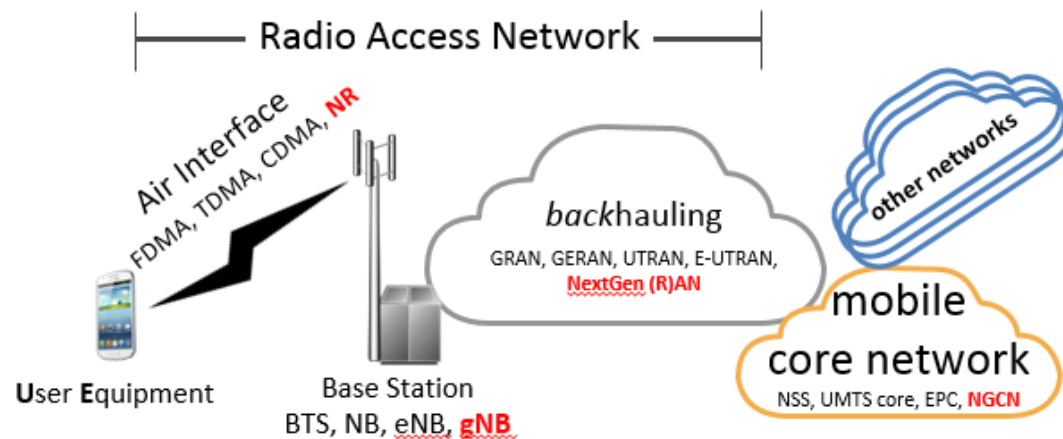


# 5G xHaul



# Reminder: 4G back/front-haul

We remember that in 4G there were backhaul and fronthaul

Backhaul transports 2 interfaces:

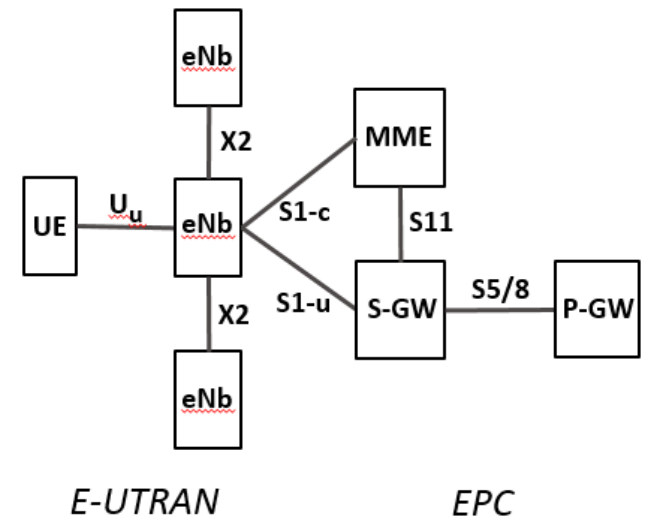
- S1 from the eNB to the core (divided into S1-U and S1-C)
- X2 between eNBs, used for
  - smooth handoff
  - CoMP

X2 data rates are lower (typically <5% of S1)  
but delay constraints may be much higher  
(< 10 ms, the lower the better)

It is expensive to physically interconnect eNBs  
so in practice, X2 is mostly a *logical* interface  
backhauled up to an aggregation point  
and then back down to another eNB

4G fronthaul is between

- **Remote Radio Unit** (AKA **Remote Radio Head**)
- **BaseBand Unit**

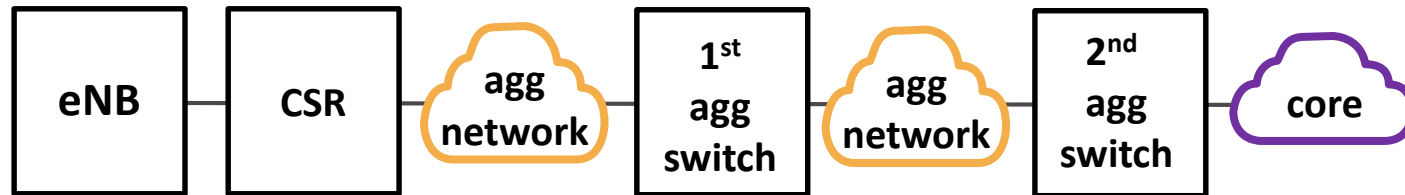


# 4G backhaul in practice

3GPP specifications do not detail *how* to backhaul S1 interfaces

In practice, there is a backhaul network with multiple network elements

- **Cell Site Router** or **Cell Site Gateway**
- 1<sup>st</sup> aggregation switch
- 2<sup>nd</sup> aggregation switch



The physical layer of the aggregation networks may be

- fiber (including **Passive Optical Networks**) mostly at 1 Gbps rates
- microwave
- DSL

in p2p, tree, ring or mesh topologies

The higher layer may be pure IP, Ethernet, or various flavors of MPLS

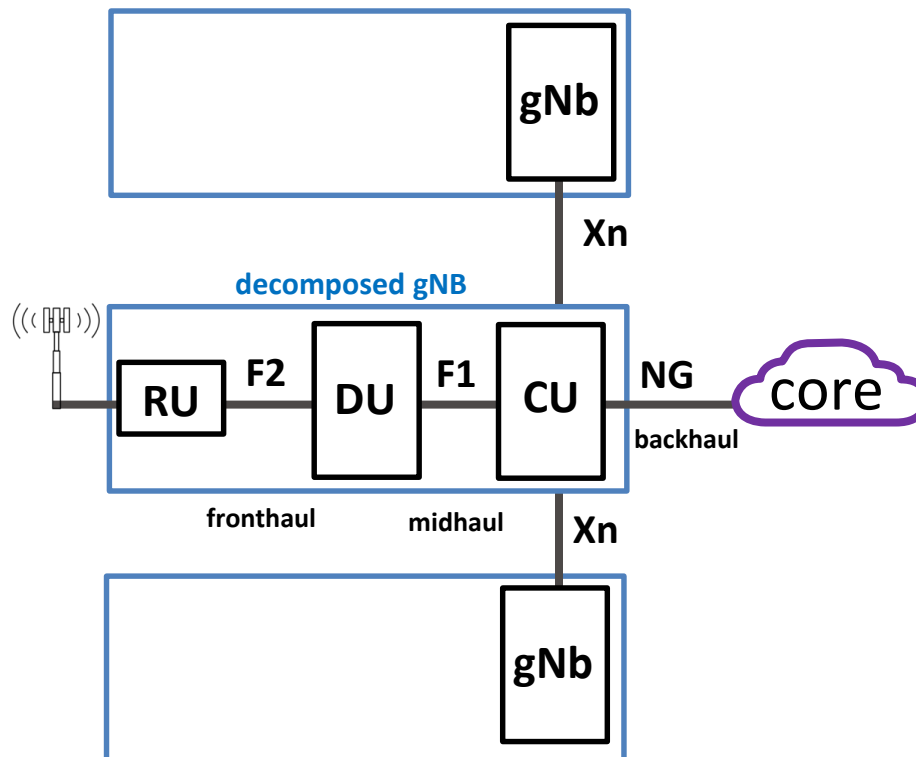
In the higher aggregation network OTN and WDM may be deployed

# 5G xHaul

At the highest level of abstraction a 5G network consists of two entities:

- 5G base-station (gNB)
- 5G core network (5GCN)

The interface between the gNBs and the core is called the NG interface but we saw that there are also *internal* F1 and F2 interfaces

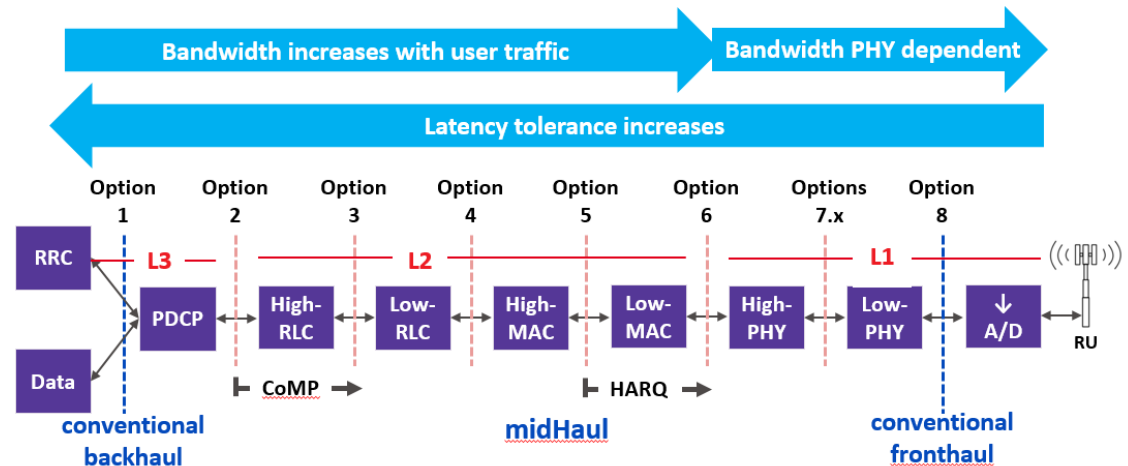


# Functional split options

We already mentioned the various functional split options

Each option presents tradeoffs of

- unit placement – real-estate, computational power, energy requirements
- transport data-rates
- latency
- reliability and resilience
- control/management

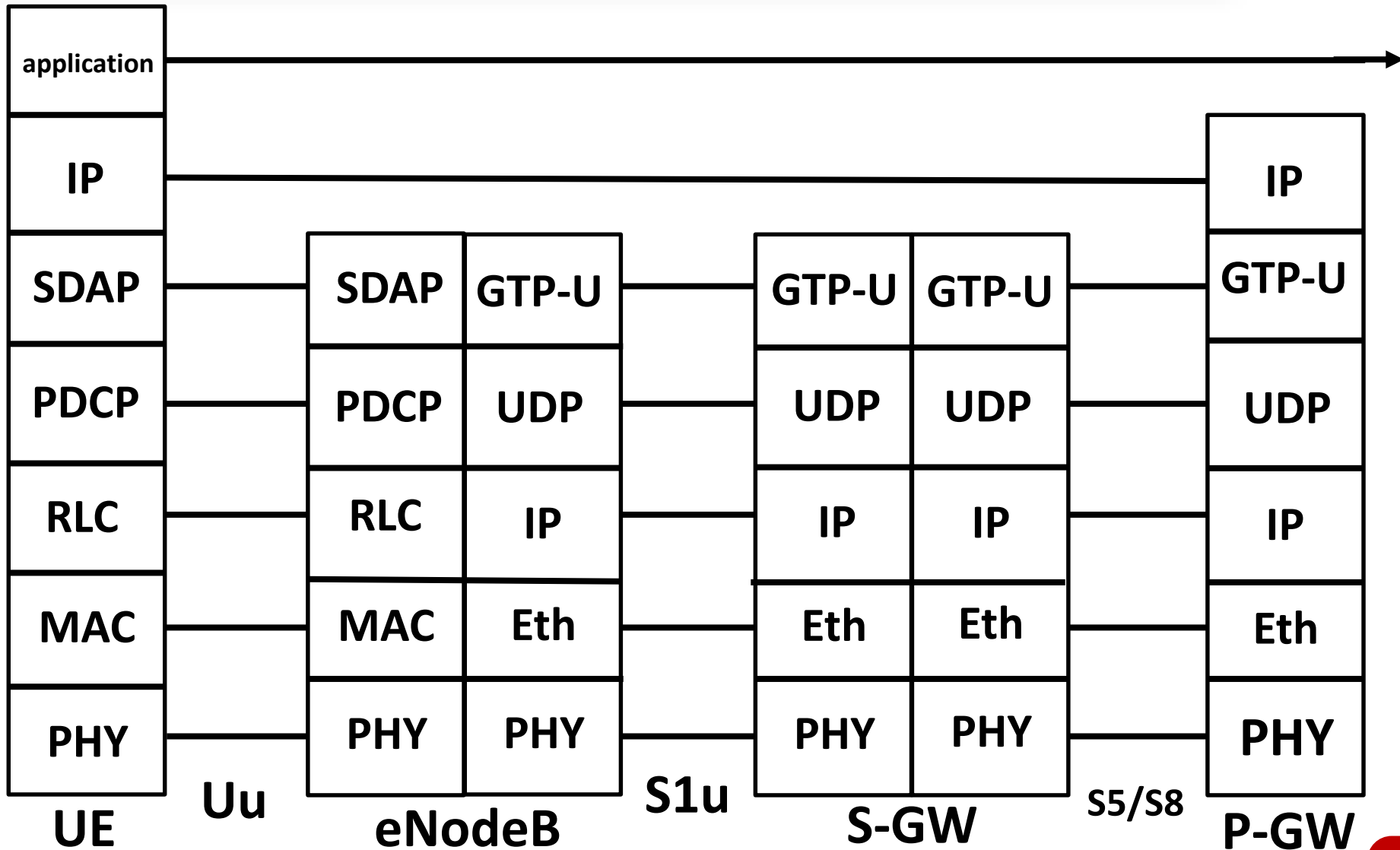


Higher number (lower level) options

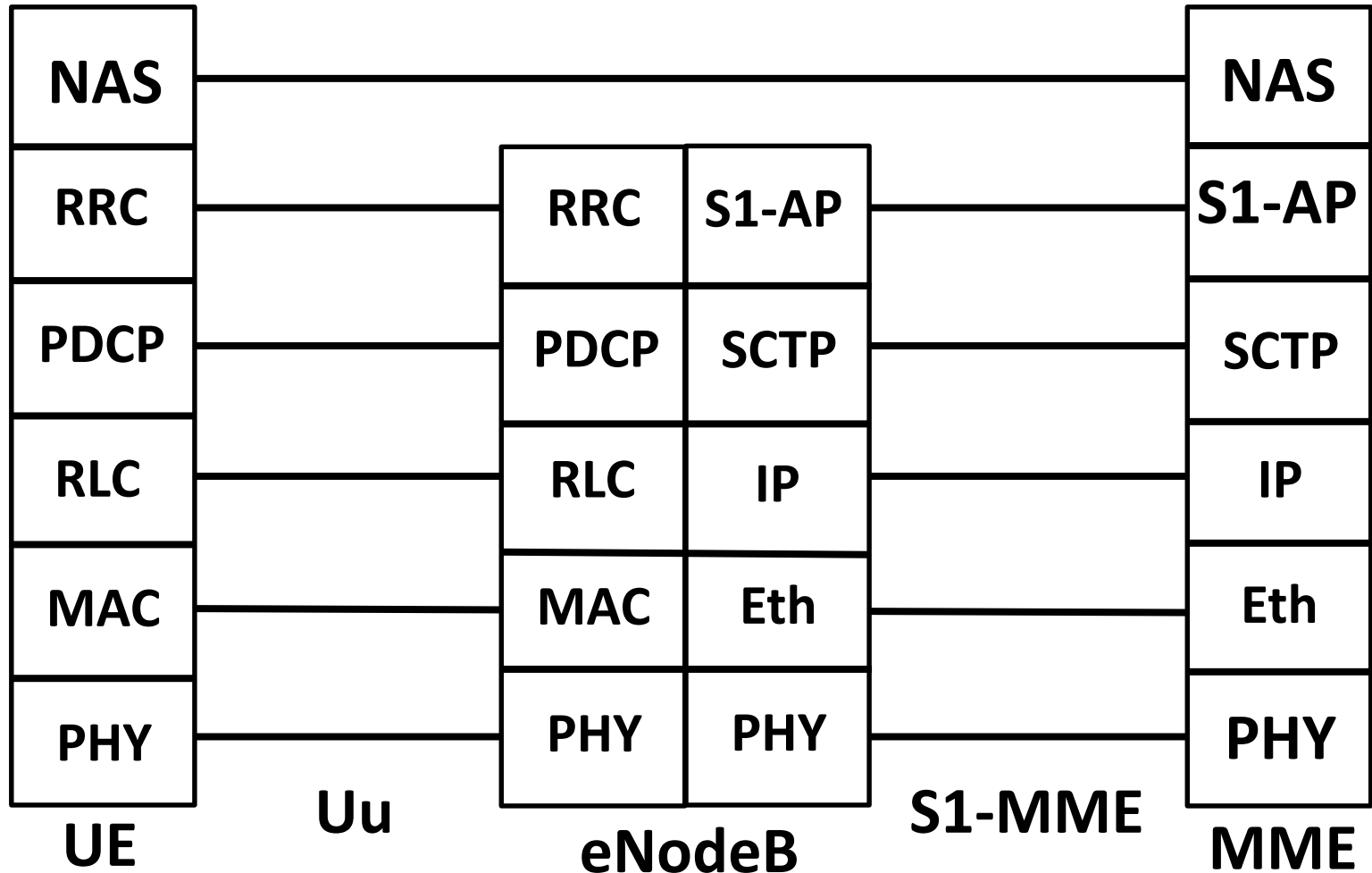
- require higher data-rates (at PHY layer independent of user traffic)
- tolerate less latency

Specific features are located before or after specific split options

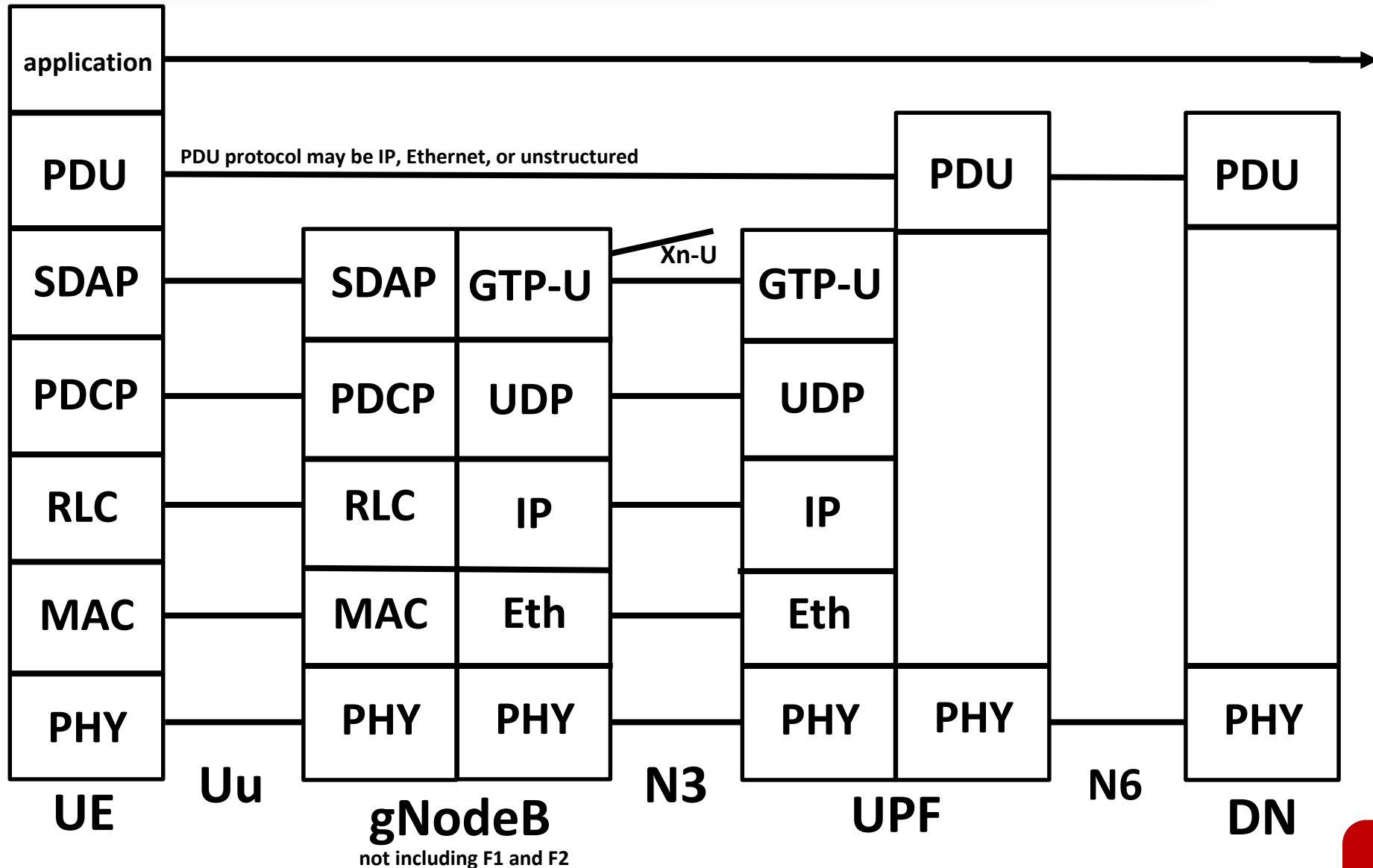
# 4G user plane stack



# 4G control plane stack

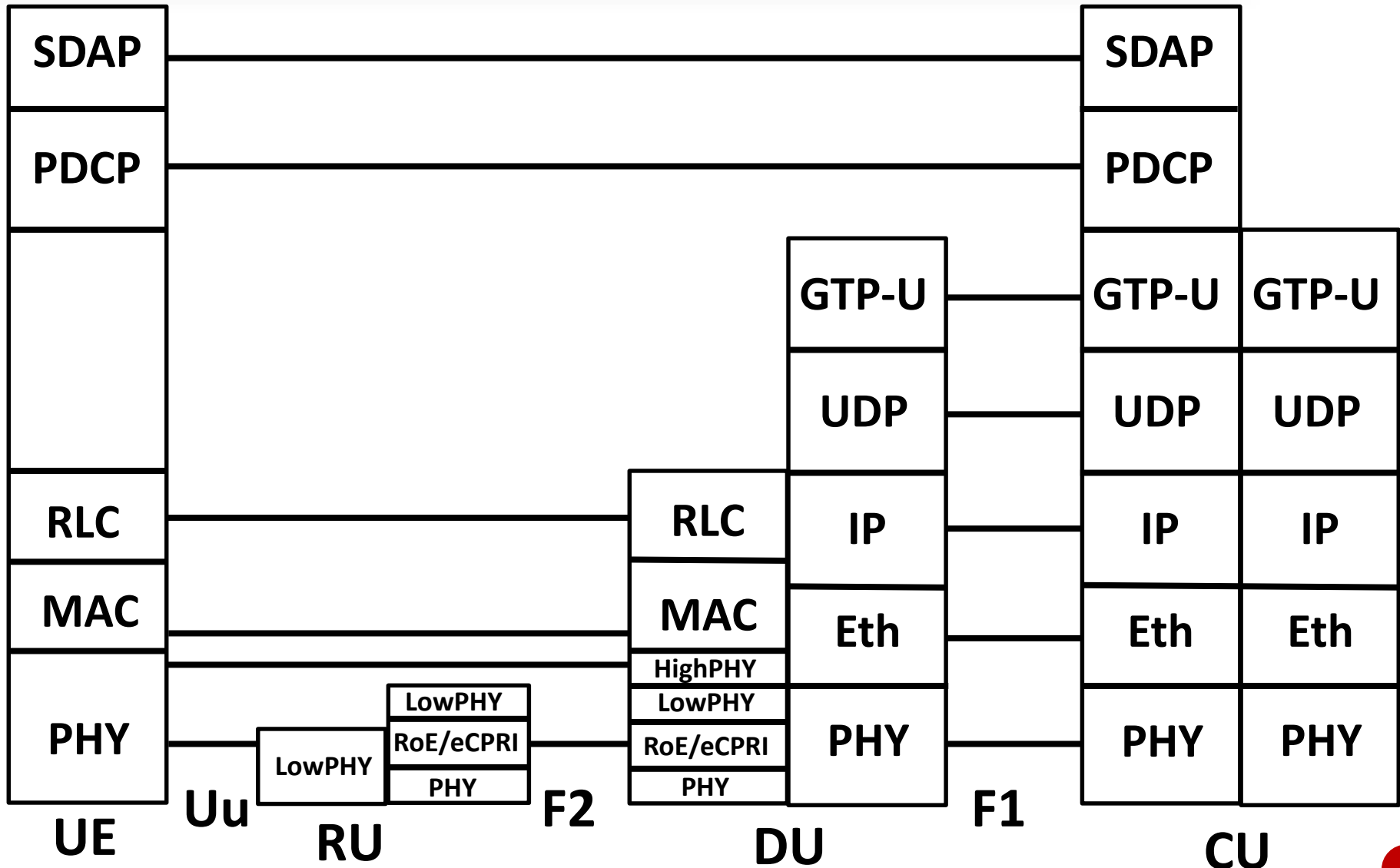


# 5G user plane stack

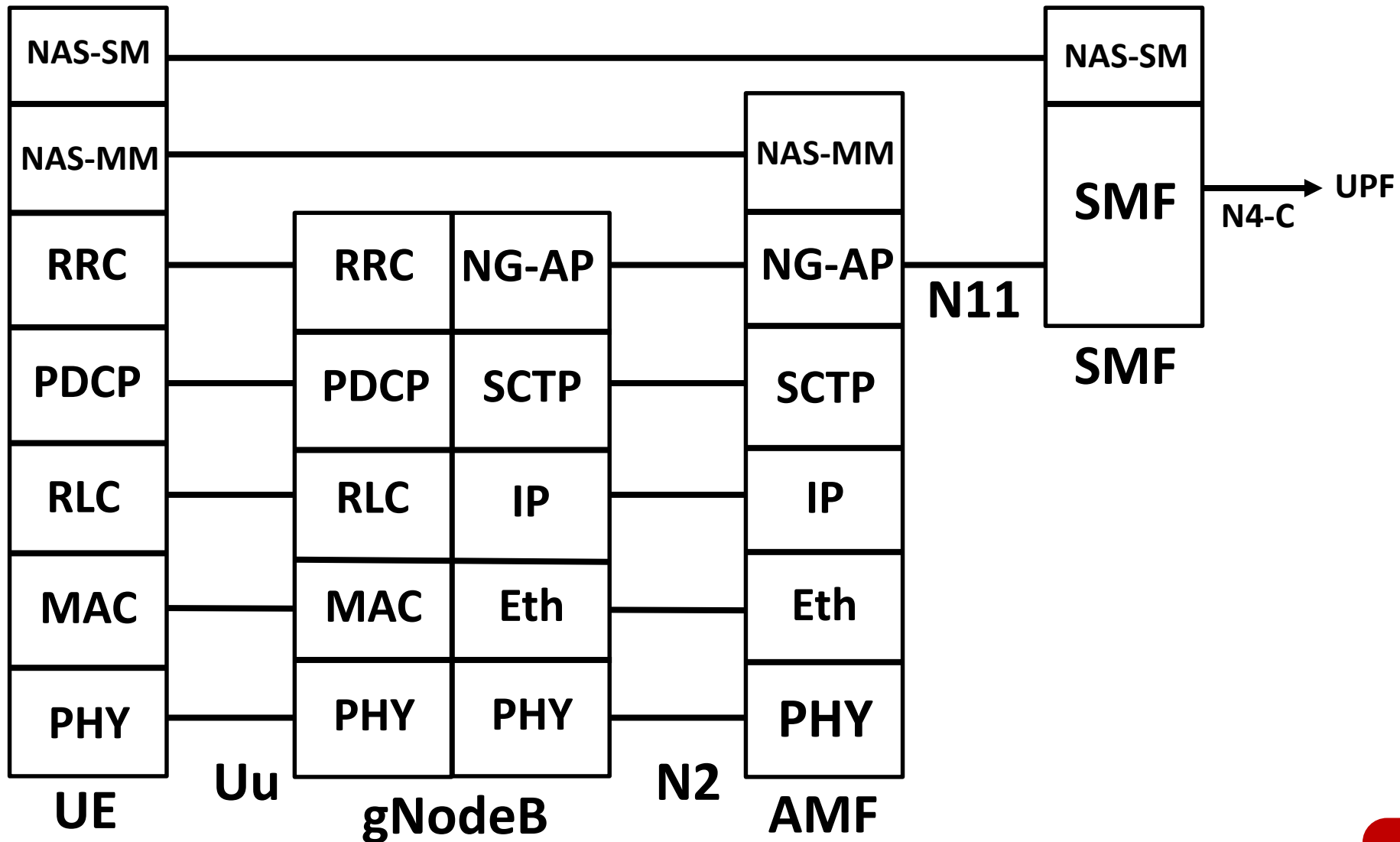




# 5G user plane stack detail



# 5G control plane stack



# Transport challenges

The main challenges for xHaul are:

- **data-rates**
  - F1 and BG estimates are 5 Gbps per cell for initial sub-6 deployments and will increase over time as users desire higher rates
  - F2 rates will start at 25G and can exceed 100 Gbps latency
- URLLC requires low end-end latencies (as low as 1 ms 1-way delay) and xhaul above HARQ point always requires low latencies!
- reliability (including very low **Packet Loss Ratios**)  
URLLC requires >5 nines reliability
- Synchronization
  - frequency accuracies of several ppb are required for FDD
  - time accuracies of 10s of nanoseconds will be required for TDD

# Why very low PLRs?

BE Internet Packet Loss Ratios can approach 1%  
Current cellular air interface induces much worse  
Carrier Ethernet can give PLRs of  $10^{-6}$  !

What applications need any better than that?

Example 1 : industrial failures

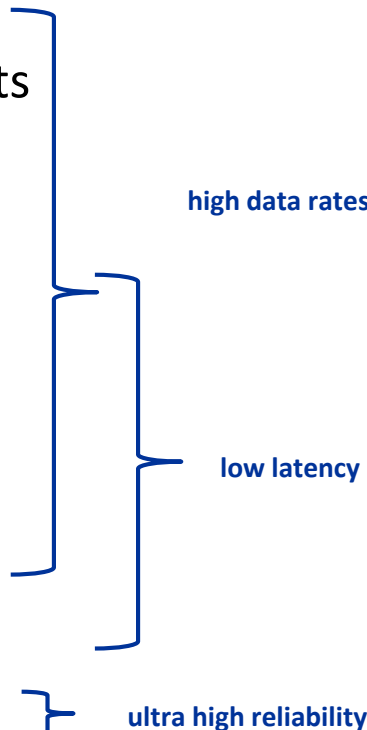
- assume that 2 consecutive packets lost to a machine in a factory may cause the machine to jam, halting production
- GbE may sustain 1.5 Mpps, or about  $10^{11}$  packets per day so a factory with 10 operational networks handles  $10^{12}$  packets per day
- a PLR of  $10^{-6}$  under IID means a 2-packet loss ratio of  $10^{-12}$  so there will be a production halt every day!

Example 2 : audio-video mastering

- mastering an 8K (7680\*4320) at 60 fps requires 3.8GBps = 2.5 Mpps
- a camera to storage PLR of  $10^{-6}$  this means 2.5 packet losses per second!
- since latency rules out TCP, recording is impossible

# Potential 5G RAN transport technologies

Recent transport innovations can assist supporting these requirements

- 10GbE, XGS-PON
    - but 10 Gbps is only satisfactory for initial 5G deployments
  - 25 GbE (802.3by), 1-lane 50 GbE (802.3cd)  
100/200/400 GbE (802.3bs, 802.3ck)
  - FlexE
  - **Mobile (Multi-access) Edge Computing**
  - Synchronization (SyncE, IEEE 1588, DGM)
  - Network slicing
  - **Time Sensitive Networking (and Deterministic Networking)**
  - **Frame Replication and Elimination (IEEE 802.1CB)**
- 
- high data rates
- low latency
- ultra high reliability

# Higher rate physical layers

Ethernet physical layer rates are typically multiples of 10  
10Mbps, 100Mbps, 1Gbps, 10Gbps, 100Gbps, ...

The present 100Gbps standard is based on 4 *lanes*\* of 25 Gbps  
so it is natural to allow the use of a single lane  
for rates higher than 10 Gbps



Single lane 25G has been a standard Ethernet rate since 2016

For yet more flexibility, the FlexE standard enables  $m \times 25G$



IEEE is working on increasing the lane speed to 50G and eventually to 100G  
resulting in 4 lanes with capacity of 200G and eventually 400G  
which FlexE can further bond together

\* depending on standard, a lane may be a pair/fiber or may be a  $\lambda$

# Time Sensitive Networking

TSN and **D**eterministic **N**etworking are Ethernet and IP/MPLS technologies that enable:

- *very low and guaranteed* packet propagation *latency*
  - time aware scheduling/queuing
  - time coordinated forwarding
  - frame preemption
- *very high reliability*
  - zero congestion loss (PLR of  $10^{-10}$ )
  - resource reservation
  - seamless redundancy
- dynamicity – flows can be removed or added w/o impacting other flows
- co-existence of TSN traffic with regular traffic
  - up to 75% express traffic

for applications such as IIoT, V2x, fronthaul

TSN is being developed for Ethernet by a task group in IEEE 802.1

DetNet is being developed for IP and MPLS by the DetNet WG in the IETF



# Some TSN Components (some already in 802.1Q-2018)



## Latency

- 802.1Qav (clause 34) Forwarding and Queuing Enhancements
- 802.1Qbu, 802.3br Frame preemption
- 802.1Qbv Scheduled traffic
- 802.1Qch (Annex Q) Cyclic queuing and forwarding
- 802.1Qcr Asynchronous shaping

## Reliability

- 802.1CB (standalone) Frame Replication and Elimination
- 802.1Qca Path control and Reservation
- 802.1Qci Per-stream filtering and policing
- 802.1AX-rev LAG revision

**we won't discuss *all* of these**

## Resource Management

- 802.1Qat (clauses 34, 35) Stream reservation protocol (SRP)
- 802.1Qcc SRP enhancements and performance improvements
- 802.1Qcw Yang data models for Qbv, Qbu, and Qci
- 802.1CS Link local reservation protocol

## Misc

- 802.1AS Timing (including 1588 profile)
- 802.1CM TSN for mobile fronthaul
- IEEE 1722 Layer 2 Transport Protocol for TSN



# Time Sensitive Networking

TSN and DetNet ask how we can improve network performance if we have highly accurate synchronization (say, better than  $1 \mu\text{sec}$ ) at network elements (Ethernet switches, IP/MPLS routers) ?

We'll see that we can

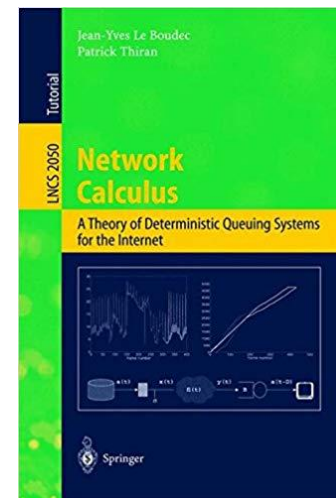
- significantly *reduce* latency
- achieve *bounded* latency

for time sensitive flows

Much of the TSN/DetNet work is based on the *Network Calculus* a mathematical theory dealing with deterministic queuing such as in communications networks

TSN and DetNet support co-existence of sensitive and non-sensitive traffic (sensitive traffic can be up to 75% of the total load)

TSN uses a control protocol (SRP) for configuring switch time behavior



# 802.1Qbu Frame preemption

The major source of residence latency for a high priority packet is its waiting in a queue for a packet being output to complete transmission

For example, assuming a 1500 B packet just started transmission the high priority packet needs to wait:

- 10 Mbps            1.7 msec
- 100 Mbps :        170  $\mu$ sec
- 1 Gbps :            17  $\mu$ sec
- 10 Gbps            1.7  $\mu$ sec
- 100 Gbps          0.17  $\mu$ sec

and the situation will be much worse with 9K jumbo packets which are being standardized

It is possible with present protocols to *run* the outgoing packet but this would require its retransmission and burden switches along the path to parse and discard it

A solution to this problem (not the most important element in TSN!) involves *preemption, frame fragmentation, and local reassembly*

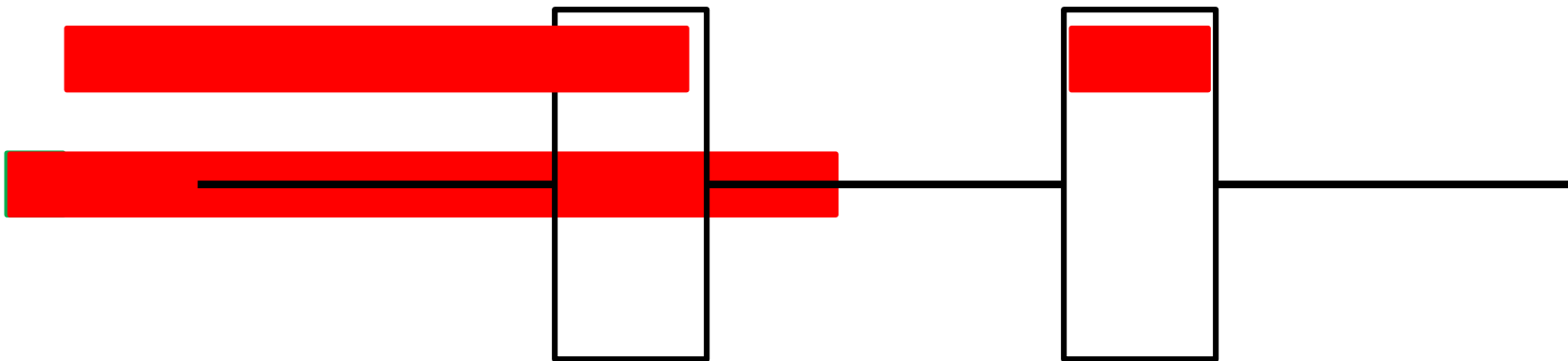
# 802.1Qbu Frame preemption (cont.)

Published in 2016 and being absorbed into 802.1Q

When express frame(s) arrives and a normal frame is being transmitted

- the packet transmission of the normal frame is temporarily suspended
- the *neighboring* switch buffers the content received
- the express frame(s) are sent and forwarded
- the transmission of the normal frame is continued
- the neighboring switch reassembles the outgoing packet and forwards

Optimal bandwidth utilization of background traffic for time aware shaping and low-latency communication in non-scheduled networks



# 802.3br Interspersing Express Traffic (IET)

802.3br provides Ethernet physical layer support for frame preemption

All frames are classified as either *express* or *preemptable*

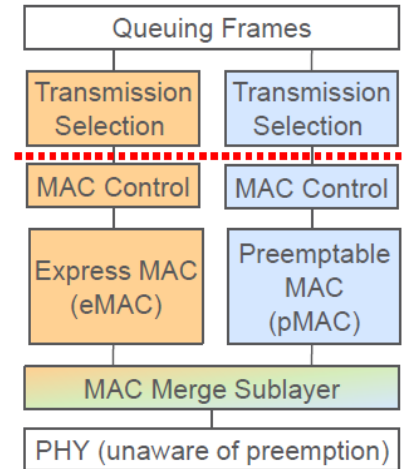
Frame could be preempted multiple times  
but no nested preemption

IET support is discovered via LLDP (new TLV)

Express frames have the usual preamble  
but a special physical start-of-frame  
SMD-e = 55 instead of SFD = AB

Frame fragments

- are at least 64 bytes (multiple of 8 except last fragment)
- have usual preamble but special start-of-frame
  - first fragment has SMD-lx = 66 CC FF 33 (for frame 0 1 2 3)
  - non-initial fragments have SDM-Cx = E1 D2 1E 2D + fragment ctr
- have a 0..3 fragment counter
- have their own (modified) FCS
- are not valid MAC frames for non-compliant devices



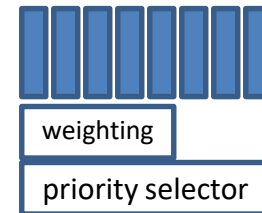
802.1Qbu  
802.3br

# 802.1 Queuing

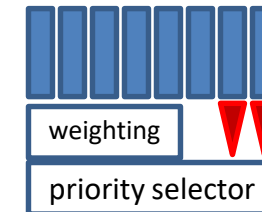
802.1Q-1988 defined *strict priority* selection



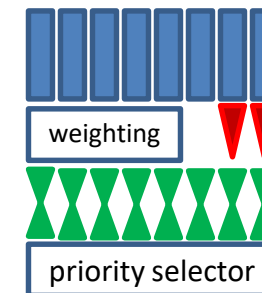
802.1Qaz (absorbed into 802.1Q-2012) added *weighted queues*



802.1Qat adds *credit-based* fair queuing (borrowed from RPR) transmit frames when non-negative *credit* credit increases when frames wait



802.1Qbv adds *time-sensitive* queues queues are Open or Closed according to *timeslots*



# 802.1Qat Credit Based Queuing

802.1Qat adds credit-based fair queuing (borrowed from RPR)

- a Credit Based Shaper (CBS) spaces out frames to reduce bursting
- a frame is only transmitted when its queue has non-negative credit
- credit
  - increases at rate `idleSlope` when frames are waiting or no frames waiting but credit is negative
  - decreases at rate `sendSlope` when frames are transmitting
  - rates are adjustable (per queue results in weighted queuing behavior)
- shaped queues have highest priority (higher than unshaped queues)
- Qat still guarantees bandwidth to the highest unshaped priority
- CBS is similar to burst rate shaper but with useful mathematical properties
  - only parameter is bandwidth
  - impact on queue of adjacent shapers = impact of 1 shaper with total BW

# 802.1Qbv Scheduled Traffic

Published in 2015 and being absorbed into 802.1Q

Qbv introduces Time Aware Traffic Shaping (Time Sensitive Queues)

- requires that every network element has highly accurate time (e.g., from 1588)
- time-gated egress CoS queues transmit one at a time based on a precise *timeslot* schedule
- implemented by circular collection of *time aware gates*

Directly timing release of packets can

support *scheduled* applications (e.g., process/vehicle control)

provide latency and PDV guarantees

completely avoid congestion

return to TDM-like determinism

Qbv retains credit-based shapers for *non-scheduled* applications

# Time Sensitive Queues

In Qbv **all** CoS queues (not just the TSN queues) are cyclically gated so non-TSN traffic is also released in timeslots

Qbv generally results in bursts of packets belonging to a stream

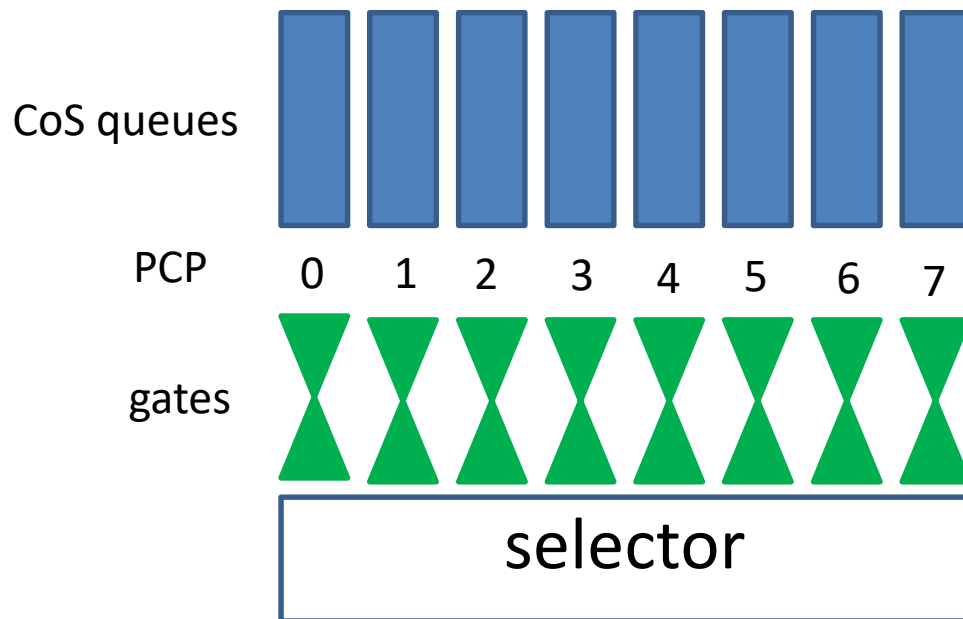
Timeslot schedules are dynamically computed by a centralized management system that configures the network nodes using SRP

Schedules are specified with up to 1 ns granularity (although implementations may be less precise) thus PDV can be reduced to about 1 ns

Schedules might require guard times between TSN timeslots unless preemption is used



# Qbv Scheduler



Gate Schedule

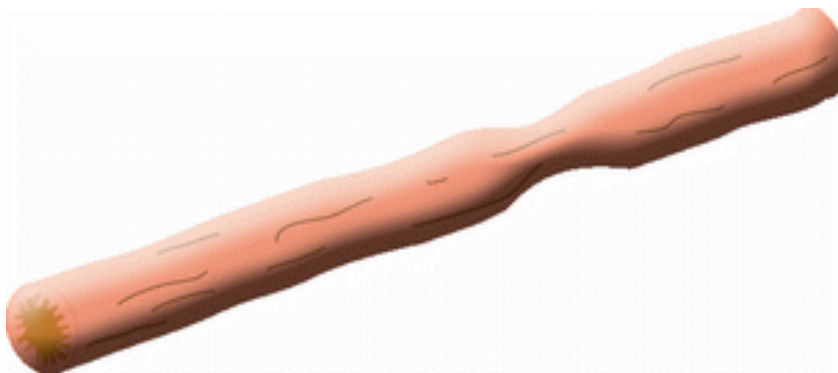
T	0	1	2	3	4	5	6	7
0	O	C	O	O	C	O	O	O
1	C	O	O	O	O	O	C	C
2	O	C	C	C	C	O	O	O
3	O	O	O	O	C	C	C	O
N	<i>repeat</i>							

# Peristaltic Queuing

What can we do with Qbv time-gated queues?

Knowing in detail each scheduled flow's behavior  
we could configure every NE in the entire network  
to be free precisely when the next scheduled packet arrives  
(this can be done using 802.1Qav)

But there is a simpler (but less optimal) method  
called peristaltic queuing



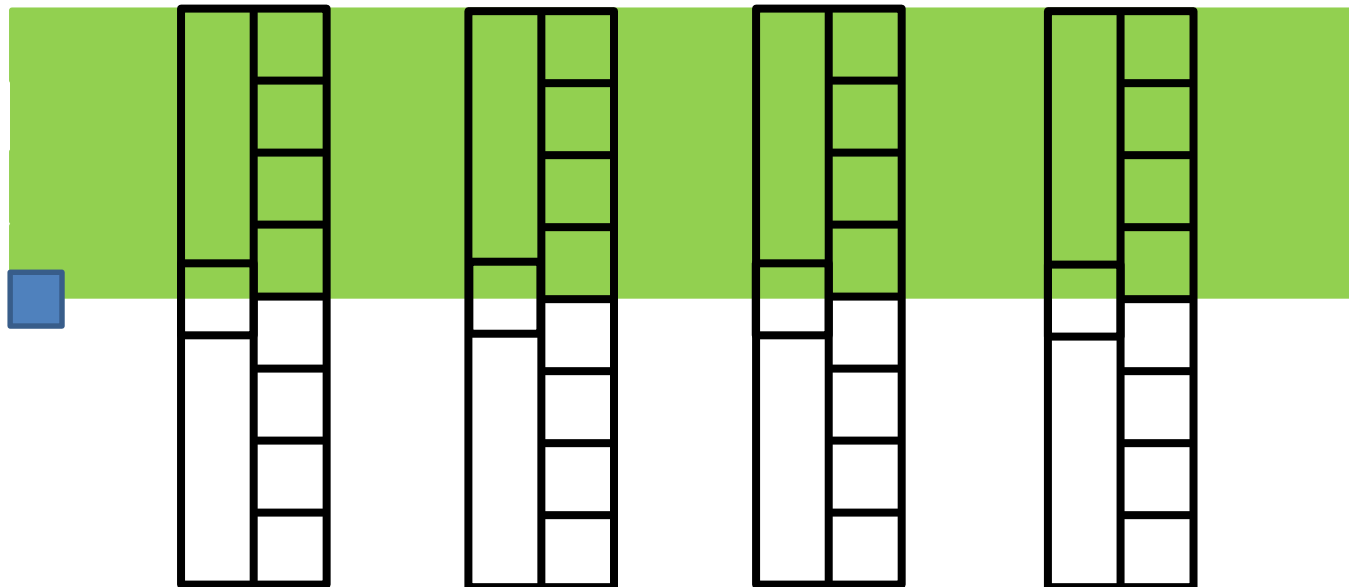
# 802.1Qci/Qch Cycling Queuing

802.1Q-2018 Annex Q (CQF, formerly called peristaltic shaping)

Exploits accurate timing to provide without intricate signaling defined (*nonoptimal*) upper latency bound

Each switch collects frames according to PCP and forwards all packets of the same traffic class in *one cycle*

Maximum bytes allowed per clock cycle configured by SRP



# Stream Reservation Protocol

**SRP** reserves network resources for flows (similar to RSVP-TE) and uses a Traffic Specification (Tspec)

802.1Qat published in 2010, absorbed into 802.1Q-2011

SRP is used to configure all of the TSN features

SRP specifies the required resources and enables their dynamic maintenance and facilitates registration, deregistration, and retention of resource reservation information in relevant network elements

- *Talker Declarations* are distributed along the path(s) to *Listeners*
- *Listener Declarations* run back and actually reserve resources

802.1Qcc proposes enhancements to SRP as defined by Qat

- support for 1000s of streams
- configurable SR (stream reservation) classes and streams
- reduce chattiness and CPU load for handling SRP
- supports L2, but probably won't be used for L3 (DetNet)

# 802.1Qav FQTSS

## Forwarding and Queuing Enhancements for Time-Sensitive Streams

Approved by AVB TG in 2009 and is now clause 35 of 802.1Q-2014

- planning and admission control (CAC) based on SRP
- procedures for mapping priorities to traffic classes
  - class A - latency  $\leq 2$  msec
  - class B - latency  $\leq 50$  msec
  - control traffic (AS and SRP)
  - best effort traffic
- precise synchronization based on 802.1AS
- CBS shaping of streams at source based on 802.1Qat
- identification of non-participating devices

# 802.1CM TSN for Fronthaul

Published in 2018, new version 2020

Joint work with the *CPRI Cooperation*

Specifies requirements for transporting time-sensitive fronthaul streams over Ethernet networks

including delay and sync (with calculations)

Specified for CPRI (*class 1*) [encapsulation not specified] and eCPRI (*class 2*)

CPRI flows (IQ data, C&M control/management, sync) are handled as separate flows

Synchronization via SyncE and ITU-T telecom profile (BC or TC)

Adopts eCPRI's *categories* (A+, A, B, C) and specifies 2 profiles

# 802.1CM Profiles

## Profile A

- use legacy Ethernet equipment
- strict priority Ethernet bridge
  - IQ data with high priority traffic class
  - C&M data with lower priority
- use 2000 bytes for all frames
- MEF 10.3 shaping

## Profile B

- assume *very minimal* TSN functionality is present
- Qbu+3br preemption enabled bridge
  - IQ data is express traffic
  - C&M data is preemptable
- 2000 bytes frames for IQ frames, flexible for others
- MEF 10.3 shaping

# Mobile Edge Computing

**Mobile (Multiaccess) Edge Computing** offers another solution to both data-rate and latency requirements

MEC enables terminating traffic close to the gNB or first aggregation nodes rather than backhauling all the way to the core

By providing processing power close to the UE network congestion and latency are both reduced

Some MEC applications

- Internet breakout
- DNS caching
- Content Delivery Networks
- mobile big data analytics
- fog networking (IoT processing)
- connected car (V2x)

MEC concepts have been absorbed into 5G's **Service Based Architecture**



# Slicing in the 5G xHaul network

5G can support coexistence of traffic with wildly diverging requirements by using *network slicing* in the air interface, xhaul, and core

- *on-demand* assignment of networking/computational resources
  - bandwidth, forwarding tables, processing capacity, etc.
- resources can be physical or virtual, but
- each slice acts as a strongly isolated network or cloud
  - isolation of management, security, and performance

3GPP defines a slice by

- Slice/Service type (SST) expected slice behaviour (features and services)
- Slice Differentiator (SD) optional information to differentiate between slices of the same Slice/Service type

Slice/Service type	SST value	Characteristics.
eMBB	1	slice suitable for the handling of 5G enhanced Mobile Broadband.
URLLC	2	slice suitable for the handling of ultra- reliable low latency communications.
MIoT	3	slice suitable for the handling of massive IoT.

# Time/Frequency Synchronization

Frequency and time accuracy requirements are defined to assure efficient and proper functioning of the air interface

We saw how the UE acquires sync from the air interface here we need to explain how the gNodeB acquires sync

RAN timing requirements are becoming stricter from generation to generation

Base stations obtain timing from the RAN (unless they have a local source of timing, e.g., GNSS)

5G requirements will be at least as strict as 4G especially in TDD modes and when location based services are used

- frequency accuracy must be better than  $\pm 50$  ppb
- time error must be several 100s of nanoseconds

So the 5G RAN must deliver ever more accurate timing!

How do we deliver sync over an asynchronous packet network ?



# Time/Frequency Synchronization

There are several important technologies:

- GNSS (GPS) timing recovery at each cell site
- SyncE – frequency synchronization of Ethernet physical layer  
recently ITU has specified **enhanced SyncE**
- 1588 (**P**acket **T**ime **P**rotocol) – time of day distribution over Ethernet or IP
  - IEEE 802 AVB WG has produced an AVB profile called 802.1AS
  - ITU-T SG15/Q13 has produced telco profiles G.8265.1, G.8275.1/2
  - new accuracy levels have been defined
- Distributed Grand Master – 1588 GMs close to the edge

# SyncE

Circuit switched Constant Bit Rate networks (e.g., TDM, SDH, OTN)  
natively distribute frequency

since the bit-rate itself is locked to a reference frequency source  
and the receiver *must* lock onto the bit-rate to correctly recover the bits

Packet Switched networks are asynchronous and do not distribute frequency

Ethernet in particular uses very low quality crystals ( $\pm 100$  ppm)

However, modern Ethernet physical layers do continuously transmit bits  
and send idle codes when there is no information to send

Synchronous Ethernet (SyncE) utilizes this fact

- source is locked to a high quality frequency reference
- each switch locks onto its input signal and recovers its clock
- each switch uses the recovered clock to forward the data

SyncE is a **physical layer** frequency distribution mechanism

it requires special hardware

but puts no further demands on (BW or computation) resources

# IEEE 1588v2 (Packet Time Protocol)

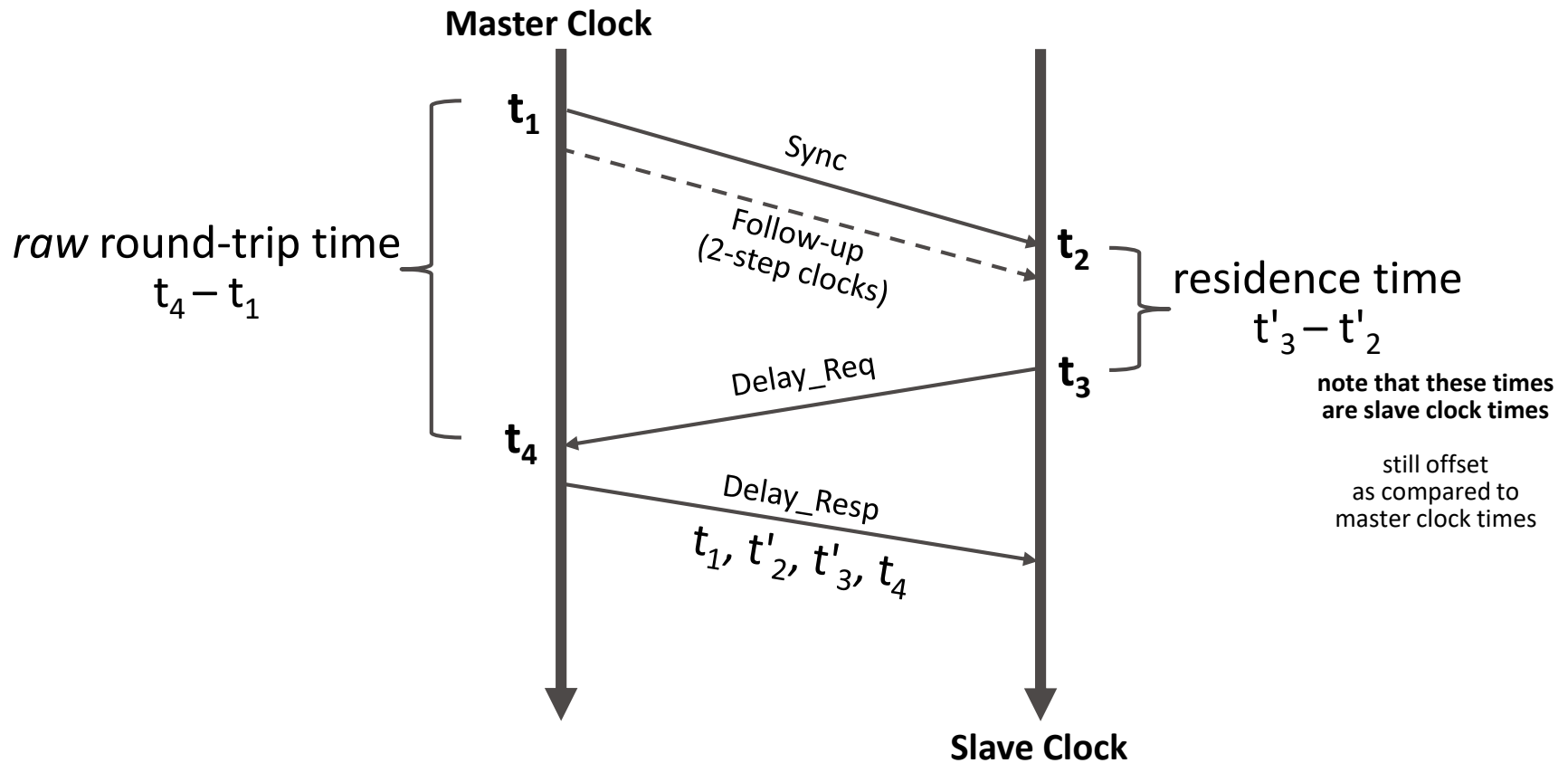
IEEE 1588v1 provided time distribution over packet networks (Ethernet, IP) for the measurement and control applications

IEEE 1588v2 is designed to distribute time over packet networks to the scale of nanoseconds and even picoseconds

Why can 1588 be much more accurate than **Network Time Protocol** (which provides accuracies of 10s of ms) ?

- hardware timestamping (half-way up the rising edge of the first bit)
- unlike NTP's client server architecture (memoryless server)
  - PTP is master-slave (master sending frequent sync updates to slaves)
- PTP separates sync messages from delay-measurement messages
- on-path support elements
  - **Transparent Clocks** (accumulates NE residence times)
  - **Boundary Clocks** (slave to higher level and master to lower level)
  - optional peer-to-peer TC mode completely compensates for PDV
- can be combined with SyncE

# PTP Operation



$$T_{\text{delay}} = \left( (t_4 - t_1) - (t'_3 - t'_2) \right) / 2$$

# Problems with PTP

Our previous slide assumed :

- small frequency difference between slave and master clocks  
so that differences in slave timescales are the same as in the master timescale
- single link between master and slave (no **P**acket **D**elay **V**ariation)  
we can overcome this by *minimum-gating*  
accepting only packets with minimal delay  
the quality of minimum-gating depends on **F**loor **P**acket **P**ercentage
- *symmetry* between directions (at least on average)  
this is not the case if switches/routers have direction-dependent loading  
but known physical asymmetry (e.g., due to path or rate asymmetry)  
can be compensated

However, 1588 has a few more tricks!

# Boundary Clock

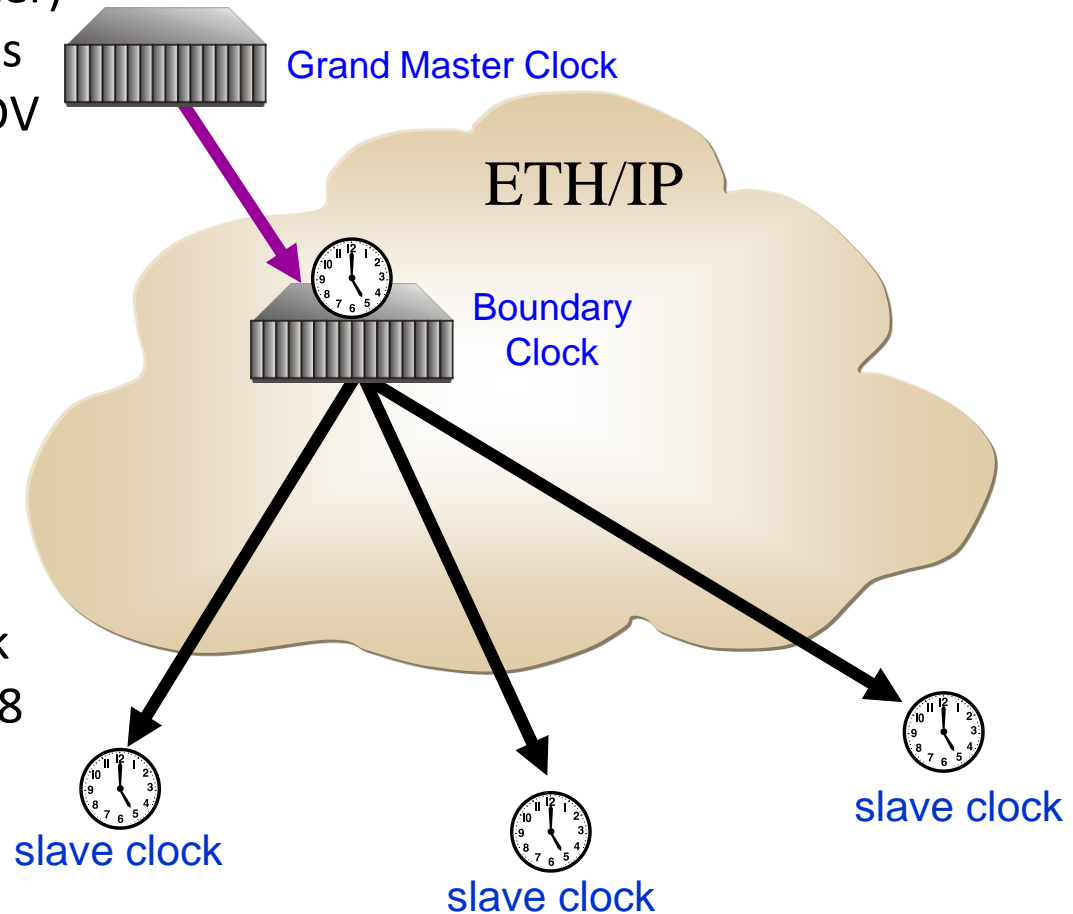
Each intermediate switch (router) adds variable residence times resulting in uncompensated PDV

More switches result in higher PDV

A BC's is like back-to-back slave clock and master clock

Once it stabilizes its own clock it can become a master clock and distribute its own 1588

Between the BC and slave there are fewer switches



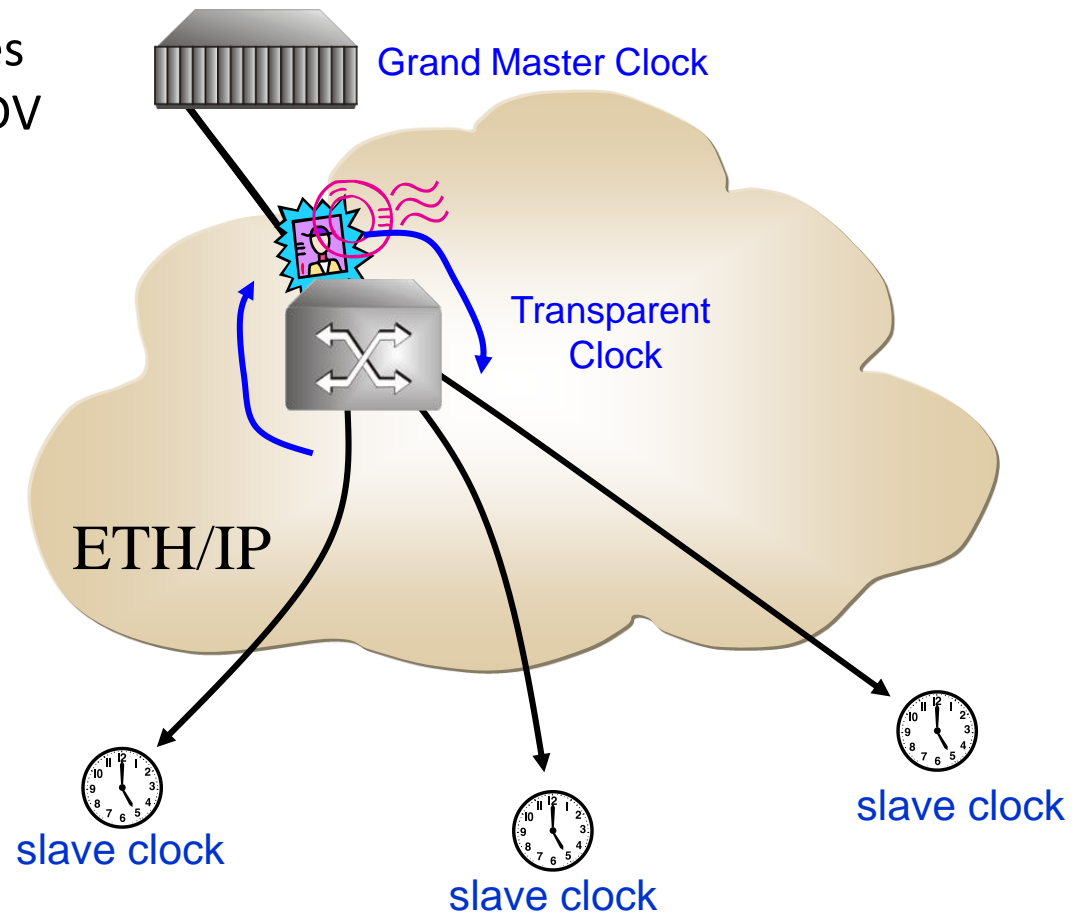


# Transparent Clock

Each intermediate switch  
adds variable residence times  
resulting in uncompensated PDV

TC accumulates these  
enabling their subtraction

Peer-to-peer TC also collects  
physical propagation times  
If done correctly  
total propagation time  
is removed



# Combining SyncE with 1588

Conventional 1588 slave clocks

stabilize their local oscillator frequency based on Sync messages

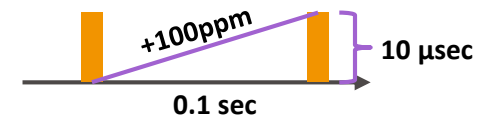
So that they can interpolate well between time updates

and extrapolate (hold-over) if delay requests become stale

However, Sync messages can not be too frequent (say 10-100 per second)

and between them phase error can accumulate

Example 100 ppm error with 10 pps can accumulate 10  $\mu$ sec error



But SyncE is *continuously* updating its frequency

so we can stabilize frequency with SyncE

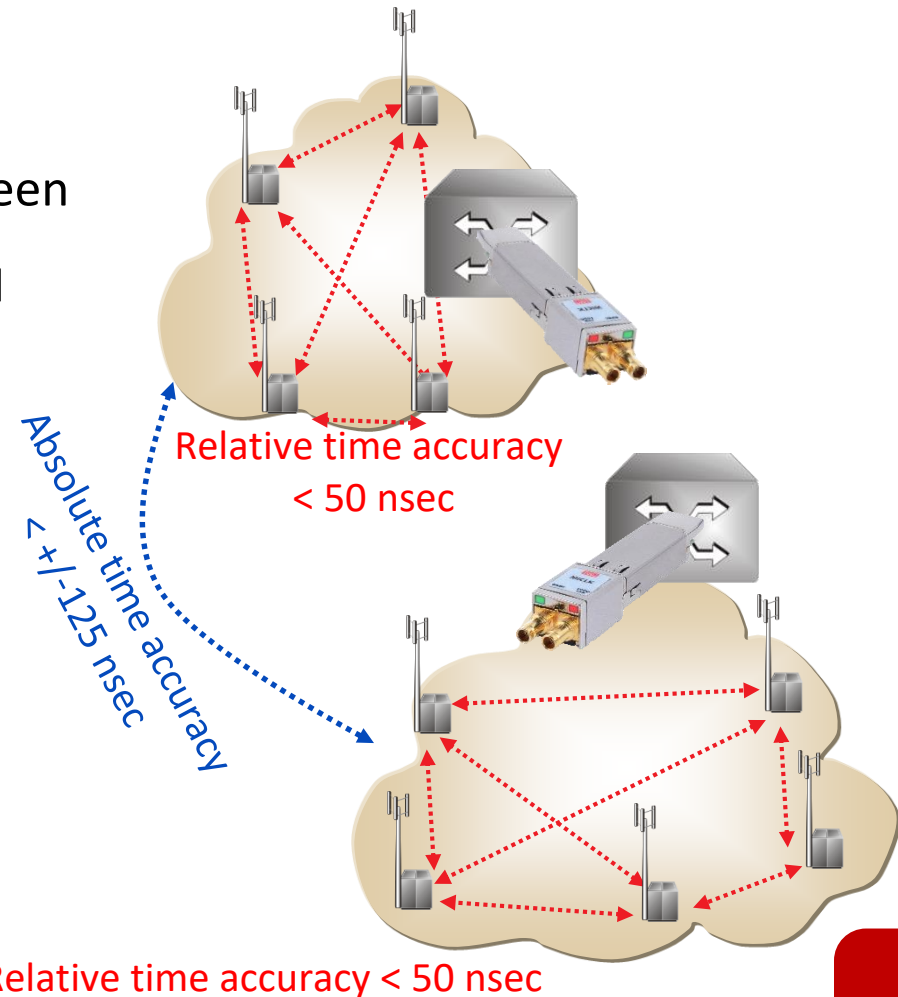
and only infrequently request time updates from 1588

# Distributed grandmaster

Instead of BCs we can distribute 1588 grandmasters that receive their timing from some other source, e.g., GNSS (GPS)

Miniature (pluggable) grandmasters can feed multiple cell sites maintaining 50 nsec accuracy between

Between sites not fed by the same DGM GPS recovery accuracy is attained



# Resilience

Multiple technologies can be used to provide high availability, including:

- Worker/Standby LAG
- BFD with Interior routing protocol mechanisms
- MPLS **F**ast **R**eRoute
- **L**oop **F**ree **A**lternates (IP **F**ast **R**eRoute, **T**opology Independent **L**FA)
- **A**utomatic **P**rotection **S**witching (many types)
- **F**rame **R**eplication and **E**limination

We'll discuss each of these briefly ...

# Worker/Standby LAG

Ethernet “link aggregation” (AKA bonding, Ethernet trunk, inverse mux, NIC teaming) enables bonding several ports together as single uplink

Defined by 802.3ad task force and folded into 802.3-2000 as clause 43

Multiple physical ports are bound into a **Link Aggregation Group** which acts as a logical port with aggregate data rate

The **Link Aggregation Control Protocol** (slow protocol - up to 5 per second) can dynamically add/remove to/from the LAG and continuously monitors to detect if changes needed

LAG can be used as a primitive protection mechanism by simply mapping all traffic onto the *worker* link and if LACP detects a failure remapping onto the *standby* link

When used this way it is called *worker/standby* LAG

The restoration time will be on the order of 1 second (3 lost packets at 5 per second is already 600 msec)

# Bidirectional Forwarding Detection

**Bidirectional Forwarding Detection** (RFC 5880) is a basic OAM protocol used to detect faults between two routers (IP, MPLS)

When BFD detects a link failure the interior routing protocol (IGP) is informed

BFD has several modes

- in *asynchronous mode* the routers send BFD packets to one another

- if a number of consecutive packets are lost a fault is declared

When *echo function* is active the receiving router performs loopback as well

The IGP will then send a link-down message to its peers so that these will change their forwarding behavior

BFD is typically configured for 10 – 50 pps

so that detection times are of the order of 100 msec

# MPLS fast reroute

RSVP-TE enables MPLS traffic engineering by fine control over placement  
specifies explicit path using information gathered from IGP  
resources may be reserved at LSRs along the way

RFC 4090 defines extensions to RSVP-TE – **Fast ReRoute (FRR)**

LSRs along the path preconfigure local bypasses (detours)

Upon detection of failure by

- BFD (specified in microseconds, typically 10s of ms) or
- RSVP hellos (RFC default is 5 ms) or
- RESV / PATH messages (driven by IGP)

upstream LSR simply enables the detour

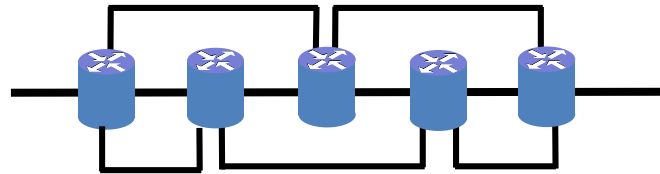
Since this is a local action, it should be fast

RFC 4090 only discusses adding FRR to RSVP-TE network

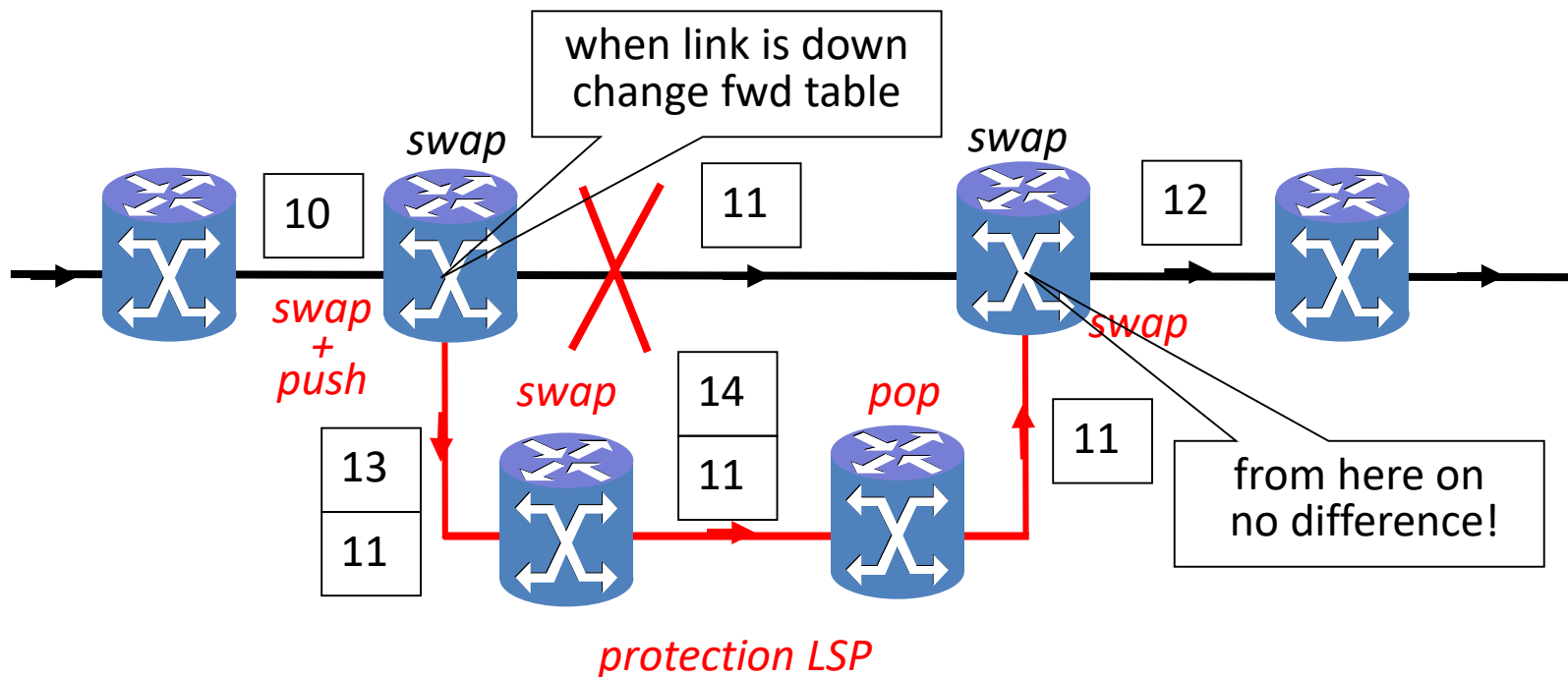
but its use with LDP is possible if there is a single label generator

# MPLS Fast ReRoute

FRR is a *local detour* method (not an end-to-end APS method)



to reroute quickly we *pre-prepare* labels for the bypass links





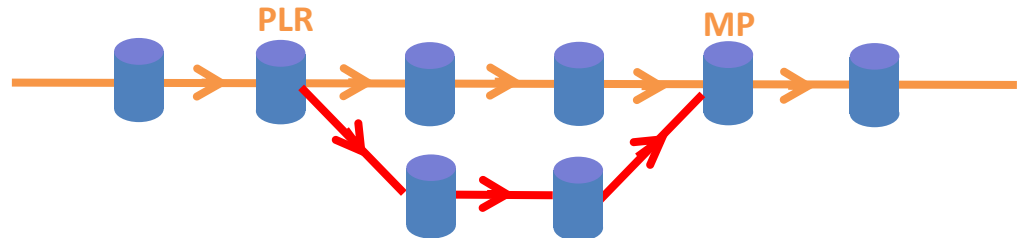
# Methods

RFC 4090 defines two different protection methods

Usually one *or* the other is employed in a given network

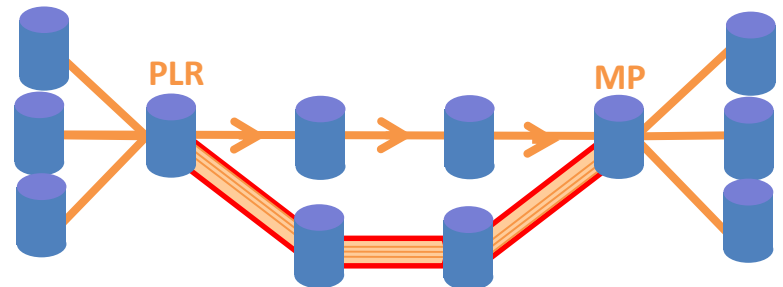
## One-to-one backup

- each LSP protected separately
- detour LSP created for each LSP at each potential PLR
- no labels pushed



## Facility backup

- backup tunnel for multiple LSPs
- bypass tunnel created at each potential PLR
- uses label stacking

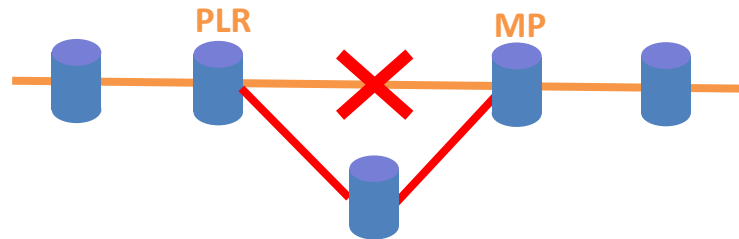


# NHOP and NNHOP

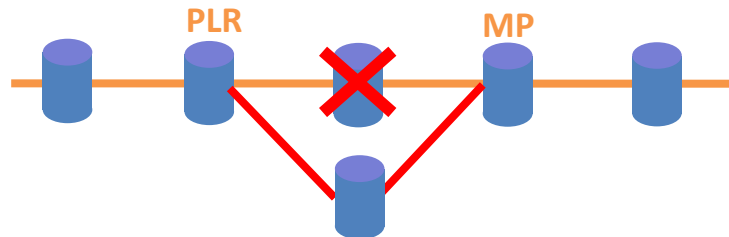
MPLS FRR can bypass a failed link or a failed node

In order to bypass a single failed link

we need an alternative path to the next hop (NHOP)



In order to bypass a single failed node, we need an alternative path to the next next hop (NNHOP)



# LFA

LFA FRR is another *precomputed path* Fast ReRoute mechanism

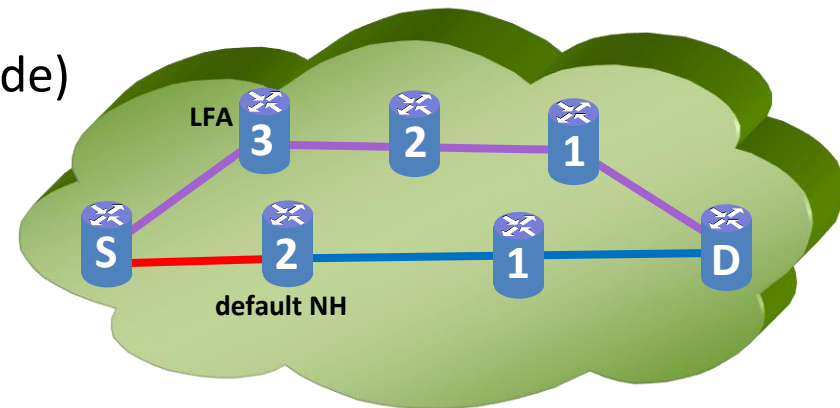
- works with plain IP or MPLS
- exploits an *alternative* to the default next hop
- the alternative forwards to the destination (in any case) without passing through the failed element (loop free)

A **Loop Free Alternative**

- with respect to an element (link/node)
- for a destination

is a router/LSR that

- is not the default next hop
- is connected to the destination
- does not forward through the element hence does not need to know about the failure)

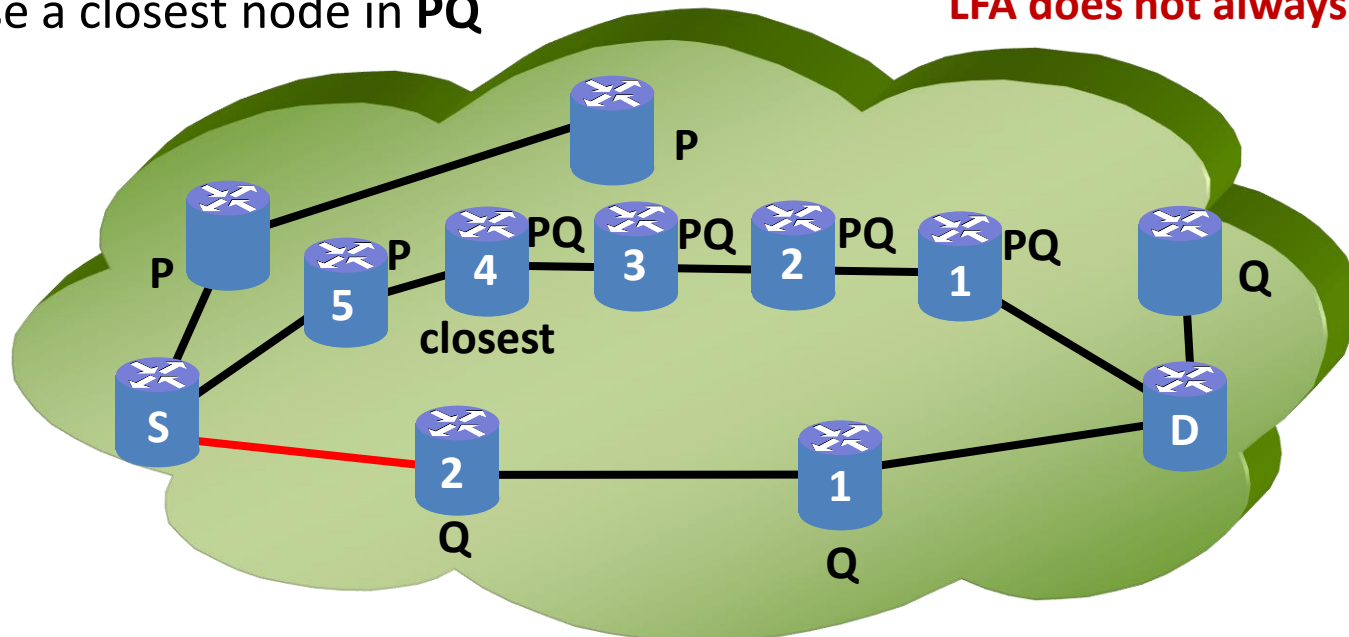


# Finding LFAs

In order to a loop free alternative node

1. find the set **P**  
all nodes still connected to the source in the event of failure
2. find the set **Q**  
all the nodes that forward to the destination without failure
3. find the set **PQ** = the intersection of **P** and **Q**
4. choose a closest node in **PQ**

**Note that PQ may be empty  
LFA does not always succeed!**



# TI-LFA

TI-LFA exploits MPLS Segment Routing  
to avoid having to open targeted LDP sessions

The source LSR knows all the labels from the IGP  
and can build the MPLS SR label stack accordingly  
using any PQ node

This capability is Topology Independent in the sense that  
a loop free backup path is found  
irrespective of the topologies before and after the failure

Using deeper label stacks affords more flexibility

Immediately upon discovering the failure  
the source LSR can use the new SR label stack  
so the protection switch time is minimal

## Automatic\* Protection Switching (APS)

is a functionality of carrier-grade transport networks such as SDH, Ethernet, MPLS-TP

APS includes :

- detection of *failures* (signal fail/degrade) on a *working channel*
- switching traffic *transmission* to a *protection channel*
- selecting traffic *reception* from the protection channel
- (optionally) reverting back to the working channel once failure is repaired

\**Automatic* means uses (at most) *control plane* protocols  
– no *management layer* or *manual operations* needed

# Protection types

We distinguish several different protection *types*

- 1+1
- 1:1
- 1:n
- m:n
- $(1:1)^n$

Each type has its applicability, advantages, and disadvantages and there are trade-offs between

- simplicity
- BW consumption
- protection switch time
- signaling requirements

# 1+1 protection

Simplest and fastest form of protection

but wasteful - only 50% of actual physical capacity is used

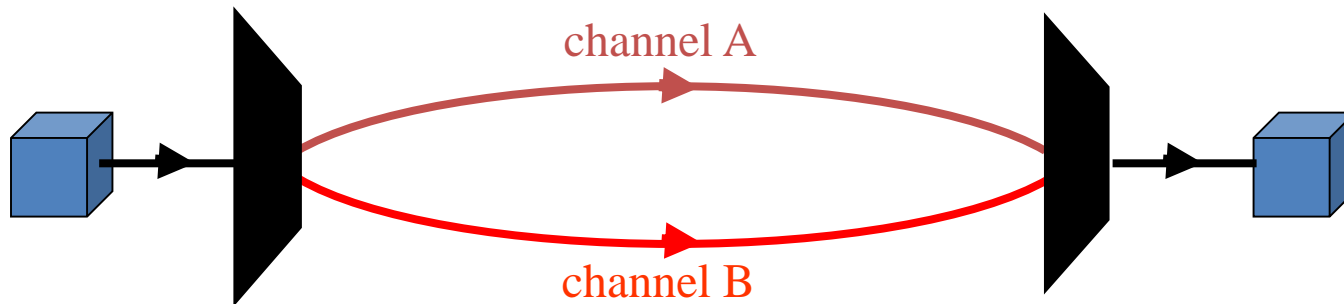
Head-end *bridge* always sends data on both channels

Tail-end *selector* chooses channel to use (based on BER, dLOS, etc.)

For unidirectional 1+1 switching there is no need for APS signaling

If non-revertive

there is no distinction between working and protection channels





# 1:1 protection

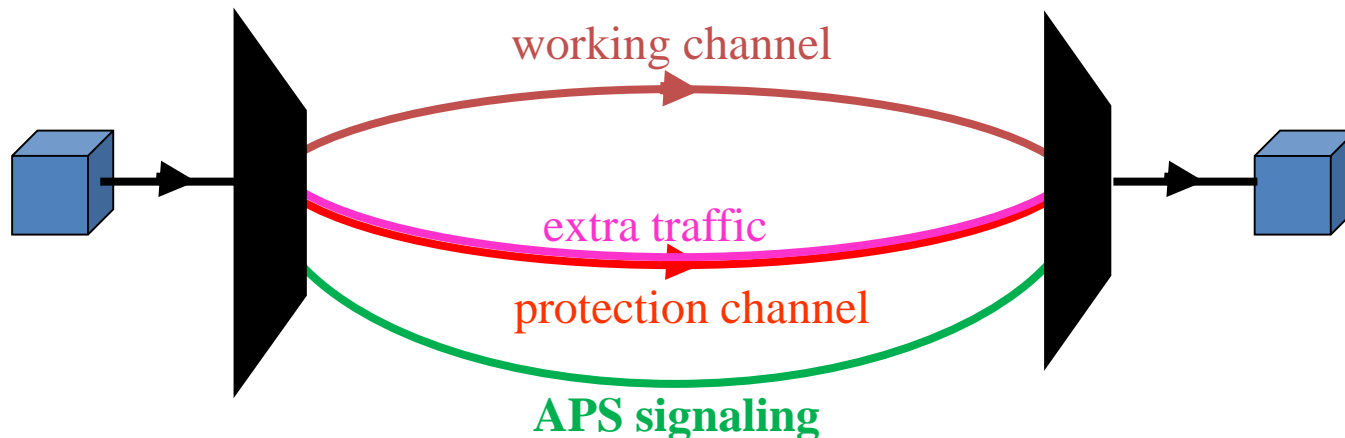
Head-end bridge usually sends data on working channel

When failure detected it starts sending data over protection channel and tail-end needs to select the protection channel

When not in use, protection channel can be used for extra traffic

However, since failure is detected by tail-end, APS signaling is needed

Protection channel should have OAM running to ensure its functionality



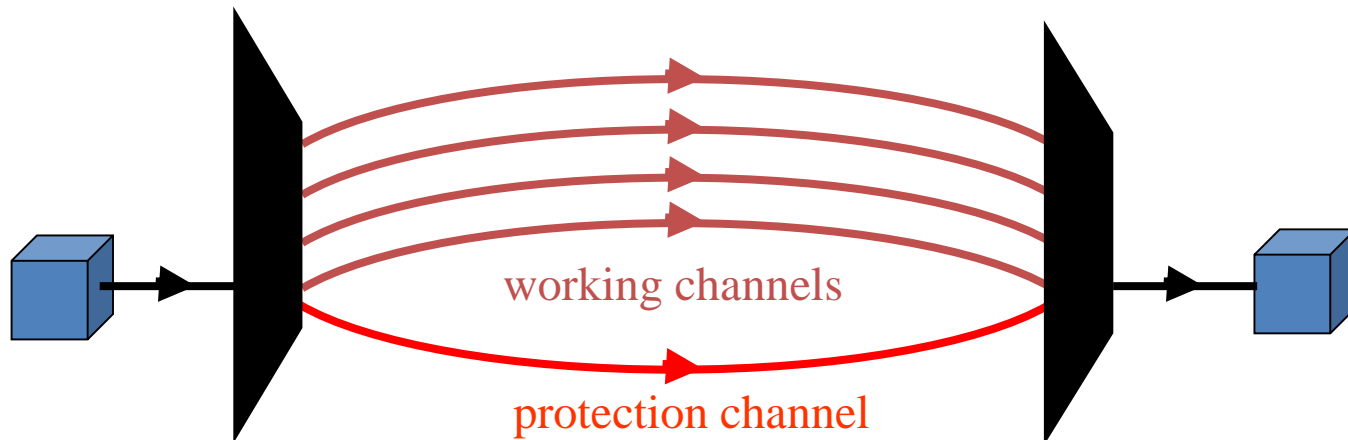
# 1:n protection

One protection channel is allocated for n working channels

Only can protect one working channel at a time

but improbable that  $> 1$  working channel will simultaneously fail

Only  $1/(n+1)$  of total capacity is reserved for protection



# m:n protection

To enable protection of more than 1 channel

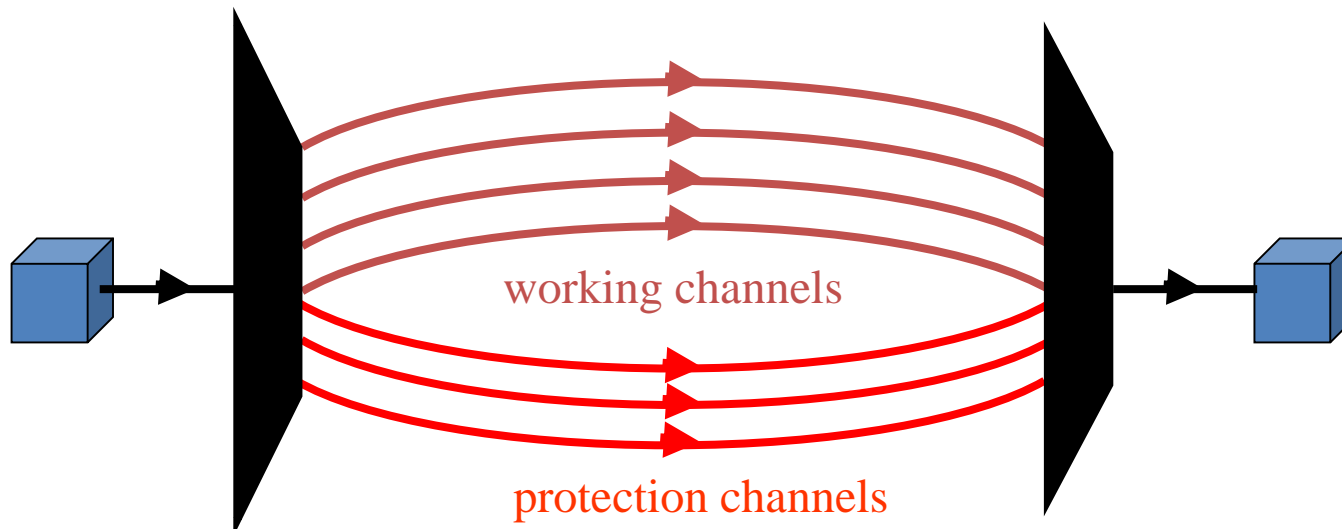
- m protection channels are allocated for n working channels ( $m < n$ )
- m simultaneous failures can be protected

Less protection capacity dedicated than for n times 1:1

When failure detected,

1 of the m protection channels need to be assigned and signaled

High complexity but conserves resources



# $(1:1)^n$ protection

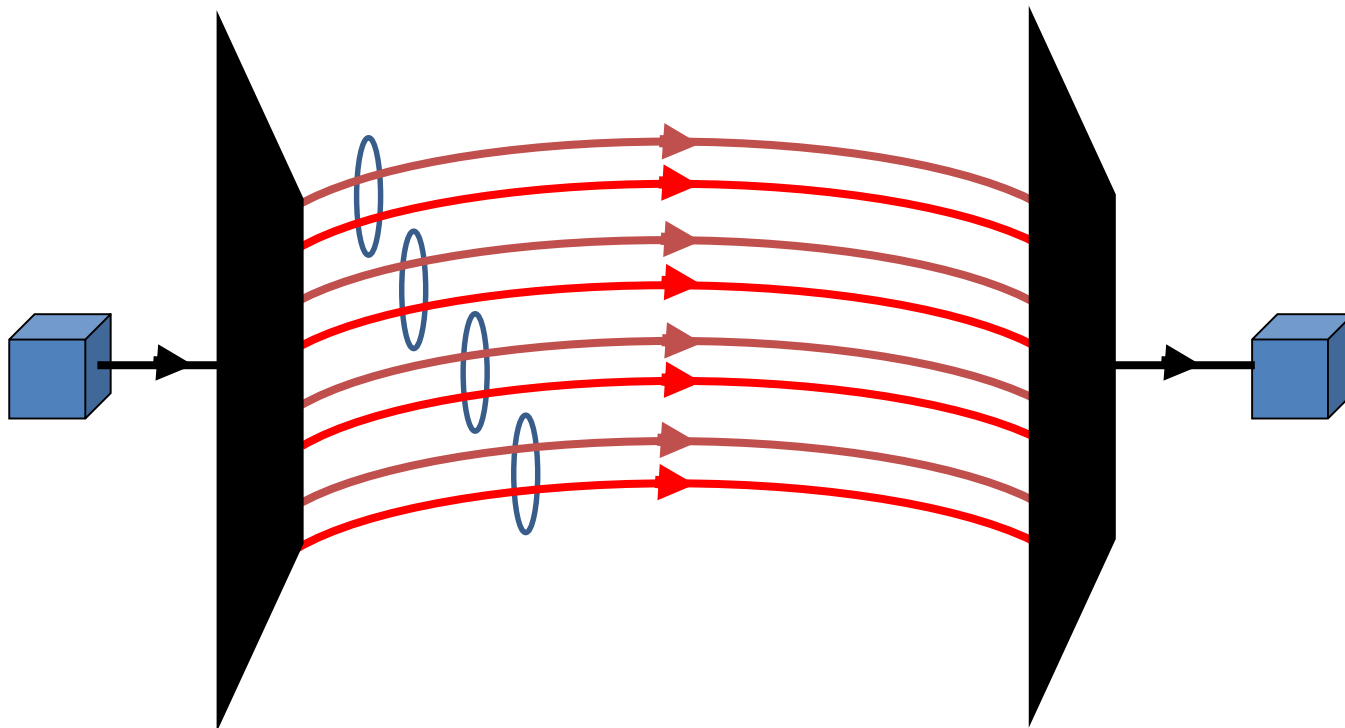
This is like n times 1:1 but the n protection channels *share bandwidth*

**Only 1 failed working channel can be protected**

This is different from 1:n since

- n protection channels are preconfigured
- n working channels need not be of the same type

Protection bandwidth must be at least that of the largest working channel



# 802.1CB Frame Replication and Elimination

Published in 2017 **Frame Replication and Elimination for Reliability**

Basically a *packet-by-packet* 1+1 protection mechanism

- no failure detection needed
- insert stream ID and sequence number into every FRER frame
- frames are replicated at source
- duplicates are eliminated at destination

Several stream identifier methods:

- DA + VID
- SA + VID
- DA + VID + IP SA + IP DA + DSCP + ...

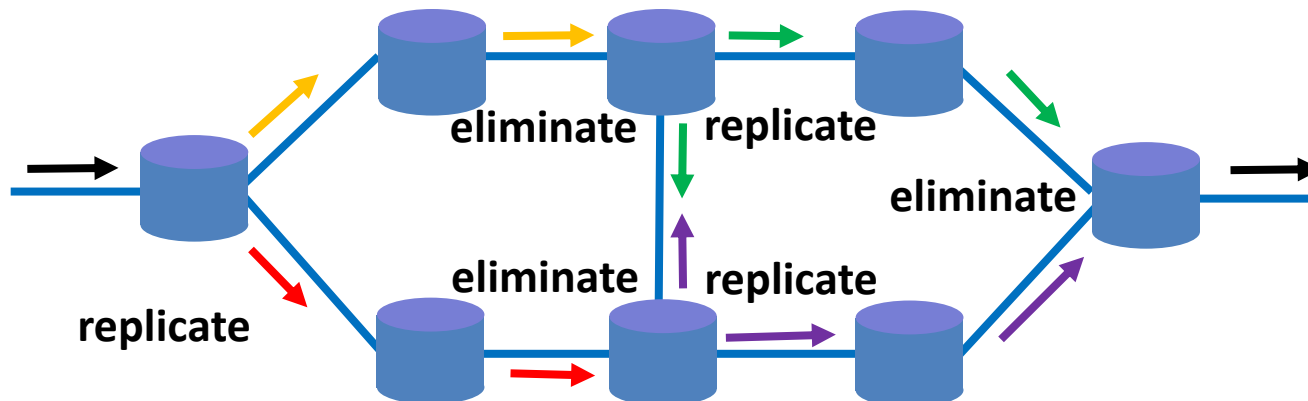
Several sequence number methods:

- new L2 sequence number tag for Ethernet frames
- HSR or PRP sequence numbers
- Ethernet PW sequence numbers

# Re-replication and Re-elimination

But replicating at source and eliminating at destination only protects against a single network failure (except source and destination points themselves which are out of scope)

FRER allows configuring replication and elimination at intermediate nodes



802.1CB does not require in-order delivery because it assumes large buffers and packets may be re-ordered