

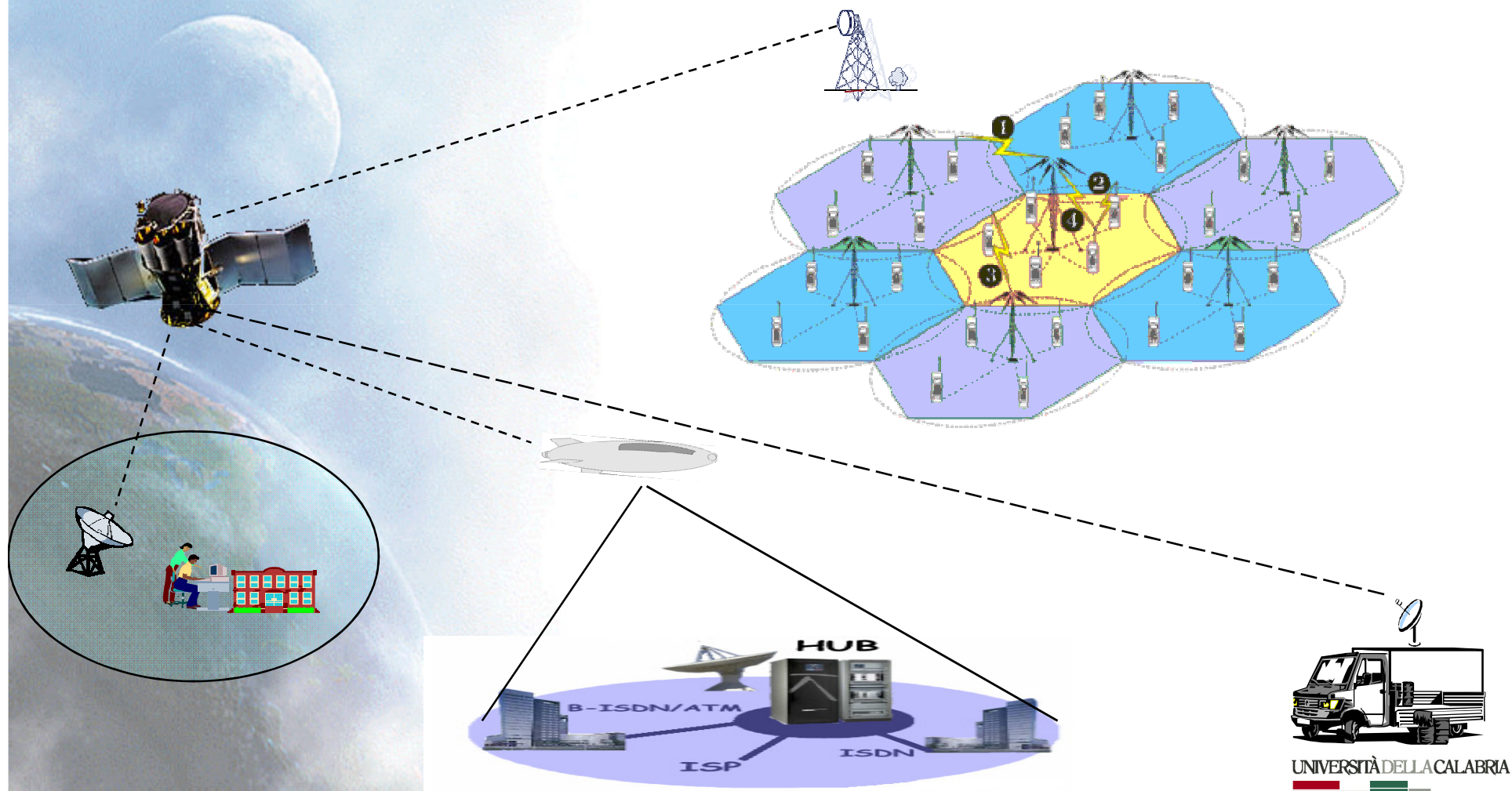
Architettura Scalable CORE (SCORE)

Sommario

- Modello Best-effort
- Nuovi Servizi
- Tecniche e meccanismi per i nuovi servizi
 - ◆ Integrated Services
 - ◆ Differentiated Services
- Architettura SCORE
 - ◆ Dynamic Packet State
 - ◆ Algoritmo di Scheduling
 - ◆ Algoritmo di Ammissione
 - ◆ Prove simulative
 - ◆ Osservazioni



Eterogeneità della rete internet



Modello Best-effort

- Oggi Internet fornisce un semplice modello di consegna dei pacchetti
- Tale modello di servizio permette ai routers di essere “stateless”, cioè, eccetto per poche informazioni, di non mantenere lo stato per il traffico
- Ciò rende Internet Scalabile e Robusta
 - ◆ Scalabile - perché la complessità dei routers non incrementa con il numero di flussi
 - ◆ Robusta – perché ci sono pochi stati da aggiornare per i routers



Nuovi Servizi

- Lo sviluppo di Internet e delle applicazioni multimediali hanno portato ad aver bisogno di servizi più performanti del best-effort:
- ◆ **Servizi Garantiti:** dare la possibilità di garantire parametri prestazionali come banda e ritardo per ogni flusso nella rete
- ◆ **Servizi Differenziati:** che permettono di fornire differenziazione di servizi in base a parametri (banda, rate di perdita, etc.) per aggregati di traffico

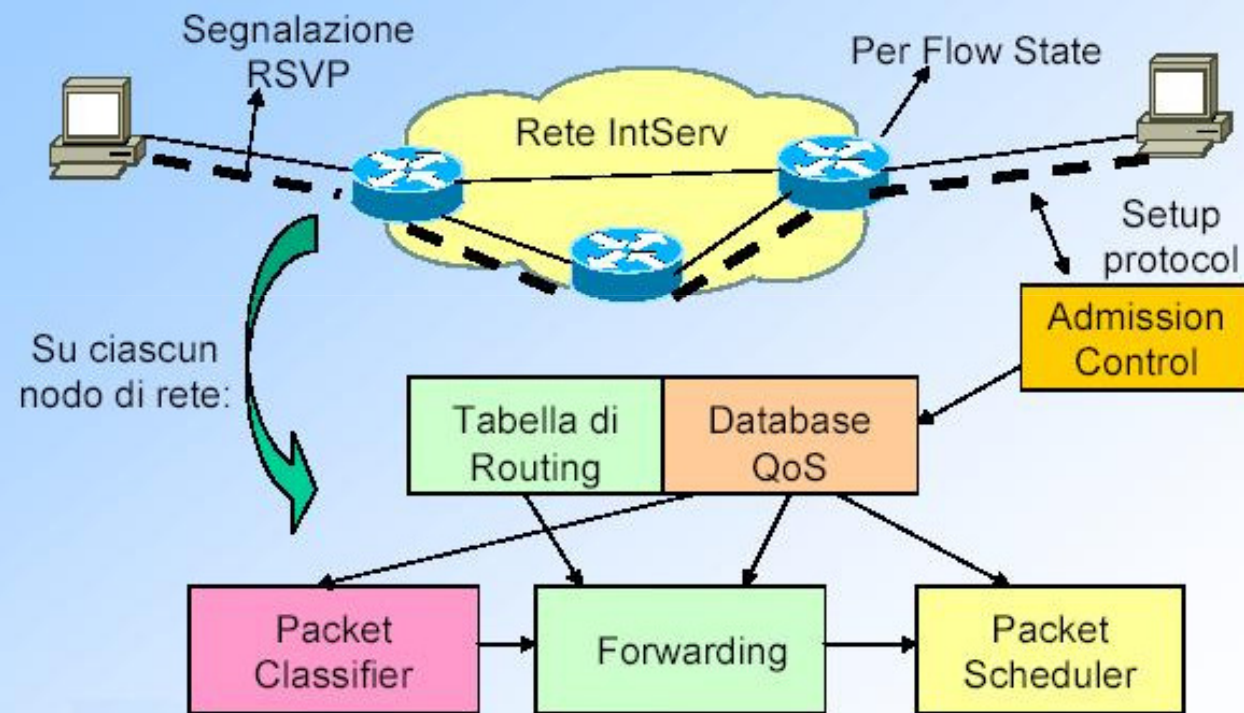


Architetture di QoS

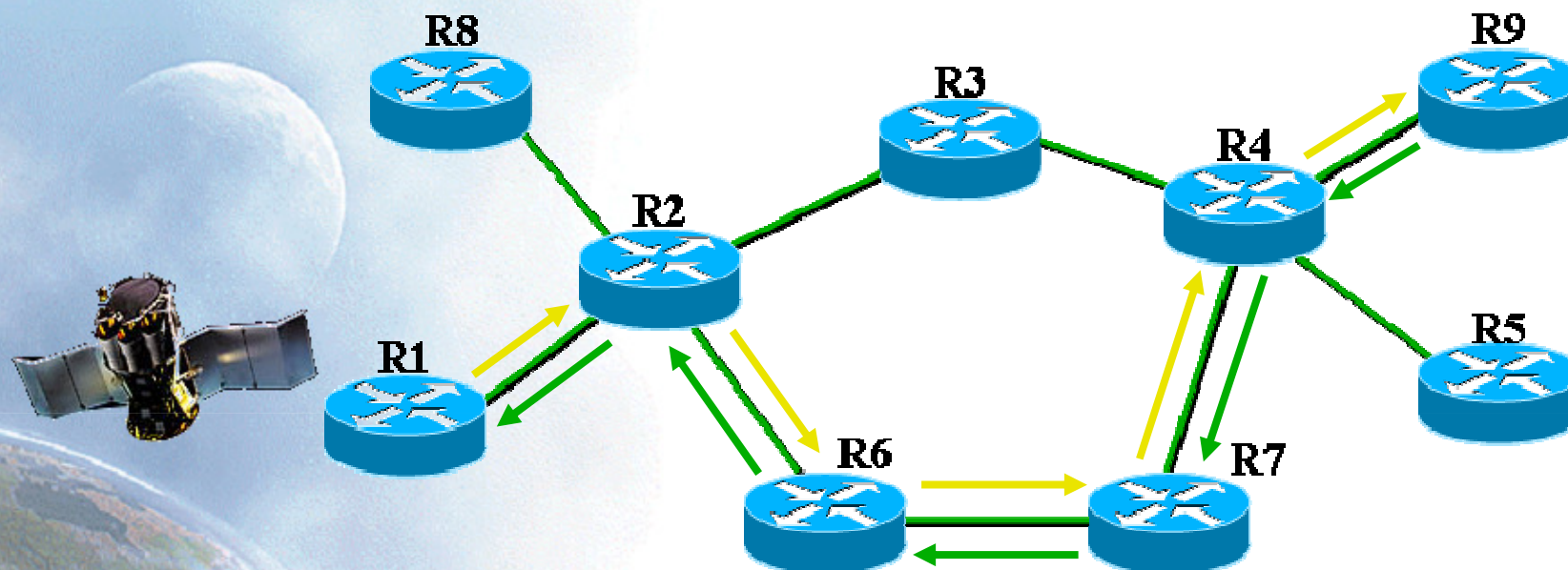
- Sono nate architetture per la Qualità del Servizio (QoS)
- Integrated Services (IntServ) (Statefull)
 - ◆ Guaranteed Services (GS)
 - ◆ Controlled Load Services (CLS)
- Differentiated Services (DiffServ) (Stateless)
 - ◆ Expedited Forwarding (EF)
 - ◆ Assured Forwarding (AF)



Schema di funzionamento dell'IntServ



Protocollo RSVP



Setup: Path (R1->R2->R6->R7->R4->R9)



Reply: Resv (R9->R4->R7->R6->R2->R1)

Classi di traffico IntServ

- Guaranteed Services (GS)

- ◆ Quantitativa

- ◆ Bound sul ritardo di accodamento $DB = \frac{b}{B} + \frac{C}{B} + D$

- C termine dipendente dalla rate

- D termine indipendente dalla rate



- Controlled Load Services (CLS)

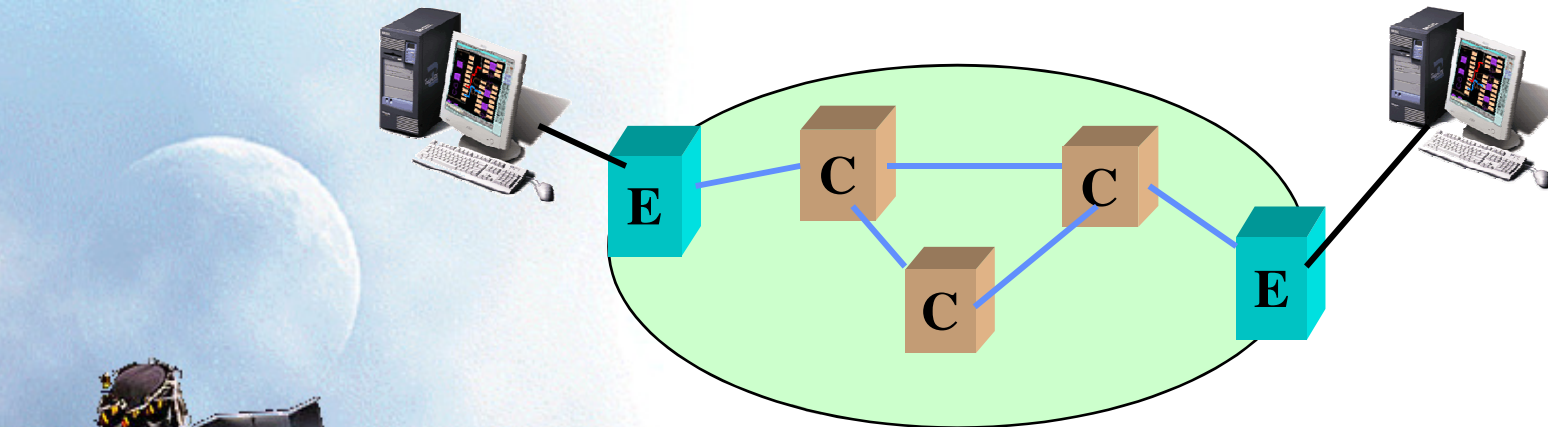
- ◆ Qualitativa

- ◆ Servizio “Better than Best-Effort”

- Best Effort

- ◆ Nessuna garanzia

DiffServ (1)



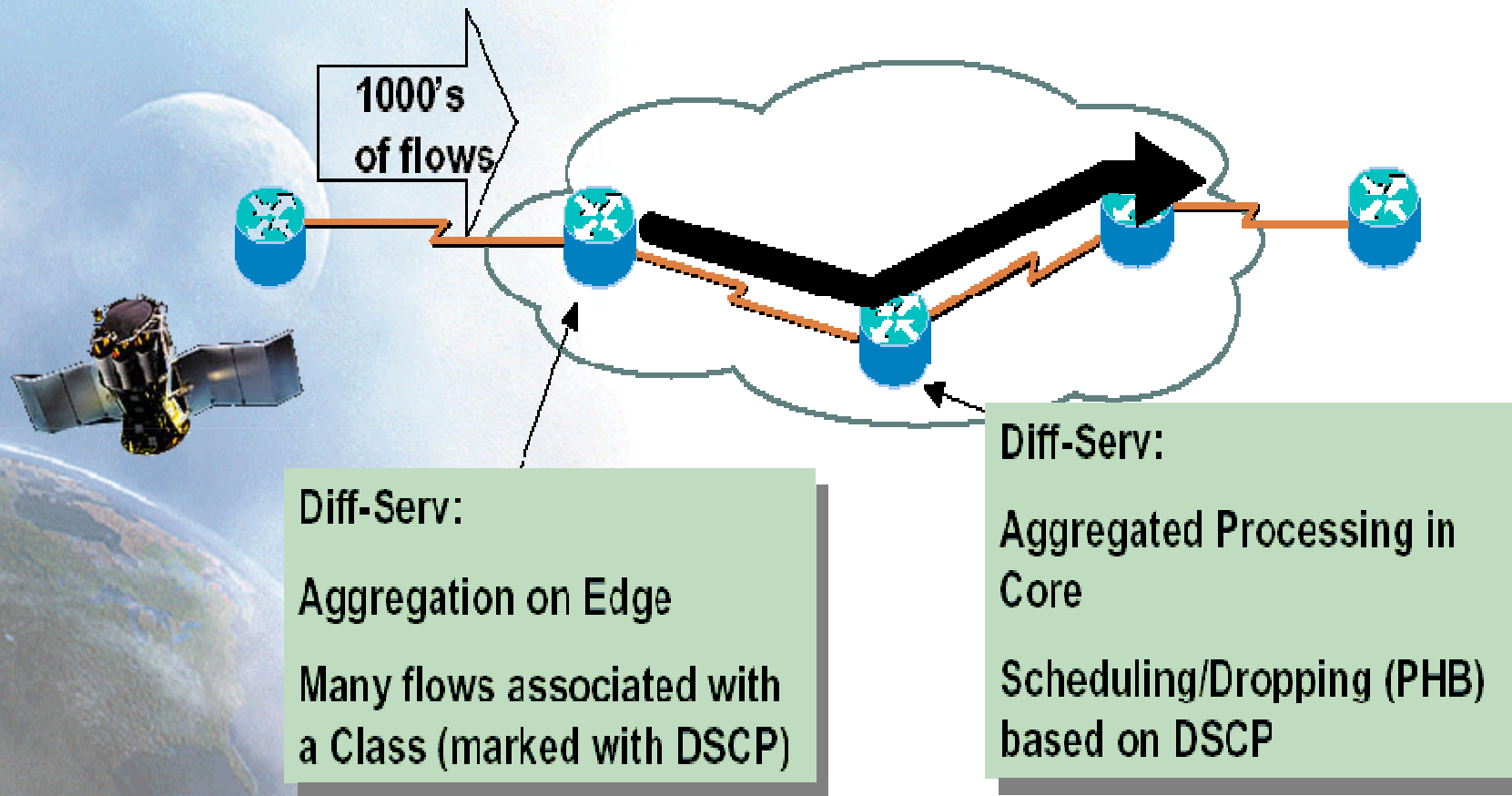
- Edge router

classificano il traffico in ingresso tramite un campo DS che associa al pacchetto un particolare comportamento di inoltro (PHB)

- Core router

applicano le strategie di scheduling previste per le diverse classi di traffico

DiffServ (2)



Classi di traffico DiffServ

- Expedited Forwarding (EF)

- ◆ Rate di uscita dal nodo non inferiore ad una rate configurabile
- ◆ Rate d'uscita indipendente dall'intensità di ogni altro traffico che transita nel nodo
- ◆ Rate di ingresso del traffico forzata ad essere sempre inferiore, o uguale, alla minima rate d'uscita garantita

- Assured Forwarding (AF)

- ◆ Offre differenti livelli di sicurezza di inoltra
- ◆ Per ogni livello è allocata in ogni nodo una certa quantità di risorse
- ◆ Ogni livello ha tre possibili livelli di precedenza di eliminazione



Domanda

- E' possibile avere il meglio dei “due mondi”, statefull e stateless?
- Fornire servizi “potenti” come quelli forniti da reti statefull
- Utilizzare tecniche scalabili e robuste come quelle usate nelle reti stateless



Soluzione proposta

- Il blocco base della soluzione proposta è il dominio Stateless Core (SCORE)
- Un dominio SCORE è definito come una regione continua nella quale solo gli edge router mantengono lo stato per flusso mentre i core router no
- Dal momento che gli edge usualmente operano a velocità più elevate e gestiscono molti meno flussi dei core router, questa architettura è altamente *scalabile*



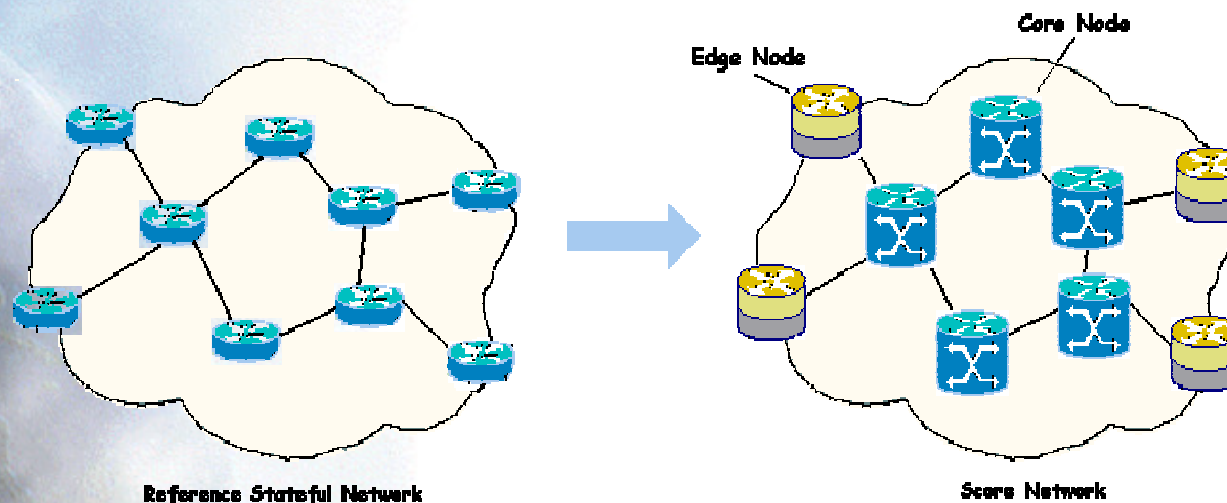
Architettura SCORE (1)

- Per raggiungere l'obiettivo di fornire servizi di rete potenti e flessibili come in una rete stateless è proposto un approccio chiamato “state elimination” e che consiste in 2 passi:
 1. Definire una rete di riferimento statefull che implementi i servizi desiderati
 2. Approssimare, o se possibile, emulare le funzionalità della rete di riferimento nella rete SCORE



Architetture SCORE (2)

- La rete SCORE ha un'architettura simile a quella Differenziata nella classificazione dei router
- Lo scopo è quello di approssimare i servizi forniti da una rete statefull



Architetture SCORE (3)

- Mostreremo come una rete SCORE può fornire ritardi end-to-end per flusso e garantire la banda come nei servizi IntServ
- Le soluzioni IntServ correnti assumono una rete “statefull” nella quale due tipi di stati per flusso sono necessari:
 - ◆ Forwarding state: che è usato dal motore di forwarding per assicurare percorsi di forwarding fissati
 - ◆ QoS state: che è usato sia dal modulo di controllo di ammissione nel piano di controllo che dal classificatore e dallo scheduler nel piano di dati



SCORE's problems

- Sebbene la SCORE ha molte vantaggi rispetto alle tradizionali soluzioni statefull essa ancora soffre delle limitazioni di scalabilità e robustezza se comparata con le soluzioni stateless
- La scalabilità è limitata dal fatto che i router sui confini devono essere statefull
- La robustezza è limitata dalla possibilità che un singolo edge o core router può malfunzionare e inserire informazioni errate nel pacchetto così da impattare sulle prestazioni di tutta la rete

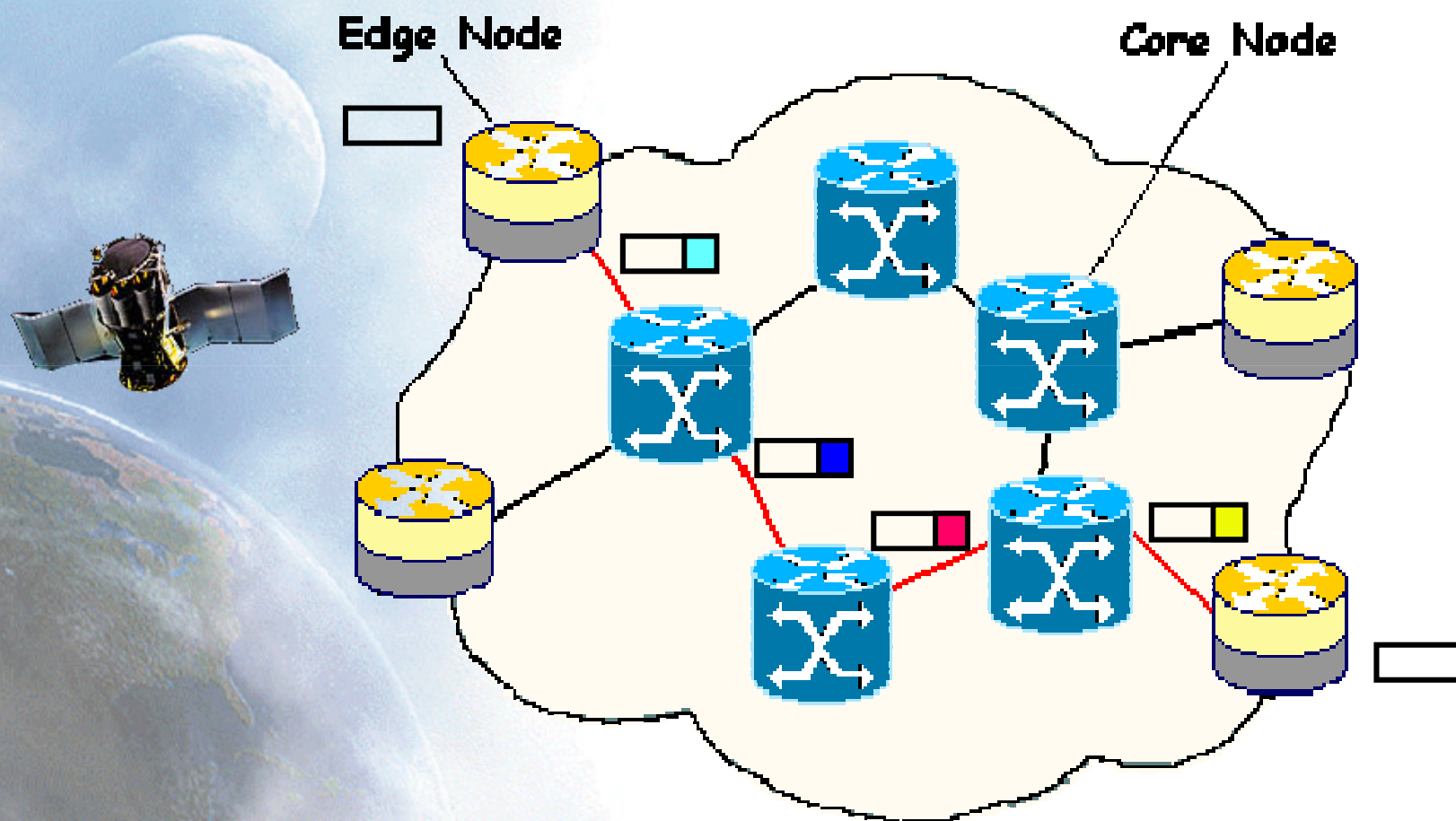


Dynamic Packet State (DPS)

- La tecnica chiave utilizzata per implementare una rete SCORE è la *Dynamic Packet State (DPS)*
 - ◆ Ogni pacchetto porta nel suo header alcuni stati che sono inizializzati dagli Ingress Router
 - ◆ I router interni processano ogni pacchetto che arriva basandosi sullo stato portato nell' header del pacchetto, aggiornano sia il loro stato sia lo stato nell' header del pacchetto prima di inoltrarlo verso un altro router
 - ◆ Alla fine l'egress router rimuove lo stato dall' header del pacchetto



Tecnica DPS



Algoritmi con DPS

- Grazie alla tecnica DPS è possibile utilizzare 2 algoritmi:
 - ◆ Uno per il piano dati per ***schedulare*** il pacchetto, chiamato ***Core Jitter Virtual Clock (CJVC)***
 - ◆ Uno per il piano di controllo per operare il ***controllo di ammissione (CAC)***.



Algoritmo di scheduling

- Il *Core Jitter Virtual Clock (CJVC)* è una variante del *Jitter Virtual Clock (JVC)*
- Il *JVC* lavora così:
 - ◆ Ad ogni pacchetto è assegnato un tempo **eleggibile** (**e**) e una **deadline** (**d**) al suo arrivo
 - ◆ Il pacchetto è trattenuto nel rate controller fin quando non diventa eleggibile
 - ◆ Lo scheduler ordina l'invio dei pacchetti eleggibili in accordo alla loro deadline
 - ◆ Per il k -esimo pacchetto del flusso i il suo tempo eleggibile $e_{i,j}^k$ e la sua deadline $d_{i,j}^k$ al j -esimo nodo sul suo percorso sono calcolati così:



Jitter Virtual Clock

$$e_{i,j}^1 = a_{i,j}^1$$

$$e_{i,j}^k = \max(a_{i,j}^k + g_{i,j-1}^k, d_{i,j}^{k-1}) \quad i, j \geq 1, k > 1$$

$$d_{i,j}^k = e_{i,j}^k + \frac{l_i^k}{r_i} \quad i, j, k \geq 1$$



- Dove l_i^k è la larghezza del pacchetto
- r_i è la rate del flusso
- $a_{i,j}^k$ è il tempo di arrivo del pacchetto al j -esimo nodo attraversato dal pacchetto
- $g_{i,j}^k$, immesso nell'header del pacchetto dal nodo precedente, è l'ammontare di tempo che il pacchetto è stato trasmesso prima della sua deadline, cioè la differenza tra la deadline del pacchetto e il suo attuale tempo di partenza al nodo j -esimo.

Core Jitter Virtual Clock (1)

- Il CJVC è basato sulla tecnica DPS
- L'idea chiave è quella di avere l'ingress router che inserisce i parametri di scheduling in ogni pacchetto
- Il router interno può allora prendere le sue decisioni basandosi su questi parametri, così da eliminare il bisogno di mantenere informazione per singolo flusso
- L'algoritmo ha bisogno di due variabili di stato per ogni flusso i:
 - ◆ r_i che è la rate riservata per il flusso i
 - ◆ $d_{i,j}^k$ che è la deadline dell'ultimo pacchetto del flusso i proveniente dal nodo j



Core Jitter Virtual Clock (2)

- Mentre è facile eliminare r_i non lo è per $d_{i,j}^k$
- La differenza è che la rate è uguale per tutti mentre la deadline è un valore dinamico che è calcolato iterativamente ad ogni nodo
- Però dal momento che $d_{i,j}^k$ è usato solamente in operazioni di massimo possiamo eliminarlo se possiamo assicurare che un altro termine in massimo non è mai più piccolo di $d_{i,j}^k$
- L'idea è quella di usare una variabile slack associata ad ogni pacchetto, denotata con δ_i^k tale che per ogni nodo interno lungo il percorso si abbia:



Core Jitter Virtual Clock (3)

$$\delta_i^1 = 0$$

$$\delta_i^k = \max \left(0, \delta^{k-1} + \frac{l_i^{k-1} - l_i^k}{r_i} - \frac{e_{i,1}^k - e_{i,1}^{k-1} - l_i^{k-1} / r_i}{h-1} \right) \quad k>1, h>1$$



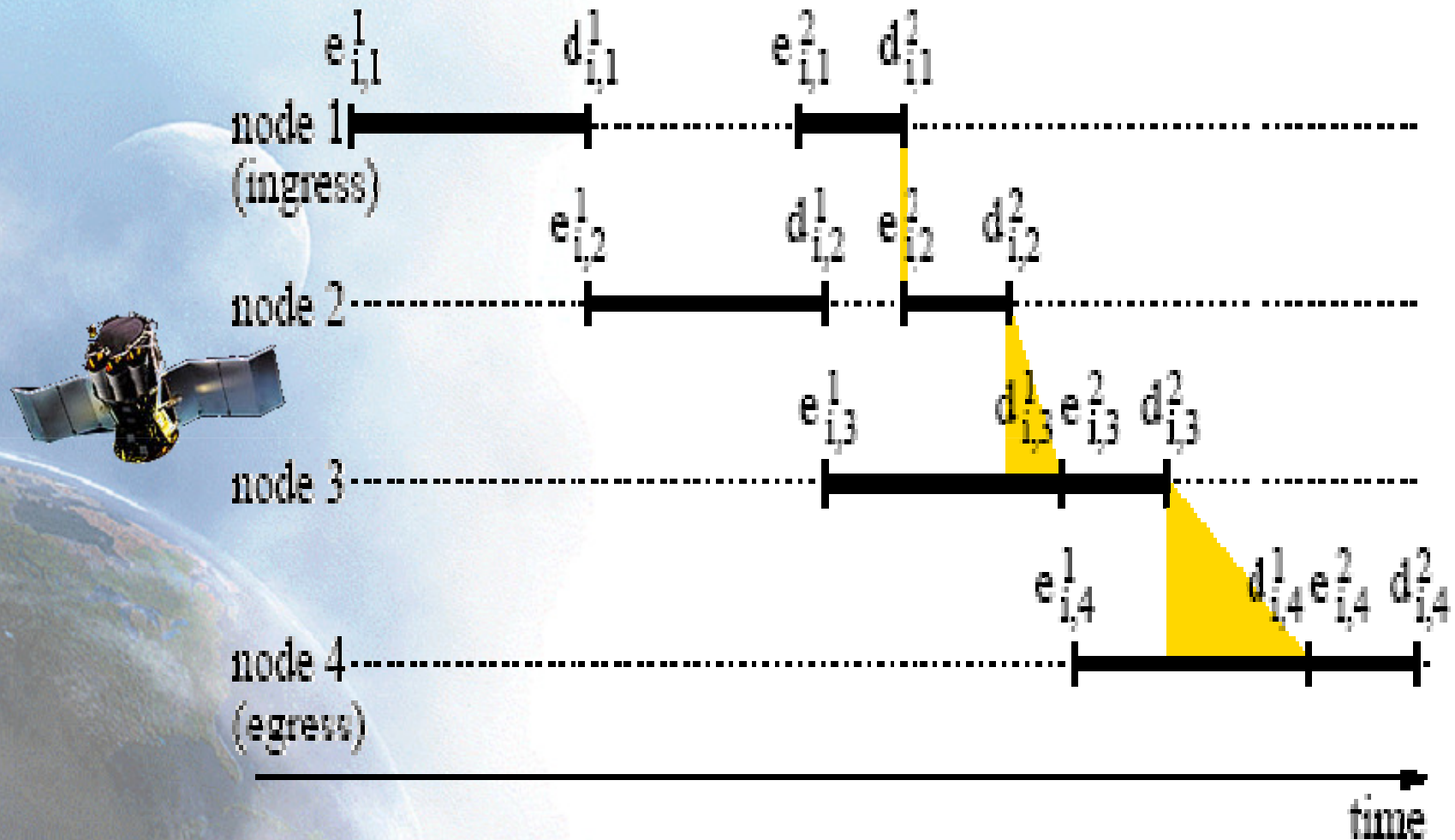
- Dove h è un contatore che viene incrementato ad ogni hop e viene calcolato usando l'algoritmo di CAC
- Di seguito è riportato l'algoritmo in pseudocodice

Core Jitter Virtual Clock (4)

- In questo modo il CJVC assicura il tempo eleggibile di ogni pacchetto p_i^k al nodo j non è più piccola della deadline del pacchetto precedente dello stesso flusso al nodo j , i.e. , $e_{i,j}^k \geq d_{i,j}^{k-1}$
- In più, lo scheduler Virtual Clock assicura che la deadline di ogni pacchetto è rispettata
- L'esempio di seguito nelle figure (a) e (b) fornisce alcune intuizioni che stanno dietro a questo algoritmo

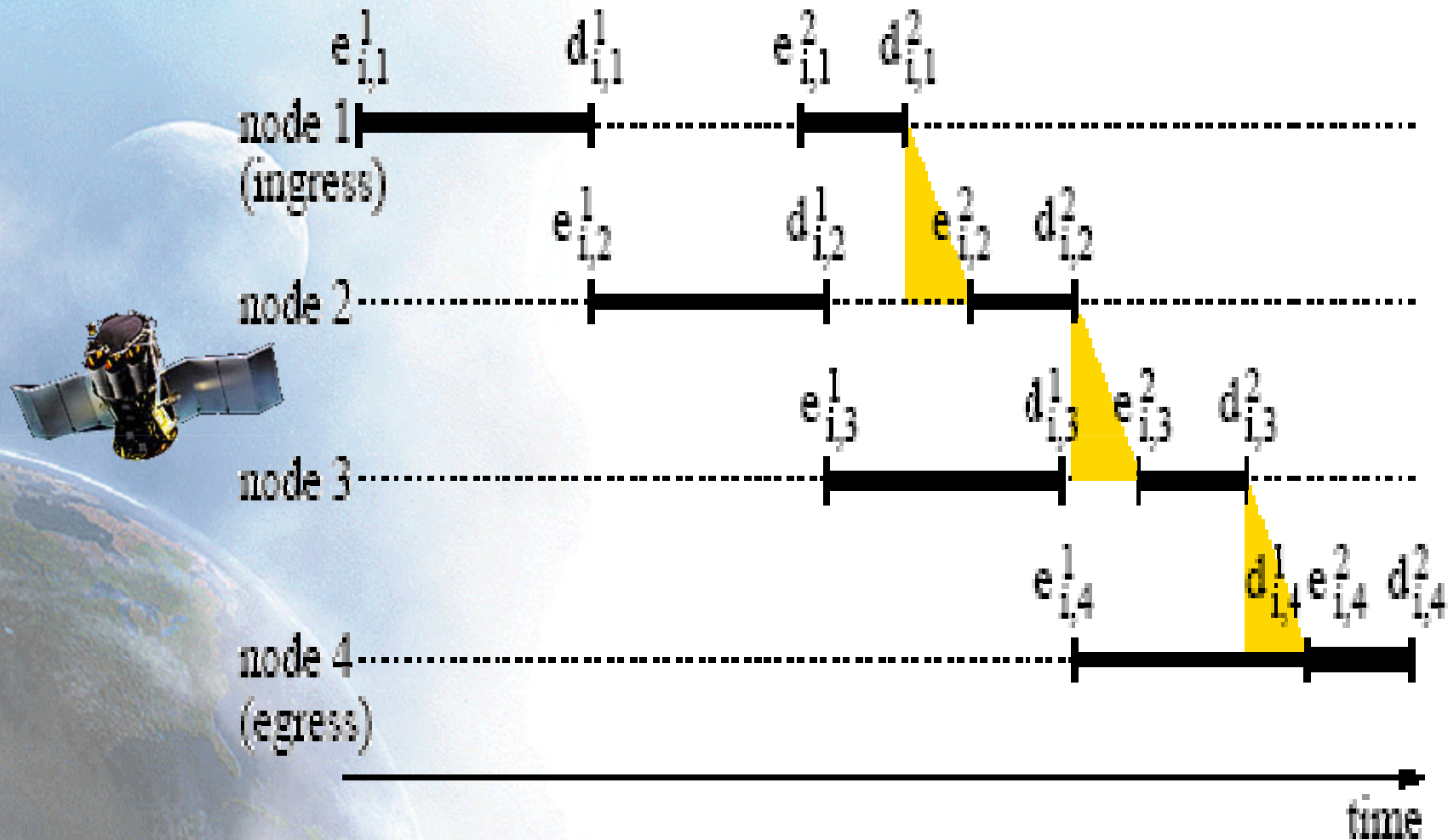


Figura (a): Jitter Virtual Clock



(a)

Figura (b): Core Jitter Virtual Clock



(b)

Osservazioni

- L'osservazione base è che con JVC non si tiene conto del ritardo di propagazione, la differenza tra il tempo eleggibile del pacchetto p_i^k al nodo j e la sua deadline al precedente nodo $j-1$, i.e. , $e_{i,j}^k - d_{i,j-1}^k$, non decresce mai lungo il percorso del pacchetto
- Consideriamo il secondo pacchetto in figura, con JVC la differenza $e_{i,j}^2 - d_{i,j-1}^2$, rappresentato dalla base del triangolo giallo, incrementa in j
- Introducendo la variabile di slack δ_i^k , il CJVC equalizza questi ritardi
- Mentre questo cambiamento può incrementare il ritardo del pacchetto agli hop intermedi non ha effetto sul limite ritardo end-to-end



Core Jitter Virtual Clock: Pseudocode

ingress node

on packet p arrival

$i = \text{get_flow}(p);$

if (first_packet_of_flow(p, i))

$e_i = \text{current_time};$

$\delta_i = 0;$

else

$\delta_i = \max(0, \delta_i + (l_i - \text{length}(p))/r_i -$
 $\max(\text{current_time} - d_i, 0)/(h - 1));$ /* Eq. (5.10) */

$e_i = \max(\text{current_time}, d_i);$

$l_i = \text{length}(p);$

$d_i = e_i + l_i/r_i;$

on packet p transmission

$\text{label}(p) \leftarrow (r_i, d_i - \text{current_time}, \delta_i);$

core/egress node

on packet p arrival

$(r, g, \delta) \leftarrow \text{label}(p);$

$e = \text{current_time} + g + \delta;$ /* Eq. (5.4) */

$d = e + \text{length}(p)/r$

on packet p transmission

if (core node)

$\text{label}(p) \leftarrow (r, d - \text{current_time}, \delta);$

else /* this is an egress node */

clear_label(p);

