Il software di controllo del sottosistema di Ottica Adattiva per LBT

Luca Fini, Alfio Puglisi, Marco Xompero, Lorenzo Busoni, Guido Agapito

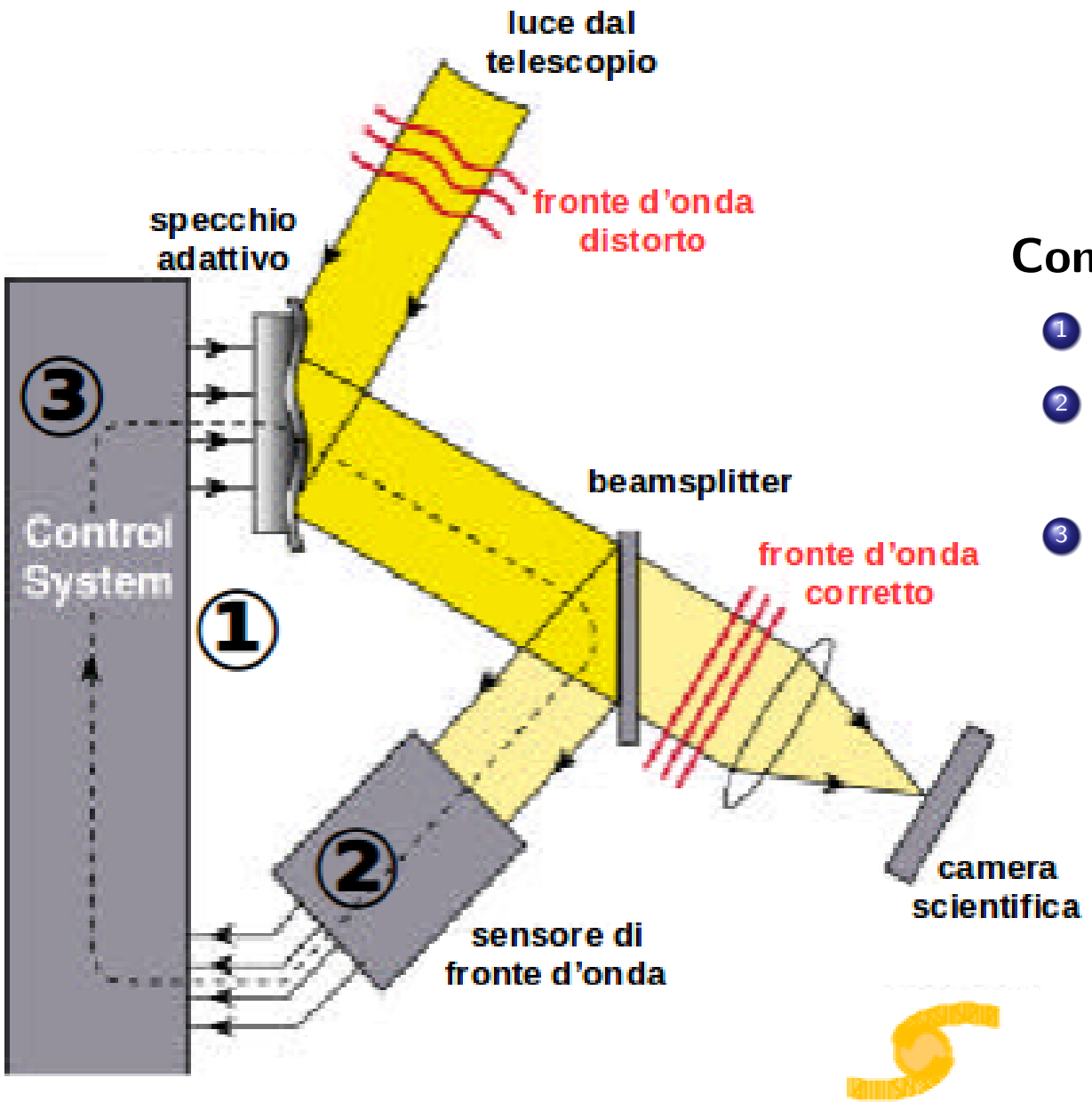
28 Novembre 2017

Viene presentata l'architettura complessiva del software di controllo del sistema di ottica adattiva per il telescopio LBT. In particolare saranno discusse le implicazioni di alcune scelte di progetto e le modifiche che si sono rese necessarie nel tempo per adeguare le funzionalità del sistema software all'evoluzione delle componenti hardware e dell'ambiente.

Sommario

- Introduzione
- Architettura del software di controllo di LBT
  - Telescope Control System
  - Sottosistema di ottica adattiva (FLAO)
  - Sottosistema ARGOS
  - Strumenti interferometrici
- Il software del sottosistema FLAO
  - Il Wavefront Sensor
  - Il Secondario Adattivo
  - Il Supervisore
  - Data flow
- Software di post-processing
- Evoluzione nel tempo del software FLAO
  - Inseguimento modifiche hardware
  - Ottimizzazioni
  - Adeguamento a standard emergenti

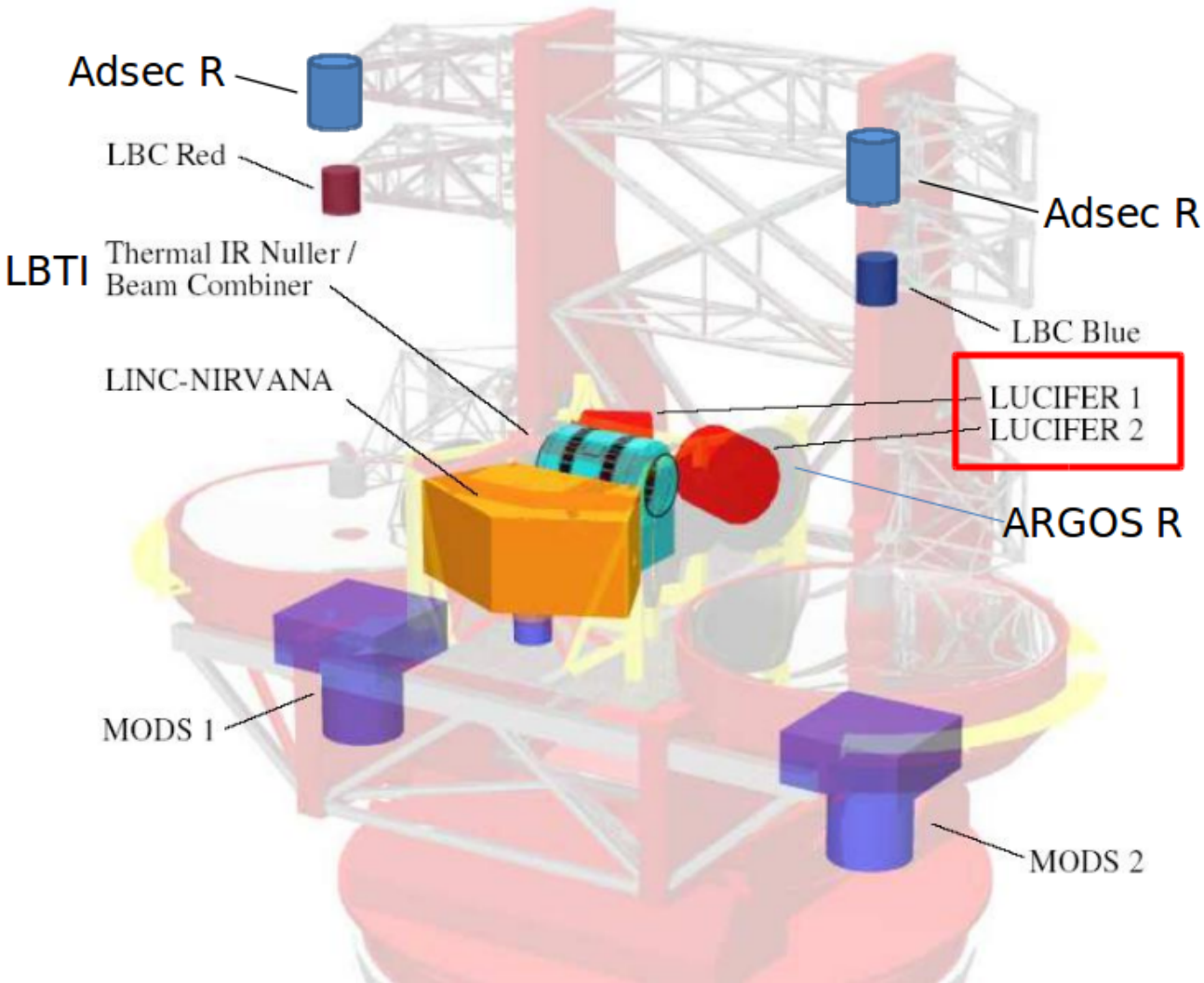
Introduzione: sistemi di ottica adattiva



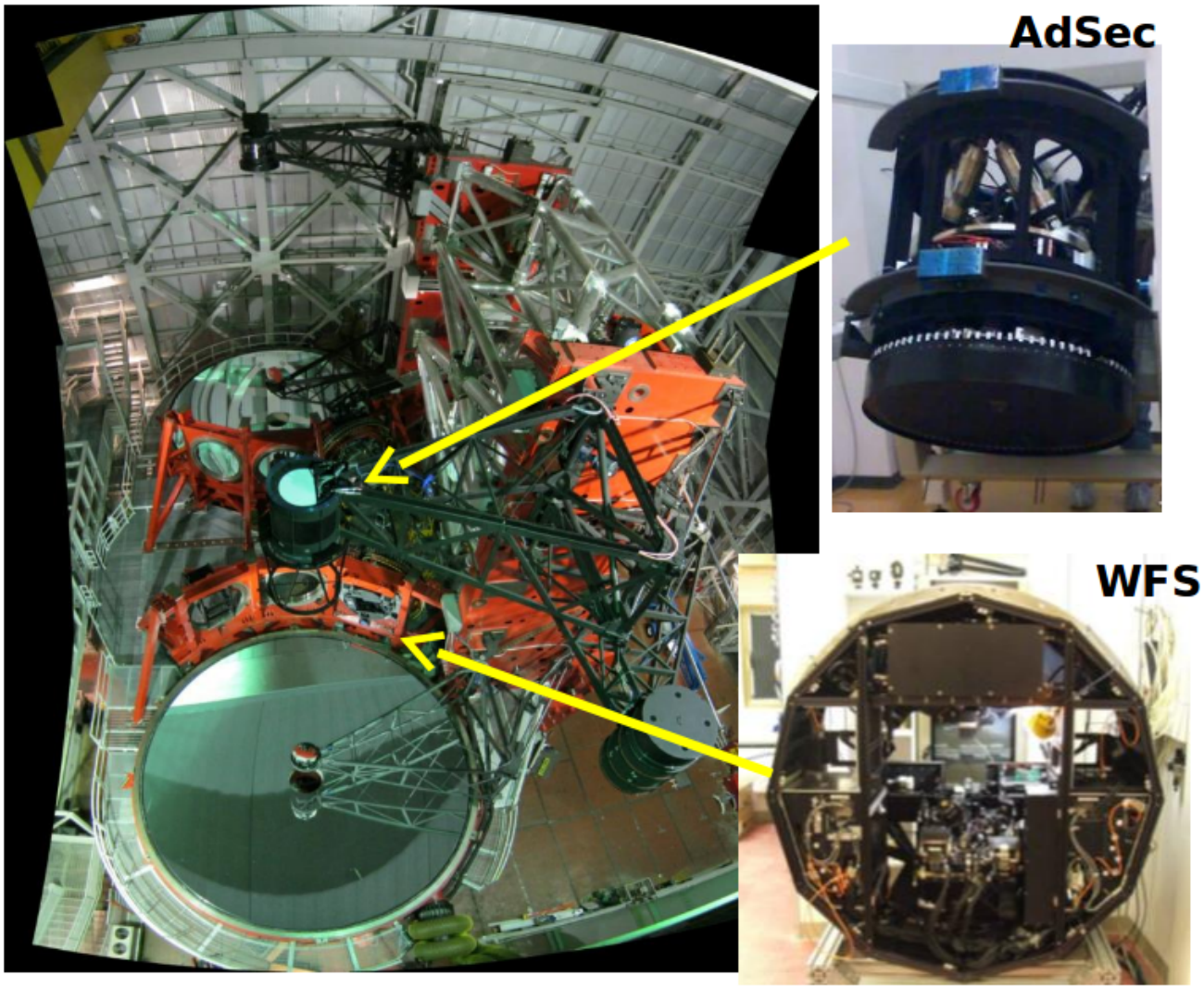
Componenti principali:

1. Specchio adattivo
2. Sensore di fronte d'onda
3. Sistema di controllo

Introduzione: Il sistema FLAO - 1



Introduzione: Il sistema FLAO - 2



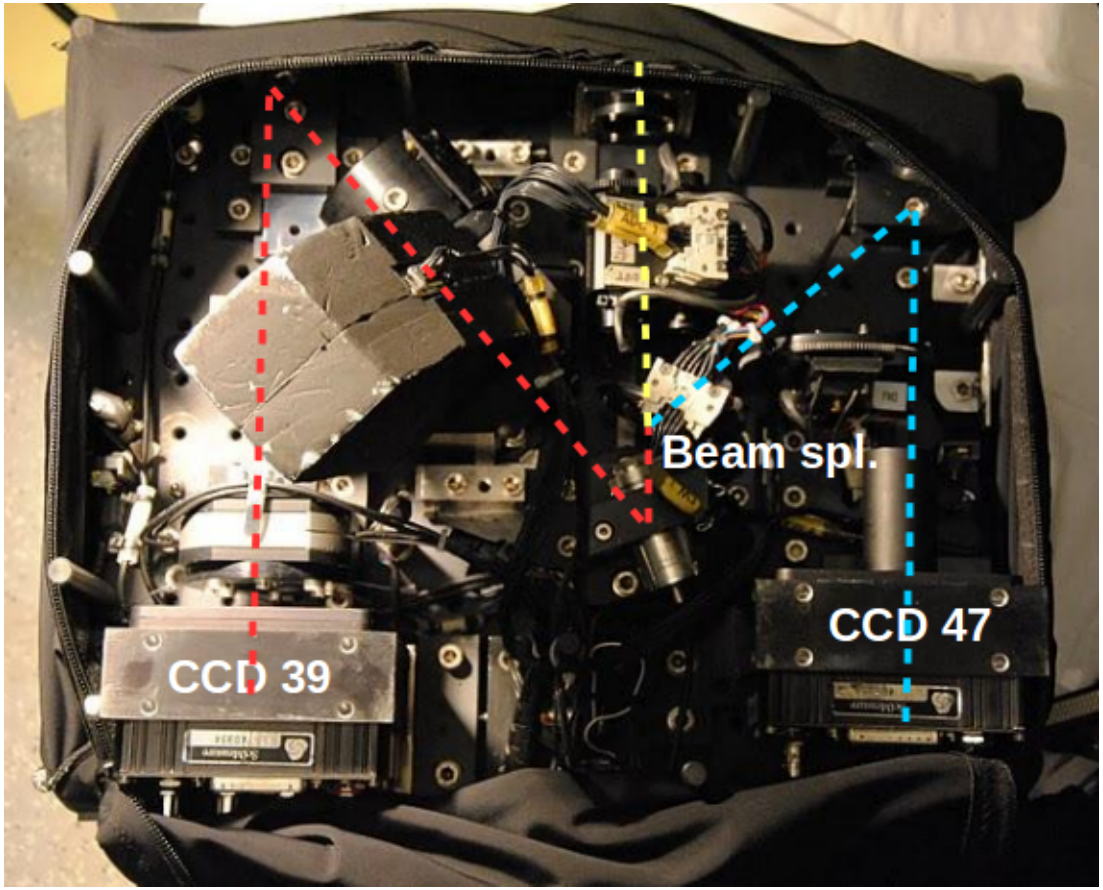
Introduzione: Il secondario adattivo

Il secondario adattivo viene utilizzato anche per osservazioni "seeing limited"

- Dati real-time via fibra ottica
- Configurazione via UDP/Ethernet
- Dati diagnostici via UDP/Ethernet

Introduzione: Il sensore di fronte d'onda

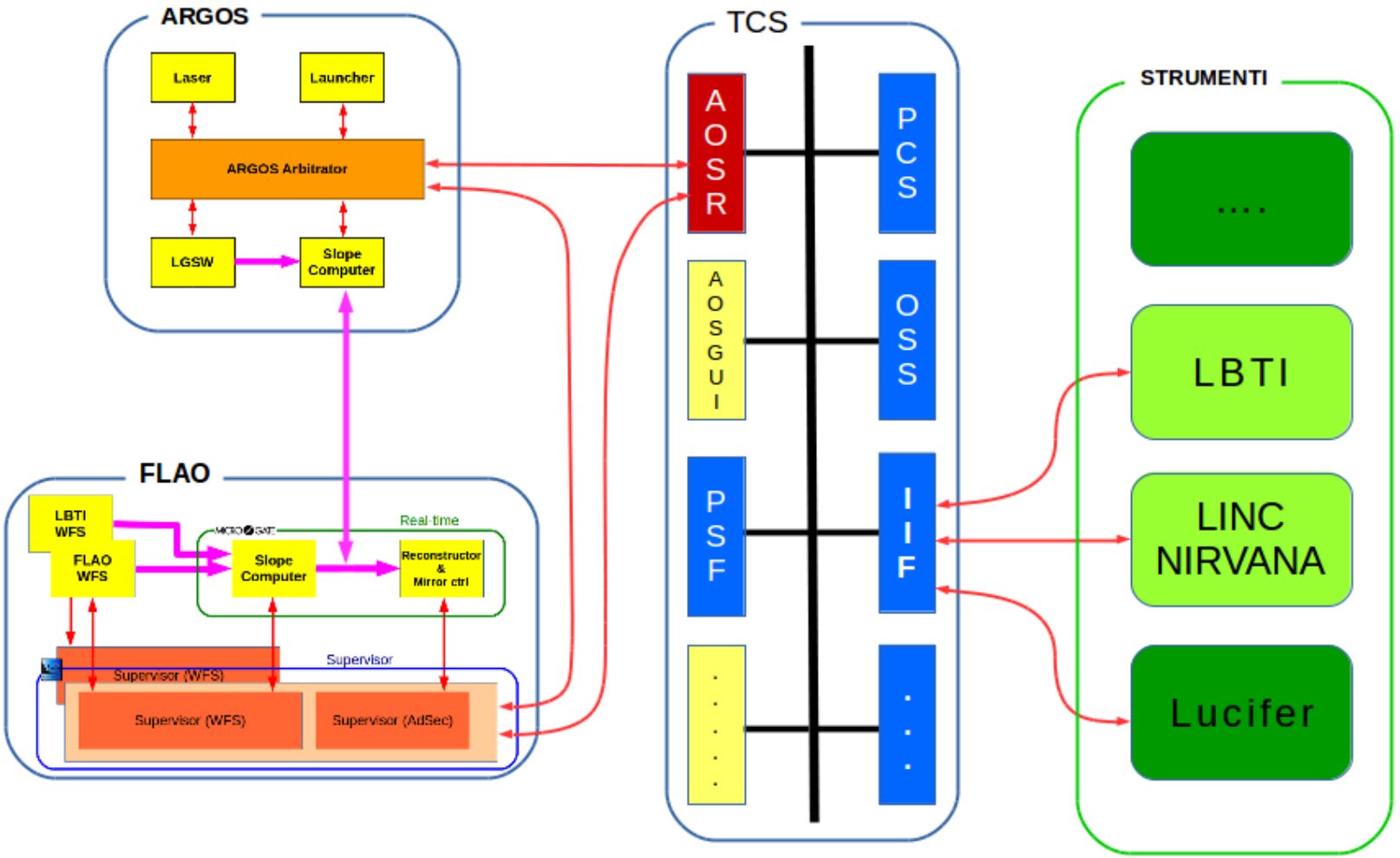
Il sensore di fronte d'onda di FLAO è del tipo "a piramide"



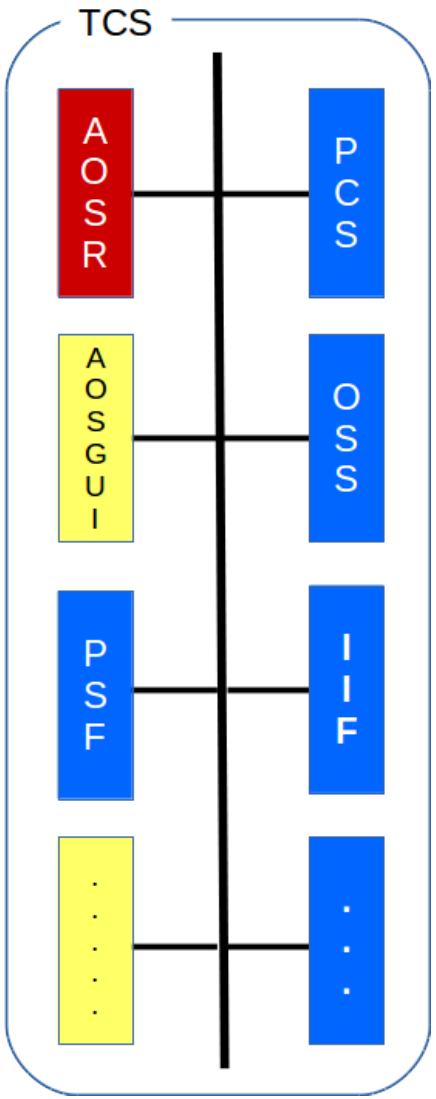
Elementi principali:

- CCD 39
- CCD 47
- 2 portafiltri
- lente di camera
- tip-tilt mirror
- movimenti x,y,z

Il software di controllo di LBT



“Telescope Control System”



Elementi rilevanti:

- **AOS** - Adaptive Optics Subsystem. Interfaccia verso i sistemi di ottica adattiva
- **PCS** - Point Control Subsystem. Controllo puntamento
- **PSF** - Point Spread Function subsystem. Controllo collimazione
- **OSS** - Optics Subsystem. Movimento elementi ottici.
- **IIF** - Instrument Interface.

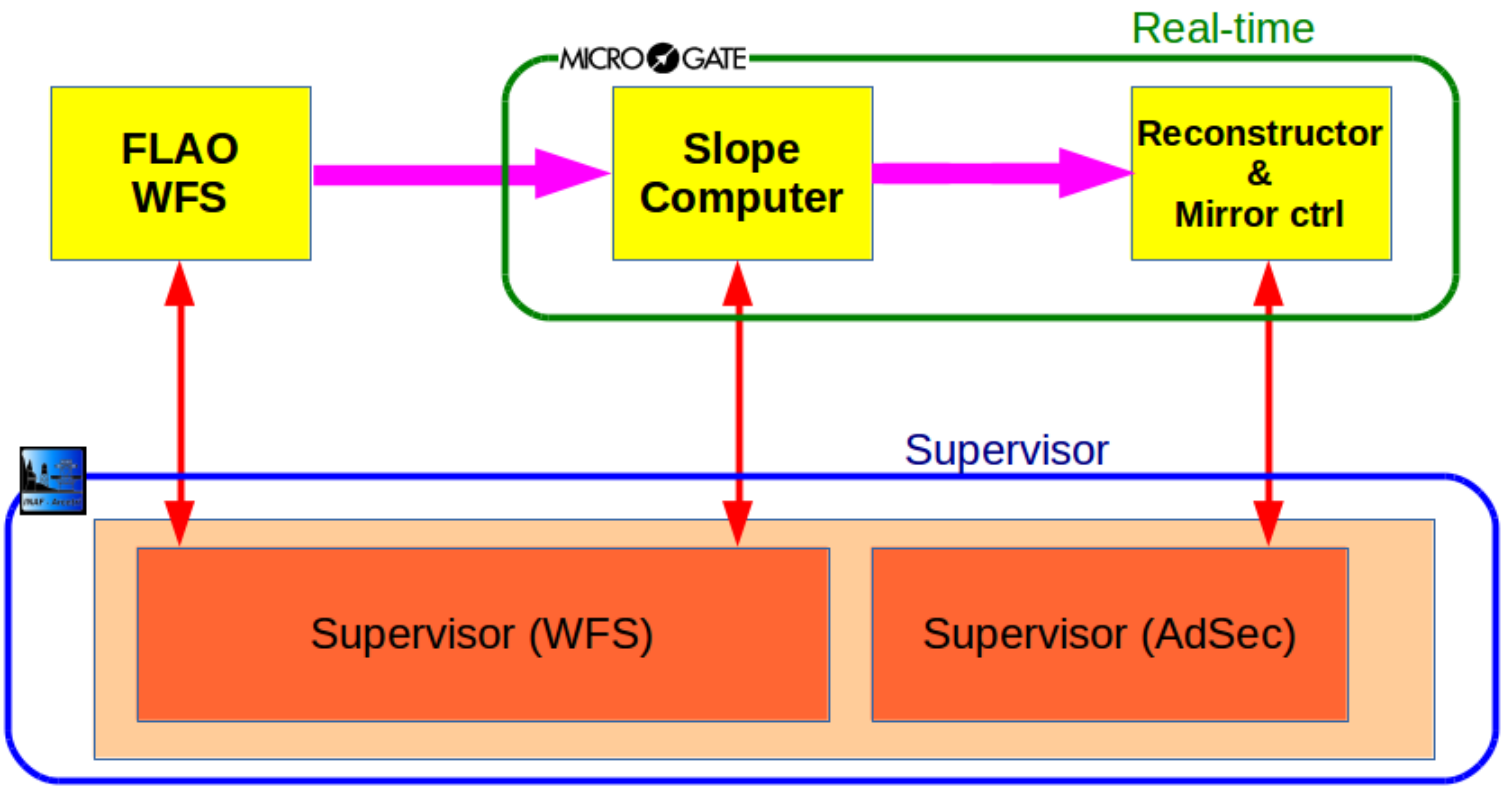
"Telescope Control System - AOS"

AOS: Modulo di interfaccia verso i sottosistemi di Ottica Adattiva

- Sottosistema standard di TCS
  - IPC: "in house"
  - *Data Dictionary*: sistema per variabili condivise
  - Ogni sottosistema definisce un set di comandi e esporta lo stato interno tramite le variabili condivise
- Comunica con FLAO tramite ICE[1] + FLAO IPC (per condivisione variabli)
- Comunica con ARGOS tramite ICE
- AOSGUI: interfaccia dell'operatore

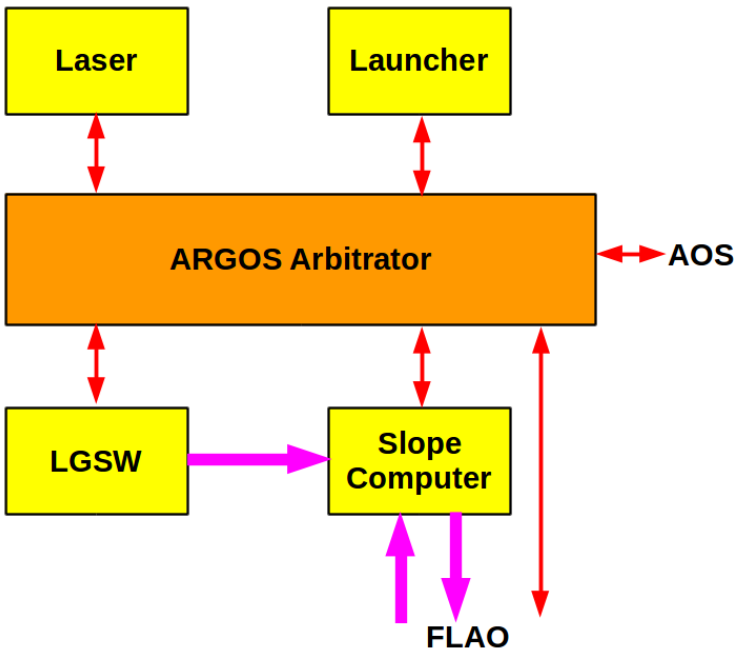
1) Internet Communication Engine: `https://zeroc.com/`

Sottosistema di Ottica Adattiva con stella naturale



Sottosistema ARGOS

Sottosistema di Ottica Adattiva con stelle laser



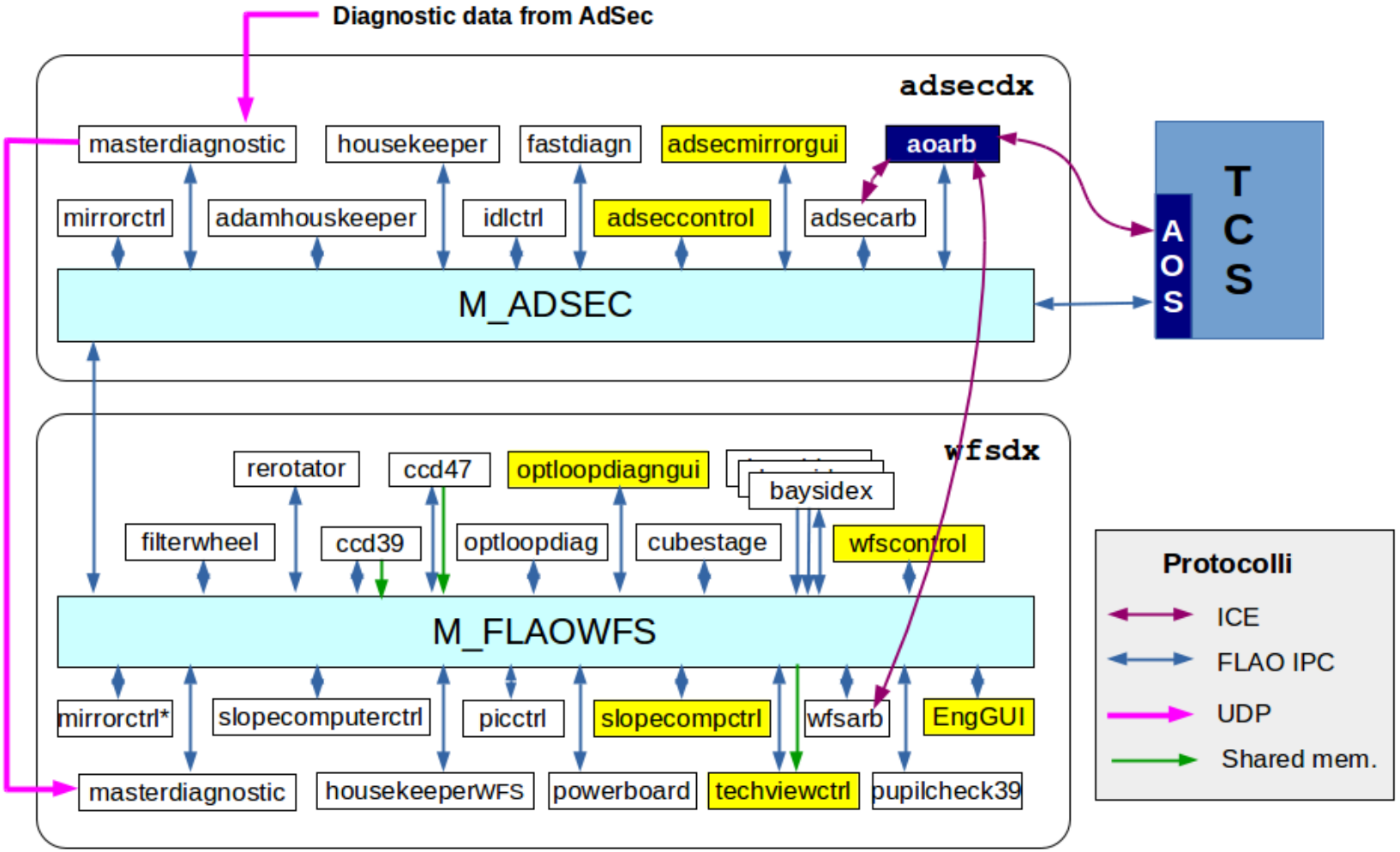
- Utilizza FLAO per la parte “stella naturale”
- Comunica con AOS e FLAO utilizzando ICE
- Uno switch hardware provvede all’instradamento dei dati real-time

Gli strumenti interferometrici

Gli strumenti interferometrici hanno interazioni assai strette con il sottosistema di Ottica Adattiva.

- **LBTI**, Large Binocular Telescope Interferometer
  - Installato alla stazione focale centrale (FLAO è collocato alla stazione anteriore)
  - Duplicazione del Wavefront Sensor di FLAO.
  - Software di controllo: *fork* della parte WFS del software di FLAO
  - È parte integrante di FLAO
- **LINC-Nirvana**, *Beam Combiner* con sistema di ottica adattiva multiconiugata.
  - Utilizza WFS e correttore propri.
  - Utilizza il secondario adattivo tramite FLAO
  - Ha richiesto la riprogettazione dell'interfaccia software dell' AdSec Arbitrator

Il supervisore di FLAO



Il supervisore di FLAO - criteri di progetto

- Sistema distribuito con processi ad "accoppiamento lasco"
- Comunicazione basata su *message passing* con server di comunicazione centralizzato (`MsgD-RTDB`)
  - Protocollo *in house* (per ragioni storiche)
  - Protocollo di basso livello (es: no XDR, solo strutture dati "semplici", per efficienza e semplicità di implementazione)
  - Supporto per IPC via *shared memory*
- Database di variabili *in house* (come sopra)
- Linguaggi di sviluppo: C/C++, Python (+ IDL!)
- Ambiente Linux
- Strumenti di sviluppo:
  - SMC, the *State Machine Compiler* (`http://smc.sourceforge.net`)
  - QT Designer
  - Subversion (inizialmente: CVS)
  - Foswiki (`http://foswiki.org`)
  - Trac (`http://trac.edgewall.org`) (inizialmente: Bugzilla)
  - Jenkins (`https://jenkins.io`)

Il supervisore di FLAO - componenti

- `MsgD-RTDB`. Message Daemon and Real-Time Database. Processo per lo scambio di messaggi e la gestione dell'archivio delle variabili condivise.
  - Tre processi interconnessi: `M_ADSEC`, `M_FLAOWFS`, `M_LBTIWFS`
- **Controllori**. Processi per la gestione dei dispositivi hardware.
- **Processi diagnostici**.
- **GUI ingegneristiche**. Interfacce grafiche per il controllo dei dispositivi.
- **Arbitratori**.
  - **WFS arbitrator**: coordinamento dei dispositivi che compongono il Wavefront Sensor.
  - **Adsec arbitrator**: coordinamento dei dispositivi che compongono il secondario adattivo.
  - **AO arbitrator**: coordinamento dei due sottosistemi: AdSec e WFS.

Data flow nel sistema FLAO

- Basso throughput
  - Informazioni di stato: processi $\Longleftrightarrow$ AdSec-RTDB/WFS-RTDB
    AdSec-RTDB $\Longleftrightarrow$ WFS-RTDB
  - Comandi: processi $\Longrightarrow$ processi
    GUI $\Longrightarrow$ processi
    processi $\Longrightarrow$ WFS/AdSec
- Medio throughput
  - Frame CCD (CCD39 e CCD47): CCD $\Longrightarrow$ GUI
- Alto throughput
  - Dati diagnostici: AdSec $\Longrightarrow$ `masterdiagnostic`
    `masterdiagnostic` $\Longrightarrow$ `fastdiagnostic`

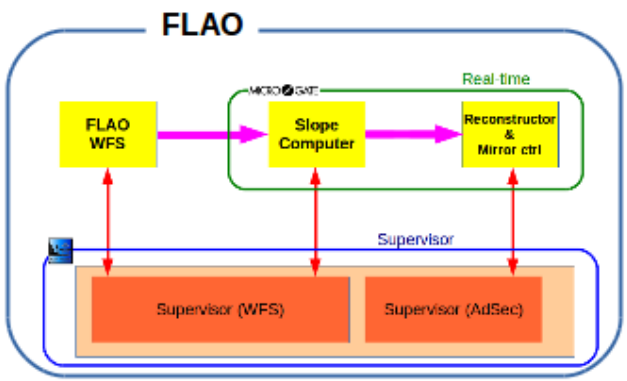
Funzioni di *post-processing*

elab-lib un "tool" generico per l'analisi dei dati del sistema FLAO.

- Accede direttamente alla strutture dati *snapshot* di FLAO:
- Dati *snapshot*: sequenze temporali limitate di:
  - *Frames*, *Slopes*, *Commands*
  - Stato telescopio (el., az., pos. rotatore, pos. secondario, ecc.)
  - Stato dispositivi ottici di FLAO (slitte, portafiltri, ecc.)
  - Immagine scientifica (opzionale)
- Utilizzazioni:
  - Valutazione prestazioni
  - Diagnostica
  - Verifica calibrazioni
  - Generazione di statistiche di prestazione
- Scritta in IDL (O.O.)
- Comprende circa 500 funzioni
- Orientata all'uso interattivo
- In corso il trasporto in Python per uso real-time

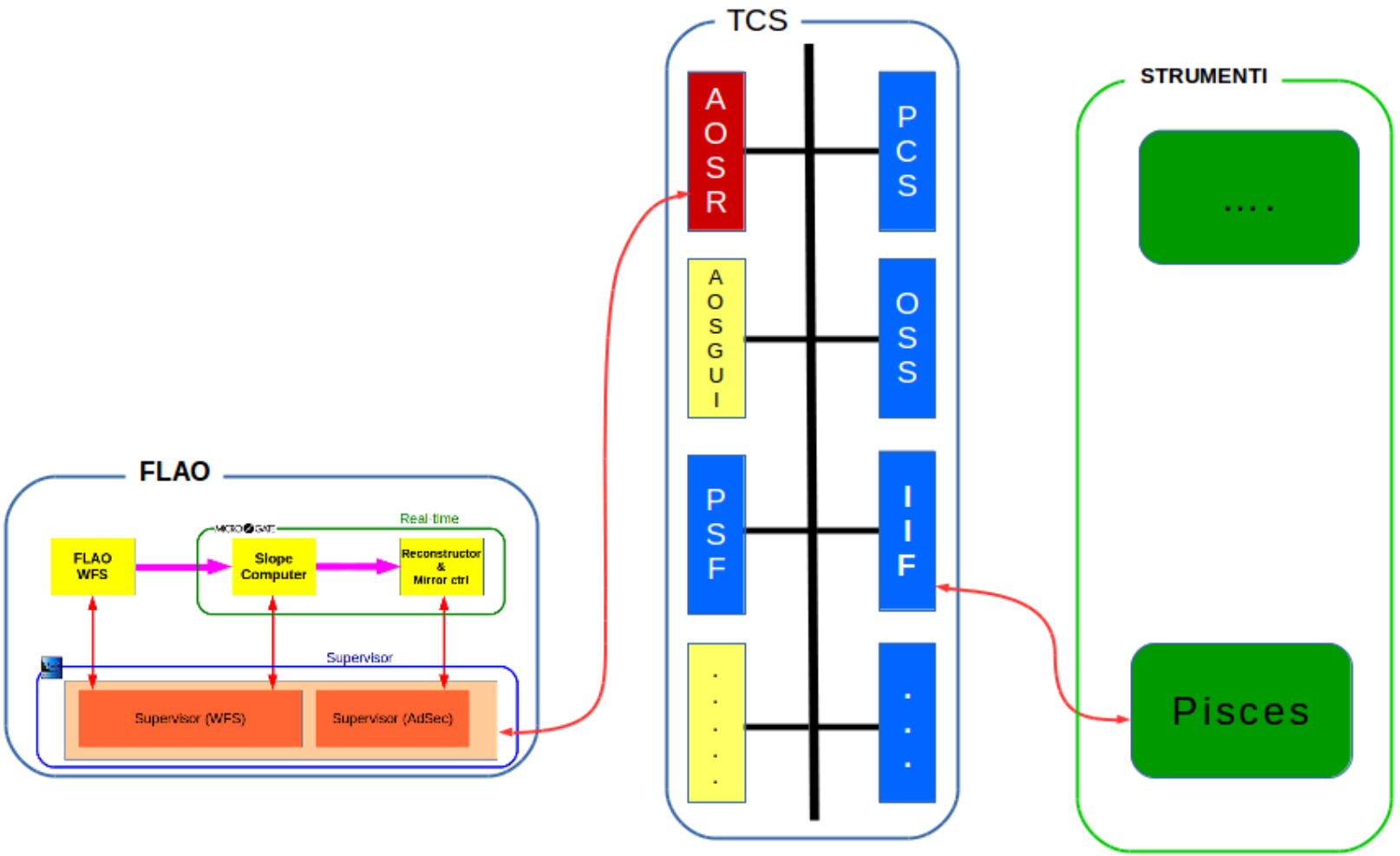
Evoluzione di FLAO - 2004. Lab. Torre ad Arcetri

- Architettura di base già consolidata
- Operazioni senza arbitratori (dalle GUI ingegneristiche)



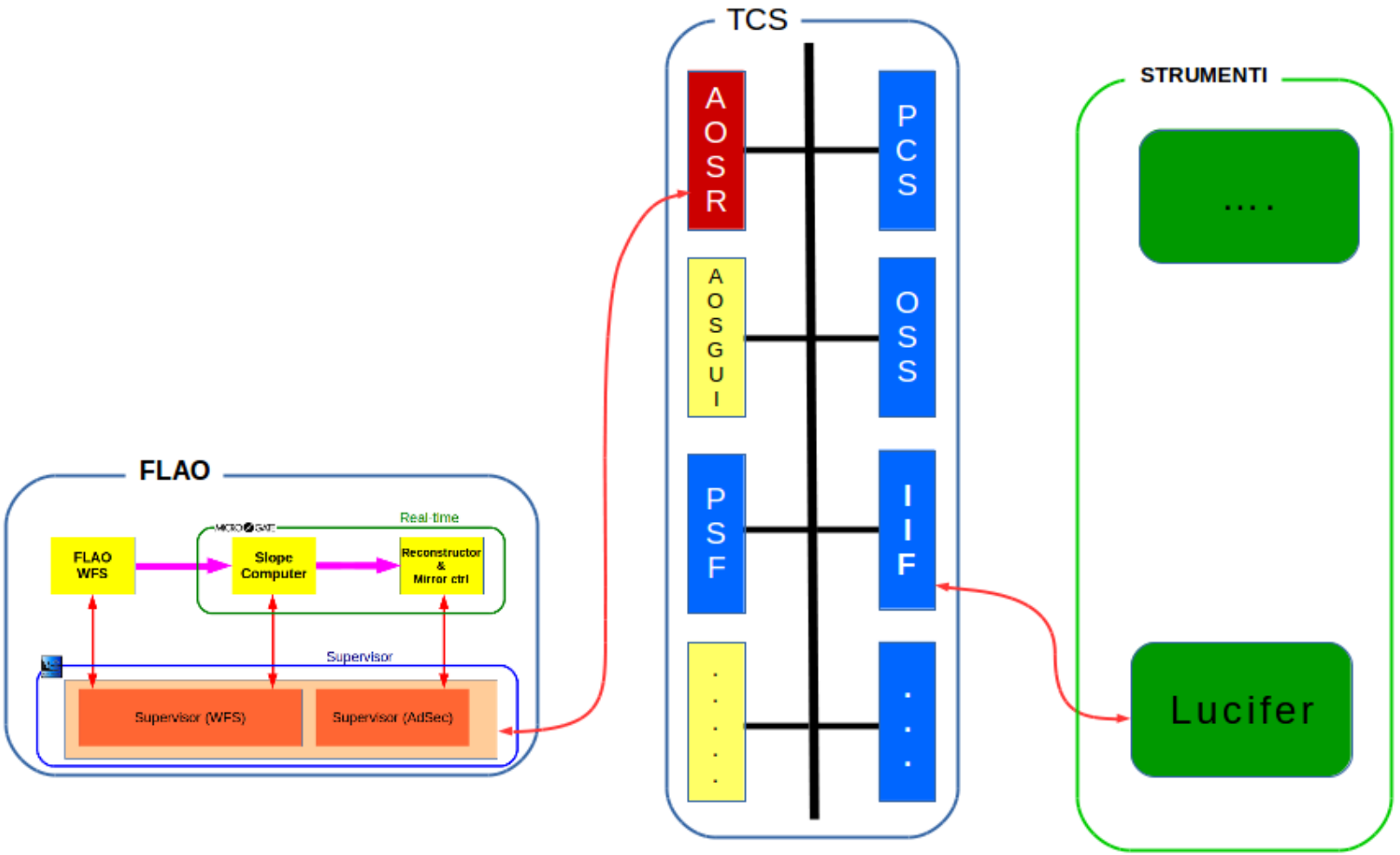
Evoluzione di FLAO - 2010. Commissioning

- Software identico alla versione di laboratorio ...
- ... più arbitratori e AOS



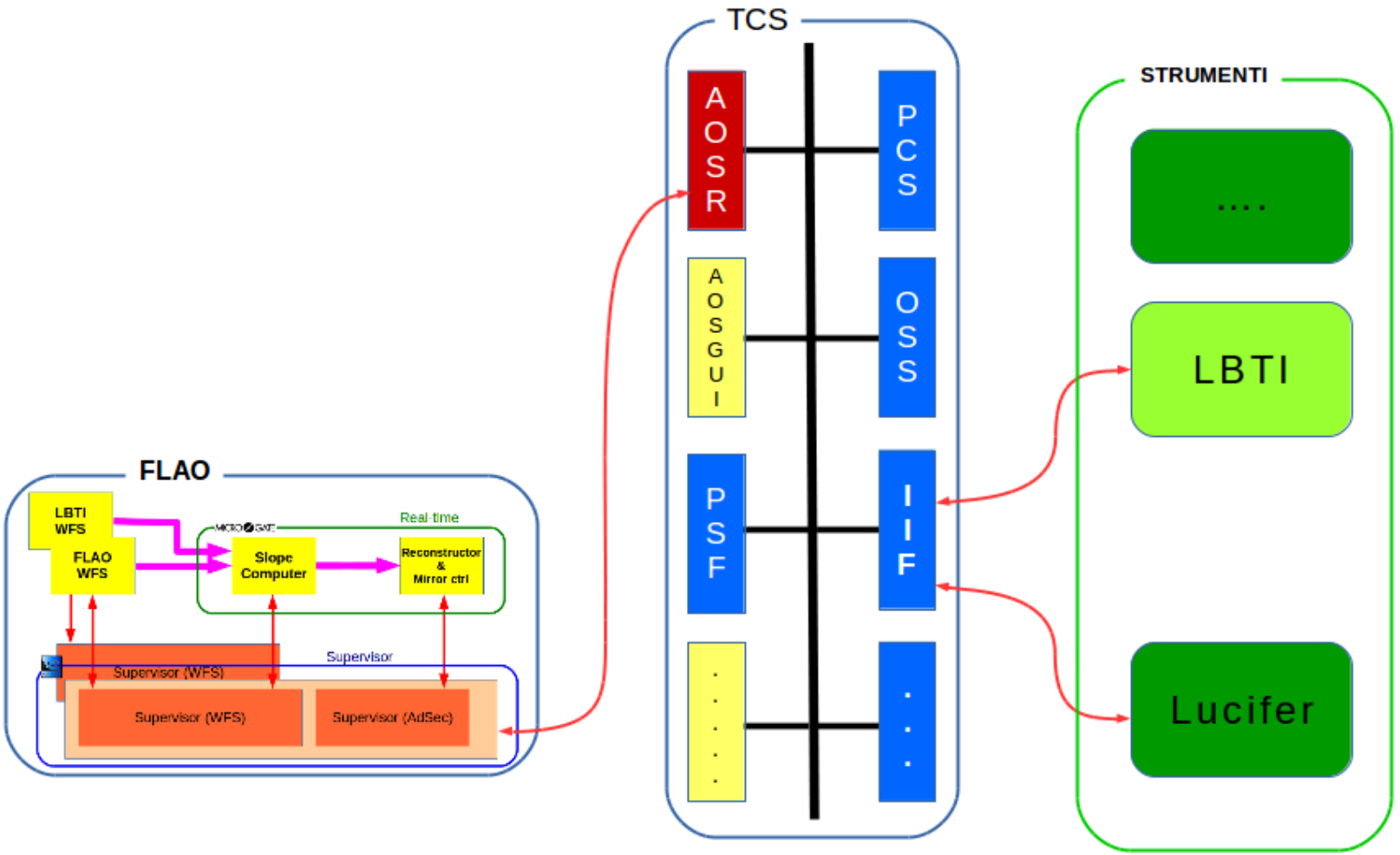
Evoluzione di FLAO - 2012. Prima scienza

- Consistenti modifiche del software durante il *commissioning*
- Strumento scientifico (Lucifer)



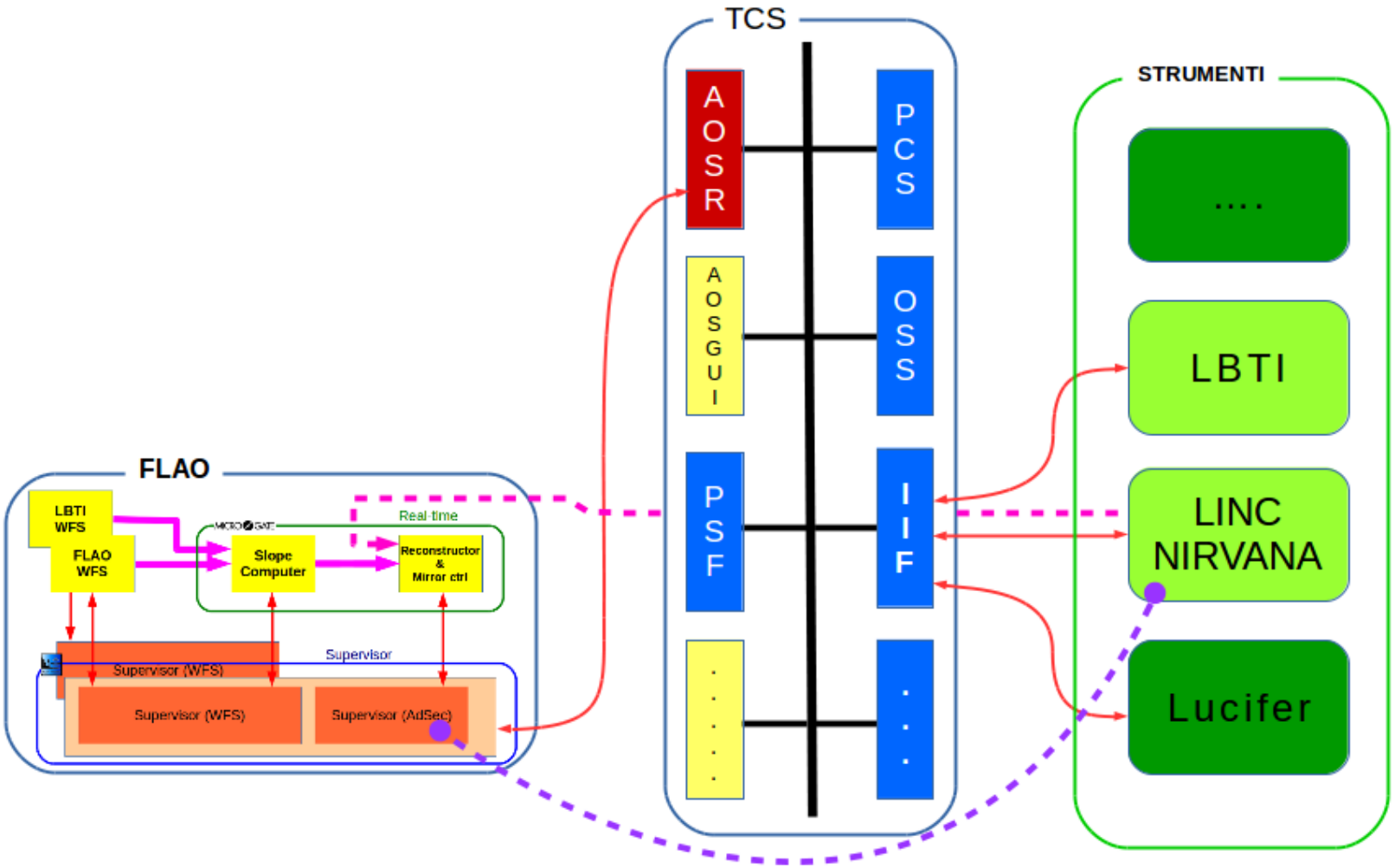
Evoluzione di FLAO - 2014. LBTI

- Duplicazione WFS e canale dati real-time
- Duplicazione Supervisor (WFS) + modifiche AOS



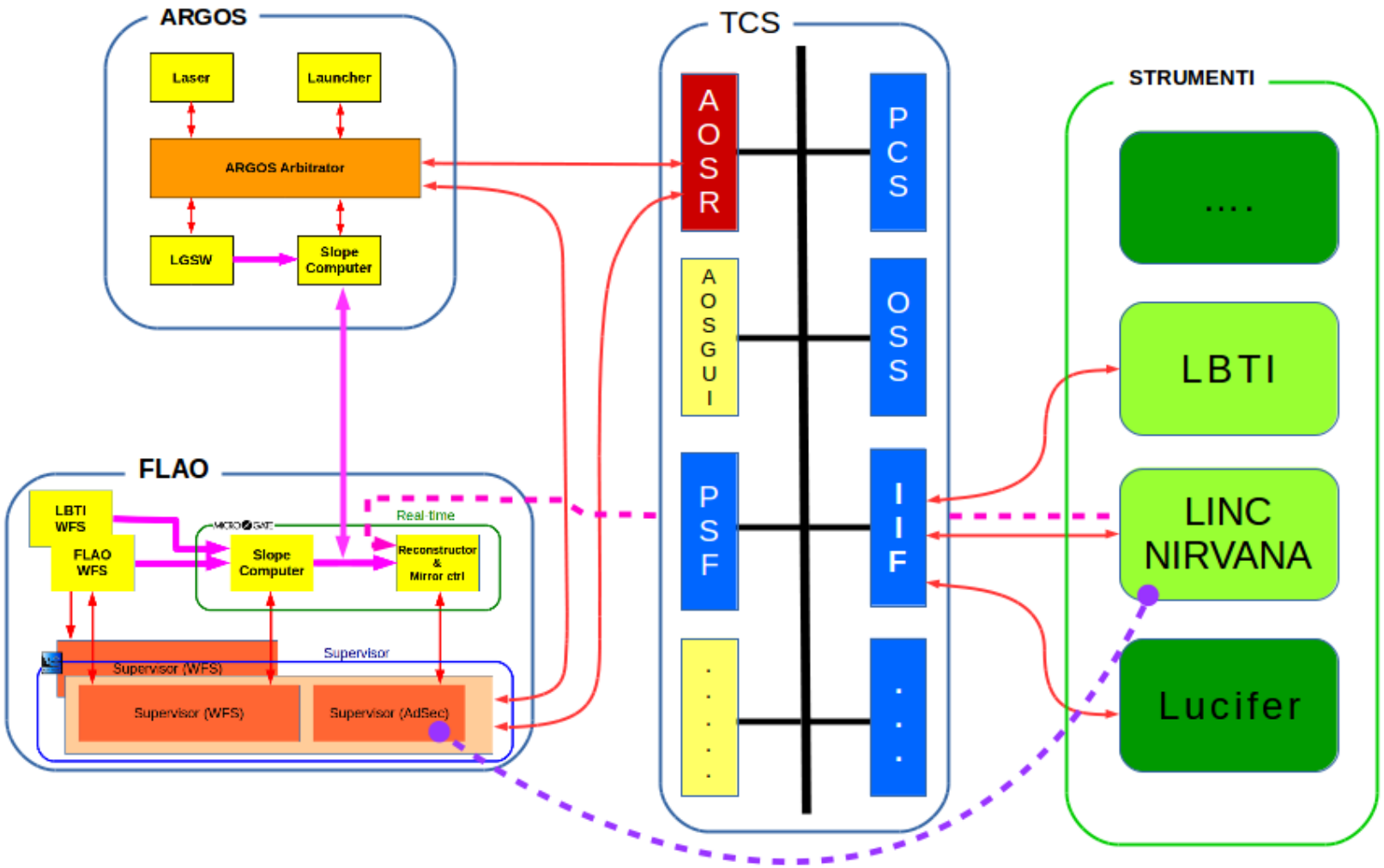
Evoluzione di FLAO - 2015. LINC-Nirvana

- Aggiunto canale real-time per secondario adattivo
- Definizione interfaccia ICE per AdSec Arbitrator



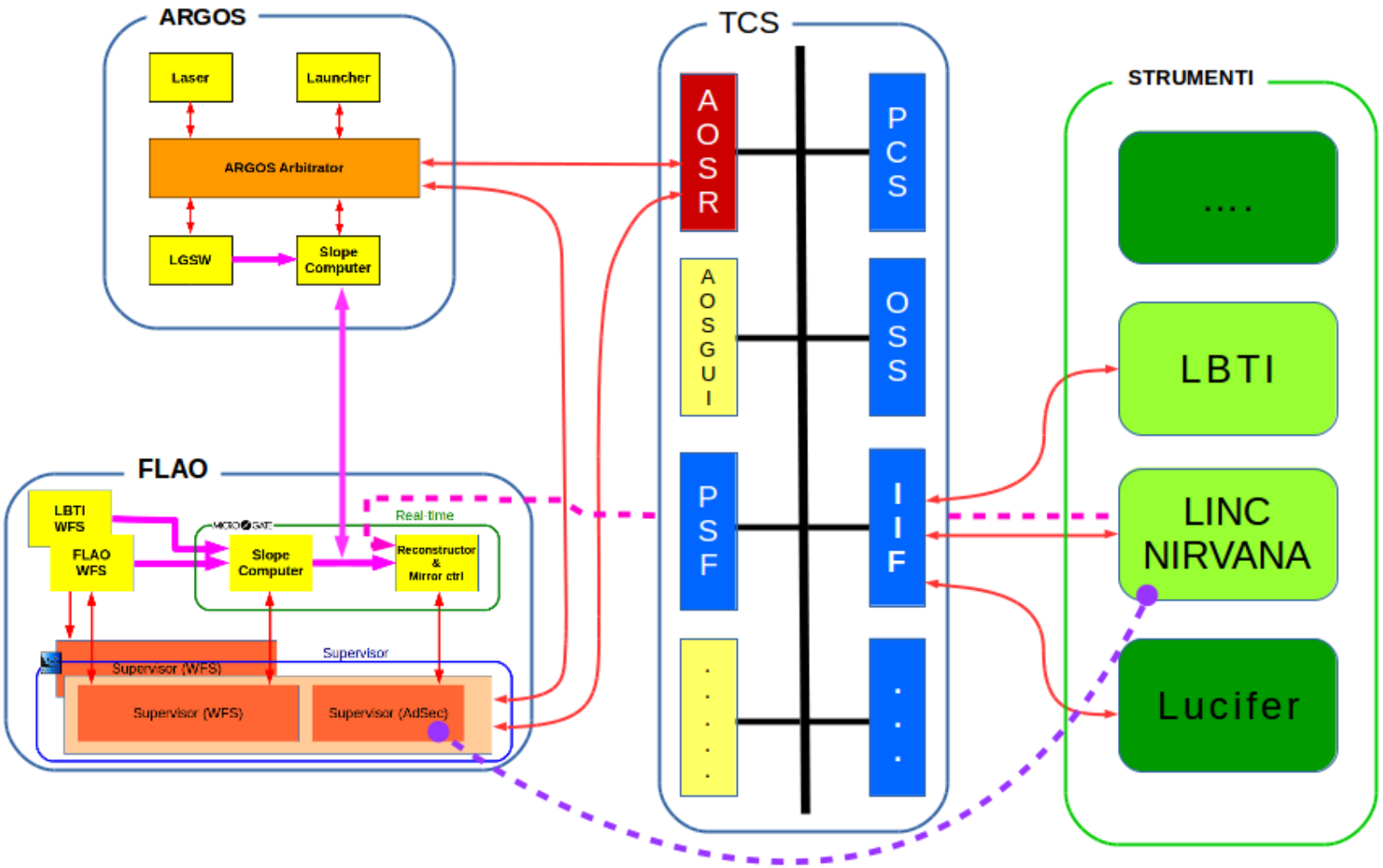
Evoluzione di FLAO - 2016. ARGOS

- Definizione interfaccia ICE per WFS Arbitrator e AO Arbitrator
- Modifica AOS per uso ICE + supporto per ARGOS



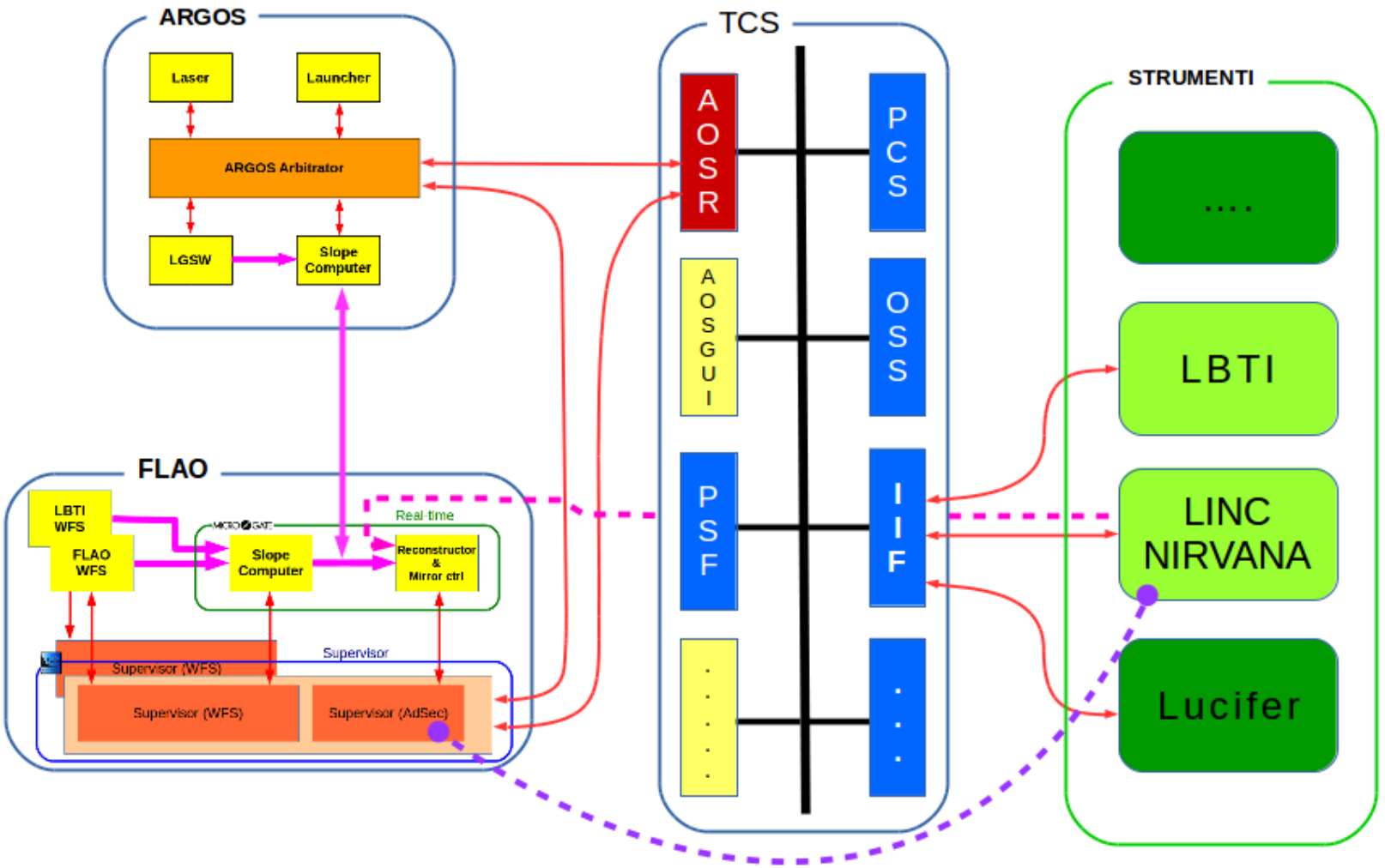
Evoluzione di FLAO - 2016. UAO

- Arbitratori riscritti in `python`
- Supporto del modo "intervention"



Evoluzione di FLAO - 2018. SOUL

- Aumentata risoluzione del CCD di WFS
- Raddoppiata frequenza operativa del secondario (1 $\rightarrow$ 2 KHz)



Conclusione - Fra la teoria e la prassi ...

Teoria	Prassi
Limitare il numero di linguaggi utilizzati	C + Python: scelta iniziale. C++: è stata una evoluzione naturale. IDL: Vogliamo rifare tutto il lavoro ben testato in laboratorio?
Il SW open-source è più affidabile e controllabile	Di nuovo: vogliamo rifare tutto il software IDL?
Usare tecnologia IPC consolidata	All'inizio del progetto stava emergendo CORBA. Ma ne esistevano due o più versioni non interoperabili ...
Usare linguaggi di alto livello dove non impedito da motivi di efficienza	Python era stato scelto per lo scopo (arbitratori), ma al momento della realizzazione avevamo disponibile un bravo programmatore C++ ...