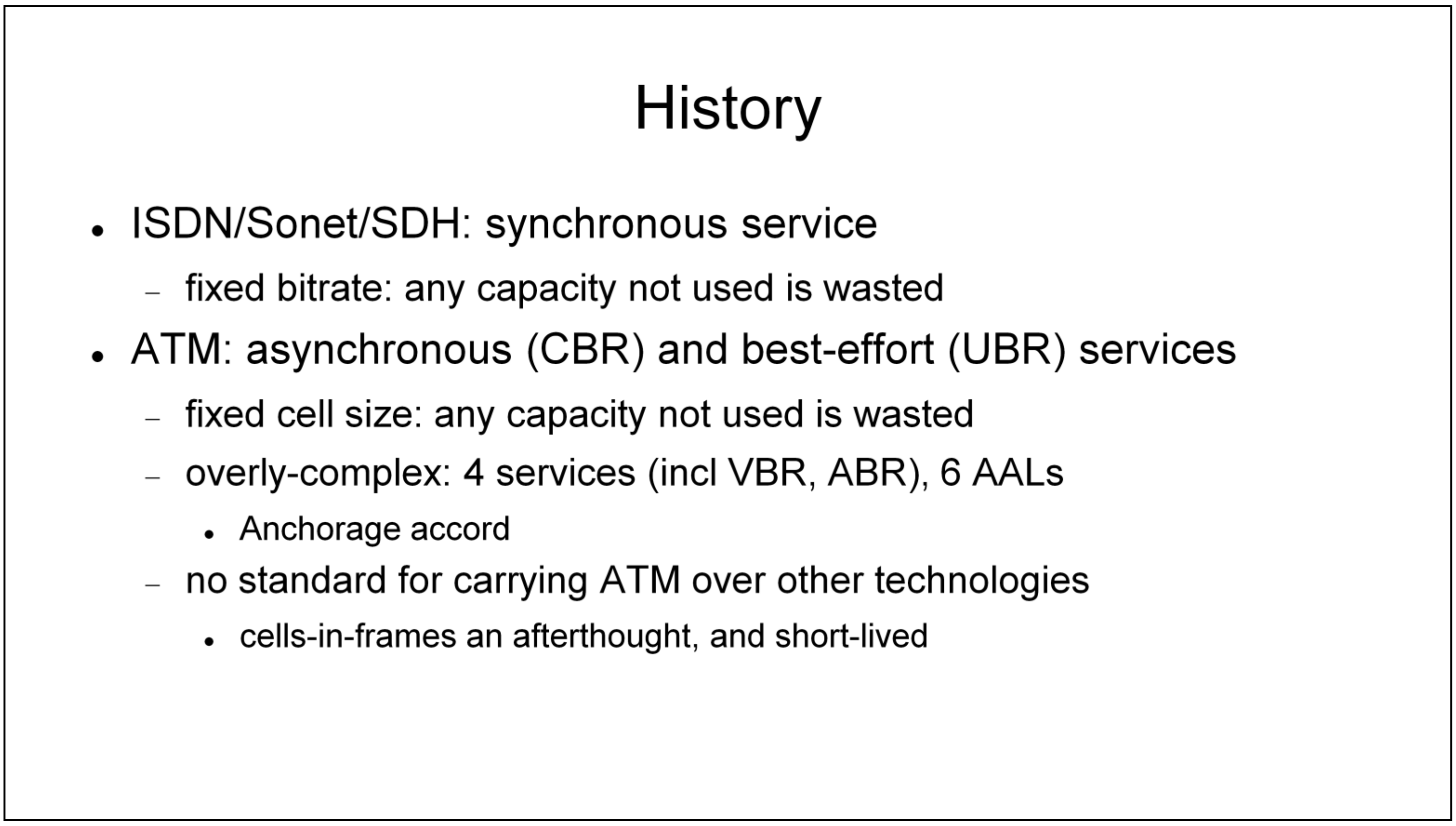
mailto: j@ninetiles.com

http://www.ninetiles.com/

IP

more up-to-date forwarding

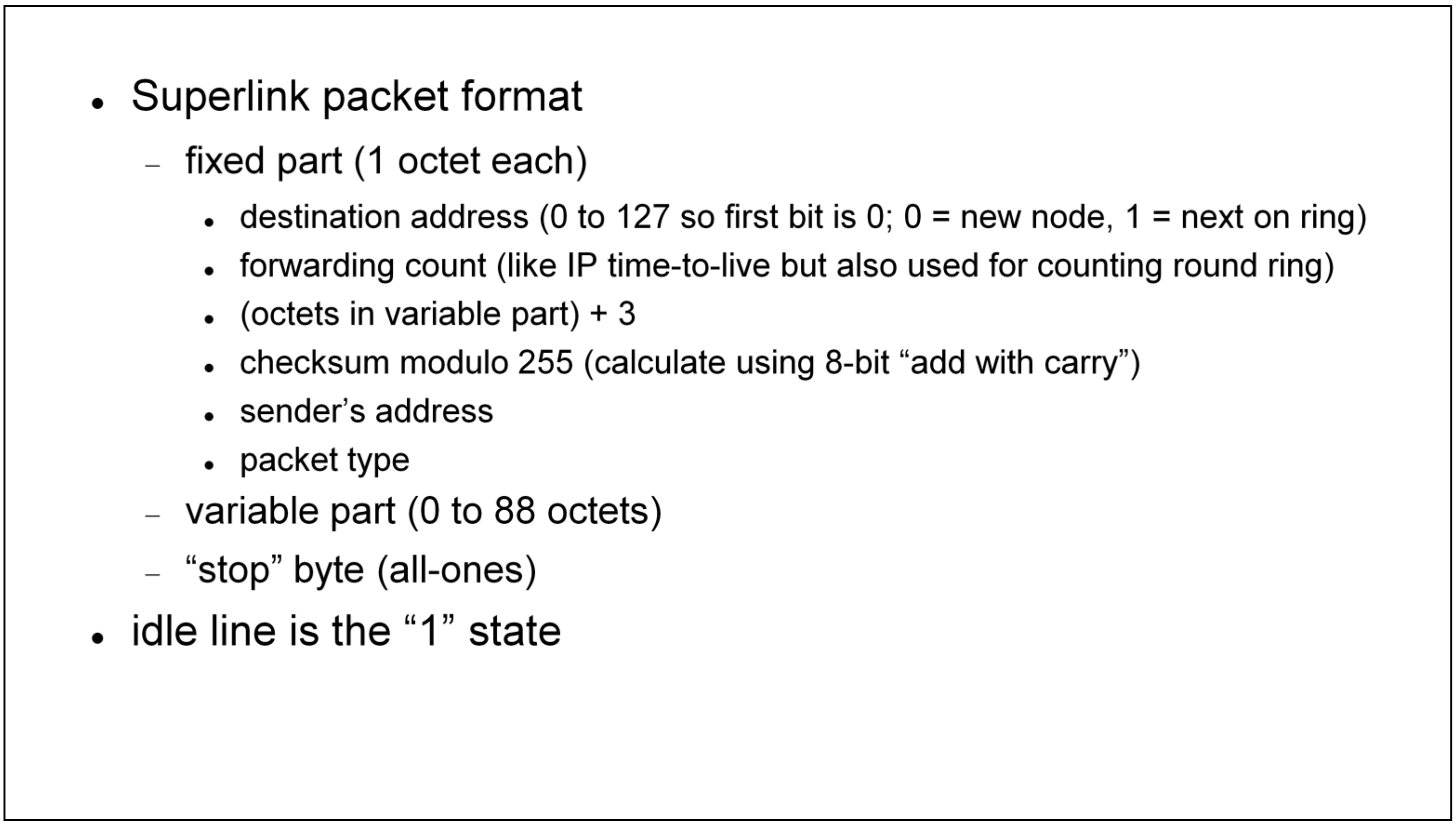
This and the following slides are included in case needed for Q&A.



History

- ISDN/Sonet/SDH: synchronous service
  - fixed bitrate: any capacity not used is wasted
- ATM: asynchronous (CBR) and best-effort (UBR) services
  - fixed cell size: any capacity not used is wasted
  - overly-complex: 4 services (incl VBR, ABR), 6 AALs
    - Anchorage accord
  - no standard for carrying ATM over other technologies
    - cells-in-frames an afterthought, and short-lived

The Anchorage Accord was an agreement made after about 7 years to stop writing new standards for ATM and concentrate on implementation. However, two areas that were still incomplete, and needed to be finished off, were how you send data and how you send voice.



- Superlink packet format
  - fixed part (1 octet each)
    - destination address (0 to 127 so first bit is 0; 0 = new node, 1 = next on ring)
    - forwarding count (like IP time-to-live but also used for counting round ring)
    - (octets in variable part) + 3
    - checksum modulo 255 (calculate using 8-bit "add with carry")
    - sender's address
    - packet type
  - variable part (0 to 88 octets)
  - "stop" byte (all-ones)
- idle line is the "1" state

The framing is similar to RS232, i.e. timing is taken from the 1-to-0 transition at the start of the frame. Whereas RS232 was specified for mechanical teletypes with timing accurate to about 3%, so the sampling point could drift away from the middle of the bit cell after about 11 bits, here we have clocks accurate to 100 ppm so frames can be up to 760 bits long.

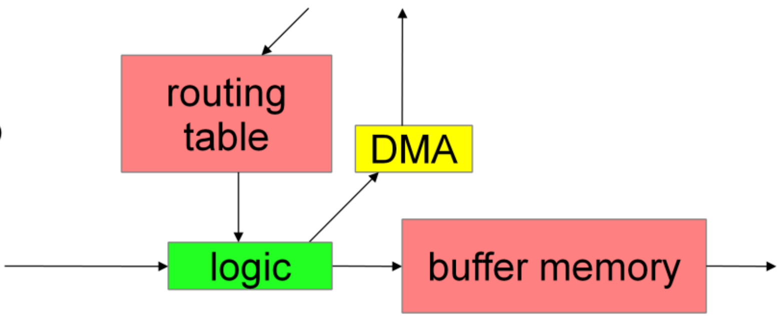
None of the octets in the fixed or variable part is allowed to be all-ones (so the "length" field in the header is redundant information; I forget why we needed both). For data packets, a "cipher key" is chosen that is XOR'd with each data byte.

A node's address is chosen when it joins the ring; it can be any value that isn't already in use, and there's a protocol that ensures addresses are unique even if the whole ring is powered on at the same time. Addresses aren't visible to users, who identify nodes by name or by position in the ring.

The ms bit of the packet type is 1 for an acknowledgement; the next is 0 for a data packet and 1 for a control packet. Each data packet is associated with a "channel"; this is similar to an ATM Virtual Channel (though it was defined about 8 years earlier) and the channel number indexes a table in the same way as the label on a Flexilink IT packet. A channel may be connected either to the host or to the node's management agent.

Control plane functions, including setting up and clearing down connections and reporting the state of the ring, are implemented by control packets. On the host interface, data on an unconnected channel is interpreted as a connection request.

- IP (connectionless) is supposed to avoid overheads of flow set-up
  - in practice the work has to be done anyway, and is less well controlled
  - Flexilink's set-up is much quicker than Dante's
  - Sockets paradigm is connection-oriented; most traffic is TCP or RTP
- Connection-oriented can be more secure
  - can check more information than will fit in a packet header
  - incoming packets with no routing table entry can simply be discarded



Dante is a system for sending audio over IP; I've been told it takes about 2 seconds to set up a flow. Connecting a flow in Flexilink takes a few tens of milliseconds.

PEANUTS
NOW SAY WE CUT AN APPLE IN HALF...
WE HAVE TWO HALVES, DON'T WE? NOW...
THAT'S FRACTIONS!! YOU'RE TRYING TO TEACH ME FRACTIONS!

This one is for those who object to anything that looks like a pre-IP telecom technology, or which they identify as "beads on a string".