Sistemi Ipermediali
Quality of Service (QoS) - II

Claudio E. Palazzi
Università degli Studi di Padova

Lucidi in parte a cura di: Marco Roccetti, Paola Salomoni, Stefano Ferretti

Resource Reservation

- Approccio pessimistico, spesso utilizzato in MM:
  - elimina conflitti sulle risorse
  - considera il caso peggiore
    - ✓ Esempio: il più lungo tempo di processing per una CPU
    - ✓ Esempio: la più alta banda
  - sotto-utilizzo della risorsa
  - utilizzata in Guaranteed Services

Resource Reservation

- Approccio ottimistico, può essere rischioso per applicazioni MM critiche:
  - considera il caso di utilizzo medio
    - ✓ Esempio: CPU allocata per il tempo di processing medio
    - ✓ Esempio: la banda alle volte può essere sotto-utilizzata
  - può capitare che la risorsa sia sovraccaricata
  - utilizzati rilevatori per determinare i casi di sovraccarico
  - risorse altamente utilizzate

Reservation Protocol

- Distribuiscono le richieste di QoS a tutti i sistemi coinvolti
  - il più delle volte orientati alla connessione
    - ✓ RSVP
  - Connessione via 2-way handshake
    - ✓ mittente
      - distribuzione delle richieste di QoS
    - ✓ ricevente
      - richieste che è in grado di soddisfare

RSVP

- RSVP è un protocollo di controllo della rete che consente alle applicazioni Internet di ottenere una certa qualità del servizio per le loro trasmissioni
  - Resource Reservation Protocol
  - anche noto come protocollo soft-state

- Questo approccio muta completamente le caratteristiche di tipo best effort fornite tipicamente da Internet poiché consente di richiedere ed ottenere riserva delle risorse su un certo path di rete

RSVP

- Protocollo di reservation di rete
- Supporta multicast
  - unicast trattato come caso particolare
- QoS fine-grained (a singoli flussi)
- QoS receiver oriented
  - il ricevente fa la sua richiesta di QoS
- soft-state (a metà tra hard state di circuit switching e “no state” di IP)
  - informazioni possono essere modificate periodicamente
- Definisce lo stato delle sessioni nei router
  - Riserva banda
  - trasparente ai router che non lo supportano

RSVP

- Come funziona:
  - Il sender invia ai destinatari un messaggio detto PATH che, percorrendo tutti router dal sender ai vari destinatari, fissa il reverse path che i messaggi di riserva-risorse dei destinatari dovranno poi percorrere (viene creato un multicast tree)
  - Ciascun router lungo il cammino prende atto delle caratteristiche del media che il sender erogherà (Tspec) e definisce la propria disponibilità di risorse (larghezza di banda, memoria per lo store&forward, ecc)

RSVP

- Come funziona:
  - Ogni receiver ricevuto il messaggio di PATH, manda attraverso il multicast tree un messaggio di riserva risorse in accordo ai propri bisogni. Il messaggio è detto RESV
    - ✓ RESV contiene il Tspec del sender ed il Rspec del receiver per specificare tipologia di richiesta
  - Ogni router sul percorso controlla RESV e tenta di allocare le risorse richieste. Se ce la fa Ok e spedisce il RESV al prossimo router sul cammino, altrimenti respinge il messaggio RESV al receiver

RSVP

- Come funziona:
  - Se tutto va bene, l'intero PATH (e relative risorse) sono riservate dal sender al receiver
  - Per mantenere valida la prenotazione, i receiver devono rinnovarla rispedendo RESV periodicamente (30 sec)
  - Se link o router falliscono non è grosso problema, perché anche il multicast tree è tenuto aggiornato
    - ✓ il sender reinvia PATH periodicamente
    - ✓ ogni nuovo invio di PATH, scatena i RESV dei receiver

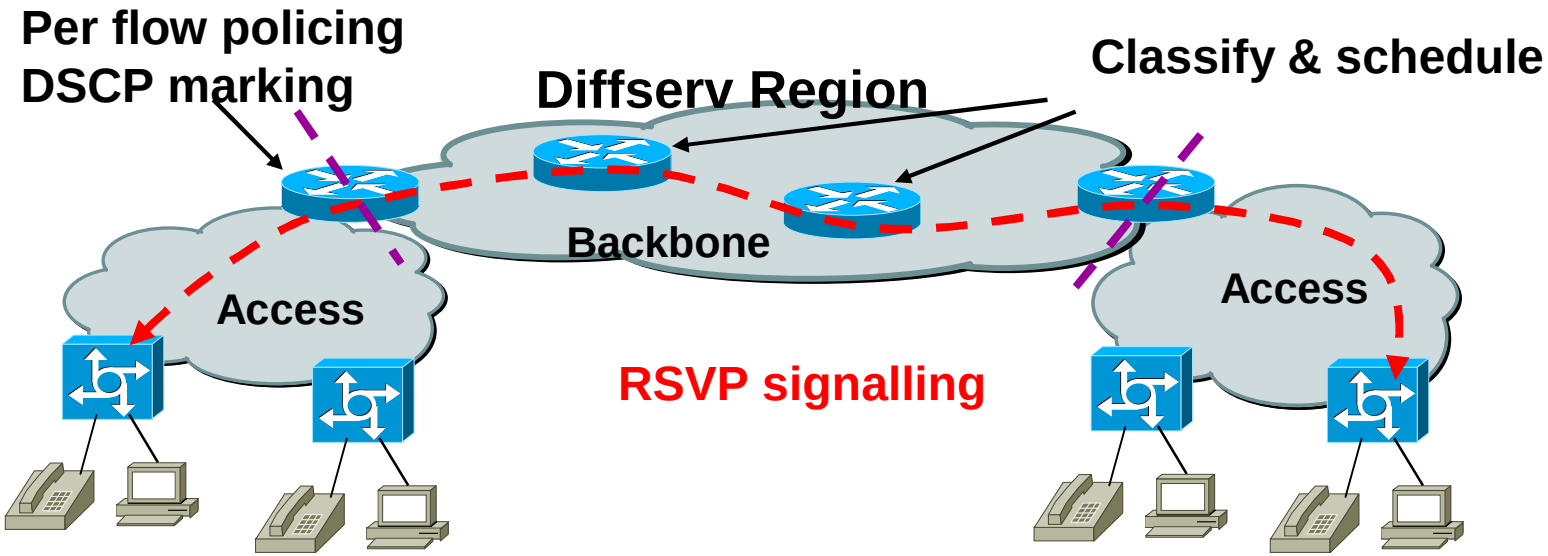
RSVP

- Come funziona:
  - E' possibile aggregazione (ottimizzazione) riserva di risorse ai router: un receiver ha già avuto accordato un ritardo garantito di 100 msec, subentra un altro receiver che vuole 150 msec sullo stesso router per lo stesso flusso, router non riserva nulla in più, è già a posto
  - E' robusto, soft state
    - ✓ lo stato s'annulla da solo, va rinfrescato periodicamente
  - Problema: non è molto scalabile

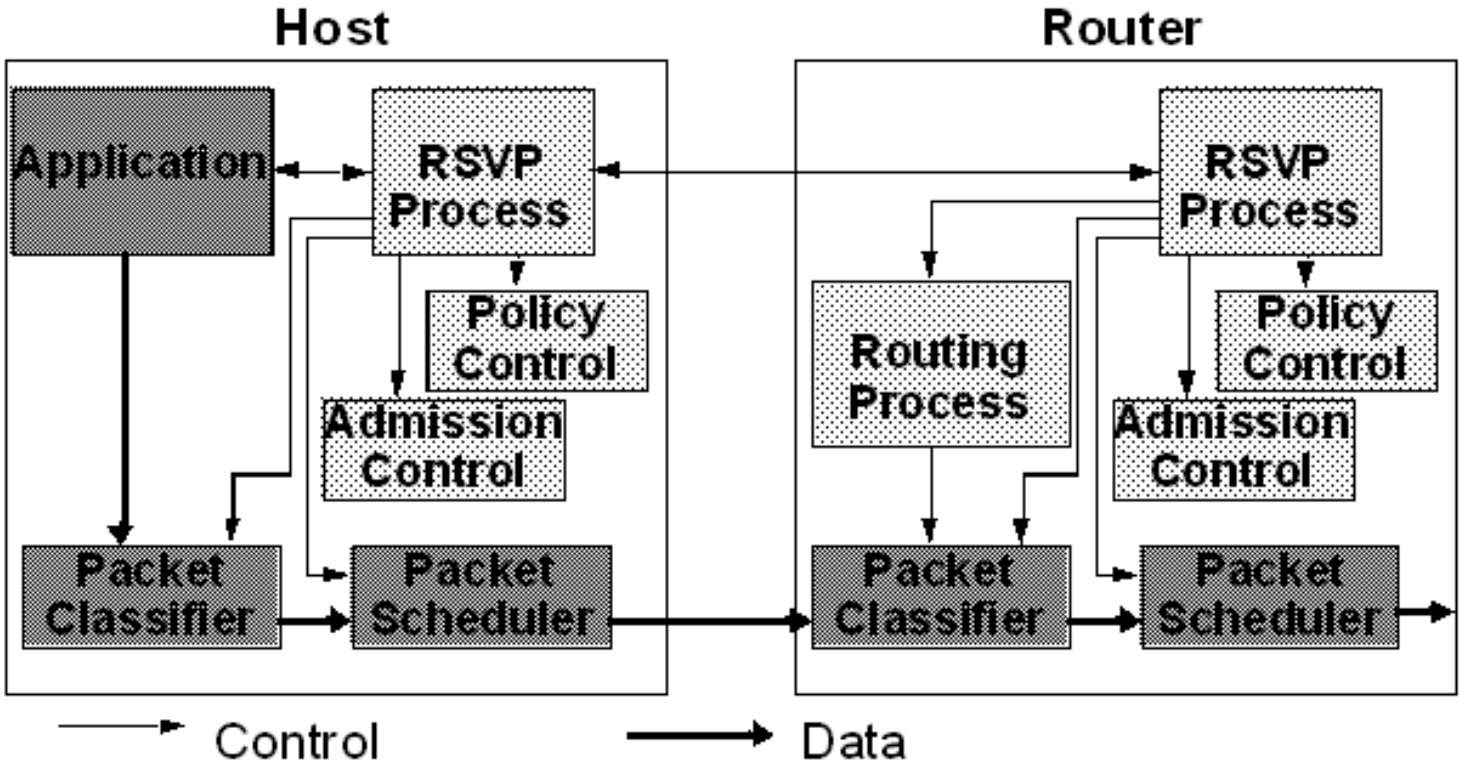
RSVP

- Non scala:
  - Link OC-3 a 2.5 Gbps che trasporta l'equivalente di connessioni audio a 64 Kbps ciascuna.
    Ne consegue $2.5 \times 10^9 / 64000 = 39.000$
  - Risultano 39000 connessioni di cui conservare lo stato ai router. Stato che va rinfrescato periodicamente. Il router deve classificare, fare il policing, gestire le code, prendere decisioni d'ammissioni, fare pricing, come descritto dopo
  - Problema ingestibile, devo andare verso QoS coarse-grained (flussi aggregati)

RSVP: usato solo per signalling in Diffserv



Componenti


Host
Router
Application
RSVP Process
Policy Control
Admission Control
Packet Classifier
Packet Scheduler
Routing Process
RSVP Process
Policy Control
Admission Control
Packet Classifier
Packet Scheduler
Control
Data

Data Path

- Packet Classifier
  - determina la classe di QoS per ogni pacchetto
- Packet Scheduler
  - gestione dell'accesso al livello-2 per garantire la QoS
  - schedula i pacchetti

Moduli di Controllo

- Admission Control
  - Gestione della Risorsa
  - Decide se il sistema è in grado di supportare il traffico richiesto
- Policy Control
  - Opzionale
  - Determina se il richiedente ha potere di fare la richiesta

Data Flow

- Sessione RSVP per un dato ricevente: flusso di dati con una particolare destinazione
  - definita da IP_addr_ricevente e da IP protocol
    - ✓ 1 ricevente per sessione
    - ✓ più potenziali mittenti per sessione
    - ✓ opzionalmente possibile considerare anche la porta (se si hanno più flussi per il ricevente)

Reservation Styles

- Insieme di opzioni che caratterizzano la riserva delle risorse
- Le *reservations* per mittenti differenti all'interno della stessa Sessione possono essere:
  - distinte
  - condivise
- Mittente considerato:
  - mittente specifico
  - *wild-card*

Reservation Styles

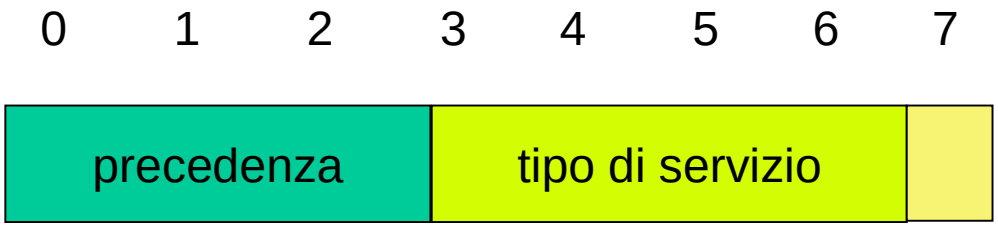
Reservation Style	Sender Selection	Reservation Type	Descrizione
Fixed Filter (FF)	**Esplicito**	**Distinto**	**Ric. Sceglie un particolare mitt. dal quale vuole ricevere**
Shared Explicit (SE)	**Esplicito**	**Condiviso**	**Esplicita scelta dei mitt., Res. Condivisa**
Wildcard Filter (WF)	**Wildcard**	**Condiviso**	**Ric. Crea una singola Res., condivisa da tutti i mitt.**

QoS per IP

- Alla base della QoS per IP ci sono alcune informazioni nell'header del pacchetto IP:
  - indirizzi sorgente e destinazione
  - protocollo (TCP, UDP…) e, in particolare,
  - i bit che possono essere dedicati alla QoS

QoS e IP

- C’è un byte per il Type of Service (TOS), così strutturato



Integrated Services (IntServ)

- Integrated Services vuole fornire garanzie ai singoli flussi di dati

- Ad oggi sono definiti due servizi
  - Guaranteed Service: che offre dei limiti quantitativi alla latenza dei flussi che rispettino le specifiche
  - Controlled Load Service: mira ad offrire un servizio equivalente in latenza e perdita di pacchetti a quello fornito da una rete best effort poco carica

IntServ

- IntServ richiede il mantenimento di informazione di stato su ogni nodo attraversato
- In genere (ma non obbligatoriamente) IntServ è associato con il protocollo di segnalazione RSVP
- (Resource ReSerVation Protocol – RFC 2205)

IntServ

- Realizzano un sistema c*onnect & reservation*
- Ad ogni flusso di dati viene associata una QoS richiesta
- Mbone: basata su un sistema IntServ per funzionare
  - Multicast backbone usato per videoconferenze (da rete di università ma non solo)
- Esistono servizi basati su questa strategia, sono invasivi, poco scalabili

Differentiated Services (DiffServ)

- Diffserv mira a fornire garanzie ad aggregazioni di flussi, Qos di tipo coarse grained
- Richiede perciò memoria di stato solo alla frontiera del dominio (dove avviene aggregazione/classificazione)
- Alla frontiera i pacchetti vengono classificati e viene loro assegnato un valore, scritto nell'header IP, chiamato DiffServ Code Point (DSCP)
- In base a tale valore i pacchetti avranno un trattamento opportuno (per hop behavior, PHB) ad ogni singolo tratto della rete

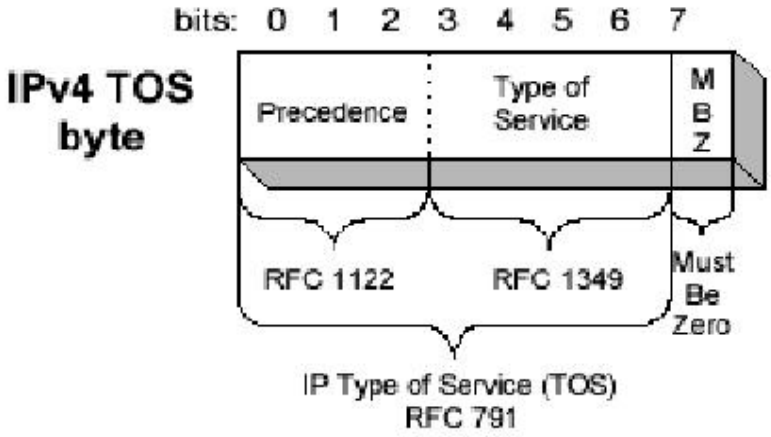
Differentiated Services (DiffServ)

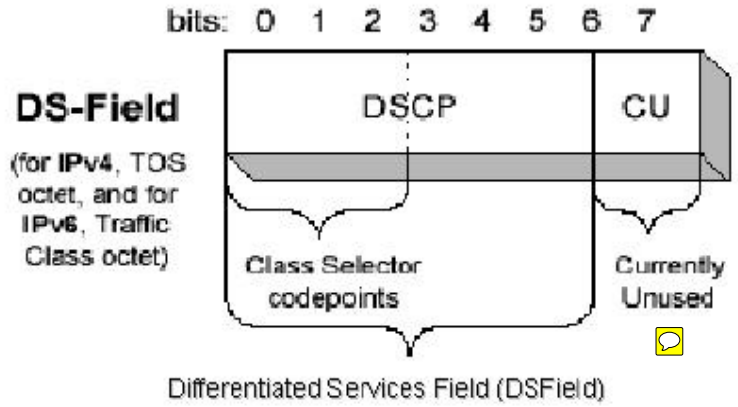
- Siccome IntServ fa fatica a gestire i flussi singolarmente, DiffServ alloca risorse ad un basso numero di classi di traffico ragionando sui singoli pacchetti (Premium e best effort)
- Piuttosto che usare protocolli appositi (come RSVP) i pacchetti vengono marcati in modo tale che i router li riconoscano come appartenenti ad un classe
- Solitamente ciò è fatto dal router al confine del dominio amministrativo in cui è ricompreso l'utente che genera pacchetti

Differentiated Services (DiffServ)

- Ad esempio, il router di confine di un dato ISP marca come Premium i pacchetti di un utente che ha pagato per quel servizio (non necessariamente tutti i pacchetti di quell'utente debbono essere Premium)
- Ai pacchetti marcati, i diversi router che li gestiranno applicheranno un particolare comportamento (per hop behavior) teso a garantire il servizio richiesto

DiffServ


bits: 0 1 2 3 4 5 6 7
IPv4 TOS byte
Precedence
Type of Service
M B Z
RFC 1122
RFC 1349
Must Be Zero
IP Type of Service (TOS)
RFC 791


bits: 0 1 2 3 4 5 6 7
DS-Field
(for IPv4, TOS octet, and for IPv6, Traffic Class octet)
DSCP
CU
Class Selector codepoints
Currently Unused
Differentiated Services Field (DSField)

DiffServ

- Due servizi (PHB), in particolare, sono stati definiti, il primo dei quali è:
  - Expedited Forwarding (EF): un servizio garantito con caratteristiche simili a quello di una linea punto-punto (minimum delay and loss)

DiffServ

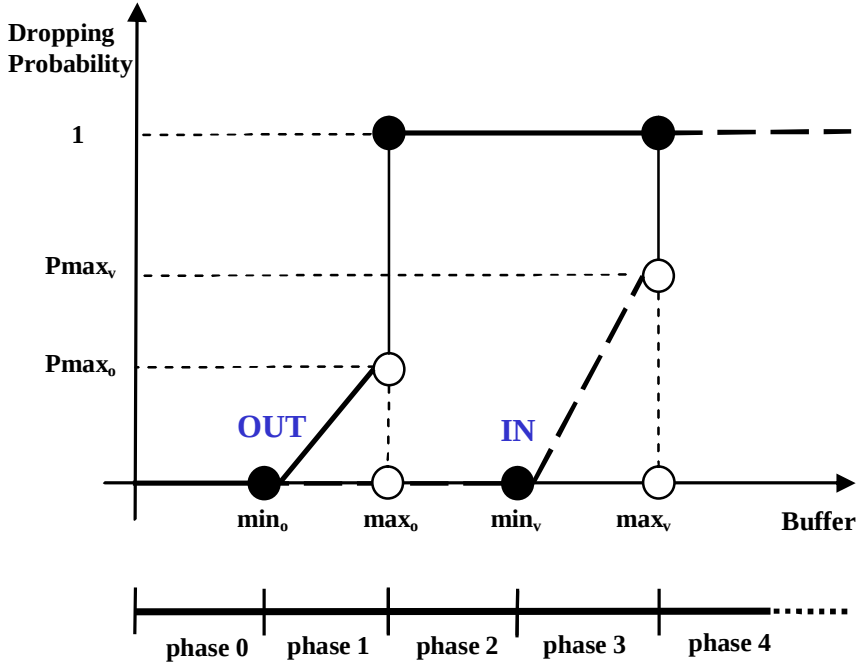
- EF potrebbe essere garantita tramite :
  - solo un numero massimo di pacchetti EF è ammesso a entrare dai border router
  - dare ai pacchetti EF stretta priorità su tutti gli altri
  - fare Weighted Fair Queuing tra i pacchetti EF e i pacchetti non EF con peso tale da garantire minimo delay e loss a EF (anche i non EF vengono trasmessi)

DiffServ

- Oltre a EF c'è anche AF:
  - Assured Forwarding (AF): offre un servizio con differenti probabilità di scarto all'interno dello stesso aggregato di pacchetti

DiffServ

- AF utilizza il seguente meccanismo:
  - dropping dei pacchetti di una data sorgente con probabilità crescente alla crescita della lunghezza media della coda (RED)
  - RED with I/O o RIO, ad esempio con due diverse curve di drop probability
    - ✓ la prima curva (OUT) ha soglia di drop più bassa
    - ✓ la seconda curva (IN) ha soglia minima più alta di out



DiffServ

- Il traffico IN:
  - non subisce tagli, cui probabilmente soggiace OUT per determinati valori di traffico
  - solo se si supera la soglia più elevata di IN, iniziano a essere scartati pacchetti di IN
  - Si parla di IN/OUT perché customer può fare contratto con la rete per una certa quantità di pacchetti AF. Tuttavia se poi manda più pacchetti di quelli specificati, quelli che eccedono sono trattati con curva di OUT

- RIO può essere generalizzata a WRED (Weighted RED): più classi di servizio servite con diverse curve di drop. Il valore del DSCP è usato per selezionare una curva di drop probability

DiffServ

- Implementano un sistema di differenziazione del traffico
- La QoS diventa una caratteristica *per classi di pacchetto* e non *per flusso*
- Si definiscono dei comportamenti *Hop-by-Hop* per poter ottenere una QoS locale
- Meno invasivi
- Più Scalabili

Caso di Studio 3 (CS3)

Giochi Online: Consistenza, Interattività ed Equità

C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti

Introduction

(CS3)

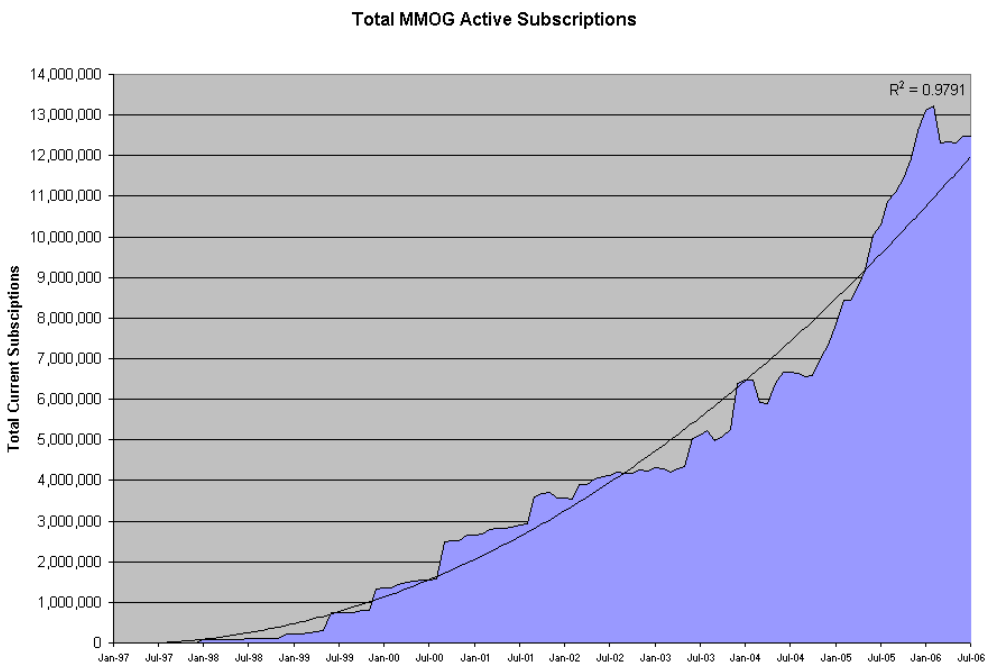
- Massive Multiplayer Online Games (MMOGs) represent one of the most challenging research areas in the field of interactive multimedia distributed systems

- Large scale game platforms need
  - ***Scalable*** distributed architectures
  - ***Communication systems*** that ensure a **rapid** game event **delivery** (network and computational issues)

- A high *playability* (interactivity) degree should be guaranteed independently of
  - user's location
  - type of connection (wired, wireless)
  - utilized device (PC, game console, PDA, cellphone)
  - number of simultaneous players

Multiplayer Online Games (MOGs): Why?

- Very challenging research area
- Shares a family of problems with several traditional research fields in computer science:
  - Distributed simulations
  - Military simulations
  - Virtual reality
  - Safety ensuring applications
  - Homeland Security
- Exploding market



Main Problem: Synchronization Offset

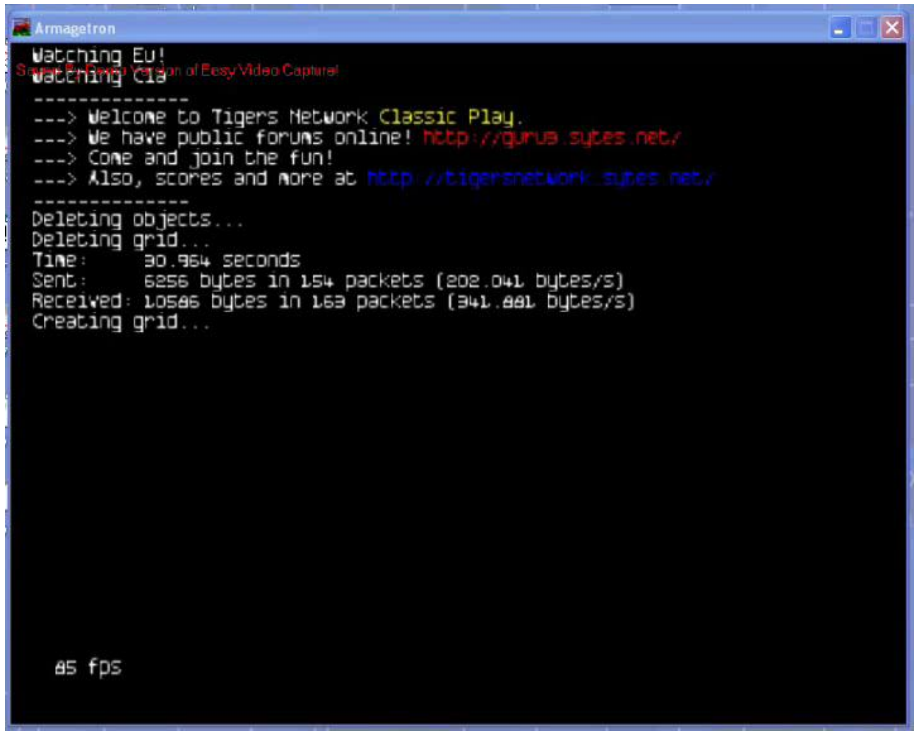
- Delays in the synchronization of all the participants are caused by:
  - Physical distance among nodes
  - Network conditions
  - Computational load

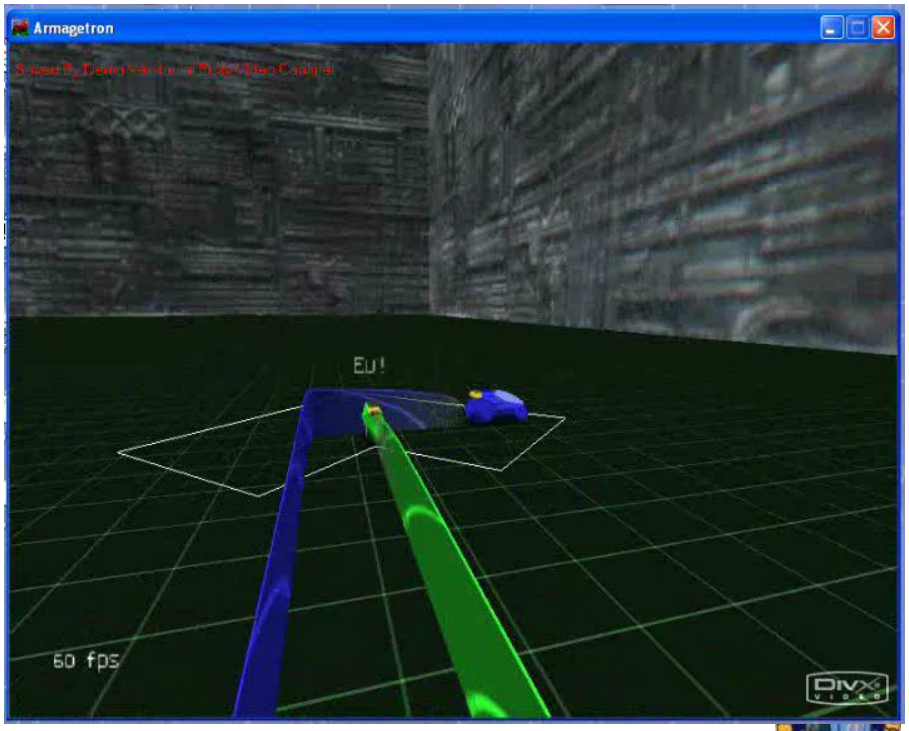
- Execution_Time – Generation_Time = Synch-Offset
  - Should remain under the human perception threshold

Example of Synchronization Offset

- Bringing games into the *online domain* is not straightforward

Armagetron

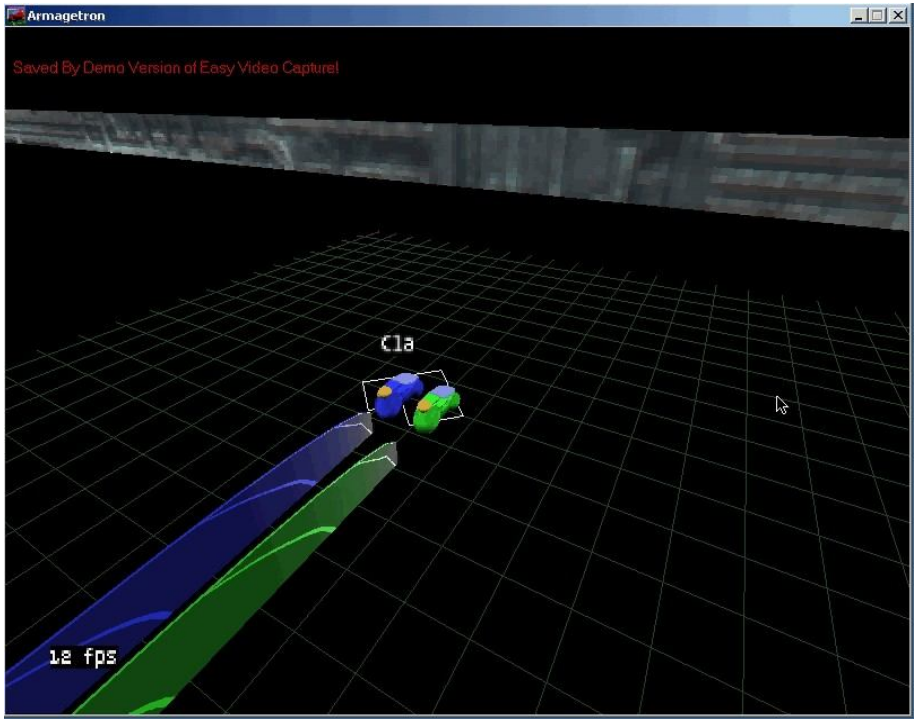




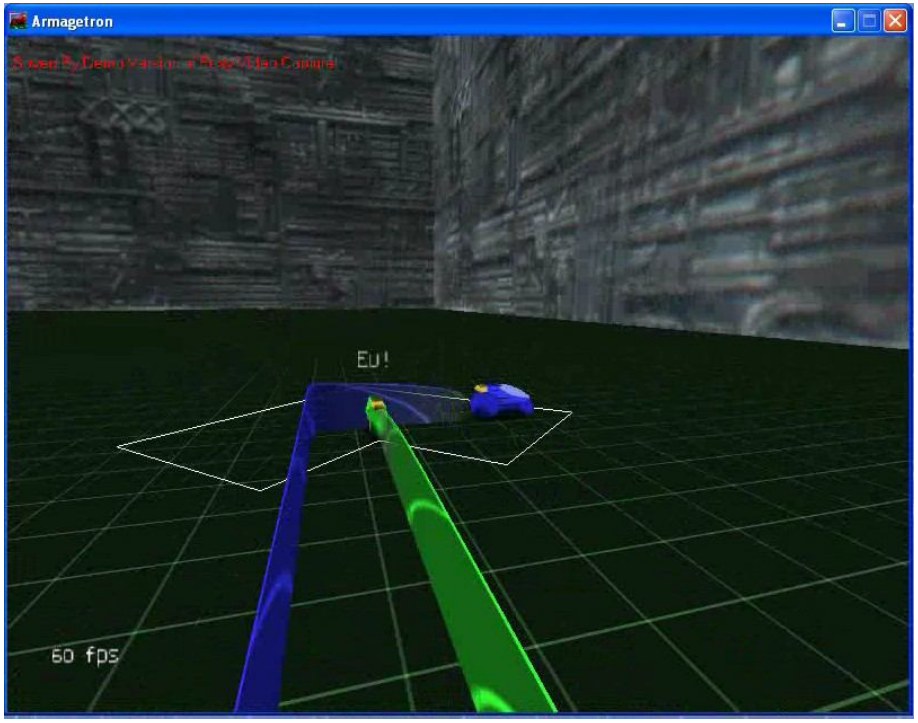
Different Rendering Times

- Client-server architecture
- Different delays between each player and the server
  - The green player is closer to the server than the blue one

GREEN PLAYER'S VIEW



BLUE PLAYER'S VIEW



Another Example of Synch Offset

Hitboxes represent where players have to aim when shooting

Network delays detach server's hitboxes from characters on screen

MMOG's Requirements

- ***Scalability**:*
  - the number of contemporary players should not be bounded
  - Massively Multiplayer Online Games (MMOGs)

- ***Interactivity**:*
  - multiplayer gaming is extremely delay-sensitive

- ***Consistency**:*
  - uniformity of game state view in all nodes

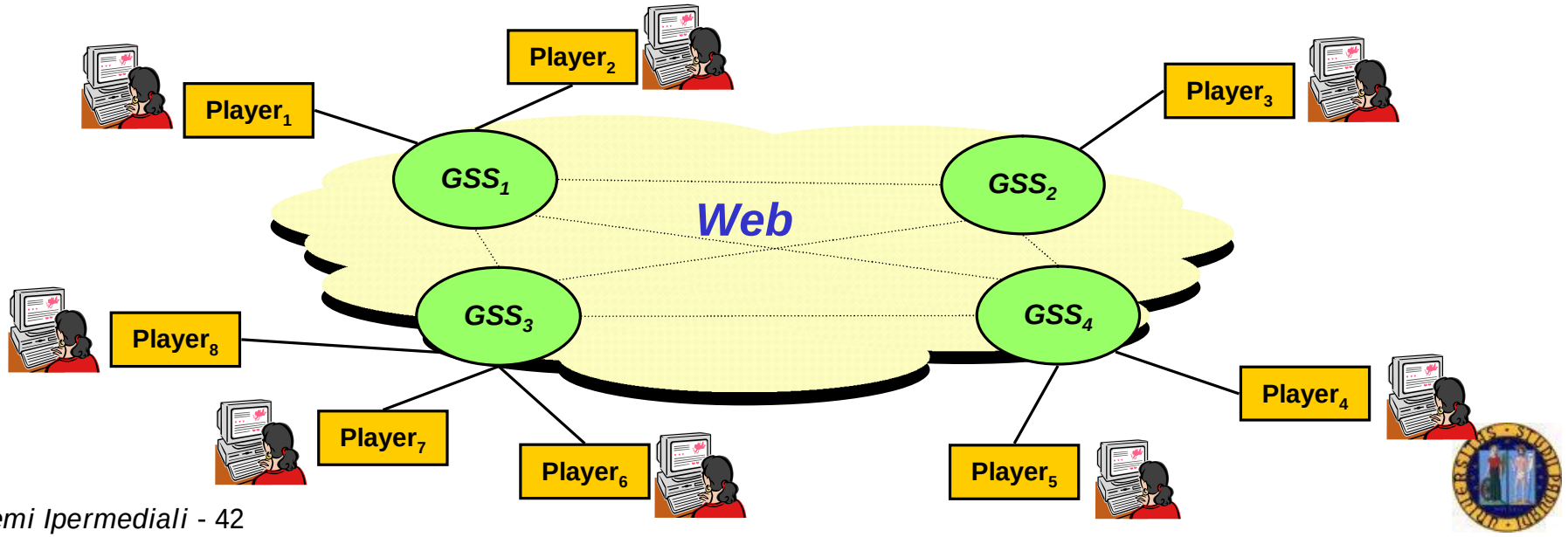
- ***(Network) Fairness**:*
  - simultaneous game evolution regardless of network conditions

Mirrored Game Server Architecture (CS3)

- To address **scalability**, a solution (in research) consists in deploying over the network several *Game State Servers* (*GSSs*)
- The *game state* is replicated at each *GSS*
- Communication:

  Player – GSS: **Client-Server**

  GSS – GSS: **P2P**



Interactivity vs Consistency (CS3)

- Interactivity: multiplayer gaming is extremely delay-sensitive
  - External stimuli have to be processed within a fixed time deadline
- Consistency: contemporary uniformity of the game state view at all the *GSSs*
  - Can be attained through *reliable*, *totally ordered* game event delivery schemes → ***interactivity*** may result ***jeopardized***
- An ***Event Delivery Synchronization Service*** is needed to maintain:
  - *Consistency* of the *game state* (among *GSSs*)
  - *Interactivity* (among players)
- Exploiting the *semantics* of the game can be put in good use to **relax constraints** (*total order* + *reliability*) and ***gain interactivity***

Relaxing the Reliability Properties in the Game: Obsolescence **(CS3)**

- The importance of **certain** game events diminishes when new game events are generated that annul the previous ones

- *Example*:
  - Given e_1, e_2 that represent two subsequent independent movements of *Alice,* if $T_g(e_i)$ is the time of generation of e_i, than $T_g(e_1) < T_g(e_2) \rightarrow e_2$ supersedes e_1

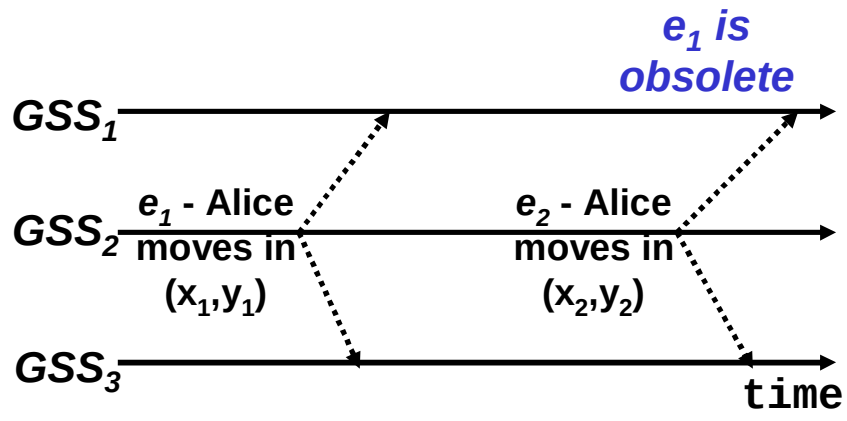
  - If a given *GSS* does not receive e_1, but receives e_2 straight after, still a consistent state is maintained at that *GSS*

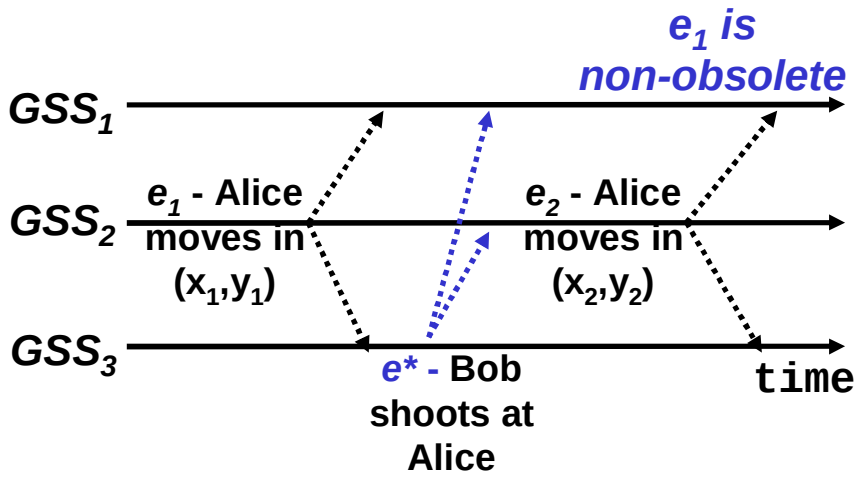
Obsolescence – Stated Simply

(CS3)

- e_2 supersedes e_1 (***e_1 is obsolete***) when the exec. of e_2 without the exec. of e_1 does not modify the final game state

- *Not all events become obsolete* during the game evolution
  - A further event *semantically correlated to* e_1 (ex: *Bob shoots at Alice*), interleaved between e_1 and e_2 may break the obsolescence relation





Trading Reliability for Interactivity (CS3)

- Each *GSS* monitors the interactivity degree of the delivered game events
  - in case of *lousy interactivity* → obsolete game events may be dropped
- This dropping approach speeds the game event processing up and decrease network traffic, thus improving interactivity
- Dropping obsolete events may generate sporadic small "jumps" in the game evolution on the user's screen
  - Becomes acceptable with online games where the main issue is to avoid the slow down of the game evolution
- ***Consistency*** of the game state view is not affected at all

Schemes for Interactivity Maintenance (CS3)

- *Interactivity Restoring*:

  in case of *disrupted interactivity* → stabilization mechanism which drops obsolete events
  - *ON-OFF*: binary dropping mechanism

- *Interactivity-Loss Avoidance* (*ILA*):

  avoids interactivity loss before it happens by discarding in advance some obsolete events with a certain probability

ON-OFF

(CS3)

- We introduce a threshold representing the limit above which the interaction among players is not guaranteed
  - *Game Interaction Threshold* (*GIT*)
- To control the interactivity degree we define
  - *Event Generation Time* $T_g(e)$: time of generation of a game event *e*
  - *Event Delivery Time* $T_d^i(e)$: time of delivery of *e* at GSS_i
  - *Game Time Difference*:

$$GTD_i(e) = T_d^i(e) - T_g(e)$$

- For each delivered event *e*
  - if $GTD_i(e) < GIT$
    - ✓ Phase OFF: normal delivery operations
  - else
    - ✓ Phase ON: discard all obsolete events

Interactivity-Loss Avoidance (ILA)

(CS3)

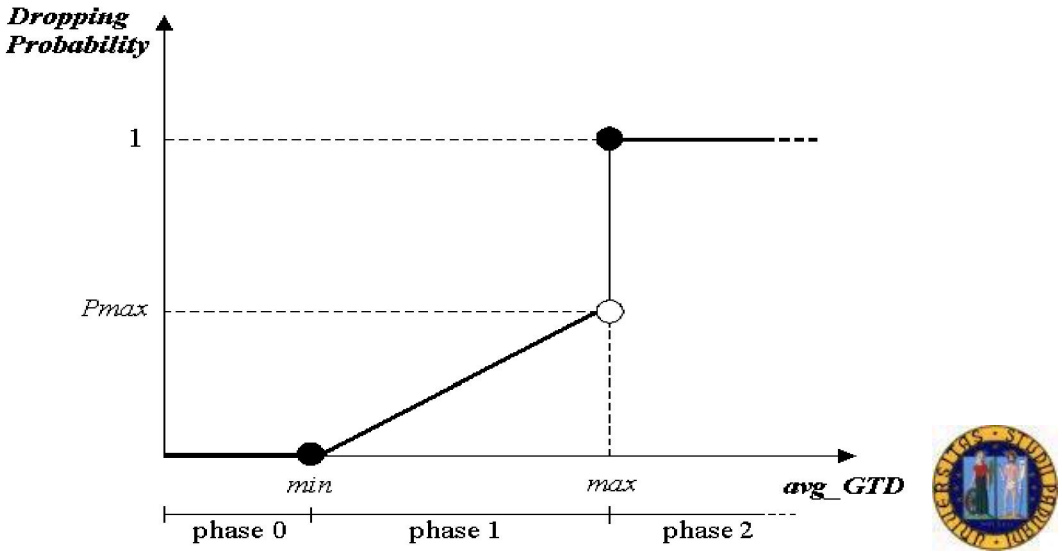
- The approach is inspired to the *Random Early Detection* (RED) algorithm, an active congestion avoidance mechanism enforced at routers
- ILA utilizes a uniformly distributed dropping function with a dropping probability which depends on the perceived responsiveness at *GSSs*
- The idea is to monitor the interactivity level of the system and, when required, preempt the loss of interactivity discarding some obsolete events

ILA Algorithm

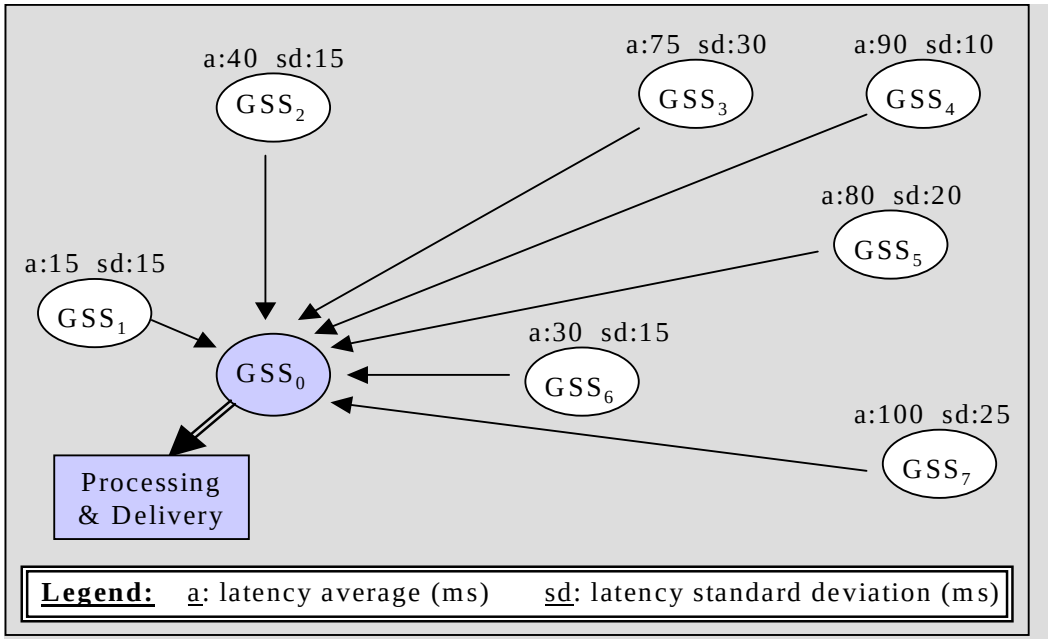
(CS3)

- Each *GSS* determines the *GTD* of each received event and feeds a low pass filter to compute an average *GTD* value
  - *avg_GTD = avg_GTD + w * (sample_GTD – avg_GTD)*
- *avg_GTD* ∈ [*0, min*): normal operations, no event drops
- *avg_GTD* ∈ [*min, max*): obsolete events are dropped with a certain probability *p*
- *avg_GTD* ∈ [*max, ∞*): *p* is set equal to 1, all obsolete events are discarded

$$p = \frac{Pmax \times (avg\_GTD - min)}{(max - min)}$$



Simulation Assessment (CS3)



Number of GSSs	Corresponding GSSs employed
4	GSS_1, GSS_2, GSS_3 and GSS_4
5	GSS_1, GSS_2, GSS_3, GSS_4 and GSS_5
6	GSS_1, GSS_2, GSS_3 GSS_4, GSS_5 and GSS_6
7	GSS_1, GSS_2, GSS_3 GSS_4, GSS_5, GSS_6 and GSS_7

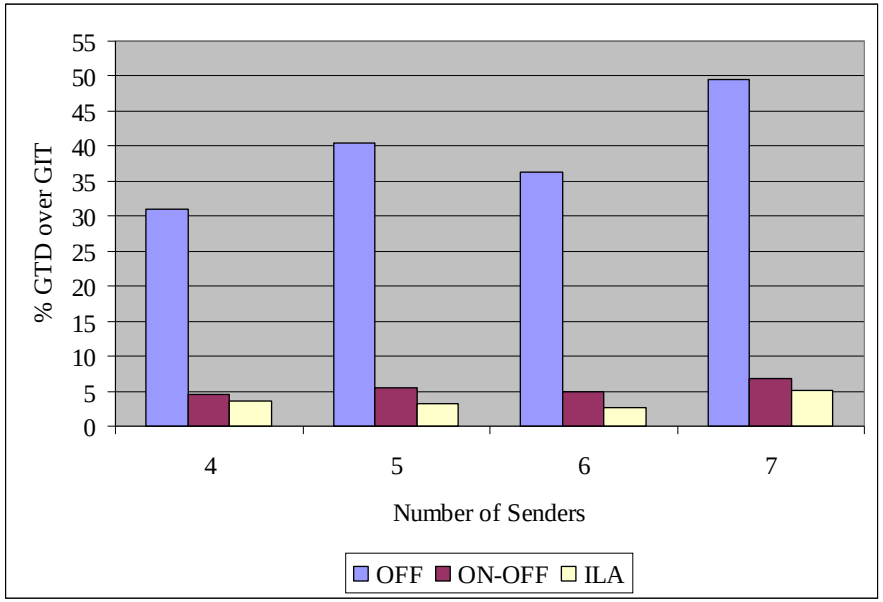
GSSs involved in the various simulations

- We compare three schemes:
  - ILA
  - ON-OFF (Restoring Approach)
  - OFF (Completely Reliable)
- Parameters inspired by the scientific literature on games

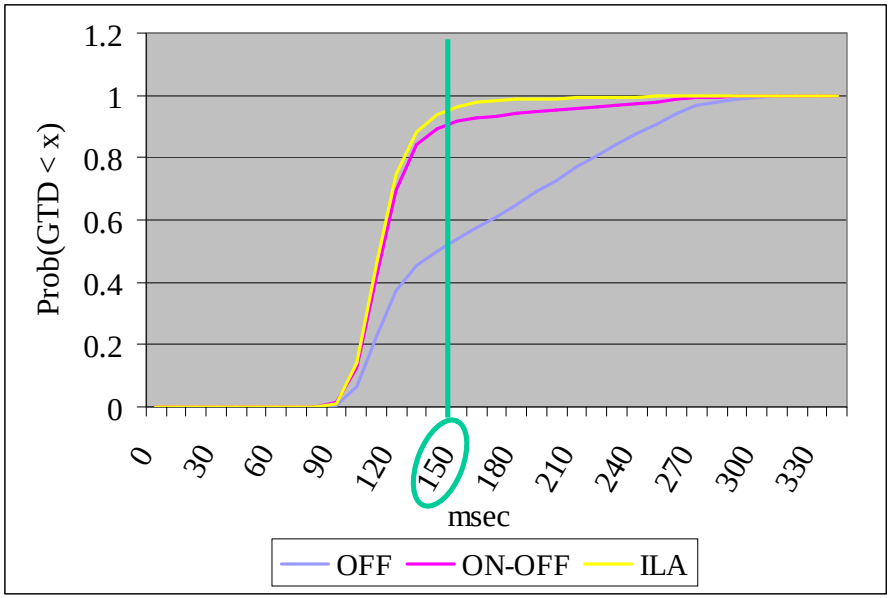
Simulation Results

(CS3)

Event percentage with GTD ≥ GIT



Cumulative function of the GTDs



- Obsolescence discarding schemes (ILA and ON-OFF) improve the interactivity degree w.r.t. traditional ones (OFF scheme)

- Percentage of events with a GTD ≤ GIT (considering GIT = 150ms):
  ILA -> 93.86% ON-OFF -> 89.40% OFF -> 49.94%

Simulation Results

(CS3)

Statistics of the various synchronization schemes

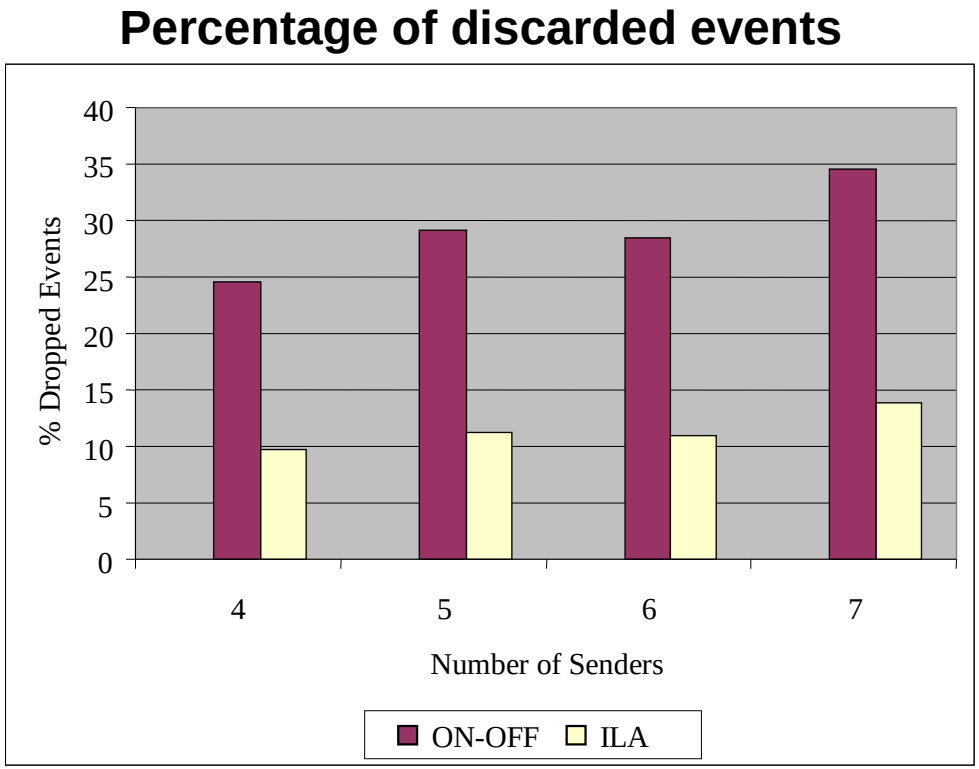
int = 30	4 GSSs			5 GSSs			6 GSSs			7 GSSs		
	OFF	ON-OFF	ILA	OFF	ON-OFF	ILA	OFF	ON-OFF	ILA	OFF	ON-OFF	ILA
MAX	324	324	325	325	324	277	318	319	278	345	345	300
MIN	88	88	86	88	88	88	87	88	88	93	93	93
AVG	142	116	111	153	120	115	148	119	114	170	130	124
ST.DEV	52	30	20	53	32	19	50	28	18	56	32	19

Comparing the values in all the four configurations of GSSs, ILA achieve better performance even than ON-OFF; in particular considering the GTDs of all the events:

- lower maximum value
- lower average (higher interactivity level)
- lower standard deviation (more uniform game evolution)

Simulation Results

(CS3)



ILA reduces to about 40% the number of dropped events w.r.t. ON-OFF

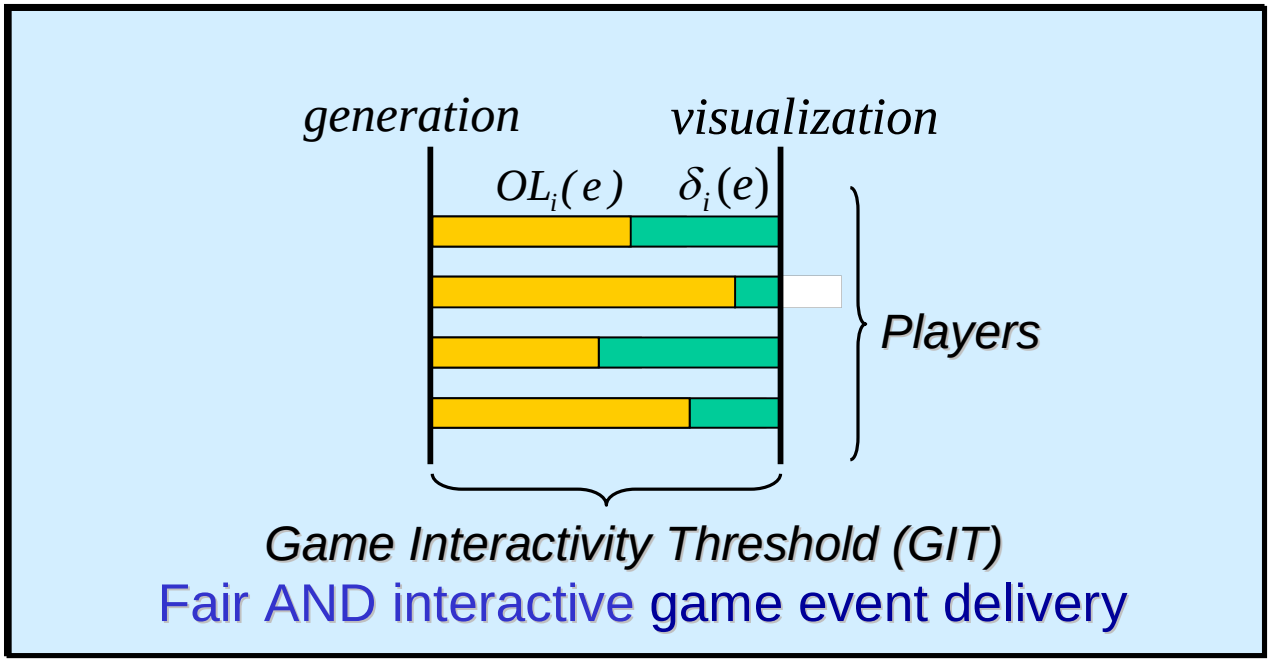
- Less "jumps" in the game evolution sequence
- game evolution fluency improved by our scheme

Results: Metrics Comparison

Network Fairness: Local Lag scheme

- ***Local lag*** delivery strategy to equalize delay differences among players for each game event *e*
  - Different Overall Latencies $OL_i(e)$

- Appropriately computes the value of the artificial delay $\delta_i(e)$ so as to guarantee simultaneous (network fair) rendering



Conclusions

(CS3)

- We presented a new scheme for a fast event delivery service among mirrored *GSSs*, aimed at supporting MMOGs
- The proposed approach exploits an event dropping mechanism, inspired by the RED algorithm
- The scheme is able to drop obsolete events when the interactivity degree results jeopardized
- This approach may be exploited to guarantee a high interactivity degree in MMOG whilst maintaining, at the same time, the consistency of the game state view
- The overall fluency of the game evolution is improved by our scheme
- Extensive simulation results confirm the efficacy of our approach

FINE Caso di Studio 3 (CS3)

Riferimenti

- "Multimedia: Computing, Communications & Applications", R. Steinmetz, K. Nahrstedt, Prentice Hall.
- www.ietf.org → numerosi RFC a riguardo di IntServ e DiffServ
- Per RSVP si vedano anche gli Appunti del Corso di Reti di Calcolatori

- C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, "On Maintaining Interactivity in Event Delivery Synchronization for Mirrored Game Architectures", *in Proc. of the 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'04), GLOBECOM 2004*, Dallas, TX, USA, Nov 2004.
  - http://www.math.unipd.it/~cpalazzi/papers/Palazzi_ILA_DALLAS.pdf