Introduction

Peer-to-peer networks:

- solution for the distribution of media content to a large number of users
- limited investments for network infrastructures by distributors
- the distribution of content relies mainly on users' resources

The distribution of a real time video stream imposes **strict performance requirements**:

- small playback delays
- few frame losses

Introduction

Users, referred to as ***peers***, receive the video stream from other peers and forward it to one or multiple peers

Overlay network over which the content is exchanged; **various topologies**:

- Tree
- Forest, i.e. multiple trees
- Mesh

Introduction

The analysis of the performance of peer-to-peer video streaming systems in the literature is carried out by means of **three techniques**:

- *trace analysis* of working systems or deployment and study of prototypal systems on specific *test beds*
- *analytical studies*
- *simulation*

Most of existing studies focus on the **analysis of full systems *as-is***, without investigating the impact on performances of changes in their operational parameters.

Goal

In our study:

- focus on **peer-to-peer video streaming systems** with **tree or forest** content distribution structure

- we provide a **sensitivity analysis** to investigate the impact of **three critical parameters**:
  - rejoin time
  - average permanence time of peers
  - playback threshold

- we carry out our study by means of a **fine-grained simulative modeling** of the peer-to-peer video streaming system

The reference system model

Our model is inspired to VidTorrent, a tree-based peer-to-peer video streaming system developed at the Massachussets Institute of Technology

However, our model is more general and it **accounts for peer-to-peer video streaming systems with the following properties**:

- a unique content distribution source
- the structure of the distribution is a tree or a forest
- at the application level, in the overlay peer-to-peer network, the content is organized into chunks of video frames referred to as *segments*
- a single *frame* can be split into a fixed number (≥1) of *sub-frames* of variable length
- users can join and leave the peer-to-peer system dynamically, even during the distribution of a video

The video stream

The **video stream** is provided by the source:

- sequence of *m* ordered frames f_i, $i=1,2,\ldots,m$

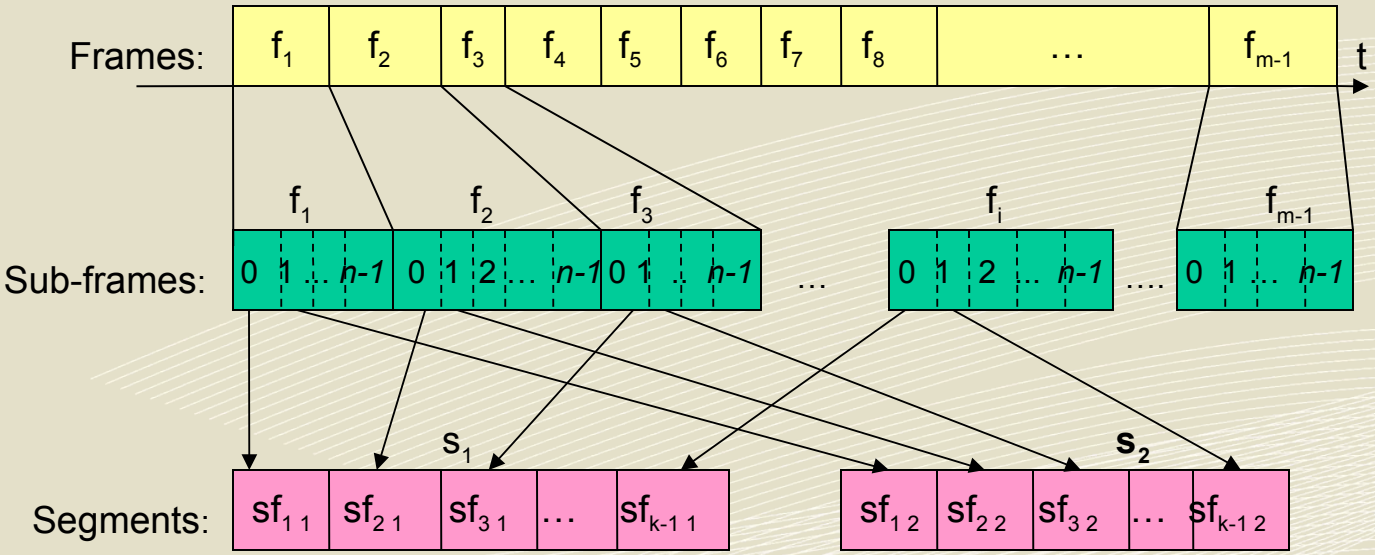
For each **frame** we know:

- start time $f_i.start$
- end time $f_i.end$

Every frame is split into *n* **sub-frames**:

- a sub-frame represents a part of the whole frame, such as, for example a single description in a Multiple Description Coding (MDC)
- subframe sf_{ij} → frame number *i*, sub-frame offset $j=1,2,\ldots,n$
- length $sf_{ij}.length$ [byte]

Segments



At the application layer, **chunks of *k* sub-frames** are organized **into segments**.

A segment s_i is assembled by grouping sub-frames having the same offset in consecutive frames.

Source and Trees

The video content is distributed among all peers through a set of ***q*** **independent trees**

The source:

- provides the video stream
- is placed at the root of each tree
- sequentially sends segments to its children at the rate determined by frame start and end times
- has a limited amount of bandwidth *Sup* [bit/s]

Only the segments composed by the sub-frames with the same *j*-th sub-frame offset are forwarded in the same tree

Trees and Peers

A **client** in the peer-to-peer video streaming system is called ***peer***

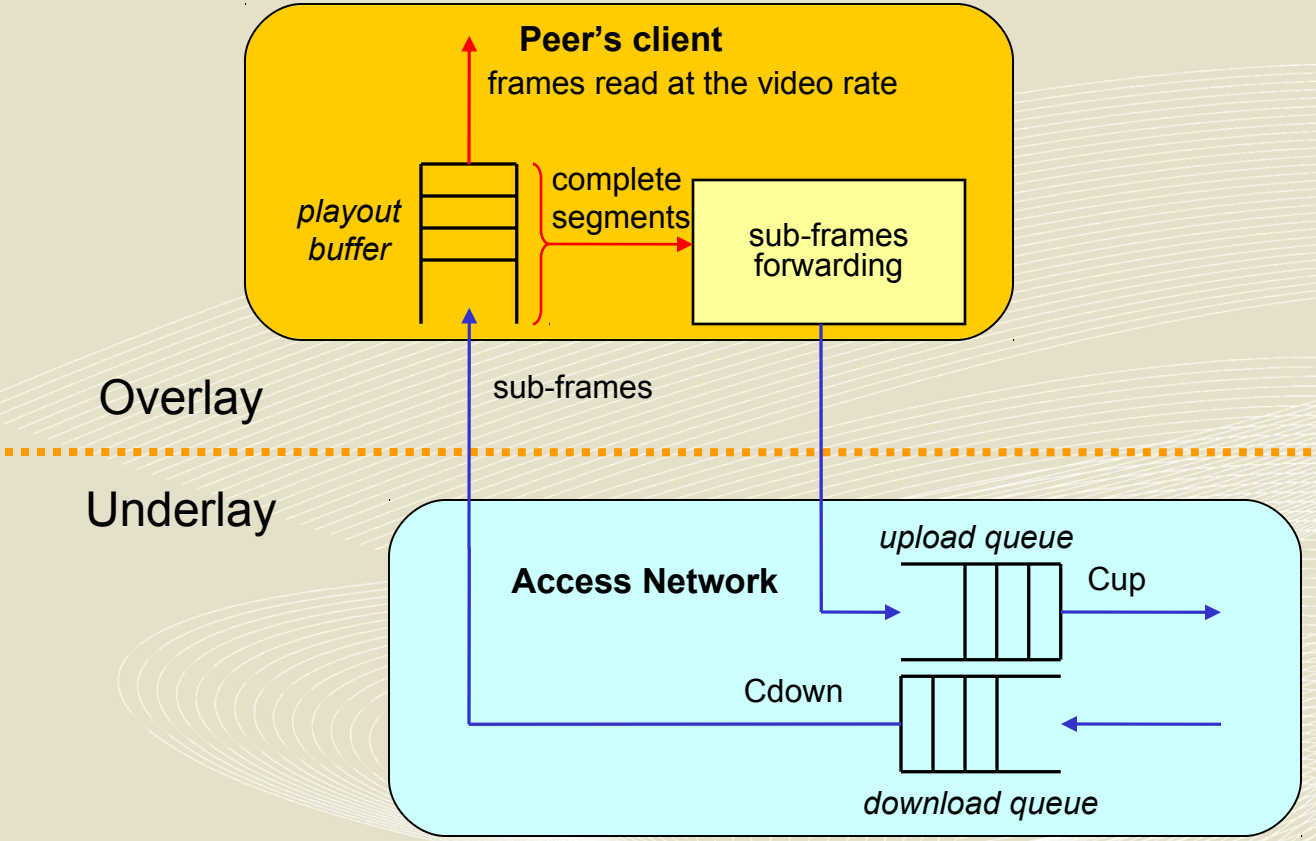
A peer p_i, in order to receive the video content, must be a node of the trees carrying the content

A peer is not required to be part of all trees

All peers, for each tree, receive segments from their parents and then send them to their children

A peer can be placed in different positions in different trees, and different trees can have different topologies

Peers' Operations



The access bandwidth of peer p_i, is referred to as $p_i.Cup$ and $p_i.Cdown$

The peer is provided with a transmission buffer and a reception buffer

Peers' Operations

Each peer performs the following **actions**:

- the download queue, where the packets from parent peers are received, is emptied at a rate determined by $p_i.Cdown$
- all sub-frames received from the download queue are stored in the playout buffer, that reassembles the stream
- as soon as all the sub-frames forming a segment are received, the segment is sent multiple times to all the children peers. These packets are transmitted through the upload link
- the frames stored in the playout buffer are sequentially extracted by the client's player at the rate determined by the video stream

Playout Buffer

The playout buffer is responsible for the re-assembly of the video stream (segments are carried by multiple packets in the underlay network)

The **playout buffer** has a **finite length**, *PBlength*, measured in segments

When a sub-frame is received, it is placed in the correct position in the playout buffer

When a complete segment is reassembled in the playout buffer, it is sent to the children

When all the frames of a segment have been read by the player, a position in the playout buffer is freed

A peer starts playing the video stream as soon as a **playback threshold** *PBTh*, measured in seconds, is reached

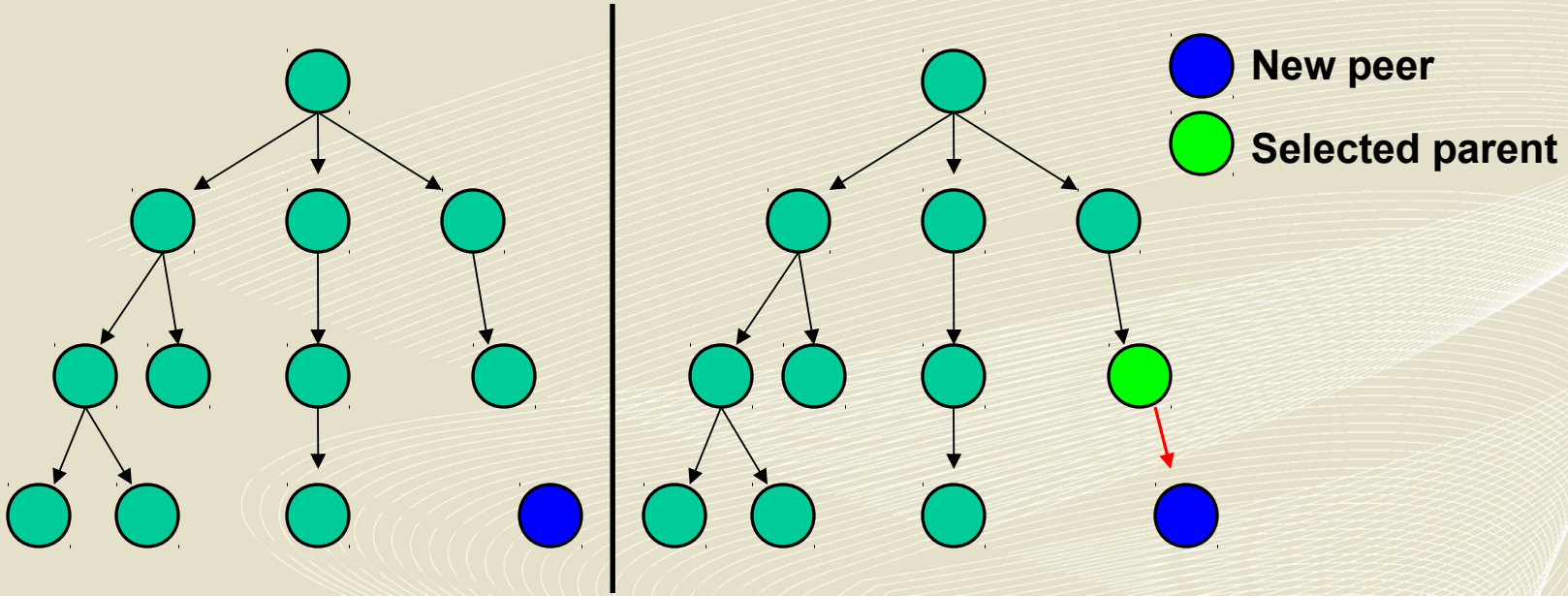
Join

When a new peer wants to receive the video stream, it must **become part of one or multiple distribution trees**

In particular, for peer p_i:

- depending on the free download bandwidth, the maximum number of sub-frames per frame to be received is computed
- the sub-frame offsets are randomly chosen
- for every chosen sub-frame offset, the peer selects a parent in the tree of that offset
- for each tree, the parent is randomly chosen among the peers at the highest level in the tree (nearer to the source) with a sufficient amount of free upload bandwidth
- **after a time interval t_{JOIN}**, the new node starts receiving the sub-streams from its new parents. This interval models the time required for the identification and the selection of a new peer

Join



Leave

A peer can leave the system unpredictably and without notification

Whenever this happens, its children – and, iteratively, all their grandchildren in the same tree – cease to receive the sub-stream

After a time interval t_{REJOIN}, the orphan peers start the join procedure for each sub-stream they are no longer receiving

This time interval can represent, for example, the time required by a keep-alive failure detection mechanism for the identification of the leave of a parent

Only direct children of the dead peer try to rejoin the trees in new positions, while all the isolated trees move together with their ancestors

Performance Analysis

Peers dynamically join and leave

The system, in the steady state, has a **number *N* of simultaneously active peers**

The time spent by a client in the system is exponentially distributed with **average duration equal to $1/\mu$**

Joins are independent Poisson events with a total average rate of joins equal to Λ, such that $N = \Lambda / \mu$

Analysis Parameters and Indexes

The selected **performance parameters** are:

- ***rejoin time***, $t_{JOIN}+t_{REJOIN}$, required by a peer to rejoin a tree when its parent leaves the system
- ***average permanence time of peers*** in the system, $1/\mu$, measured from peer's first join to its leave
- ***playback threshold***, *PBTh*, that must be reached in peer's playout buffer before the video stream is locally played

We measure system performance by means of the following indexes:

- ***playback delay***, defined as the time elapsing from the instant in which the source provides the content to the instant in which a client reads it from the peers' playout buffer
- ***received frames and sub-frames ratios*** for each peer, measured by considering the presence or absence of sub-frames in the playout buffer at their playback time

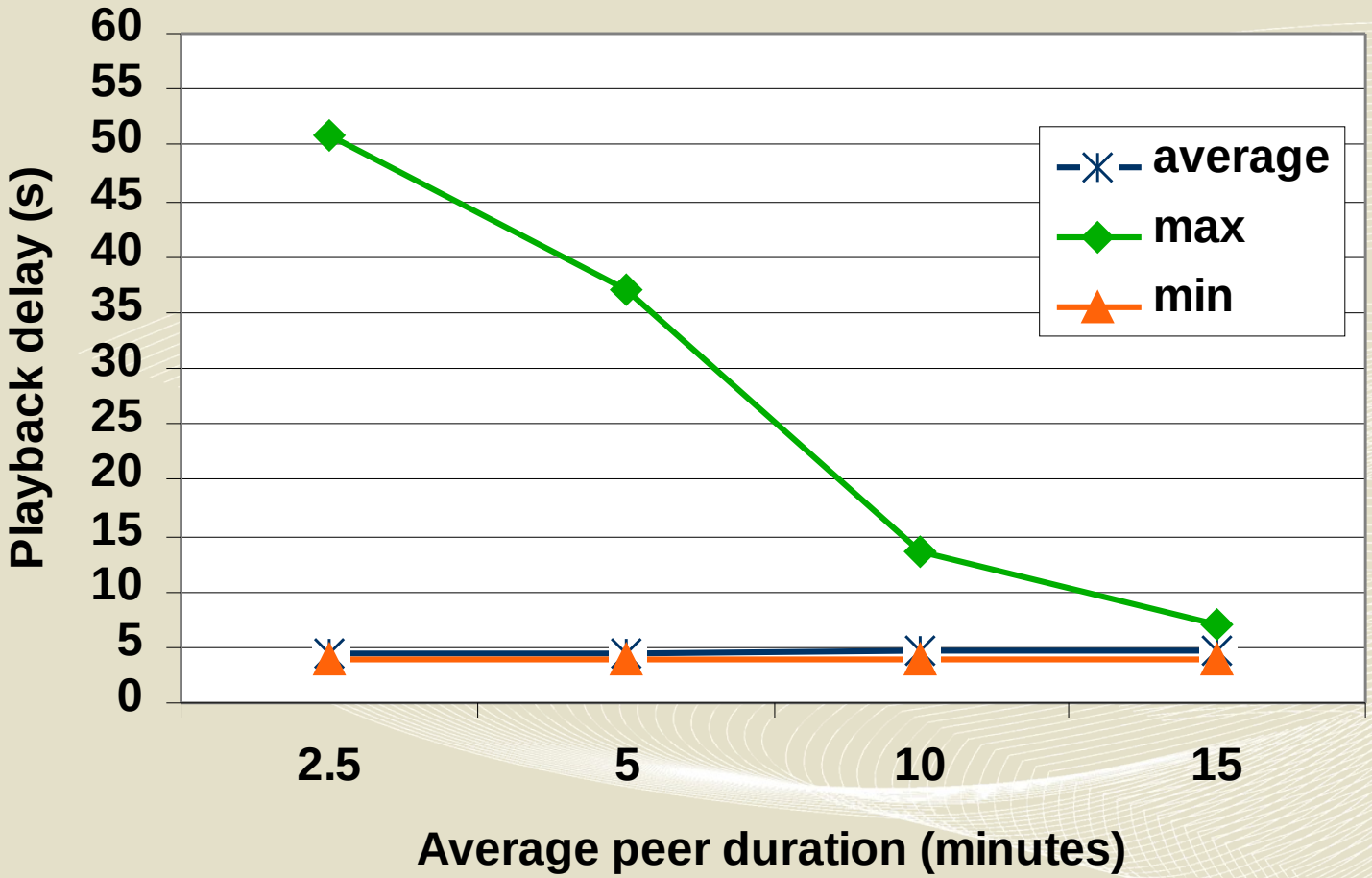
Simulation Parameters

- Real **video trace** of a soccer match with a duration of **40 minutes**, coded with a Multiple Description Coding with **4 descriptions per frame** (*n* = 4)
- Average **rate of the video stream**: **820.5 kbps**
- **Frame duration**: **40 ms**
- Each **segment** comprises *k* = **20 sub-frames**
- A total number of *q* = **4 trees / sub-streams** are used
- Source's bandwidth *Sup* chosen in such a way that it can provide up to 20 sub-streams (i.e. up to 5 complete streams) simultaneously
- The **playout buffer length** has been set equal to **160 s** (*PBLength* = 800)
- Total ***rejoin time***: **100 s**
- ***Average permanence time of peers***: **15 minutes**
- ***Playback threshold***: **4 s**

Simulations have been carried out in **two different scenarios**:

- *Symmetric*: download **7** Mbps – upload **7** Mbps
- *Asymmetric*: download **7** Mbps – upload **1** Mbps

Results – Permanence Time of Peers (I)



Goal

Performances are greatly affected by the upload bandwidth of the access networks of peers

In our study:

- we propose a set of **rewarding techniques** able to cleverly exploit the use of peers' upload bandwidth
- focus on **peer-to-peer video streaming systems** with **tree or forest** content distribution structure
- we provide a **sensitivity analysis** to investigate the impact of:
  - the **dynamicity of the system**, i.e. the existence of peers joining and leaving the system
  - the **time** required by the rewarding protocol to **reorganize the overlay topology**
  - the **number of peers** in the system
- we carry out our study by means of a **fine-grained simulative modeling** of the peer-to-peer video streaming system

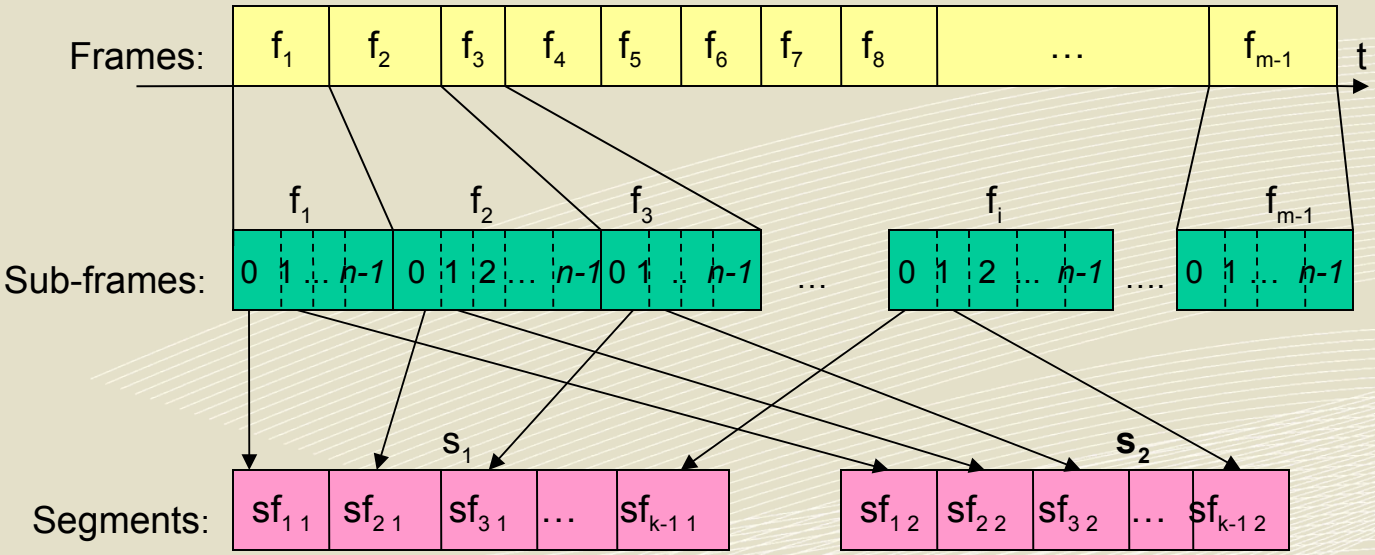
The reference system model

Our model is inspired to VidTorrent, a tree-based peer-to-peer video streaming system developed at the Massachussets Institute of Technology

However, our model is more general and it **accounts for peer-to-peer video streaming systems with the following properties**:

- a unique content distribution source
- the structure of the distribution is a tree or a forest
- at the application level, in the overlay peer-to-peer network, the content is organized into chunks of video frames referred to as *segments*
- a single *frame* can be split into a fixed number (≥1) of *sub-frames* of variable length
- users can join and leave the peer-to-peer system dynamically, even during the distribution of a video

Segments



At the application layer, **chunks of *k* sub-frames** are organized **into segments**

A segment s_i is assembled by grouping sub-frames having the same offset in consecutive frames

Source and Trees

The video content is distributed among all peers through a set of ***q*** **independent trees**

The source:

- provides the video stream
- is placed at the root of each tree
- sequentially sends segments to its children at the rate determined by frame start and end times
- has a limited amount of bandwidth *Sup* [bit/s]

Only the segments composed by the sub-frames with the same *j*-th sub-frame offset are forwarded in the same tree

Trees and Peers

A **client** in the peer-to-peer video streaming system is called ***peer***

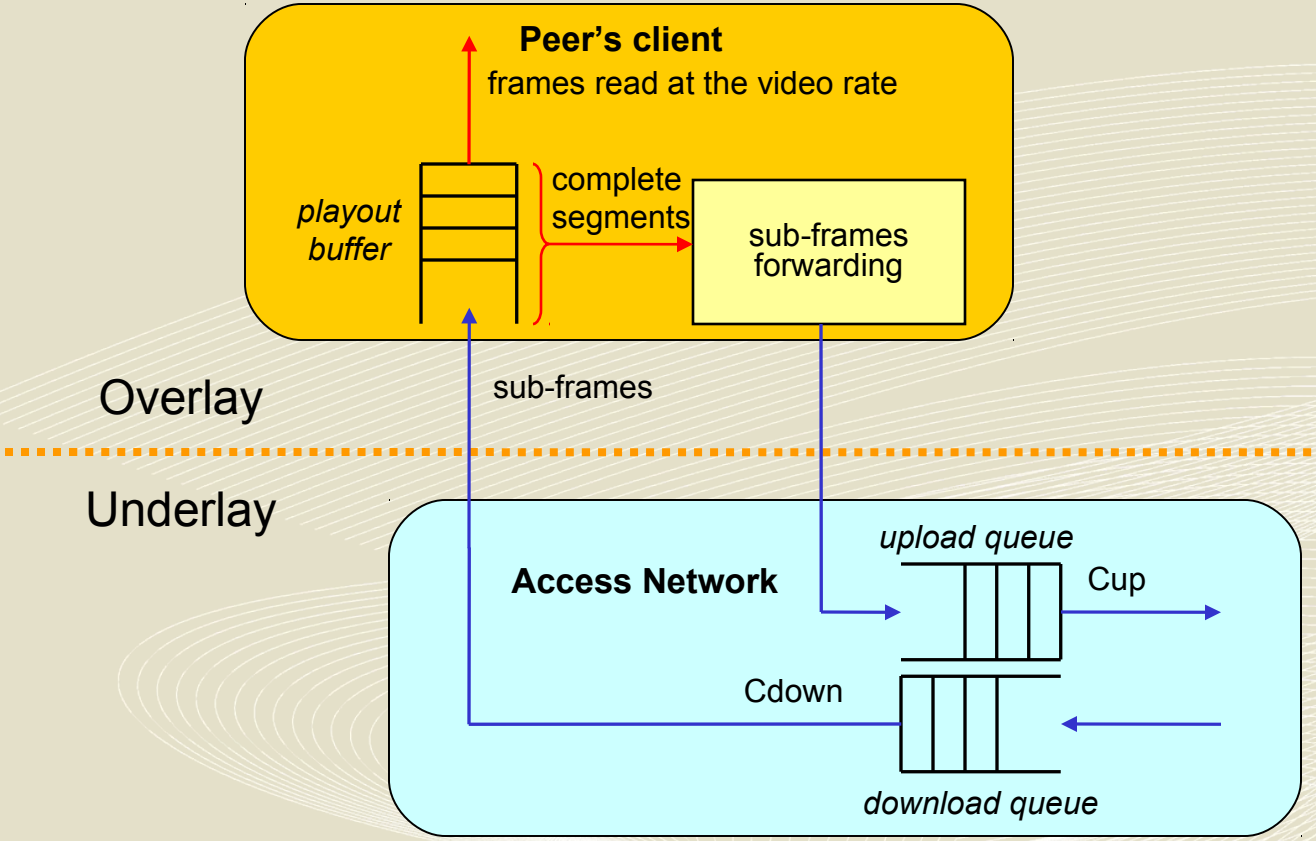
A peer p_i, in order to receive the video content, must be a node of the trees carrying the content

A peer is not required to be part of all trees

All peers, for each tree, receive segments from their parents and then send them to their children

A peer can be placed in different positions in different trees, and different trees can have different topologies

Peers' Operations



The access bandwidth of peer p_i, is referred to as $p_i.Cup$ and $p_i.Cdown$

The peer is provided with a transmission buffer and a reception buffer

Peers' Dynamicity

Peers dynamically join and leave

The system, in the steady state, has a **number *N* of simultaneously active peers**

The time spent by a client in the system is exponentially distributed with **average duration equal to $1/\mu$**

Joins are independent Poisson events with a total average rate of joins equal to Λ, such that $N = \Lambda / \mu$

Standard Join

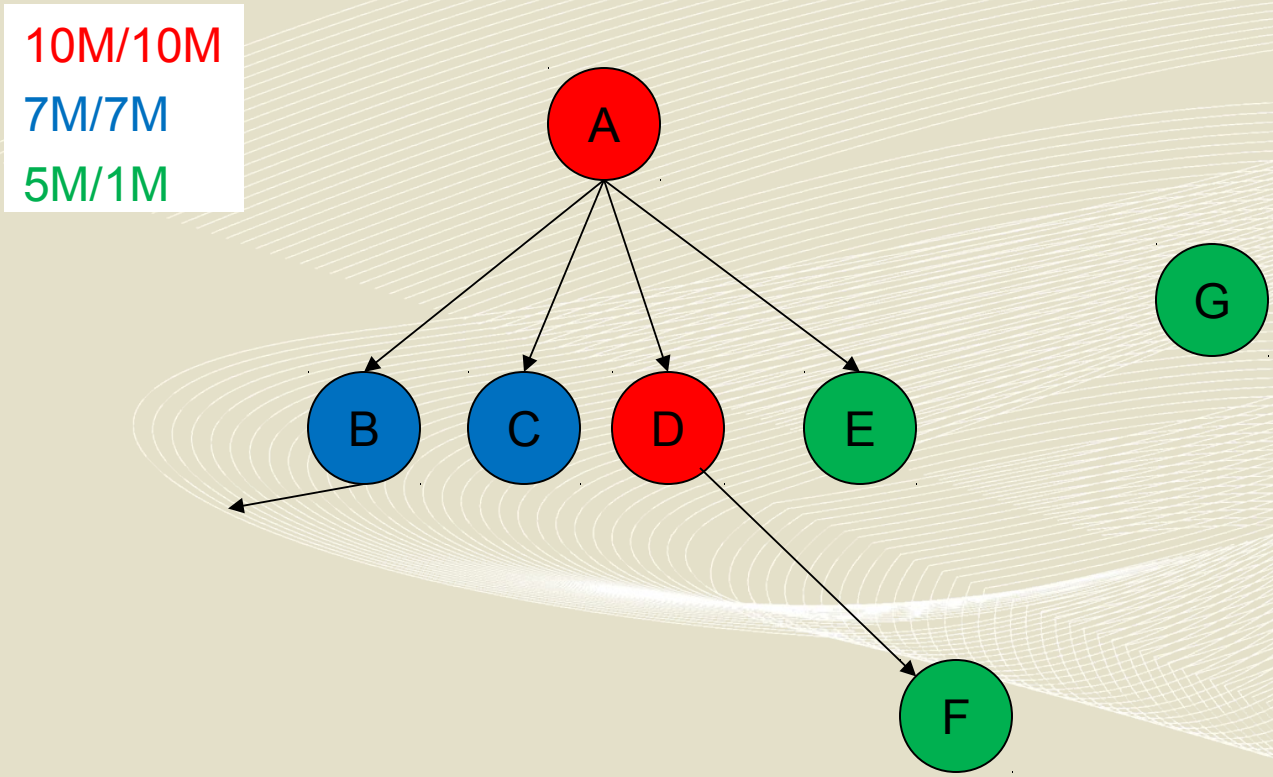
When a new peer wants to receive the video stream, it must **become part of one or multiple distribution trees**

In particular, for peer p_i:

- depending on the free download bandwidth, the maximum number of sub-frames per frame to be received is computed
- the sub-frame offsets are randomly chosen
- for every chosen sub-frame offset, the peer selects a parent in the tree of that offset
- for each tree, the parent is **randomly chosen among the peers at the highest level in the tree** (nearer to the source) with a sufficient amount of free upload bandwidth
- **after a time interval t_{JOIN}**, the new node starts receiving the sub-streams from its new parents. This interval models the time required for the identification and the selection of a new peer

Standard Join

The parent is **randomly chosen among the peers at the highest level in the tree** (nearer to the source) with a sufficient amount of free upload bandwidth

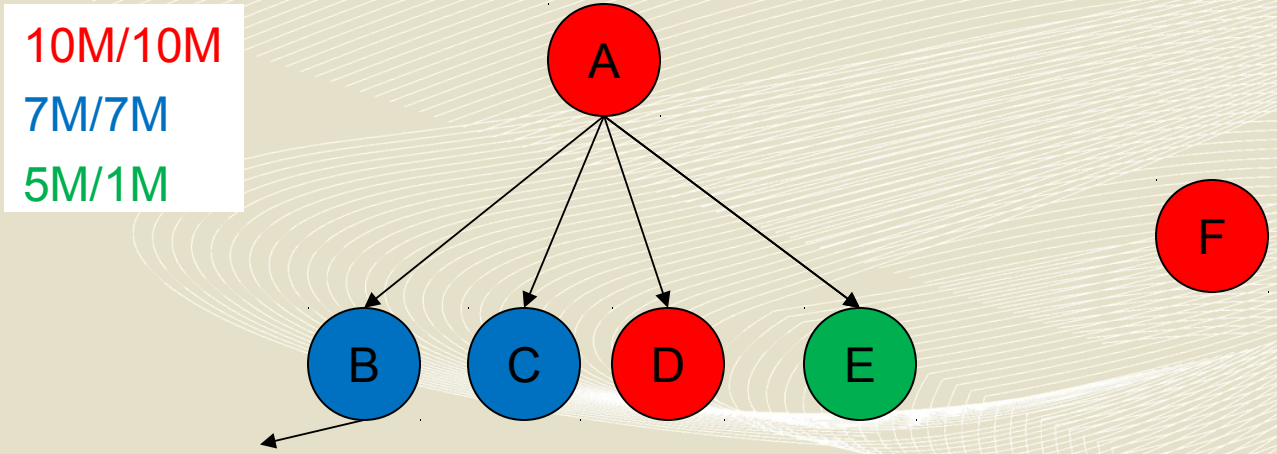


Join with Base Optimization

New peers are placed at the highest position in each tree **such that no peers with lower upload bandwidth are at higher levels**

The **new peer replaces an already existing peer** with lower upload capacity

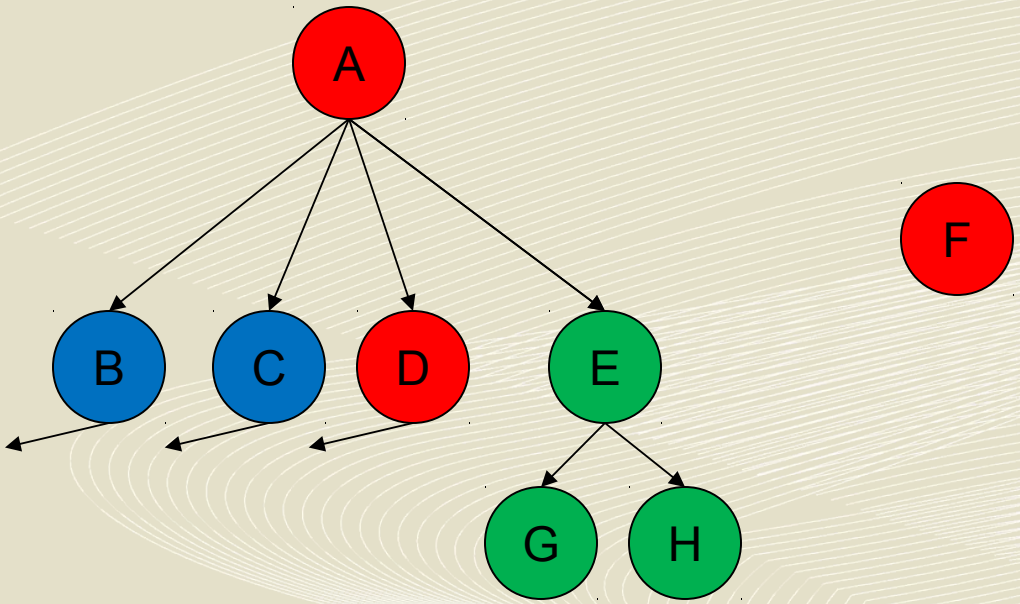
The nodes start receiving the sub-stream from their new parents after a **time interval** t_{REJOIN_REW}



Join with Advanced Optimization

Not only the disconnected peer starts a join procedure, but that **the join procedure is also started by its direct children**

10M/10M
7M/7M
5M/1M



Standard Leave

A peer can leave the system unpredictably and without notification

Whenever this happens, its children – and, iteratively, all their grandchildren in the same tree – cease to receive the sub-stream

After a time interval t_{REJOIN}, the orphan peers start the join procedure for each sub-stream they are no longer receiving

This time interval can represent, for example, the time required by a keep-alive failure detection mechanism for the identification of the leave of a parent

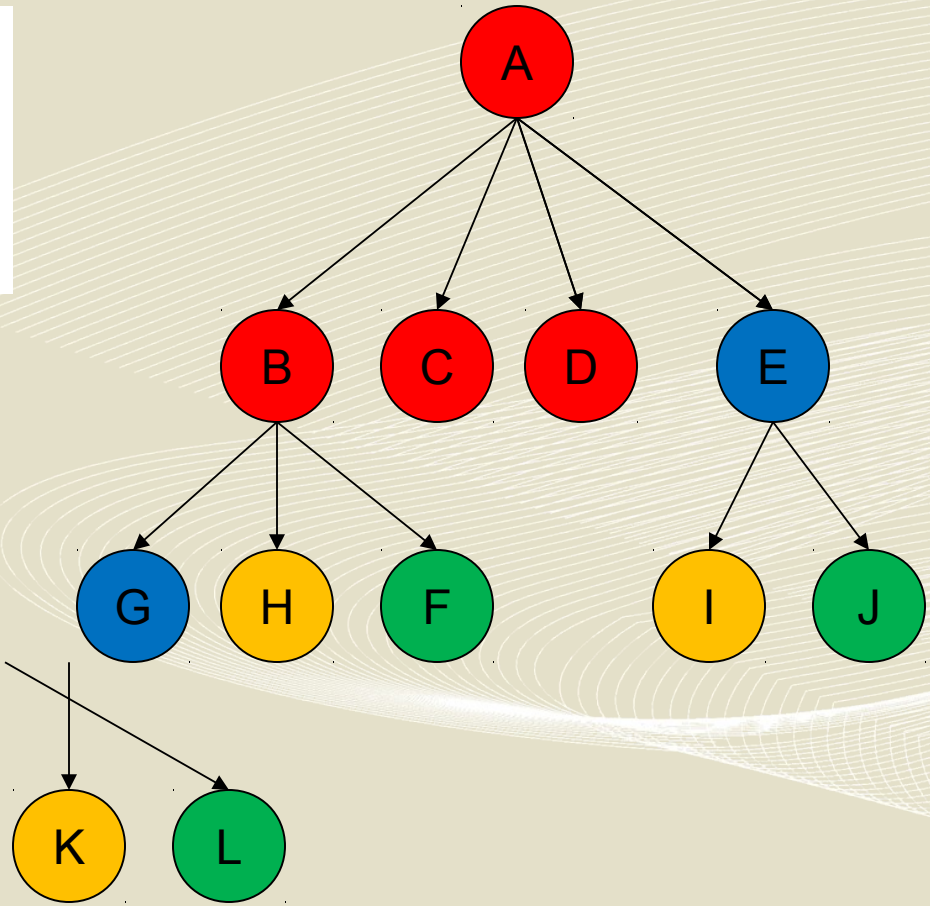
Only direct children of the dead peer try to rejoin the trees in new positions, while all the isolated trees move together with their ancestors

Standard Leave

After a time interval t_{REJOIN}, the orphan peers start the join procedure

for each sub-stream they are no longer receiving

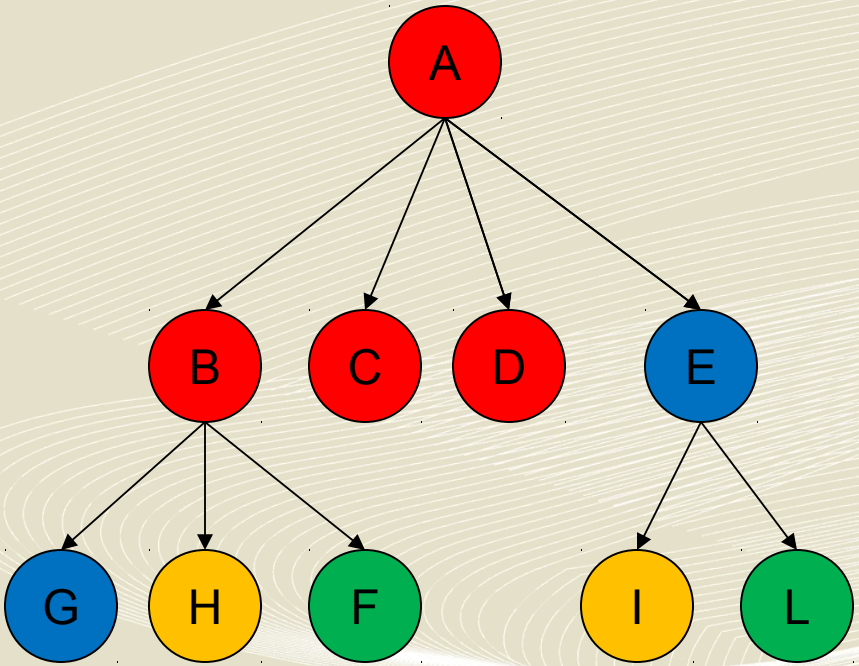
10M/10M
7M/7M
7M/1M
5M/1M



Optimized Leave

In addition to the activities prescribed by the *standard leave* procedure it first **identifies an already existing peer for replacing the peer that has left** the system.

10M/10M
7M/7M
7M/1M
5M/1M



Analysis Parameters and Indexes

The selected **performance parameters** are:

- **average number of peers**, *N*, that are concurrently in the peer-to-peer video streaming system
- ***rewarding rejoin time***, t_{REJOIN_REW}, required by a peer to rejoin a tree when it is disconnected from its current parent and reassigned to a new parent by the rewarding techniques
- ***average permanence time of peers*** in the system, $1/\mu$, measured from peer's first join to its leave

We measure system performance by means of the following indexes:

- ***playback delay***, defined as the time elapsing from the instant in which the source provides the content to the instant in which a client reads it from the peers' playout buffer
- ***received frames and sub-frames ratios*** for each peer, measured by considering the presence or absence of sub-frames in the playout buffer at their playback time

Simulation Parameters

- Real **video trace** of a soccer match with a duration of **36 minutes**, coded with a Multiple Description Coding with **9 descriptions per frame** (*n* = 9)
- Average **rate of the video stream**: **943 kbps**
- **Frame duration**: **33.3 ms**
- Each **segment** comprises *k* = **20 sub-frames**
- A total number of *q* = **9 trees / sub-streams** are used
- Source's bandwidth *Sup* chosen in such a way that it can provide up to 20 sub-streams (i.e. up to 5 complete streams) simultaneously
- The **playout buffer length** has been set equal to **133 ms** (*PBLength* = 1800)
- Total rejoin time: 100 s
- Playback threshold: 3.33 s
- ***Rewarding rejoin time: 100 s***
- ***Number of peers: 250***
- ***Average permanence time of peers: 15 minutes***

Simulation Parameters

The **upload and download access bandwidth** for joining peers, have been set according to the following probability distribution:

50% – C_{DOWN}= 7 Mbps, C_{UP}= 1 Mbps;

30% – C_{DOWN}= 20 Mbps, C_{UP}= 1 Mbps;

10% – C_{DOWN}= 8 Mbps, C_{UP}= 1 Mbps;

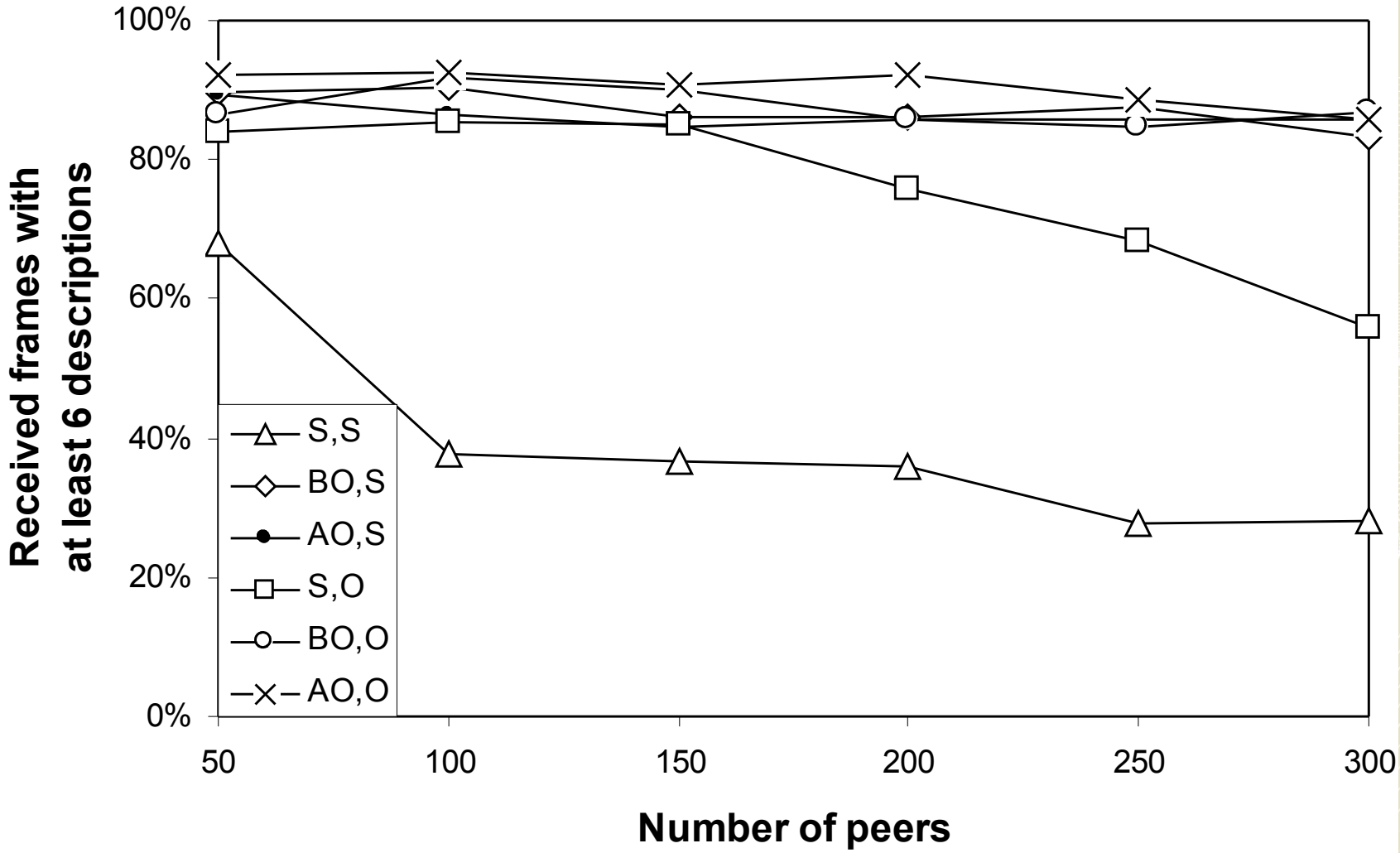
10% – C_{DOWN}= 10 Mbps, C_{UP}= 10 Mbps;

Scenarios

Simulations have been carried out in **6 different scenarios**:

- *standard* join, *standard* leave (**S,S**);
- *base optimization* join, *standard* leave (**BO,S**);
- *advanced optimization* join, *standard* leave (**AO,S**);
- *standard* join, *optimized* leave (**S,O**);
- *base optimization* join, *optimized* leave (**BO,O**);
- *advanced optimization* join, *optimized* leave (**AO,O**).

Results – Average Number of Peers



Results – Conclusions

The use of the proposed **rewarding techniques allows gaining a remarkable increase in the quality** of the video stream received by the users

The use of a rewarding technique **is beneficial only if the time required** by a peer, in order to identify a new parent when it is disconnected, **does not exceed a maximum threshold**

As far as **playback delay** is concerned, in a non-optimized scenario the systems exhibits a **sharply unfair behavior** when peers stay in the system for a short time.The use of **rewarding techniques greatly reduces this problem**.

Results – Conclusions

The performances that can be achieved by the use of the *base* version of the *join optimization* are **not improved by the use of more complex techniques**

The use of rewarding techniques **allows keeping the system scalable**, i.e. the quality of the video stream is not affected by an increase in the number of users

Introduction

Peer-to-peer networks:

- solution for the distribution of media content to a large number of users
- limited investments for network infrastructures by distributors
- the distribution of content relies mainly on users' resources

The distribution of a real time video stream imposes **strict performance requirements**:

- small playback delays
- few frame losses

Introduction

Users, referred to as ***peers***, receive the video stream from other peers and forward it to one or multiple peers

Overlay network over which the content is exchanged; **various topologies**:

- Tree
- Forest, i.e. multiple trees
- Mesh

Introduction

The analysis of the performance of peer-to-peer video streaming systems in the literature is carried out by means of **three techniques**:

- *trace analysis* of working systems or deployment and study of prototypal systems on specific *test beds*
- *analytical studies*
- *simulation*

Most of existing studies focus on the **analysis of full systems *as-is***, without investigating the impact on performances of changes in their operational parameters.

Goal

In this study:

- Focus on **peer-to-peer video streaming systems** with **tree or forest** content distribution structure

- We analyze the impact of two optimization techniques aiming at reducing the negative effects of user's leaves
  - *nearly-permanent* nodes, i.e., peers with a smaller-than-average leave rate
  - placement of high-bandwidth peers in higher levels of trees (*rewarding*)

- We carry out our study by means of a **fine-grained simulative modeling** of the peer-to-peer video streaming system

The reference system model

Our model is inspired to VidTorrent, a tree-based peer-to-peer video streaming system developed at the Massachussets Institute of Technology

However, our model is more general and it **accounts for peer-to-peer video streaming systems with the following properties**:

- a unique content distribution source
- the structure of the distribution is a tree or a forest
- at the application level, in the overlay peer-to-peer network, the content is organized into chunks of video frames referred to as *segments*
- a single *frame* can be split into a fixed number (≥1) of *sub-frames* of variable length
- users can join and leave the peer-to-peer system dynamically, even during the distribution of a video

The video stream

The **video stream** is provided by the source:

- sequence of *m* ordered frames f_i, $i=1,2,\ldots,m$

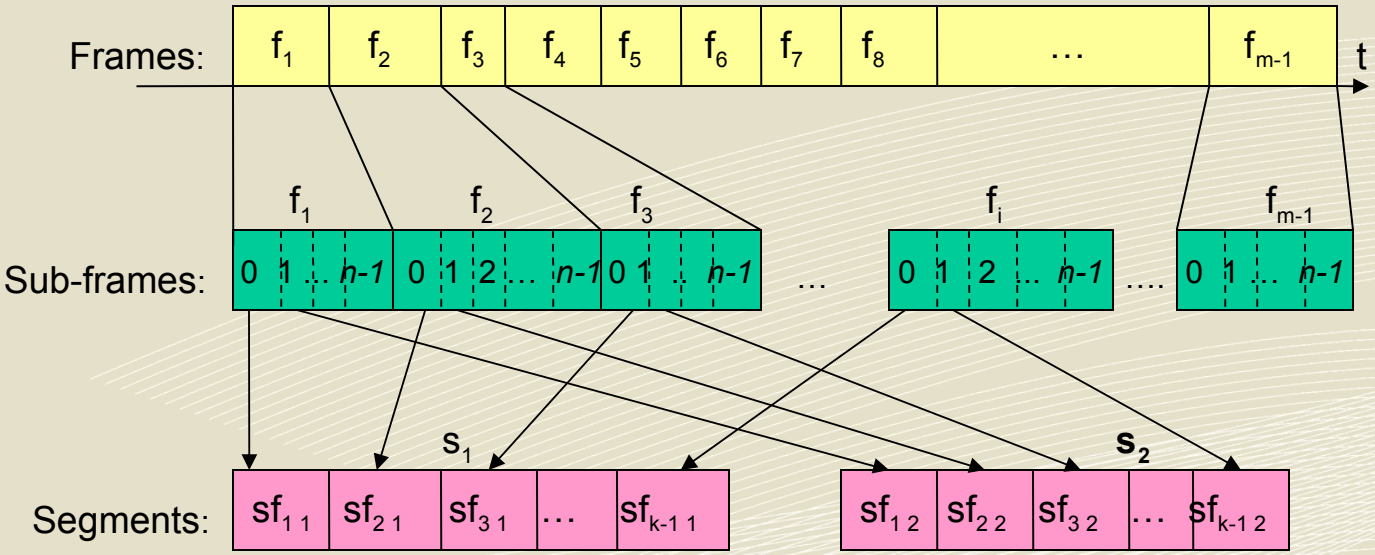
For each **frame** we know:

- start time $f_i.start$
- end time $f_i.end$

Every frame is split into *n* **sub-frames**:

- a sub-frame represents a part of the whole frame, such as, for example a single description in a Multiple Description Coding (MDC)
- subframe sf_{ij} → frame number *i*, sub-frame offset $j=1,2,\ldots,n$
- length $sf_{ij}.length$ [byte]

Segments



At the application layer, **chunks of *k* sub-frames** are organized **into segments**.

A segment s_i is assembled by grouping sub-frames having the same offset in consecutive frames.

Source and Trees

The video content is distributed among all peers through a set of ***q*** **independent trees**

The source:

- provides the video stream
- is placed at the root of each tree
- sequentially sends segments to its children at the rate determined by frame start and end times
- has a limited amount of bandwidth *Sup* [bit/s]

Only the segments composed by the sub-frames with the same *j*-th sub-frame offset are forwarded in the same tree

Trees and Peers

A **client** in the peer-to-peer video streaming system is called ***peer***

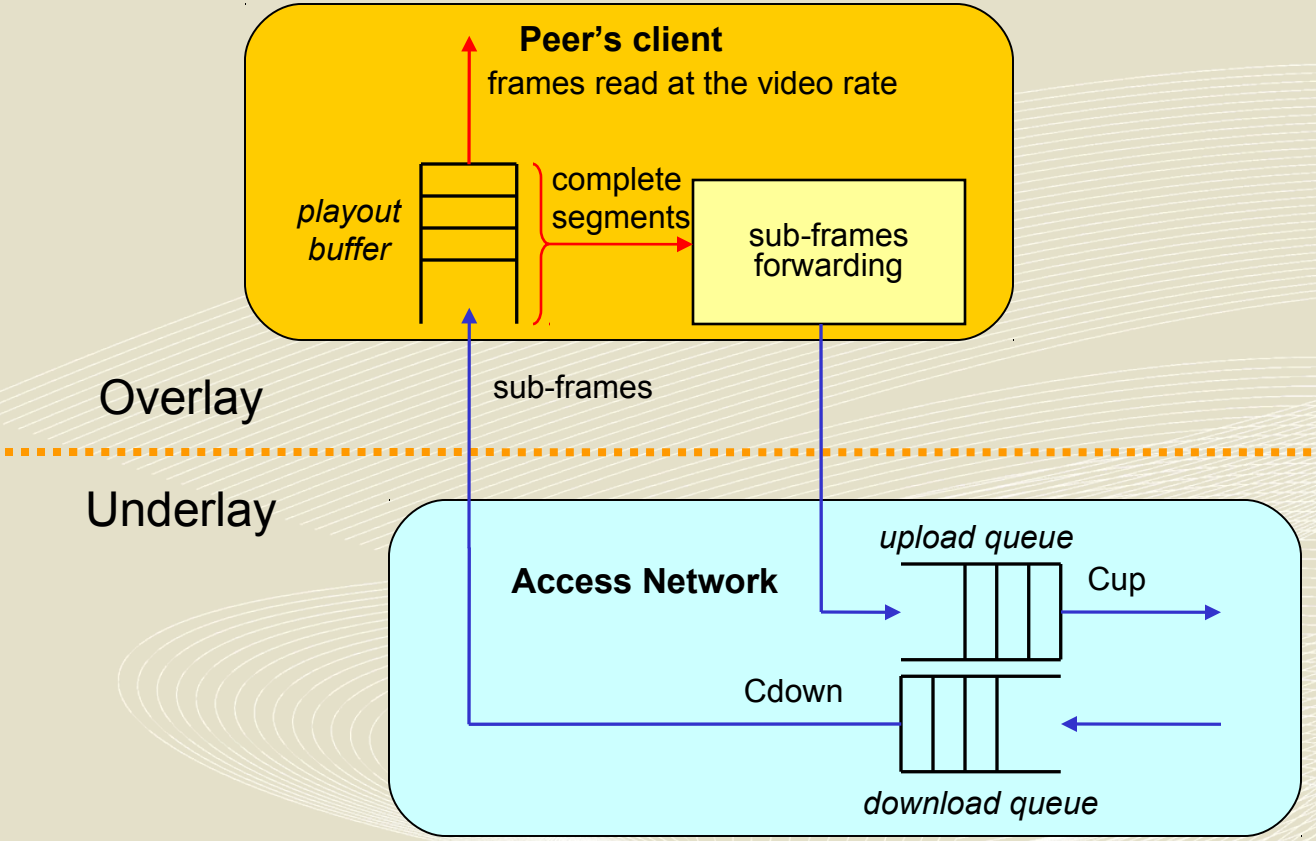
A peer p_i, in order to receive the video content, must be a node of the trees carrying the content

A peer is not required to be part of all trees

All peers, for each tree, receive segments from their parents and then send them to their children

A peer can be placed in different positions in different trees, and different trees can have different topologies

Peers' Operations



The access bandwidth of peer p_i, is referred to as $p_i.Cup$ and $p_i.Cdown$

The peer is provided with a transmission buffer and a reception buffer

Peers' Operations

Each peer performs the following **actions**:

- the download queue, where the packets from parent peers are received, is emptied at a rate determined by $p_i.Cdown$
- all sub-frames received from the download queue are stored in the playout buffer, that reassembles the stream
- as soon as all the sub-frames forming a segment are received, the segment is sent multiple times to all the children peers. These packets are transmitted through the upload link
- the frames stored in the playout buffer are sequentially extracted by the client's player at the rate determined by the video stream

Playout Buffer

The playout buffer is responsible for the re-assembly of the video stream (segments are carried by multiple packets in the underlay network)

The **playout buffer** has a **finite length**, *PBlength*, measured in segments

When a sub-frame is received, it is placed in the correct position in the playout buffer

When a complete segment is reassembled in the playout buffer, it is sent to the children

When all the frames of a segment have been read by the player, a position in the playout buffer is freed

A peer starts playing the video stream as soon as a **playback threshold** *PBTh*, measured in seconds, is reached

Peers' Dynamicity

Peers dynamically join and leave

The system, in the steady state, has a **number *N* of simultaneously active peers**

The time spent by a client in the system is exponentially distributed with **average duration equal to $1/\mu$**

Joins are independent Poisson events with a total average rate of joins equal to Λ, such that $N = \Lambda / \mu$

Standard Join

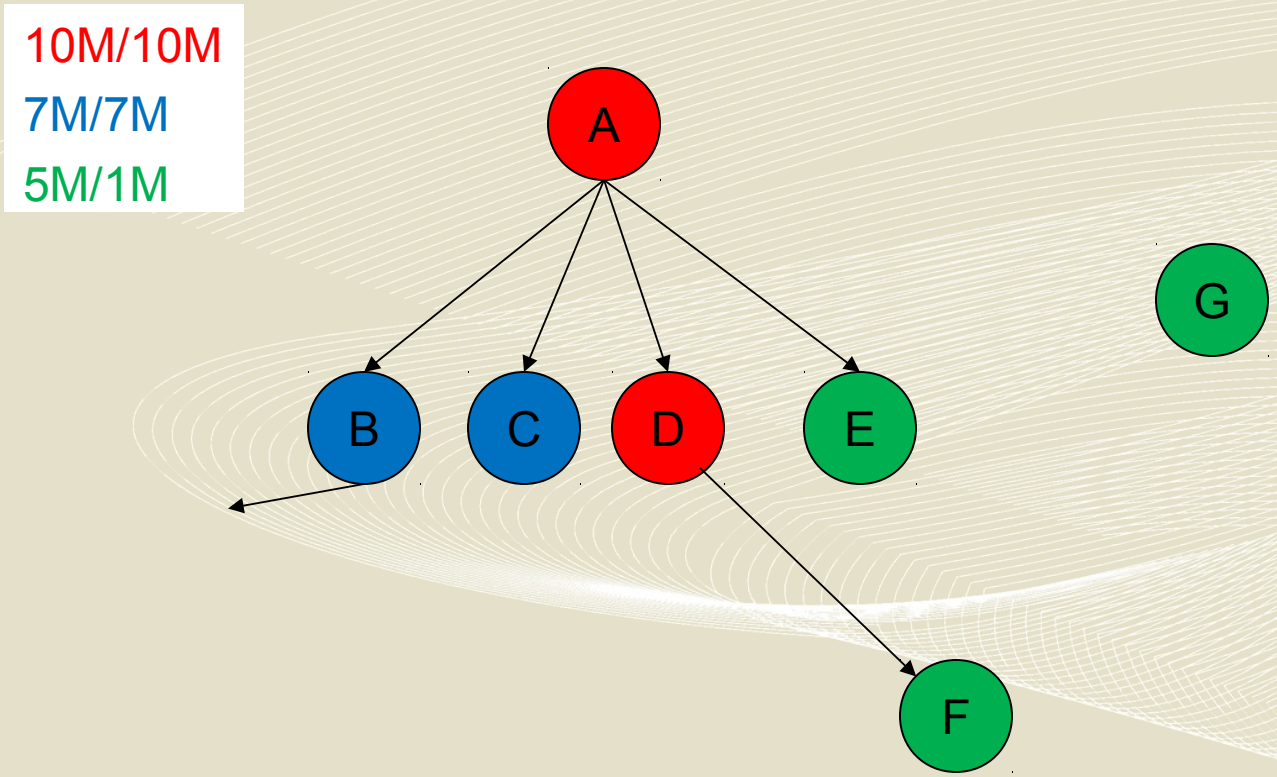
When a new peer wants to receive the video stream, it must **become part of one or multiple distribution trees**

In particular, for peer p_i:

- depending on the free download bandwidth, the maximum number of sub-frames per frame to be received is computed
- the sub-frame offsets are randomly chosen
- for every chosen sub-frame offset, the peer selects a parent in the tree of that offset
- for each tree, the parent is **randomly chosen among the peers at the highest level in the tree** (nearer to the source) with a sufficient amount of free upload bandwidth
- **after a time interval t_{JOIN}**, the new node starts receiving the sub-streams from its new parents. This interval models the time required for the identification and the selection of a new peer

Standard Join

The parent is **randomly chosen among the peers at the highest level in the tree** (nearer to the source) with a sufficient amount of free upload bandwidth

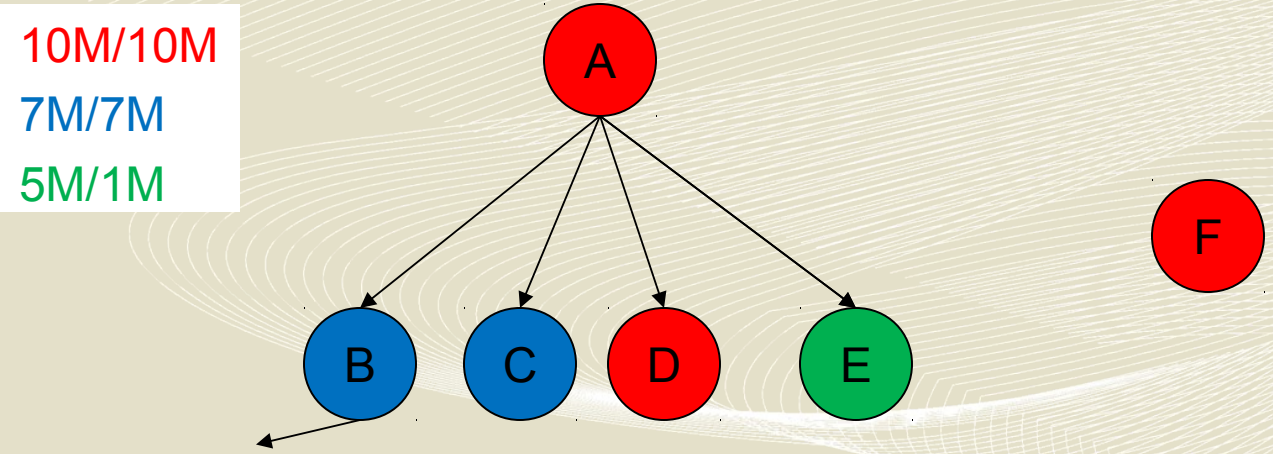


Optimized Join

New peers are placed at the highest position in each tree **such that no peers with lower upload bandwidth are at higher levels**

The **new peer replaces an already existing peer** with lower upload capacity

The nodes start receiving the sub-stream from their new parents after a **time interval** t_{REJOIN_REW}



Standard Leave

A peer can leave the system unpredictably and without notification

Whenever this happens, its children – and, iteratively, all their grandchildren in the same tree – cease to receive the sub-stream

After a time interval t_{REJOIN}, the orphan peers start the join procedure for each sub-stream they are no longer receiving

This time interval can represent, for example, the time required by a keep-alive failure detection mechanism for the identification of the leave of a parent

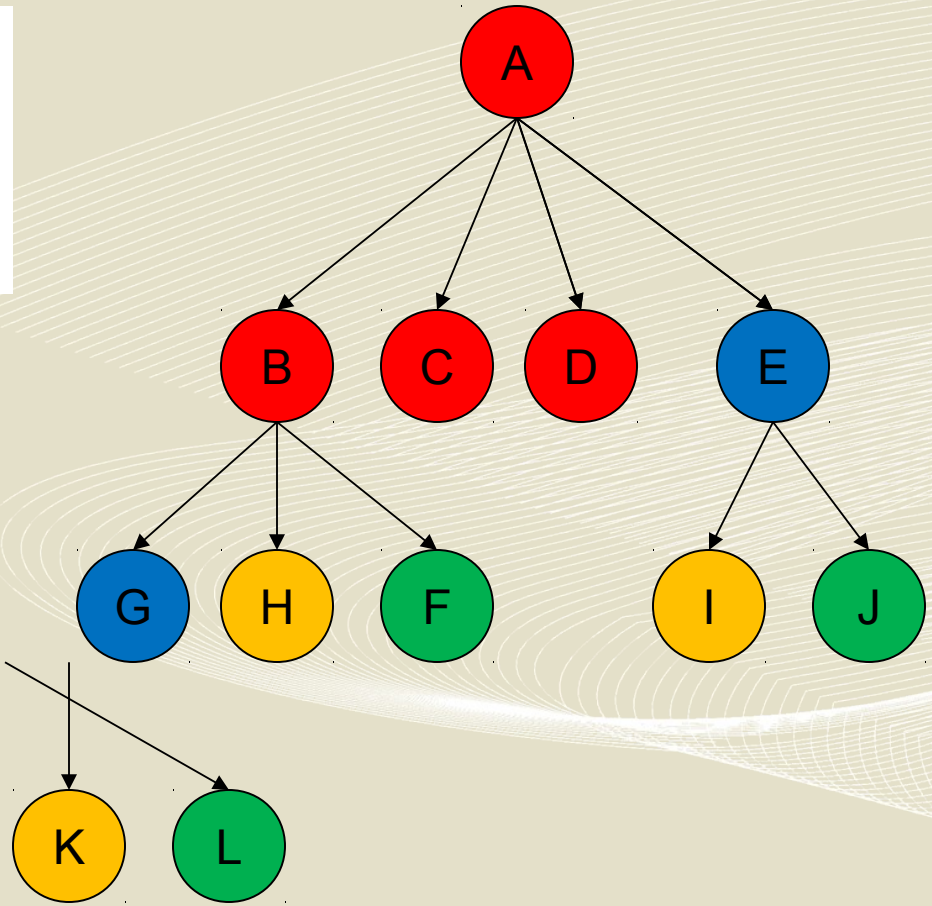
Only direct children of the dead peer try to rejoin the trees in new positions, while all the isolated trees move together with their ancestors

Standard Leave

After a time interval t_{REJOIN}, the orphan peers start the join procedure

for each sub-stream they are no longer receiving

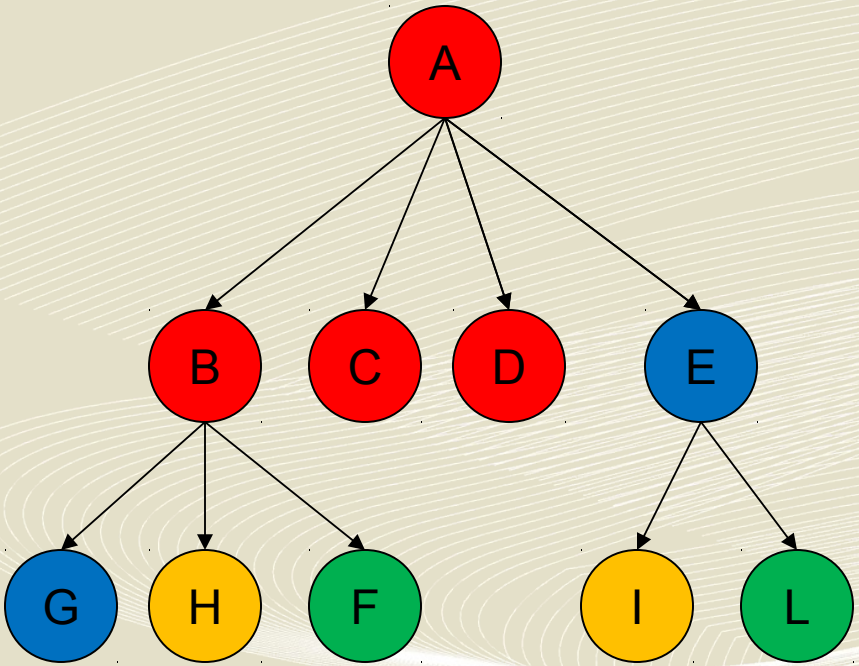
10M/10M
7M/7M
7M/1M
5M/1M



Optimized Leave

In addition to the activities prescribed by the *standard leave* procedure it first **identifies an already existing peer for replacing the peer that has left** the system.

10M/10M
7M/7M
7M/1M
5M/1M



Nearly-permanent peers

Special nodes, referred to as **nearly-permanent peers**, are modelled in our reference system:

- peers whose *average permanence* time in the system, $1/\mu$, is more than one order of magnitude higher with respect to the duration of the media stream
- from our point of view, *higher* with respect to the observation time of a simulation

Additional parameter:

- **number of nearly-permanent peers**, N^{PERM}, that are permanently in the peer-to-peer video streaming system
- the number of standard peers is $N - N^{PERM}$

Nearly-permanent peers

In our scenarios, **two situations** have been analyzed:

1. The N^{PERM} nearly-permanent peers are peers **randomly chosen** among those in the system

1. Nearly-permanent peers are the N^{PERM} peers having the **highest upload capacity**
   - → all nearly-permanent peers goes in the **highest positions** of the trees
   - This state is likely to be beneficial, since it allows the formation of **short and stable trees**

Analysis Parameters and Indexes

The selected **performance parameters** are:

- **average number of peers**, *N*, that are concurrently in the peer-to-peer video streaming system
- **number of nearly-permanent peers**, N^{PERM}, in the peer-to-peer video streaming system
- **average permanence time of peers** in the system, $1/\mu$, measured from peer's first join to its leave

We measure system performance by means of the following indexes:

- ***playback delay***, defined as the time elapsing from the instant in which the source provides the content to the instant in which a client reads it from the peers' playout buffer
- ***received frames and sub-frames ratios*** for each peer, measured by considering the presence or absence of sub-frames in the playout buffer at their playback time

Simulation Parameters

- Real **video trace** of a soccer match with a duration of **36 minutes**, coded with a Multiple Description Coding with **9 descriptions per frame** (*n* = 9)
- Average **rate of the video stream**: **943 kbps**
- **Frame duration**: **33.3 ms**
- Each **segment** comprises *k* = **20 sub-frames**
- A total number of *q* = **9 trees / sub-streams** are used
- Source's bandwidth *Sup* chosen in such a way that it can provide up to 20 sub-streams (i.e. up to 5 complete streams) simultaneously
- The **playout buffer length** has been set equal to **133 s** (*PBLength* = 1800)
- Join time: 500 ms
- Total rejoin time: 100 s
- Playback threshold: 3.33 s
- Rewarding rejoin time: 500 ms
- ***Number of peers: 50***
- ***Number of nearly-permanent peers: 10***
- ***Average permanence time of peers: 15 minutes***

Simulation Parameters

The **upload and download access bandwidth** for joining peers, have been set according to the following probability distribution:

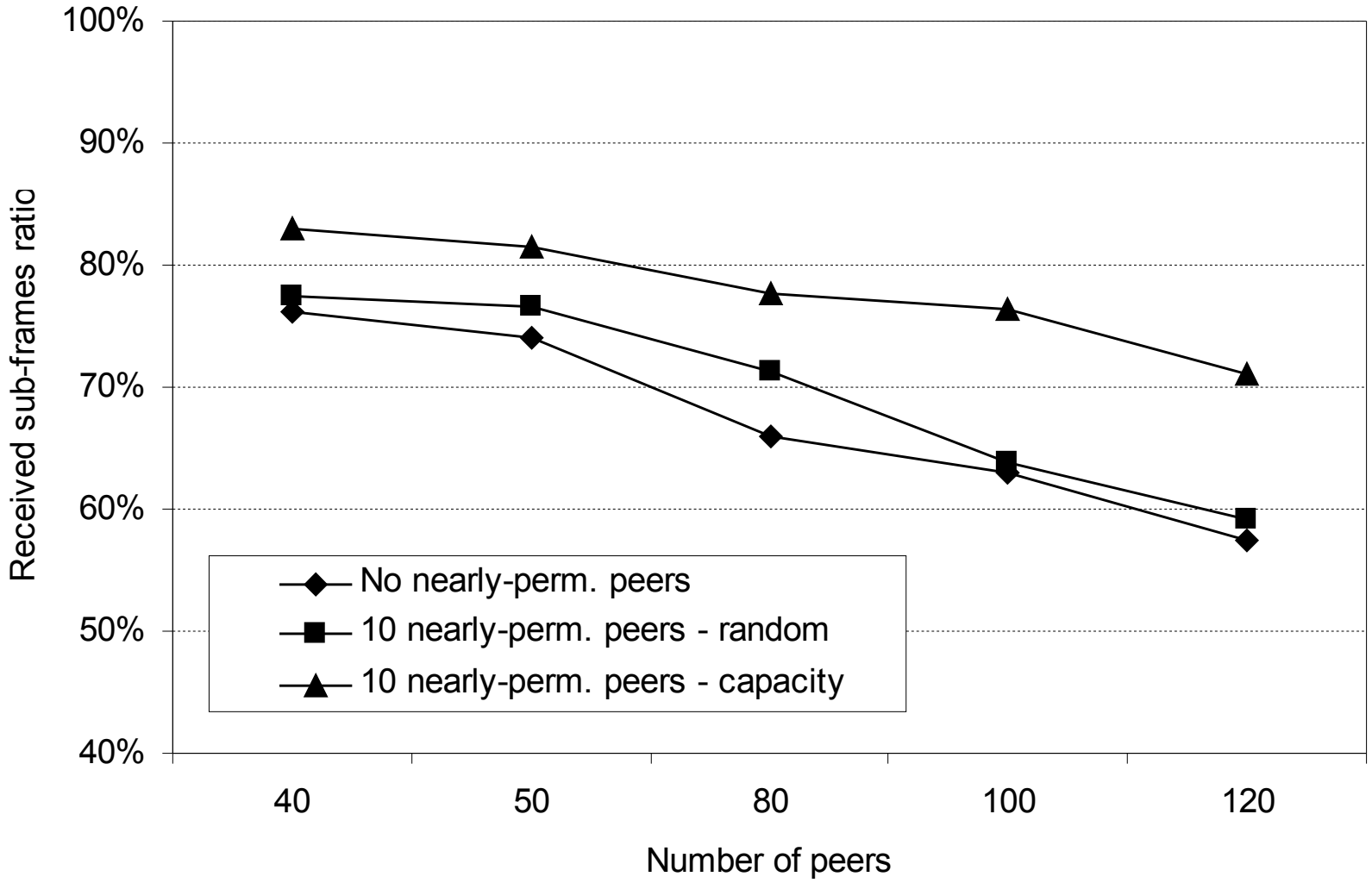
50% – C_{DOWN}= 7 Mbps, C_{UP}= 1 Mbps;

30% – C_{DOWN}= 20 Mbps, C_{UP}= 1 Mbps;

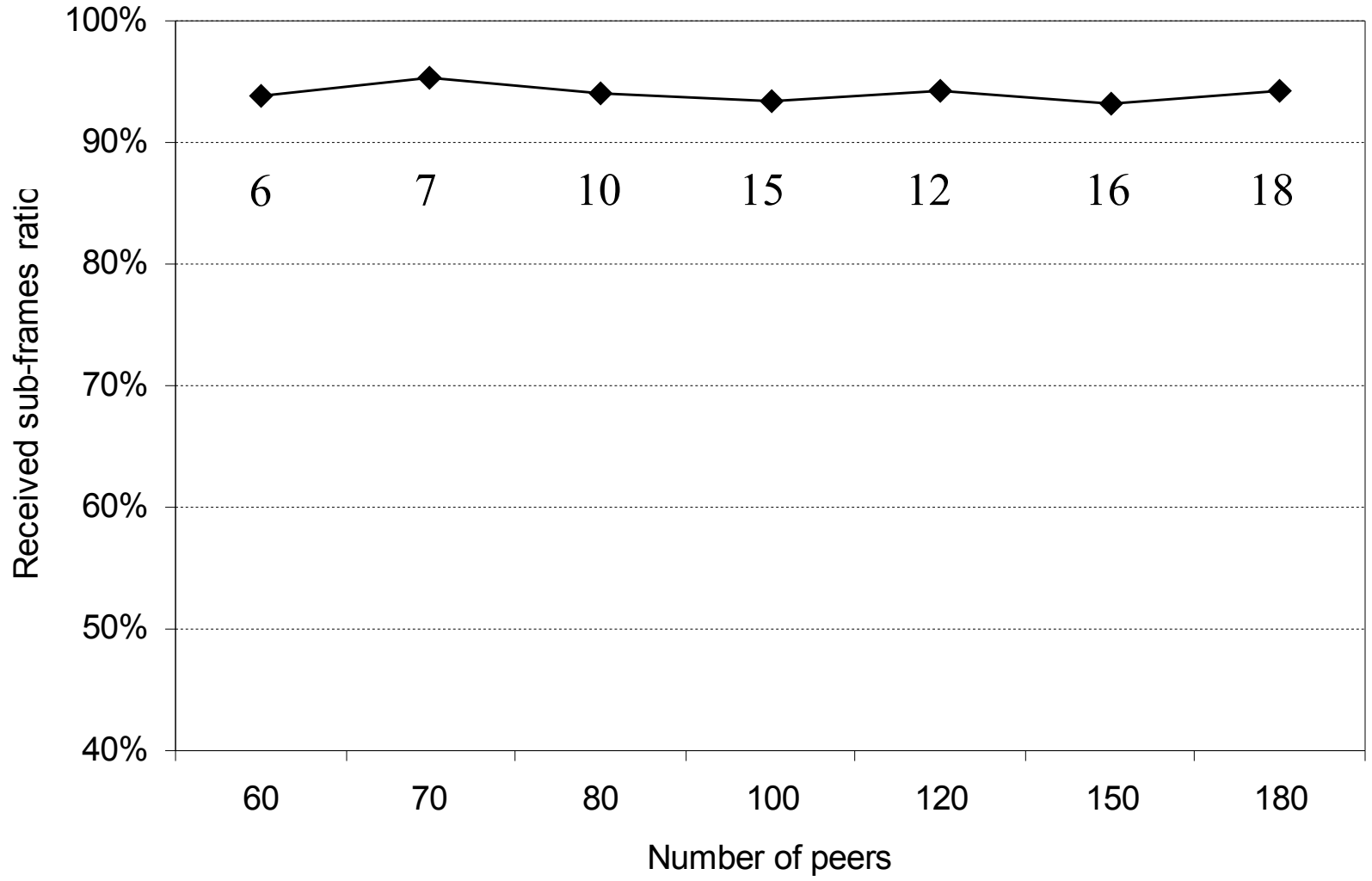
10% – C_{DOWN}= 8 Mbps, C_{UP}= 1 Mbps;

10% – C_{DOWN}= 10 Mbps, C_{UP}= 10 Mbps;

Results – Number of peers



Results – Number of Nearly-Permanent Peers



Results – Conclusions

The presence of **nearly-permanent peers** allows increasing performances

However, these peers must be **chosen according to their upload bandwidth**, otherwise, their utilization is not significantly beneficial

The number of nearly-permanent peers needed to provide scalability is **a percentage of the total number of concurrently active peers** (10%-15% in the examined scenarios)