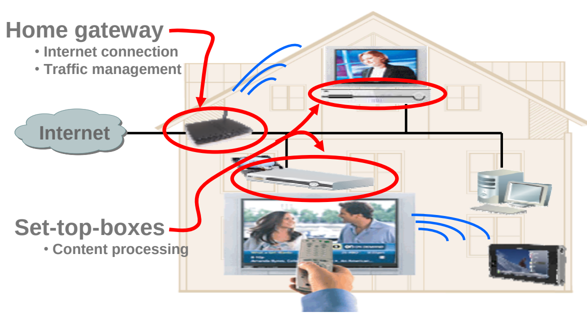
THOMSON
images & beyond

Nano Data Centers
(and application to VoD)

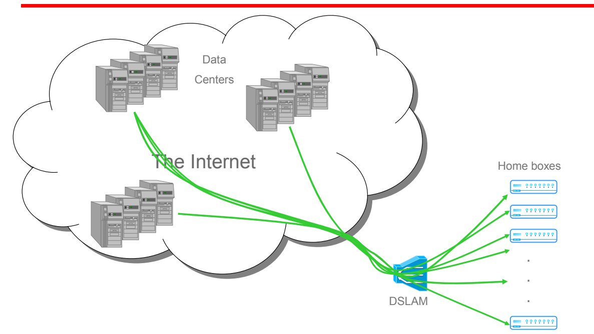
Christophe Diot
Thomson, Paris France
www.thlab.net

Work with: Pablo Rodriguez (Telefonica), Laurent Massoulie, Kyoungwong Suh (UMASS Amherst), Matteo Varvello, Christoph Neuman, Jim, Don, etc.

Gateway vs. Set-top-box



The current model: Data centers



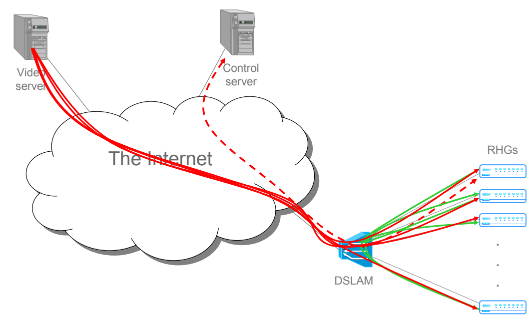
Limitation

- Expensive
  - High capital investment
  - Customer generally pays per byte
- Location constraints in order to be “central”
- Requires a lot of redundancy to be robust
  - Electricity shortage
  - Content availability
- Power, power, power
- New service deployment is slow
  - ISPs not encouraged to take risks, nor to deploy new services

The nano data center

- Add memory to edge boxes, i.e. home gateways, CPEs, Set-Top-Boxes, wireless APs
- Virtualize box architecture
- Manage millions of boxes as a single server using P2P infrastructure
- Take advantage of content locality
- Two phases process (called push-to-peer)
  - Step1: push content from caches when bandwidth is cheap (or create content locally)
  - Step2: serve content requests from peers

The nano data center model



7

The nano data center

- **PROS**
  - Regular network topology that should make it easier to manage and predict/control performance (QoS)
  - Always-on boxes
  - Highly scalable and robust by design
  - Cheap for ISPs
  - Localized & personalized services
- **CONS**
  - Uplink bandwidth often limited
  - Millions of boxes to manage using P2P (up to 2000 below a DSLAM)
  - Cost of gateways
  - Incentive?
  - Privacy?

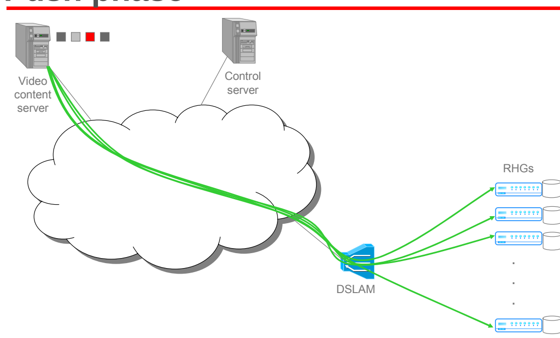
8

Outline

- **The Push-to-Peer architecture**
  - VoD on Nano Data Centers
- **Push strategies**
  - Full striping
  - Coding
- **Issues**

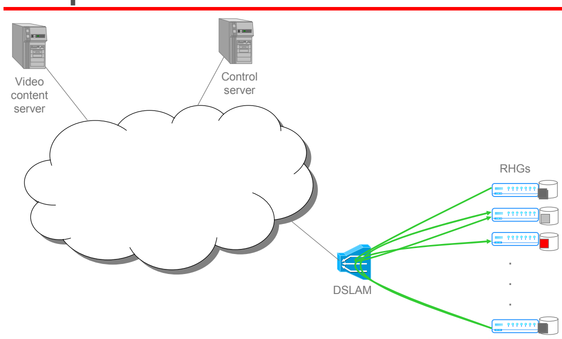
9

Push phase



10

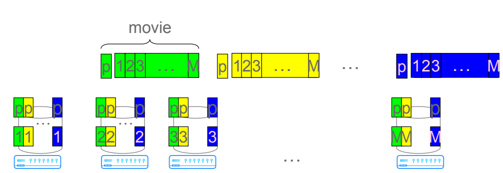
Pull phase



11

Full striping – push phase

- ***M* boxes; each stores the beginning of each movie plus a fraction *1/M* of each movie**



12

Full striping – pull phase

- **Movie playback request from box *m* creates *M-1* TCP connections ☹**
- **TCP connections handled in Processor Sharing fashion**
- **Boxes can only serve K_{max} requests. When K_{max} is reached:**
  - request enqueued (waiting model), or
  - request dumped (blocking model)

13

Optimality properties

- **For uniform users' "taste" and waiting model, startup delays remain bounded as long as**

$$\sum_j d_j L_j (1 - 1/M) < M \times B_{up}$$

Rate of demand for movie j

Size of movie j

Box uplink b/w

- **Optimal among placement schemes that store only one copy of each movie**
- **Optimality too for « flash crowd » demand**

14

Coding – push phase

- **Erasure codes created so that movie can be reconstructed from data on any collection of *y+1* boxes, *y+1<M***
- **Total storage for movie *j*: $C*L_j$, where *C= M/(y+1)***
- **Benefits**
  - Need only *y* TCP connections in parallel to reconstruct the movie
  - Any *y* boxes will do

15

Coding – pull phase

- **Each download request is broken into *y* TCP connections**
- **Each sub-request placed on one of the *y* least loaded boxes, if there are *y* boxes with less than K_{max} jobs**
- **If not,**
  - Waiting model: sub-requests placed in a queue
  - Blocking model: download request dumped

16

Optimality properties

- **For uniform users' "taste" and waiting model, startup delays remain bounded if**

$$\sum_j d_j L_j (1 - C/M) < M \times B_{up}$$

Rate of demand for movie j

Size of movie j

Box uplink b/w

- **Optimal among placement schemes that store only *C* copies of each movie**
- **Quasi-optimality for « flash crowd » demand scenario (optimal for M large)**

17

Performance analysis – waiting model

- **Semi-classical queuing system**
  - «Processor Sharing» server that accepts at most K_{max} jobs
  - Additional jobs wait in FIFO queue
- **Used "Heavy Traffic asymptotics" to derive approximation of the waiting time.**

$$P(W > t) \approx e^{-(1-\rho)\left[K_{\max} + t\left(2E(\sigma)/E(\sigma^2)\right)\right]}$$

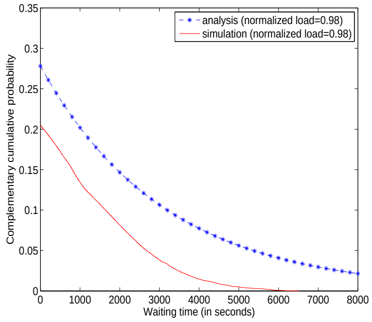
Waiting time in queue

Normalized load

Normalized job size

18

Time to view (full striping)



- 128 boxes
- Video encoding rate 2Mbps
- Upstream bandwidth 1Mbps
- Size of video 2Gbytes

THOMSON

Dimensioning

- **Parameters:**
  - Number of boxes = 100
  - Uplink Bandwidth = 1Mb/s
  - Encoding rate = 1.5Mb/s
  - Number of movies = 500 (currently AT&T catalog)
  - Movie duration = 2 hours (10 Gb)
  - Request rate = 1 movie per box every 140 minutes
- **In full striping mode:**
  - Required memory per box = 6 GB
  - Maximum number of requests handled by a box = 66
  - Normalized load = 0.9
  - Probability of experiencing delay = 0.001

THOMSON

Summary of Push-to-Peer

- **Optimal content placement and pull policies for peers under common control using erasure codes**
- **Formal proof of optimality and performance models**

K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, M. Varvello ,
"Push-to-Peer Video-on-Demand system: design and evaluation",
IEEE JSAC, December, 2007

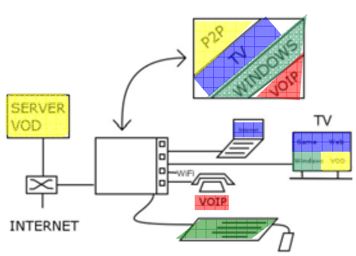
THOMSON

Issues to be addressed

- **Management**
  - Box virtualization
  - Control server
  - Utilization with CDNs
- **Exploiting user preferences**
  - Design of optimal placement schemes taking into account on movie popularity
  - Manage tension between individual and global preferences
  - Data mining and user profiling
- **Content access on multiple DSLAMs**
  - Account for network locality
- **Other applications: IPTV, Games, UGS, etc.**

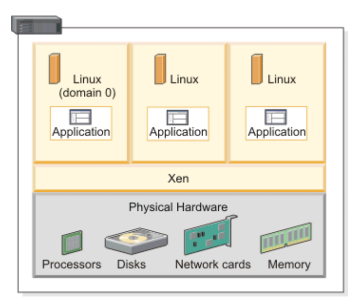
THOMSON

Virtualization



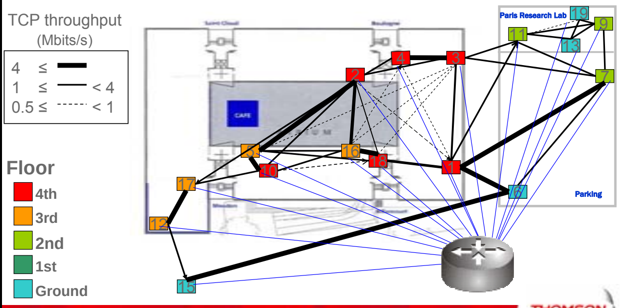
THOMSON

XEN



THOMSON

Testbed



THOMSON

The future of nano data center

- We are doing it!
  - There should be a demo at NAB 08
- Funded EU FIRE project

26

THOMSON