PREDICTIVE COST-BASED SCHEDULING FOR SCALABLE MEDIA STREAMING

Mohammad Alsmirat and Nabil J. Sarhan

ICME 2008 Conference Presentation

June 25, 2008

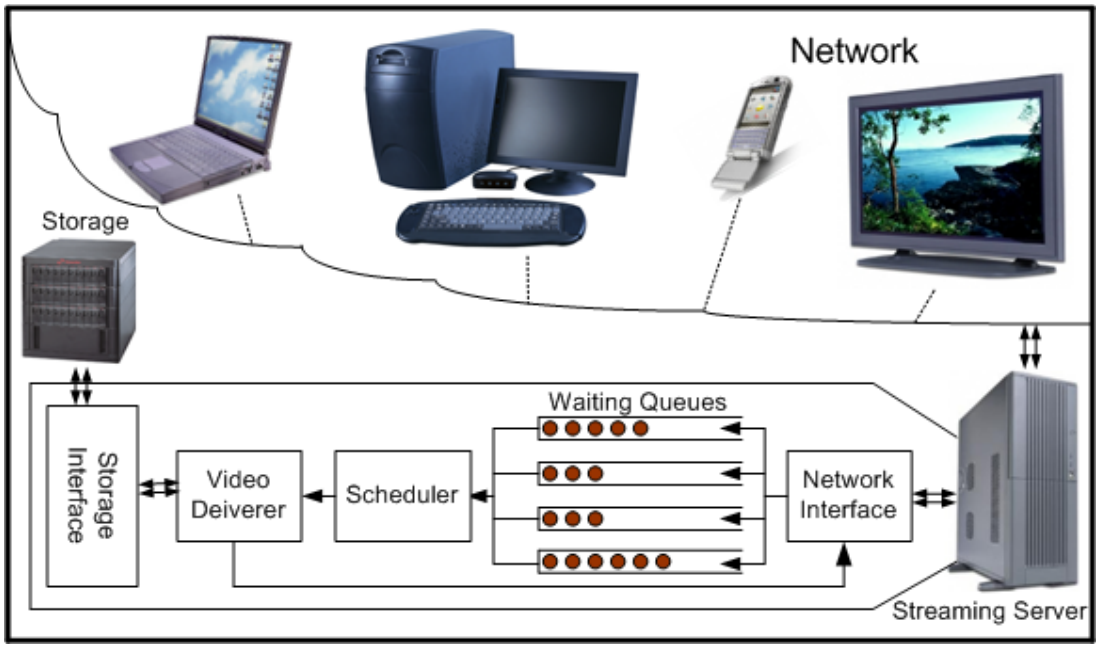
This work has been supported in part by NSF grant CNS-0626861.
This paper was published as [11].

Outline

- Introduction
- Related Work
- Predictive Cost-based Scheduling
- Evaluation Methodology
- Main Results
- Conclusions

Introduction

- The distribution of streaming media faces a significant scalability challenge.
- This challenge has been addressed by Scalable Media Delivery *and* Request Scheduling.

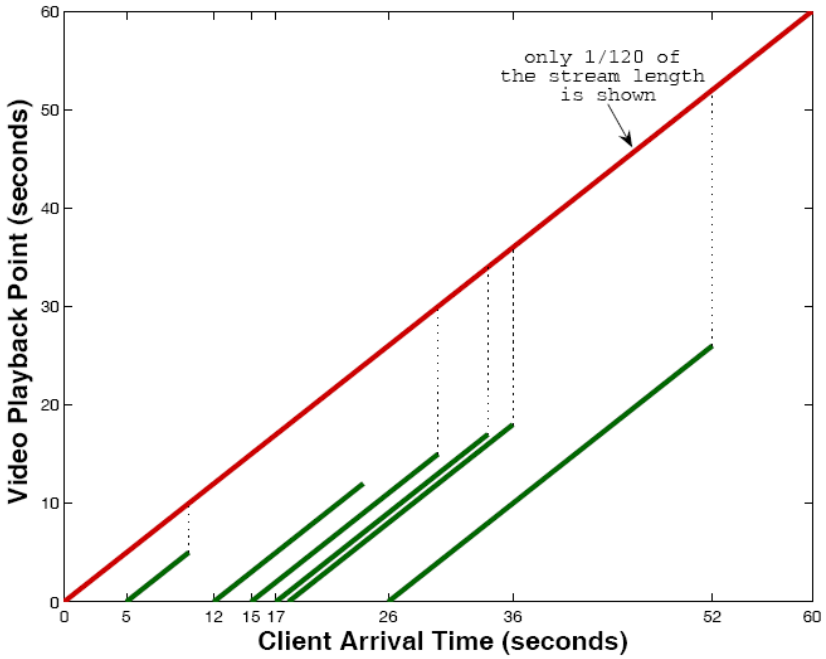


Introduction (cont…)

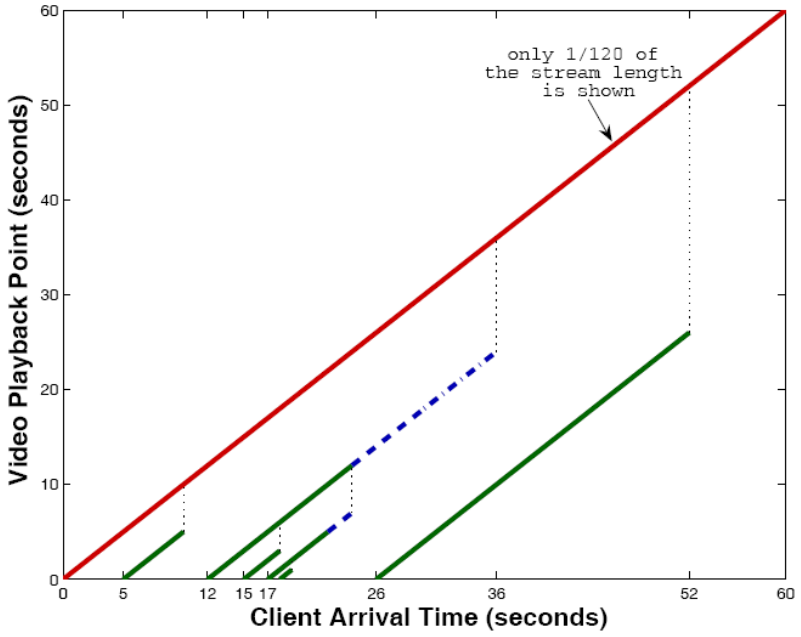
- The choice of a scheduling policy can be more important than the choice of a stream merging technique.
- *Minimum Cost First (MCF)* is a cost-based scheduling policy [5].
  - Significantly outperforms all other scheduling policies.
- We propose a new cost-based scheduling policy, called *Predictive Cost-Based Scheduling (PCS)*.
- We present and analyze two alternative implementations of PCS.
  - We use waiting-time predictability as an additional metric.

Related Work (cont…)

- Scalable Video Delivery Strategies:
  - Periodic Broadcasting [4]
  - Stream Merging [2,1,5,8,9,10,2]



Patching



ERMT

Related Work (cont…)

- Request Scheduling
  - One waiting queue for each video
  - Main Scheduling Policies:

Policy	Selects the queue with
First Come First Serve (FCFS) [6]	oldest request
Maximum Queue Length (MQL) [6]	largest number of requests
Minimum Cost First [5]	least cost (per request)

Related Work (Cont…)

- *MCF* is the best existing scheduling policy.
  - It selects the video requiring the least cost.
- The length of the required stream is directly proportional to the cost of servicing that stream.
  - The server allocates a channel for the entire time the stream is active.
- *MCF-P (P for "Per")* is the best implementation of MCF.
  - It selects the video with the least cost per request.

Proposed Predictive Cost-based Scheduling (PCS)

- We propose a new cost-based scheduling policy, called *Predictive Cost-Based Scheduling (PCS)*.
- Like MCF, PCS is cost-based, but it
  - predicts future system state.
  - uses the prediction results to potentially alter the scheduling decisions.

Proposed Predictive Cost-based Scheduling (PCS) (Cont…)

- How does it work?
  - When a channel becomes available, determine the video V_{Now} that is to be serviced tentatively at the current scheduling time (T_{Now}) and its associated delivery cost ($Cost_{Now}$) .
  - Before actually servicing that video,
    - Predict the system state at the next scheduling time (T_{Next}).
    - Estimate the delivery cost ($Cost_{Next}$) at that time

Predictive Cost-based Scheduling (PCS) (cont…)

- if $Cost_{Next} < Cost_{Next}$

  Do not service any request at T_{Now}

- else

  Service the requests for video V_{Now} immediately.

Predictive Cost-based Scheduling (PCS) (cont…)

- To reduce possible server underutilization
  - PCS delays the service of streams only if the number of available server channels (*freeChannels*) is smaller than a certain threshold (*freeChannelThresh*).

freeChannelThresh

- The free channel threshold is to be computed dynamically.

```
currDefectionRate =
                defectedCustomers/servedCustomers;
if (currDefectionRate < lastDefectionRate) {
    if (last action was decrement and freeChannelThresh > 2)
        freeChannelThresh − −;
    else if (last action was increment)
        freeChannelThresh + +;
} else if (currDefectionRate > lastDefectionRate){
    if (last action was increment and freeChannelThresh > 2)
        freeChannelThresh − −;
    else if (last action was decrement)
        freeChannelThresh + +;
}
lastDefectionRate = currDefectionRate;
```

Alternative Implementations

- We present two alternative implementations of PCS:
  - *PCS-V* (V stands for Video)
  - *PCS-L* (L stands for Length)

PCS-V

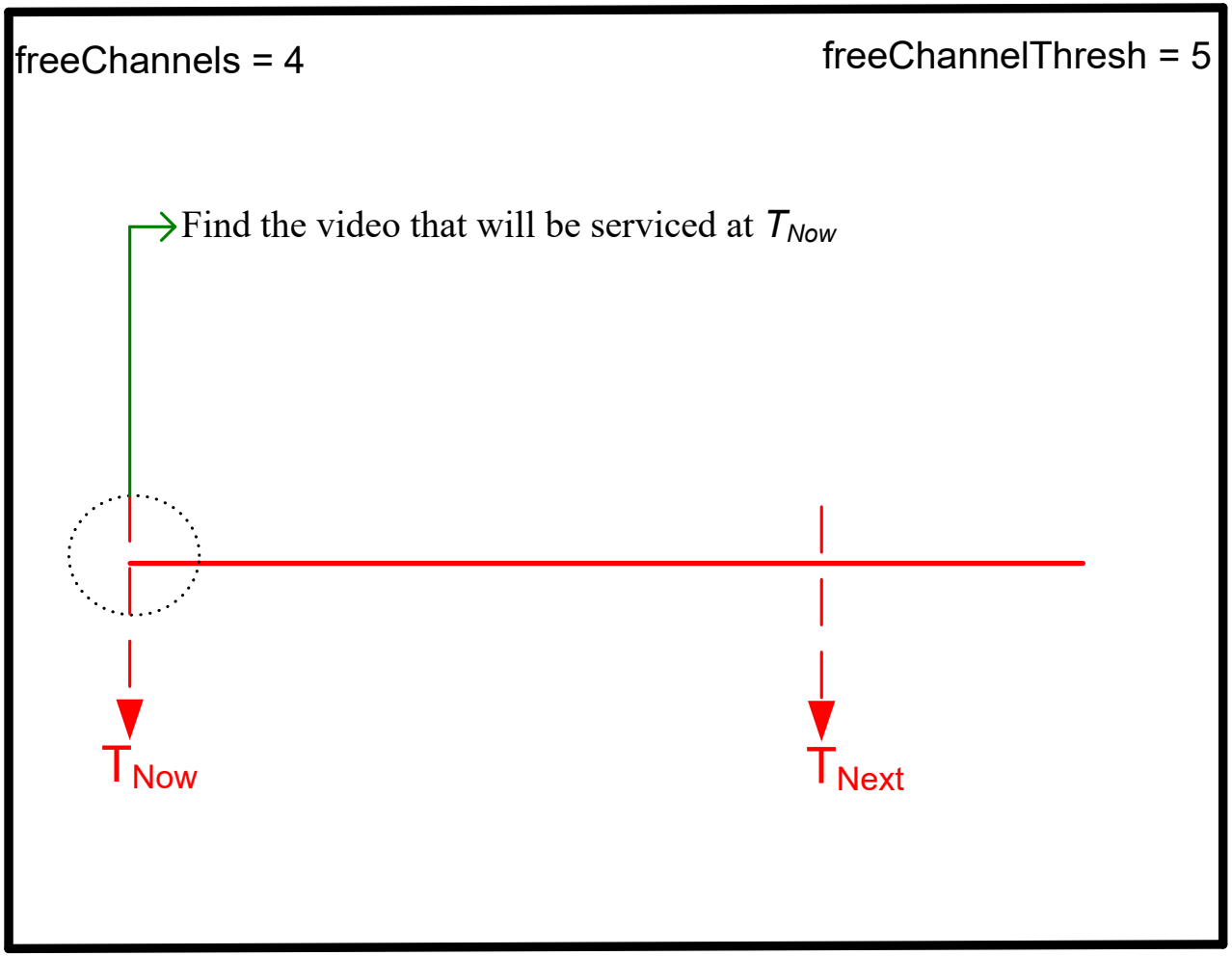
- PCS-V predicts the video to be serviced at the next scheduling time and simply uses its required stream length.
- The video prediction is done by utilizing detailed information about the current state of the server
  - In a manner similar to our waiting-time prediction approach in ACM Multimedia 2007 [8,10].
  - This information includes
    - # waiting requests for each video
    - Completion times of running streams
    - Statistics such as the average request arrival rate for each video

PCS-V

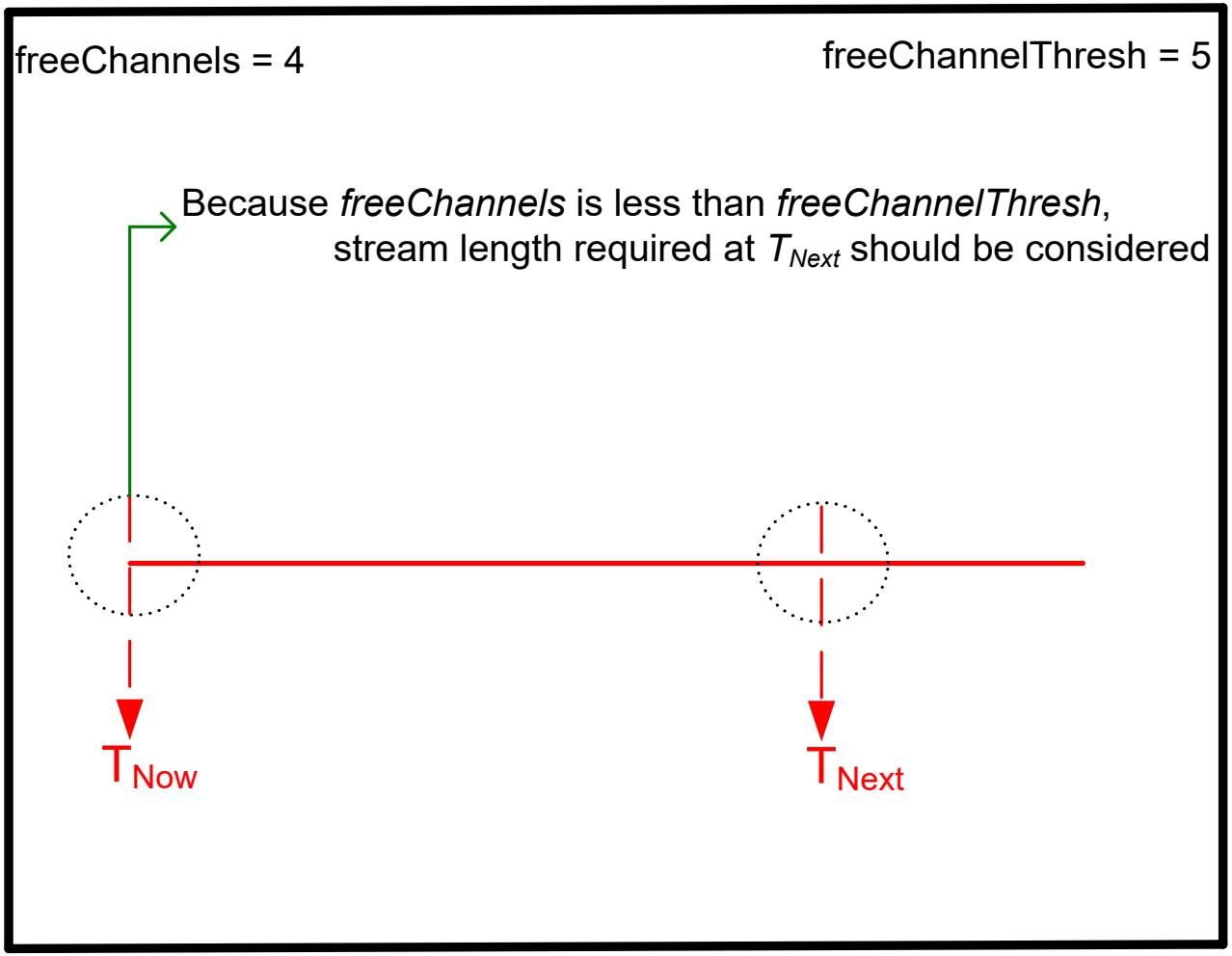
- Simplified Algorithm:

V_{Now} = find the video that will tentatively be serviced at T_{Now};
if $(freeChannels \geq freeChannelThresh)$
    Service the requests for V_{Now};
else {
    $currStreamLen$ =
        find required stream length to service V_{Now} at T_{Now};
    V_{Next} =
        find the video that is expected to be serviced at T_{Next};
    $nextStreamLen$ =
        find required stream length to service V_{Next} at T_{Next};
    if $(currSreamLen \leq nextStreamLen)$
        Service the requests for V_{Now};
}

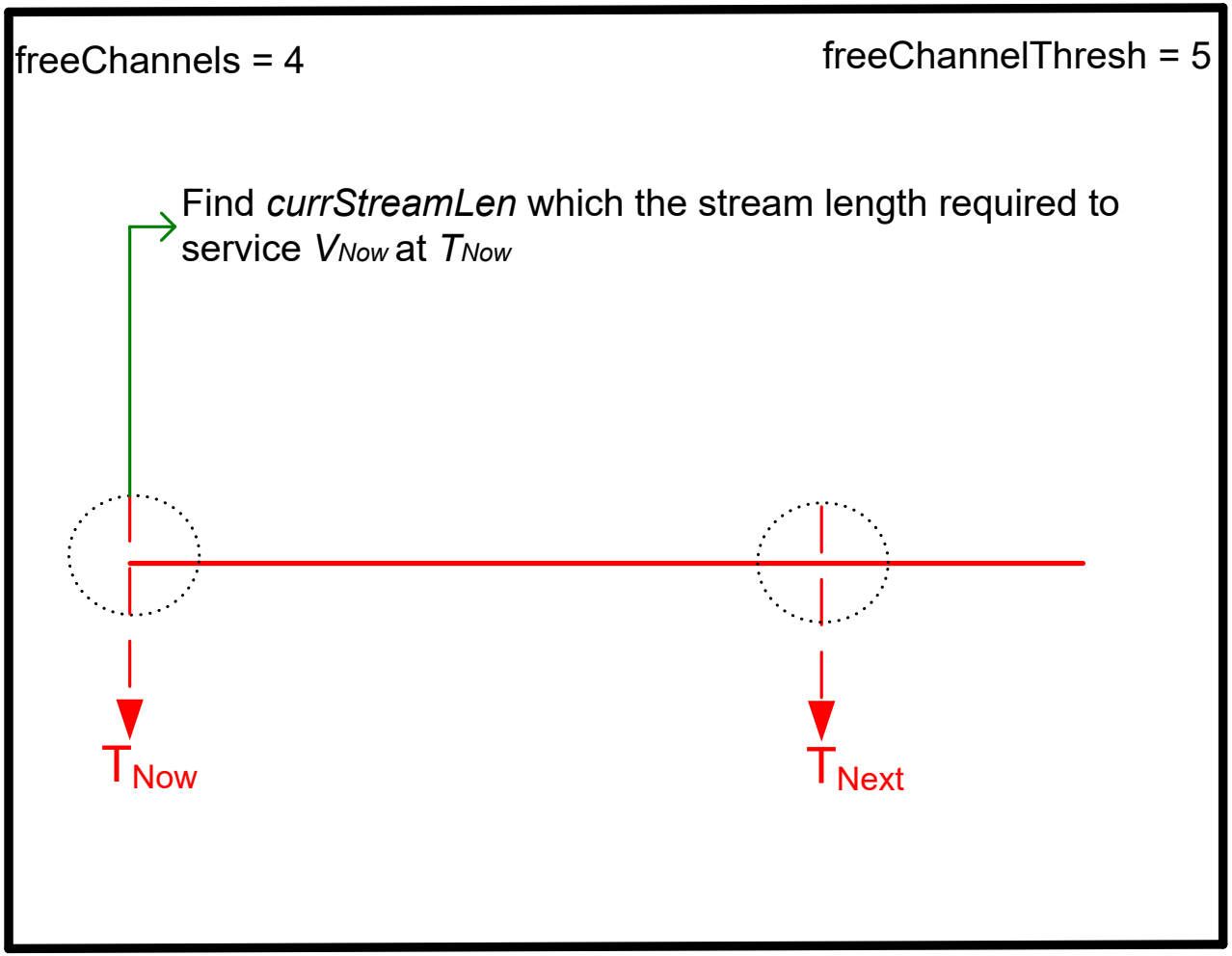
PCS-V Demo


freeChannels = 4
freeChannelThresh = 5
Find the video that will be serviced at T_{Now}
T Now
T Next

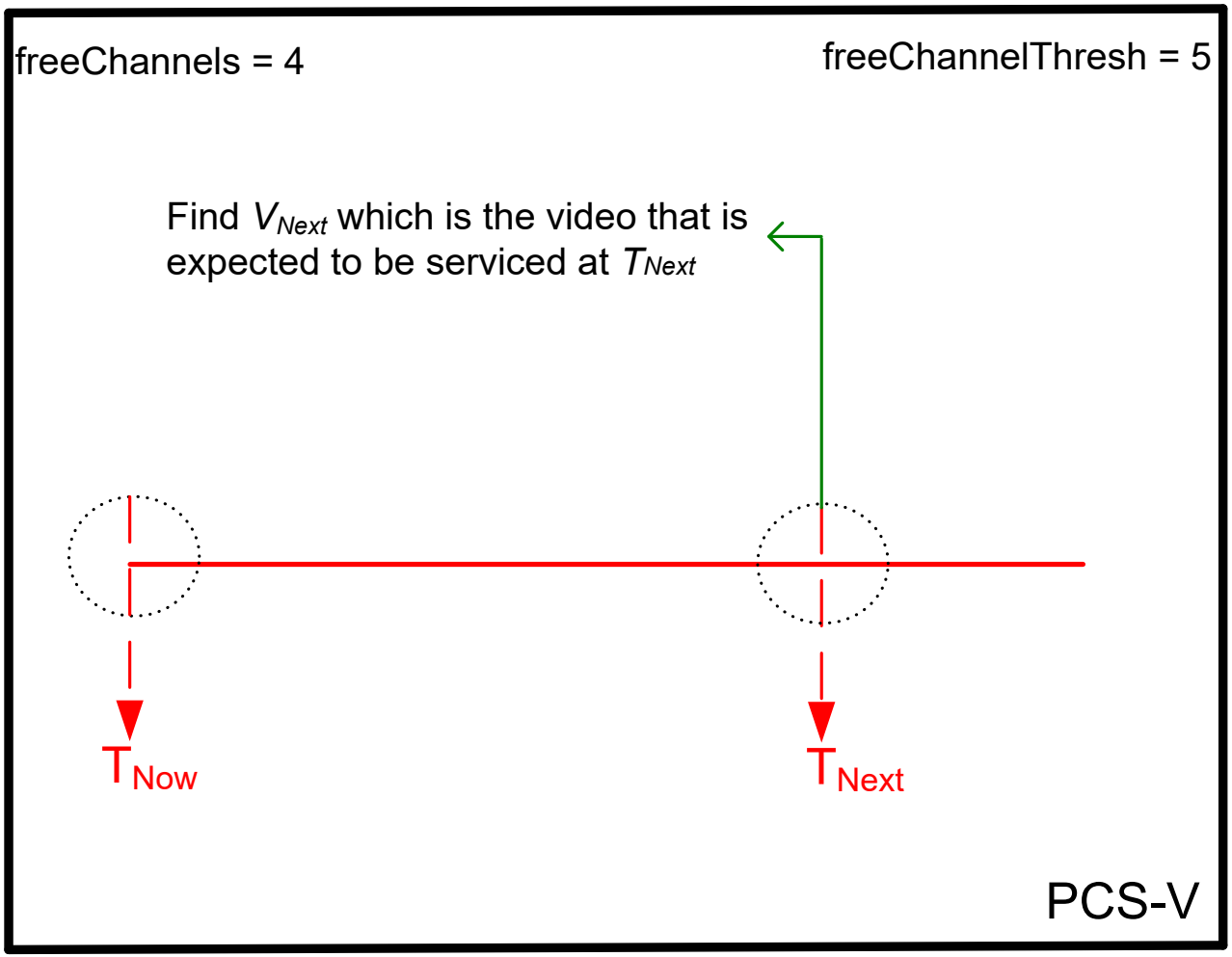
PCS-V Demo


freeChannels = 4
freeChannelThresh = 5
Because *freeChannels* is less than *freeChannelThresh*,
stream length required at T_{Next} should be considered
T Now
T Next

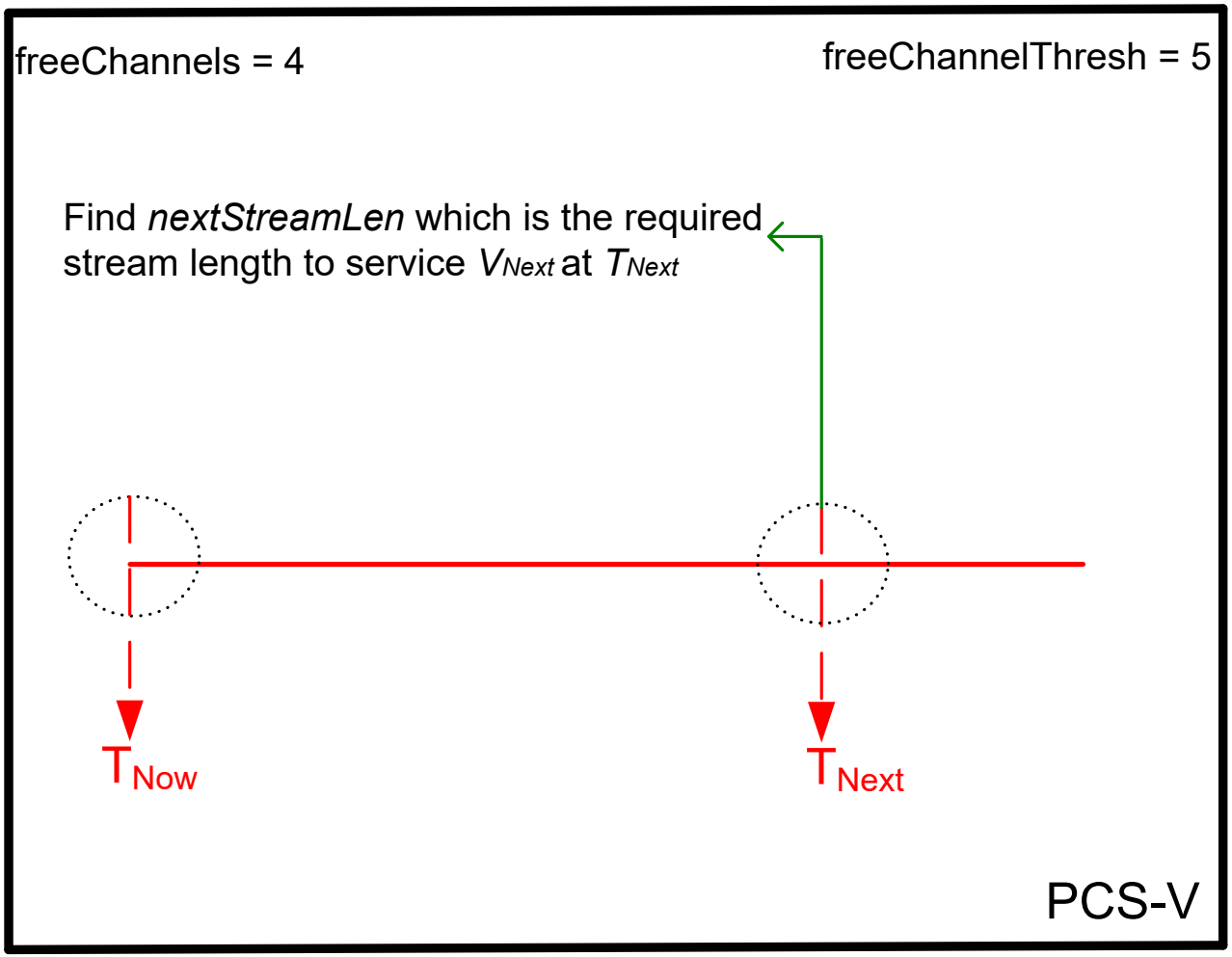
PCS-V Demo


freeChannels = 4
freeChannelThresh = 5
Find currStreamLen which the stream length required to service V_{Now} at T_{Now}
T_{Now}
T_{Next}

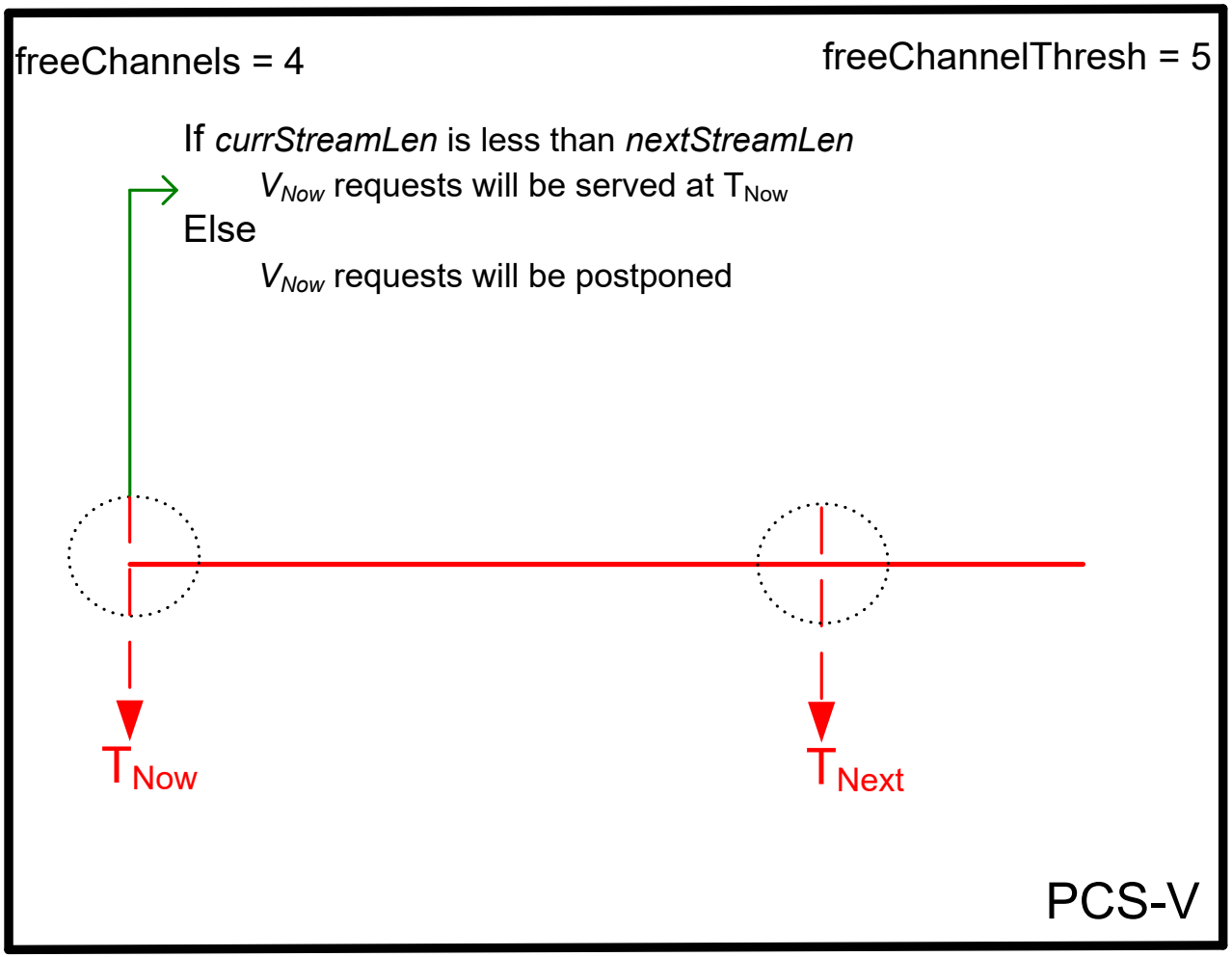
PCS-V Demo


freeChannels = 4
freeChannelThresh = 5
Find V_{Next} which is the video that is expected to be serviced at T_{Next}
T Now
T Next
PCS-V

PCS-V Demo


freeChannels = 4
freeChannelThresh = 5
Find *nextStreamLen* which is the required stream length to service V_{Next} at T_{Next}
T_{Now}
T_{Next}
PCS-V

PCS-V Demo

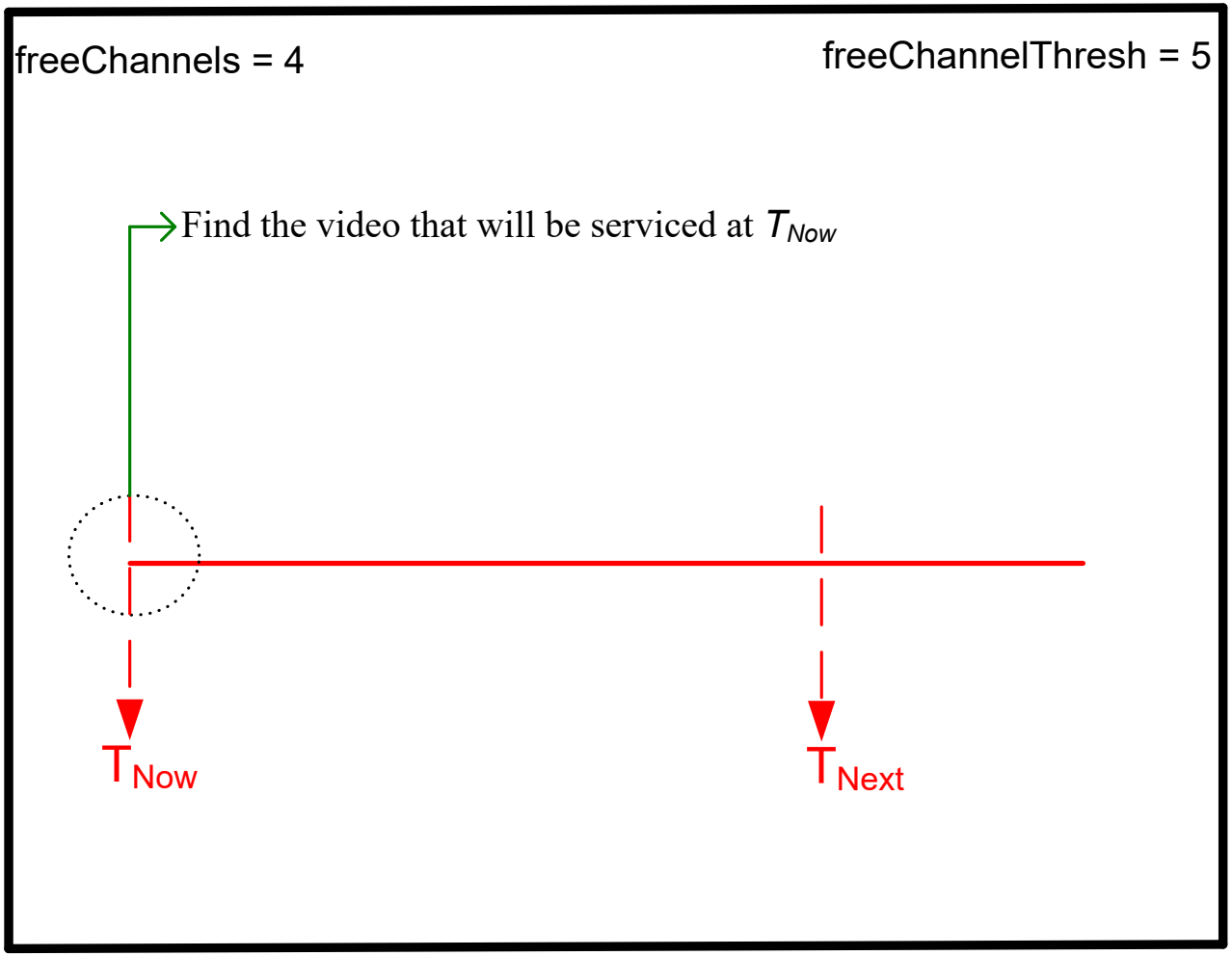

freeChannels = 4
freeChannelThresh = 5
If currStreamLen is less than nextStreamLen
V Now requests will be served at T Now
Else
V Now requests will be postponed
T Now
T Next
PCS-V

PCS-L

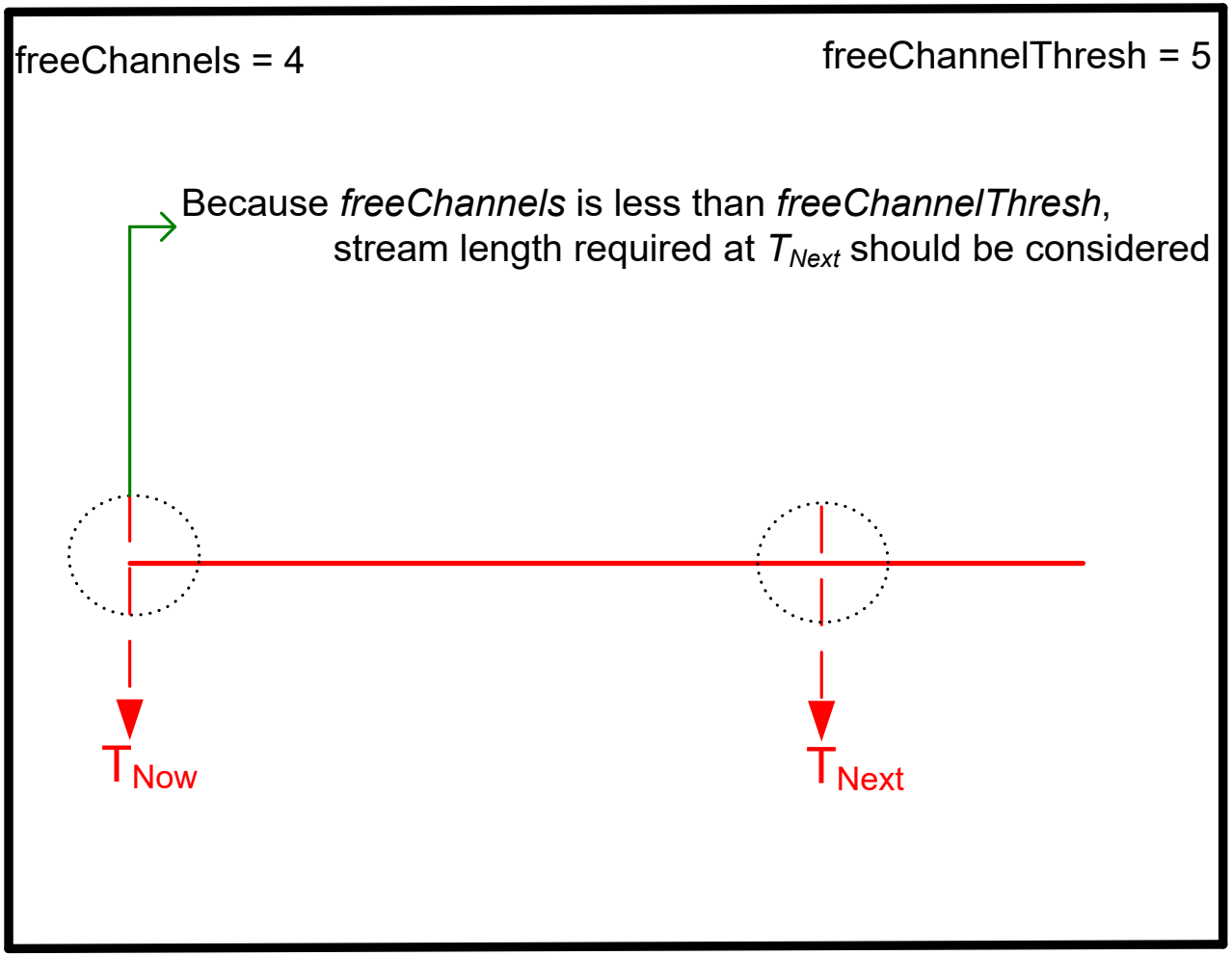
- PCS-L computes the expected required stream length at the next scheduling time based on
  - the lengths of all possible video streams
  - their probabilities

```
V_Now = find the video that will tentatively be serviced at T_Now;
if (freeChannels ≥ freeChannelThresh)
    Service the requests for V_Now;
else {
    currStreamLen =
            find required stream length to service V_Now at T_Now;
    Calculate objective function for each video at T_Next;
    Sort videos from best to worst according to objective function;
    expectedStreamLen = 0; // initialization
    // loop to find expected stream length at T_Next
    for (v = 0; v < N_v ; v++){ // for each video
        nextStreamLen =
                find required stream length to service v at T_Next;
        Prob(video v is selected) =
                Prob(no other video with better objective is selected)
                 * Prob(video v has at least one arrival);
        expectedStreamLen +=
                Prob(video v is selected ) * nextStreamLen;
    }
    if (currSreamLen ≤ expectedStreamLen)
        Service the requests for V_Now;
}
```

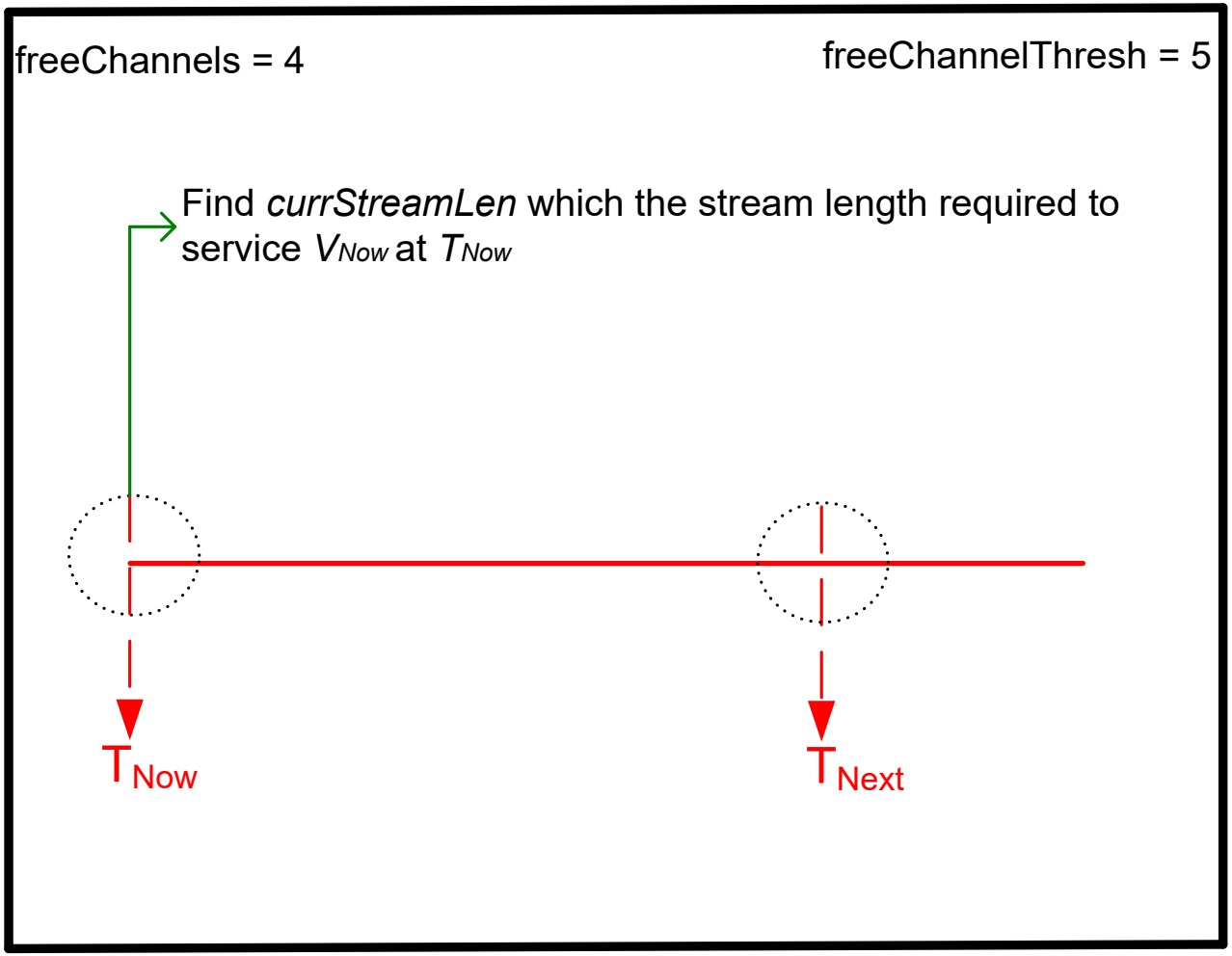
PCS-L Demo


freeChannels = 4
freeChannelThresh = 5
Find the video that will be serviced at T_{Now}
T Now
T Next

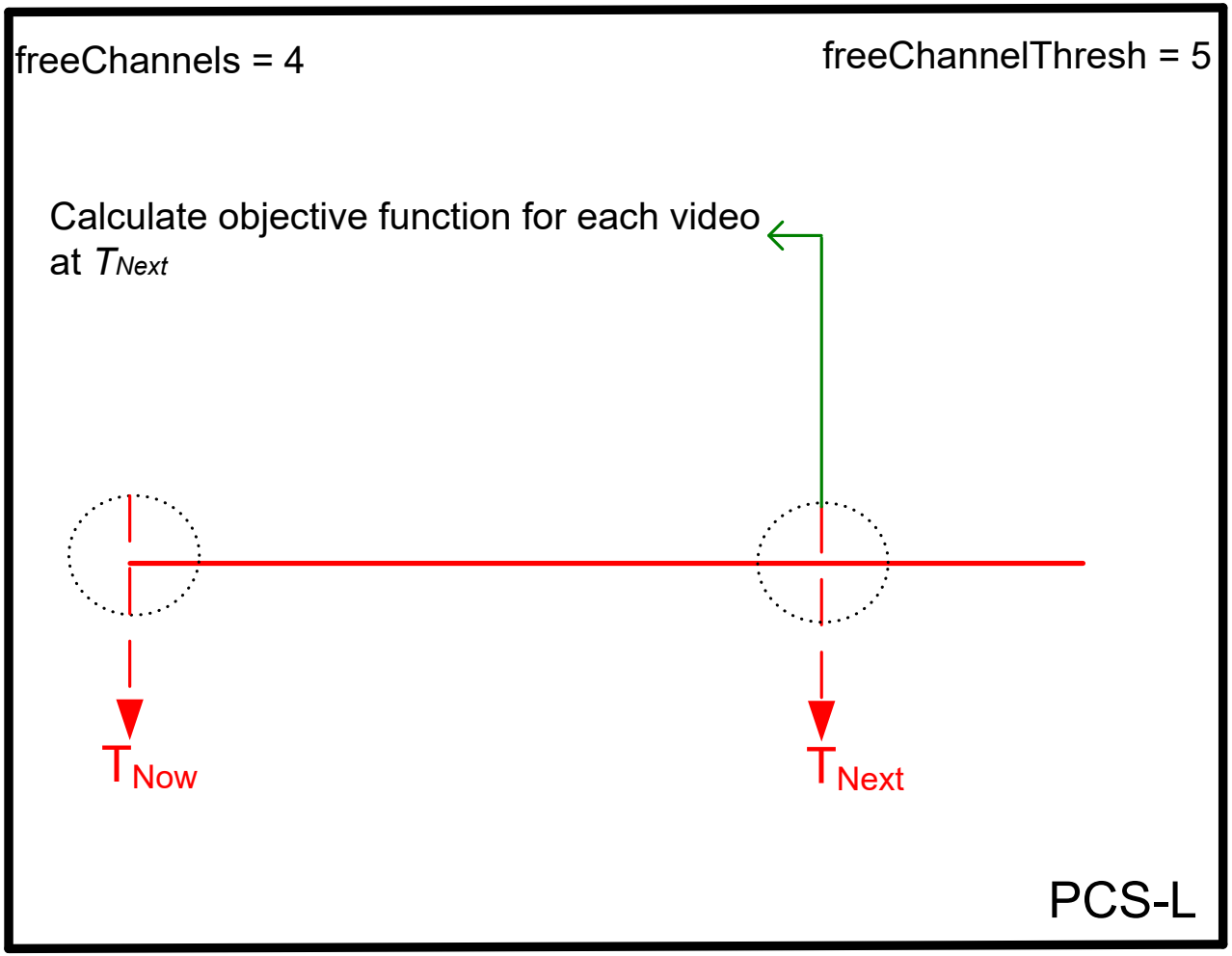
PCS-L Demo


freeChannels = 4
freeChannelThresh = 5
Because *freeChannels* is less than *freeChannelThresh*,
stream length required at T_{Next} should be considered
T Now
T Next

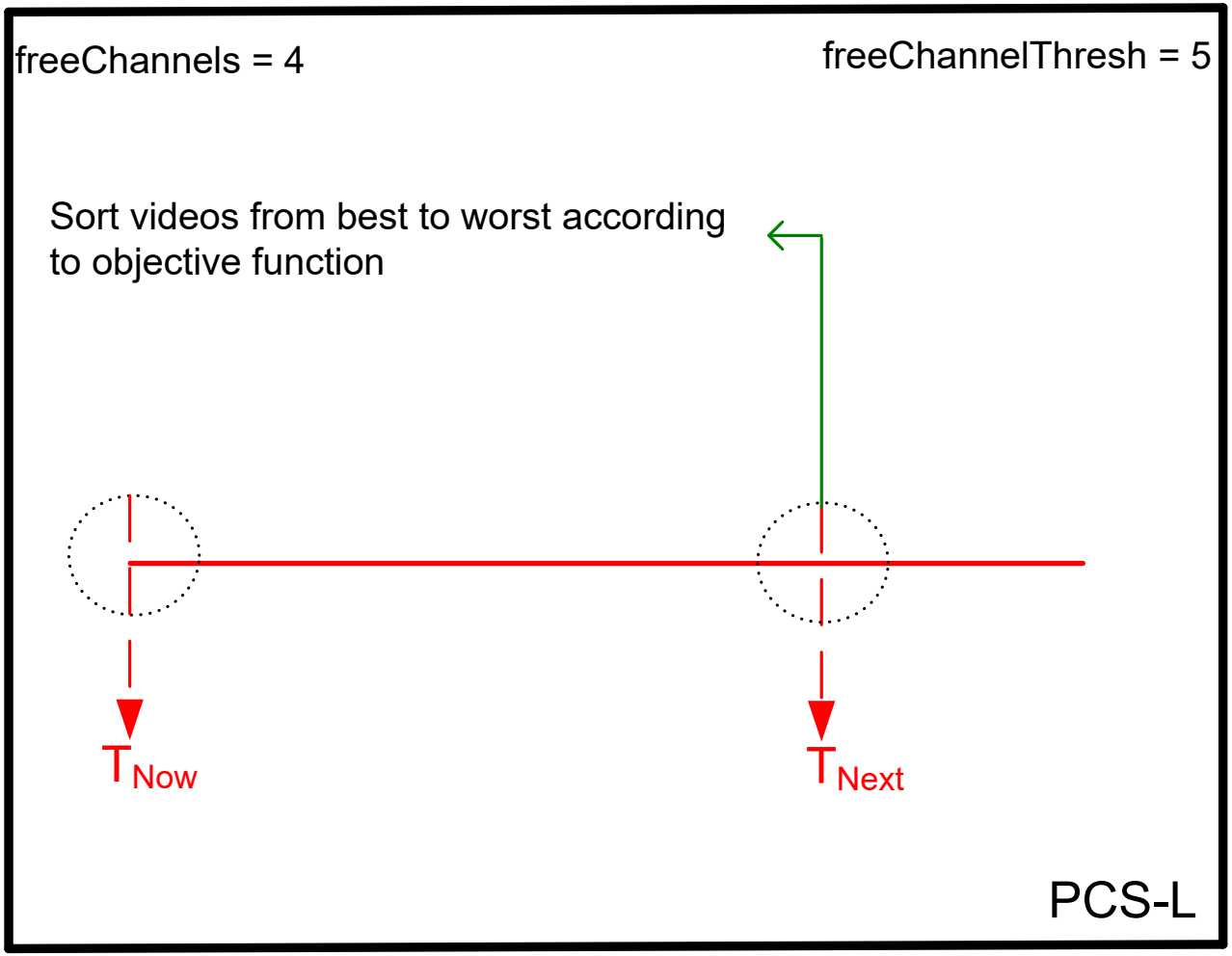
PCS-L Demo


freeChannels = 4
freeChannelThresh = 5
Find currStreamLen which the stream length required to service V_{Now} at T_{Now}
T_{Now}
T_{Next}

PCS-L Demo


freeChannels = 4
freeChannelThresh = 5
Calculate objective function for each video at T_{Next}
T_{Now}
T_{Next}
PCS-L

PCS-L Demo


freeChannels = 4
freeChannelThresh = 5
Sort videos from best to worst according to objective function
T Now
T Next
PCS-L

PCS-L Demo

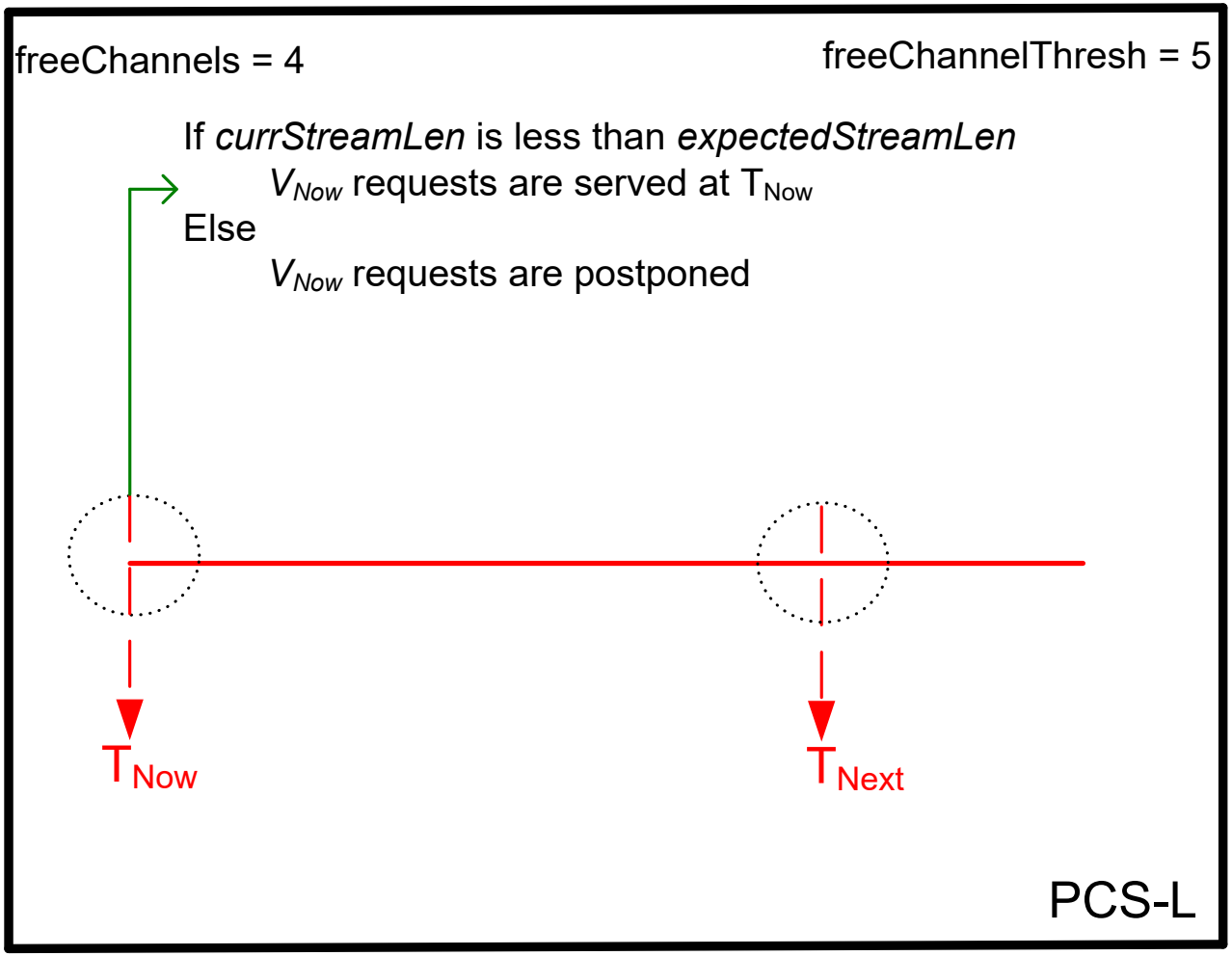
freeChannels = 4

freeChannelThresh = 5

Initialization *expectedStreamLen* = 0;
For each video in the sorted list
    nextStreamLen = find required stream length to service the video at T_{Next};
    Prob(this video is selected) =
        Prob(no other video with better objective is selected)
            * Prob(this video has at least one arrival);
    expectedStreamLen += Prob(this video is selected) * *nextStreamLen*;

PCS-L

PCS-L Demo


freeChannels = 4
freeChannelThresh = 5
If currStreamLen is less than expectedStreamLen
V_Now requests are served at T_Now
Else
V_Now requests are postponed
T Now
T Next
PCS-L

Evaluation Methodology

- We study the effectiveness of the proposed policy through simulation.
- The shown table summarizes the workload characteristics used.

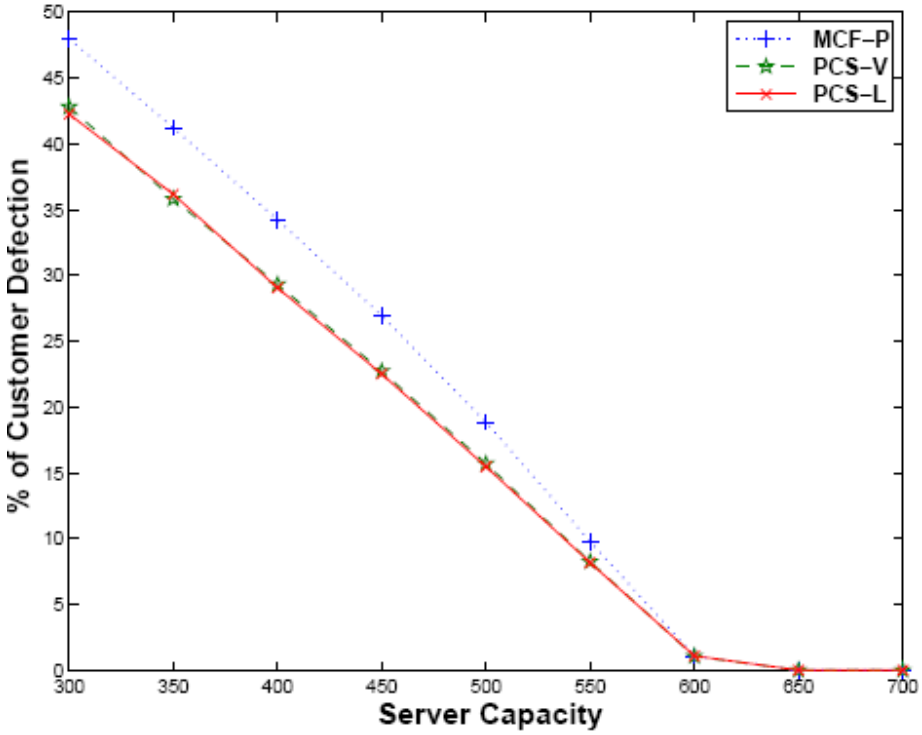
Parameter	**Model/Value(s)**
Request Arrival	Poisson Process
Request Arrival Rate	40 Req./min
Server Capacity	300 to 750 channels
Video Access	Zipf-Like ($\theta = 0.271$)
Number of Videos	120
Video Length	120 min
Waiting Tolerance Model	A, B, C
Waiting Tolerance Mean	$\mu_{tol} = 30$ sec

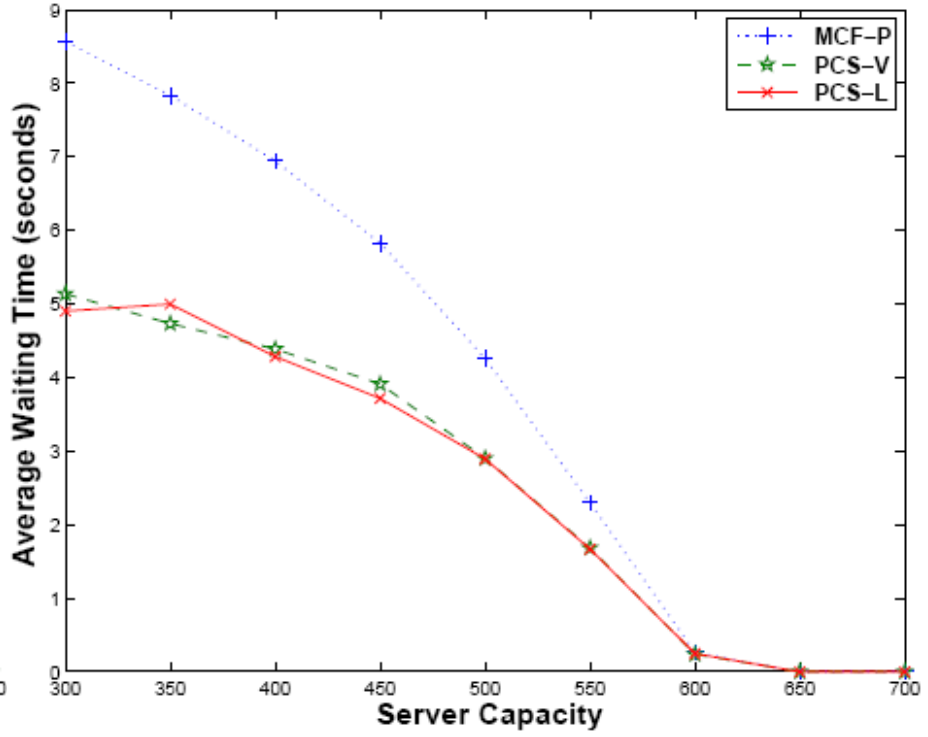
Evaluation Methodology (cont…)

- The analyzed performance metrics include
  - customer defection probability
  - average waiting time
  - Unfairness against unpopular videos
- The waiting-time predictability is analyzed by two metrics:
  - waiting-time prediction accuracy
  - % clients receiving expected waiting times

Results Presentation

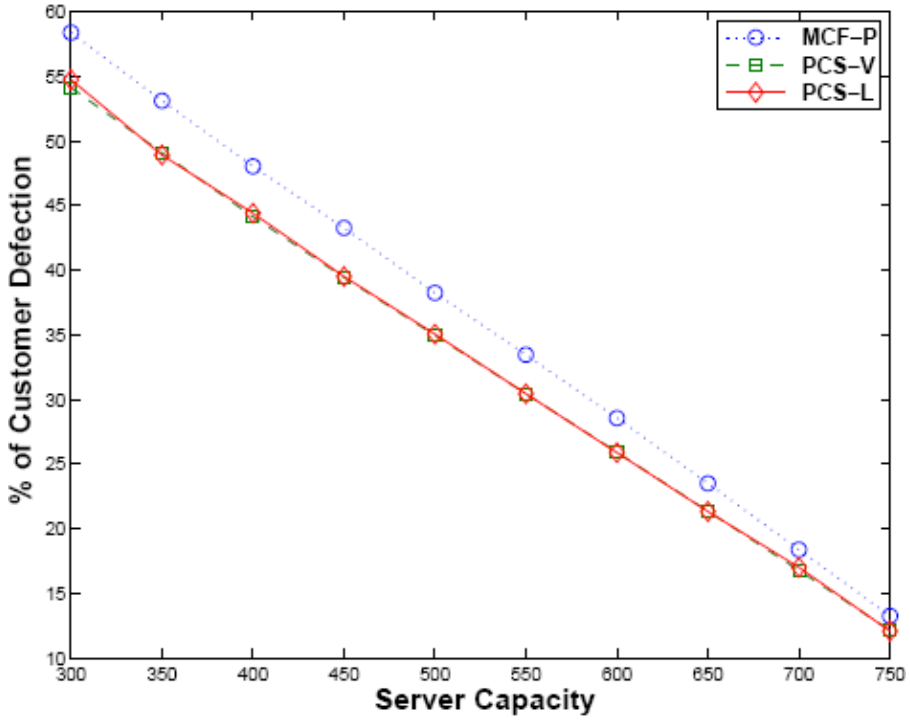
Effectiveness of PCS-V and PCS-L [*ERMT*]

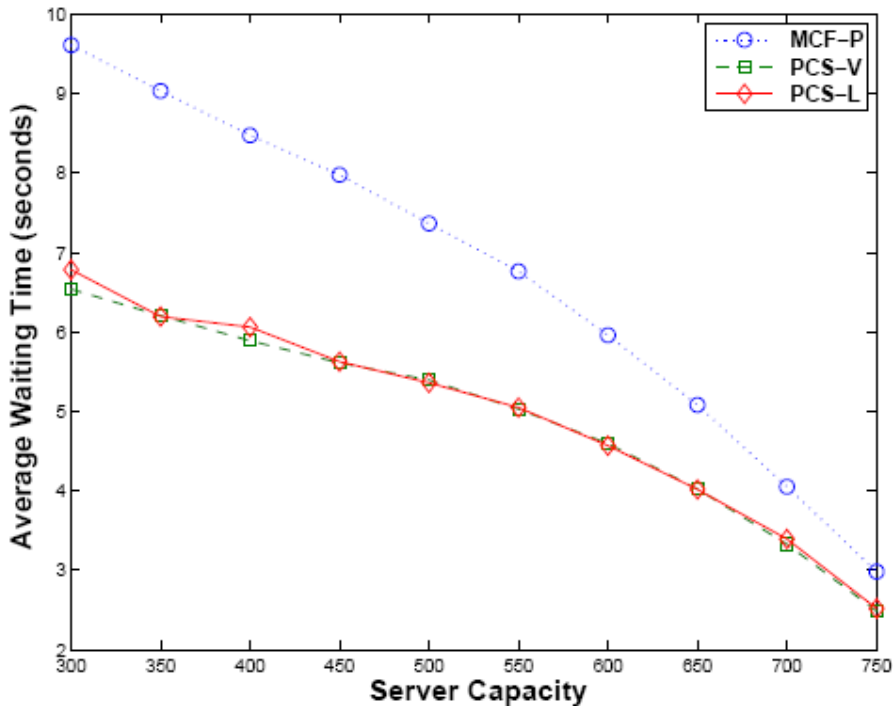




Results Presentation

Effectiveness of PCS-V and PCS-L [*Patching*]

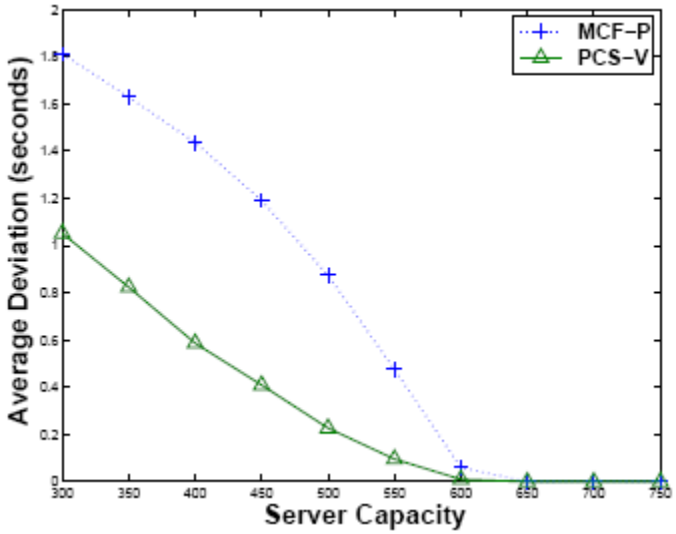


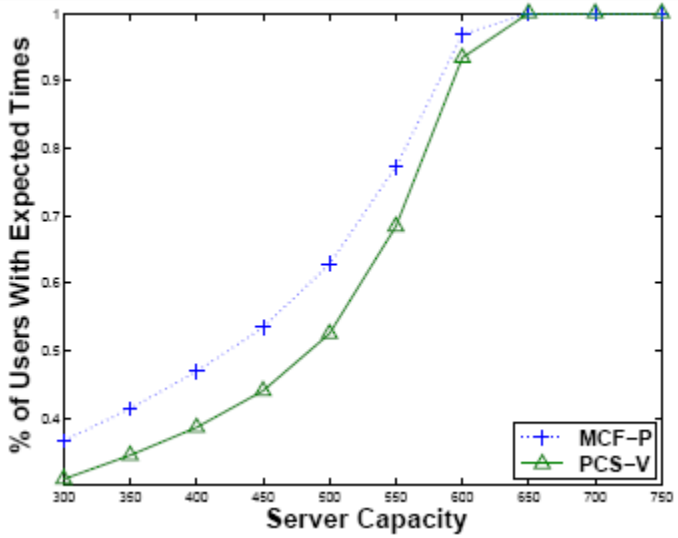


Results Presentation

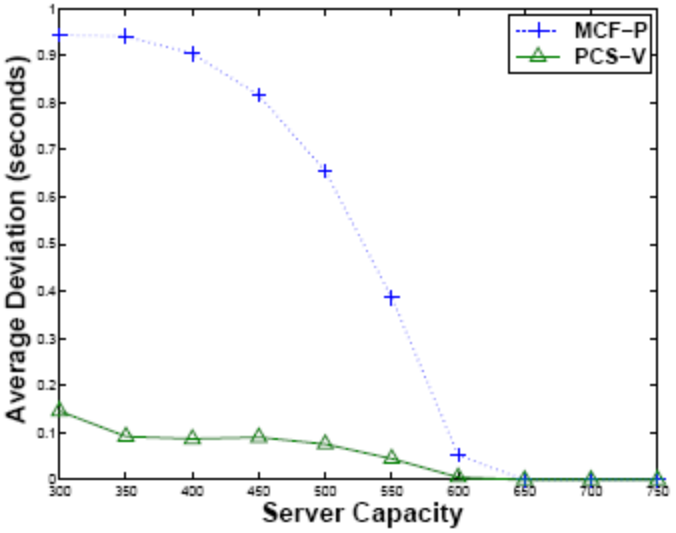
Predictability of PCS-V vs. MCF-P[ERMT, WP = $0.5\mu_{tol}$]

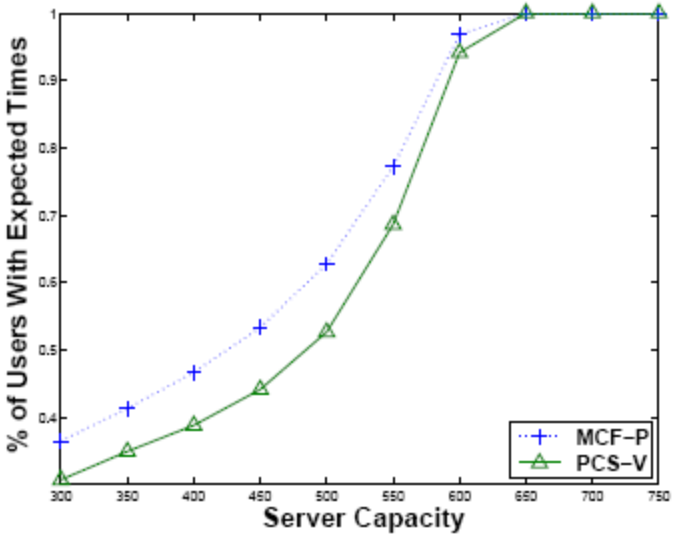
Defection Model B





Defection Model C





Conclusions

- The results show that the proposed PCS scheduling policy outperforms the best existing policy in terms of customer defection probability and average waiting time.
- The waiting times can also be predicted more accurately with PCS.

References

[1] Y. Cai and Kien A. Hua, “Sharing multicast videos using patching streams,” *Multimedia Tools and Applications journal*, vol. 21, no. 2, pp. 125–146, Nov. 2003.

[2] D. L. Eager, M. K. Vernon, and J. Zahorjan, “Optimal and efficient merging schedules for Video-on-Demand servers,” in *Proc. of ACM Multimedia*, Oct. 1999, pp. 199–202.

[3] B. Qudah and N. J. Sarhan, “Towards scalable delivery of video streams to heterogeneous receivers,” in *Proc. Of ACM Multimedia*, Oct. 2006, pp. 347–356.

[4] L. Shi, P. Sessini, A. Mahanti, Z. Li, and D. L. Eager, “Scalable streaming for heterogeneous clients,” in *Proc. of ACM Multimedia*, Oct. 2006, pp. 337–346.

References

[5] N. J. Sarhan and B. Qudah, "Efficient cost-based scheduling for scalable media streaming," in *Proc. of Multimedia Computing and Networking Conf. (MMCN)*, January 2007.

[6] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *Proc. of ACM Multimedia*, Oct. 1994, pp. 391–398.

[7] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "The maximum factor queue length batching scheme for Video-on-Demand systems," *IEEE Trans. on Computers*, vol. 50, no. 2, pp. 97–110, Feb. 2001.

[8] M. Alsmirat, M. Al-Hadrusi, and N. J. Sarhan, "Analysis of waiting-time predictability in scalable media streaming," in *Proc. of ACM Multimedia*, Sept. 2007, pp. 727 – 736.

[9] B. Qudah and N. J. Sarhan, "Analysis of resource sharing and cache management techniques in scalable video-on-demand," in *Proc. of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Sept. 2006, pp. 327–334.

References

[10] Nabil J. Sarhan, Mohammad A. Alsmirat, and Musab Al-Hadrusi. Waiting-Time Prediction in Scalable On-Demand Video Streaming . ACM Transactions on Multimedia Computing, Communications, and Applications (ACM TOMCCAP), Volume 6, Issue 2, March 2010. DOI: https://doi.org/10.1145/1671962.1671967.

[11] Mohammad Alsmirat and Nabil J. Sarhan. Predictive Cost-Based Scheduling for Scalable Video Streaming. In Proceedings of the IEEE International Conference on Multimedia & Expo (ICME 2008), pages 857 - 860, Hannover, Germany, June 2008. Acceptance rate for oral presentations: 20%. DOI: https://doi.org/10.1109/ICME.2008.4607570.