Exact GPS simulation and its application to an optimally fair scheduler

Paolo Valente
Dip. Ingegneria dell'Informazione
Università di Pisa - Italy

2/9/2004

Subject

- Context: Packet scheduling algorithms
- System: *N packet flows* sharing a link that can transmit only one packet at a time
- Contributions: computational complexity of
  - the *simulation* of the Generalized Processor Sharing (GPS) server, and
  - the *implementation* of the Worst-case Fair Weighted Fair Queuing (WF^2Q) scheduling algorithm

reduced from *O(N)* to *O(logN)* per packet transmission time

Summary

- Background on GPS and WF^2Q
- State of the Art
- L-GPS: simulating GPS at *O(logN)* cost
- L-WF^2Q: implementing WF^2Q at *O(logN)* cost

GPS definition

- The GPS server serves all *backlogged* flows *simultaneously*, providing each flow *i* an amount of service:

$$dW_i(t) = \frac{\phi_i}{\Phi(t)} dW(t)$$

- ϕ_i: weight of *flow i*
- $\Phi(t)$: sum of ϕ_i of the flows backlogged at time t
- $dW(t) = \underline{R(t)} \cdot dt$: total amount of service provided by the system at time *t*

GPS benefits

- Due to its perfectly fair allocation, the GPS server can be used as a reference system for:
  - Evaluating the fairness of practical packet schedulers
  - Implementing fair packet schedulers through on-line simulation of a GPS server
- Fairness measure: maximum per-flow *deviation* with respect to the GPS service

WF^{2}Q

- No packet scheduler can avoid a *minimum* deviation equal to one maximum packet size

- WF^{2}Q guarantees the minimum deviation

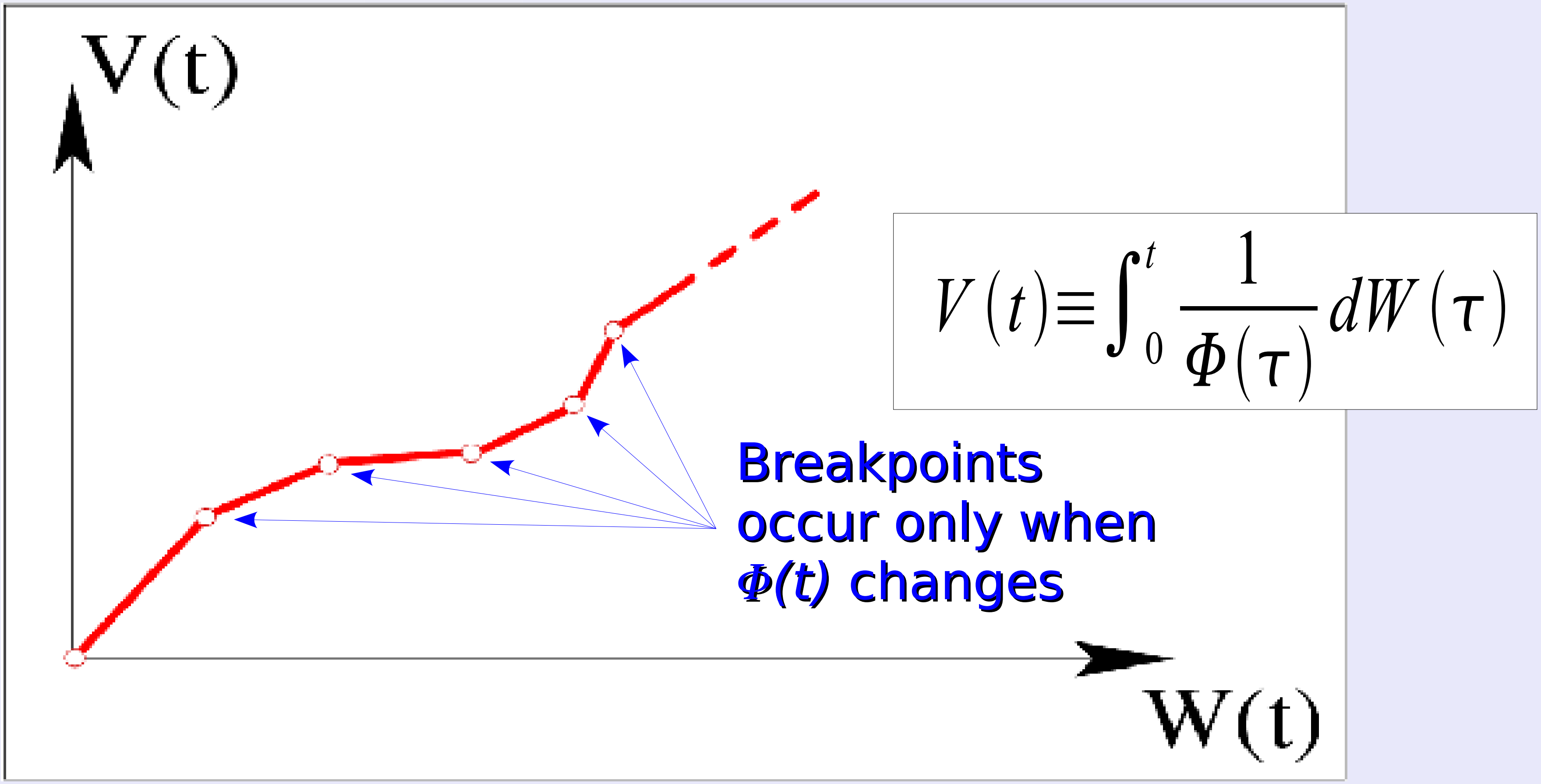
- WF^{2}Q internally simulates a GPS server by tracking a function called *system virtual time*
  - Timestamping arriving packets
  - Choosing next packet to transmit

System virtual time 1/2

- System virtual time function:

$$V(t) \equiv \int_0^t \frac{1}{\Phi(\tau)} dW(\tau)$$

System virtual time 2/2



- Hereafter we will report *W(t)* instead of *t* on the x-axis
  - Since *W(t)* is an increasing function of time, there is a one-to-one correspondence between *t* and *W(t)*

Summary

- Background on GPS and WF^2Q
- State of the Art
- L-GPS: simulating GPS at *O(logN)* cost
- L-WF^2Q: implementing WF^2Q at *O(logN)* cost

Summary

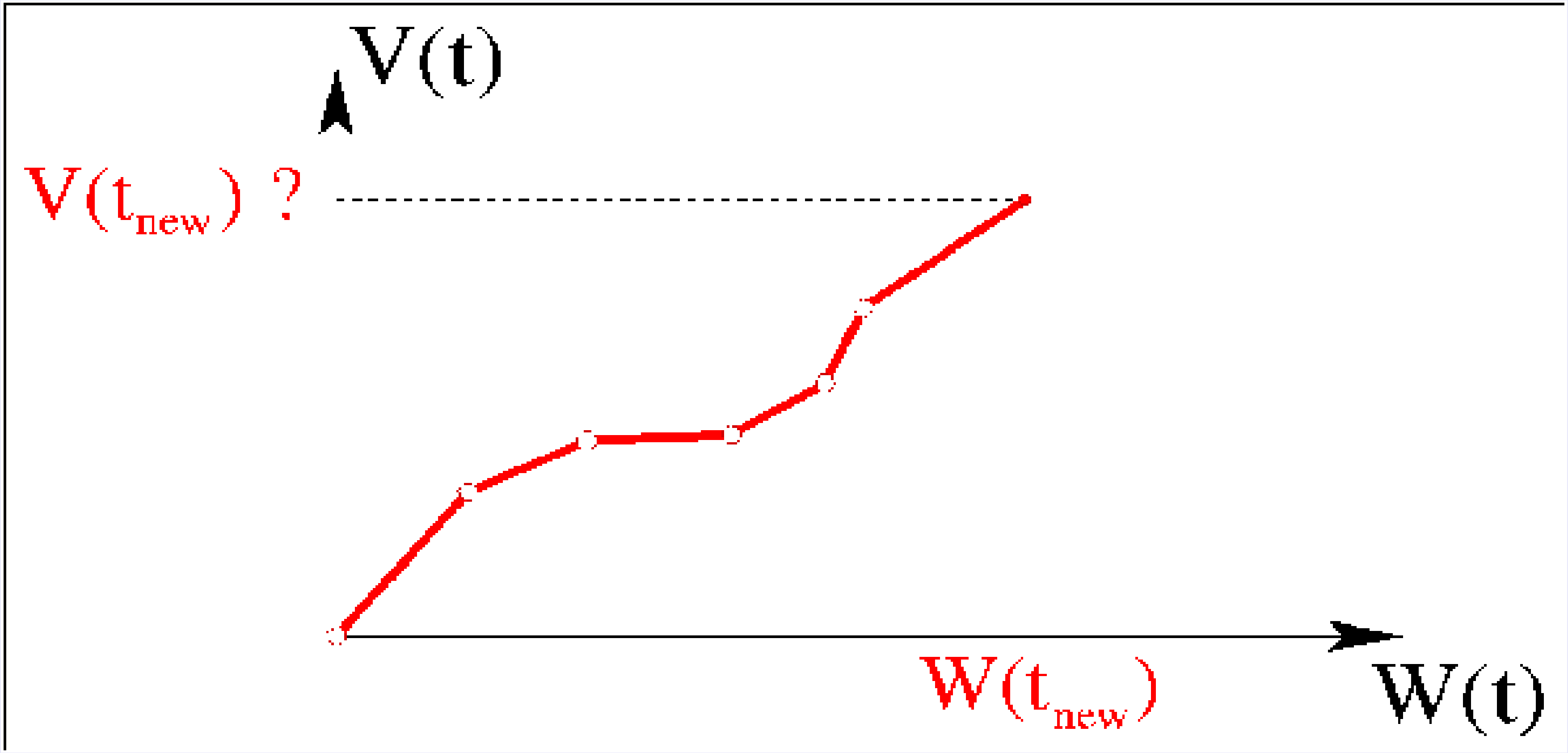
- State of the Art:
  - GPS emulation
  - GPS simulation

GPS emulation

- To the literature, all packet schedulers, apart from WF²Q, exhibit *O(N)*, or, worse yet, un-bounded deviation with respect to the GPS service
- One of them, called Worst-case Fair Weighted Fair Queueing Plus (WF²Q+) has *O(1)* deviation with respect to the *minimum* service guaranteed by the GPS server ...
- ... but *O(N)* deviation when some flows are idle

GPS simulation

- Provided that *W(t)* is known at any time instant, compute $V(t_{new})$ at a generic time instant t_{new}



Algorithms for simulating GPS

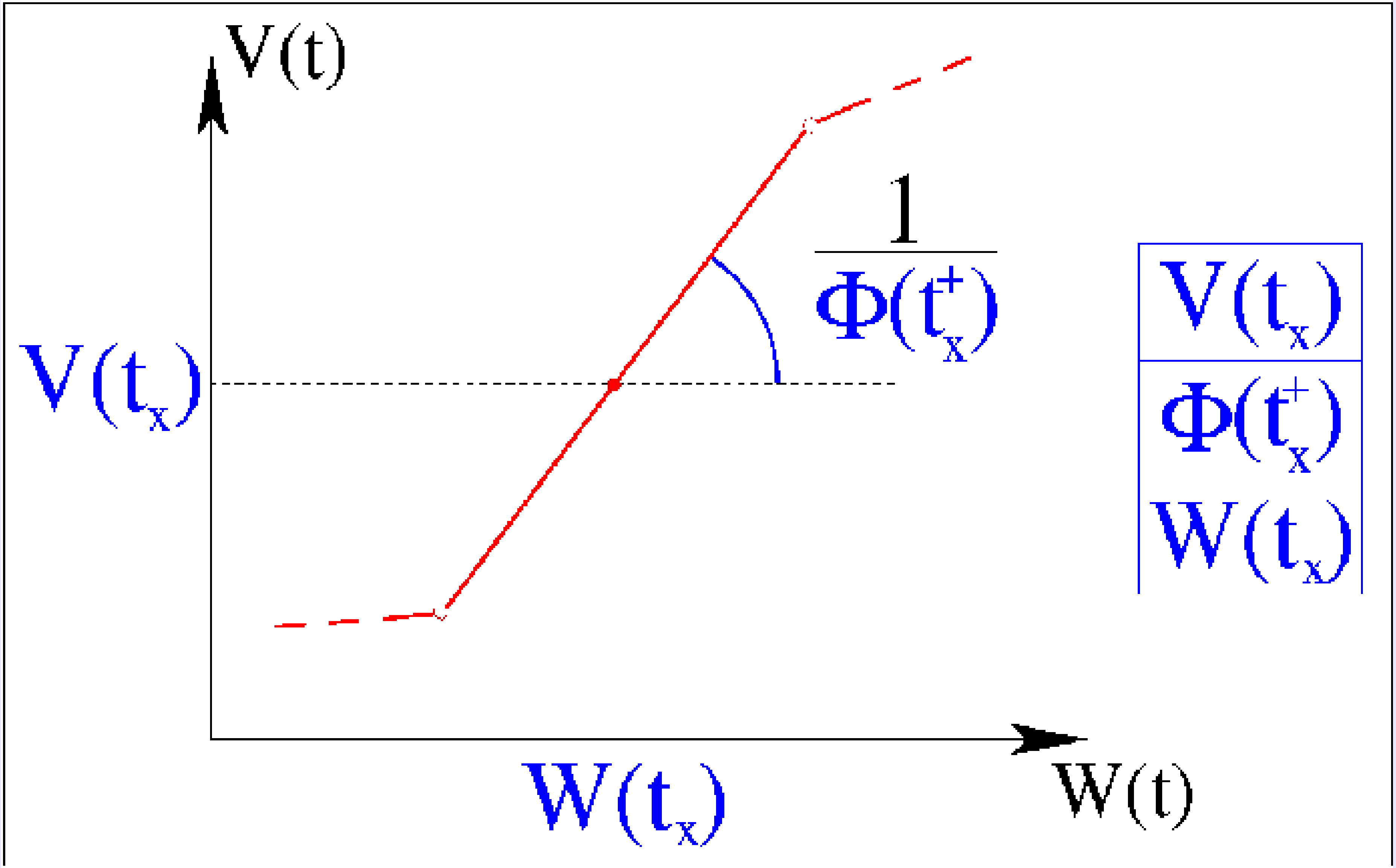
- *Two* algorithms in the past literature:

  1) The *classical* algorithm [Parekh and Gallager, 1992]

  2) Another algorithm [Greenberg and Madras, 1992] recently *re-discovered* [Zhao and Xu, 2004]

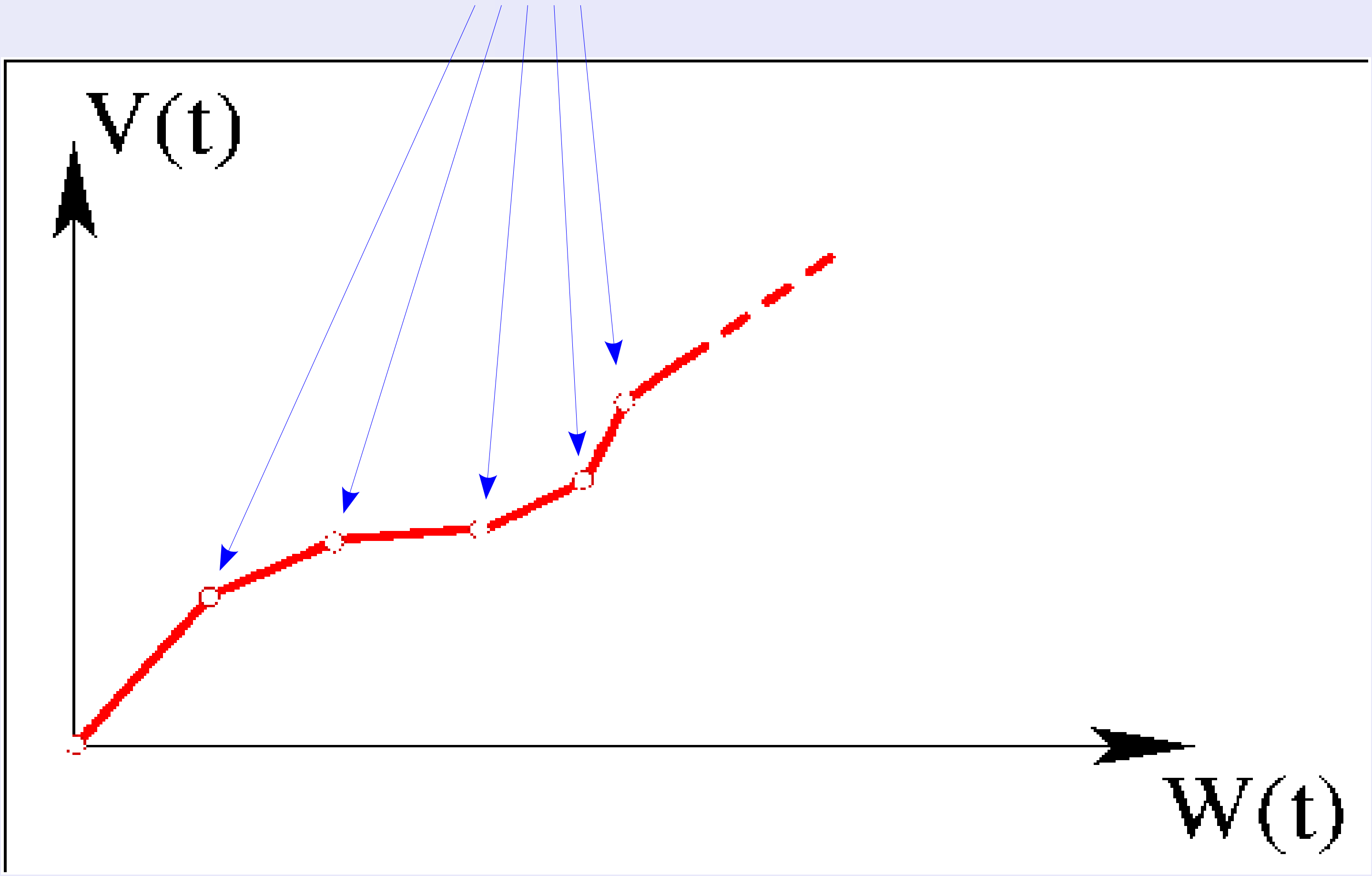
- For describing both of them, we will use the concept of <u>state of the GPS server</u>

State of the GPS server



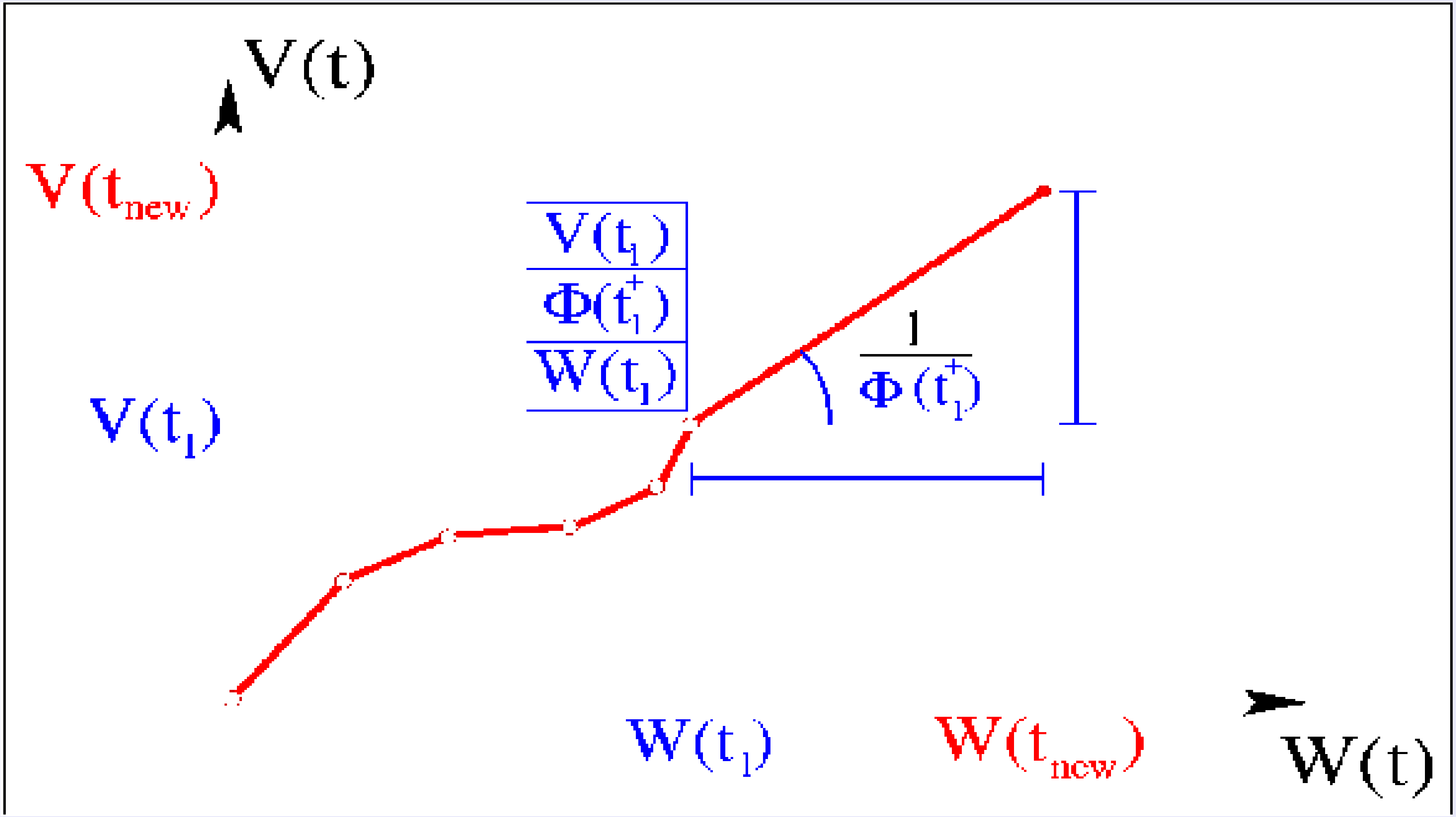
Classical algorithm 1/2

- Update the *state variables* each time *Φ(t)* changes



Classical algorithm 2/2

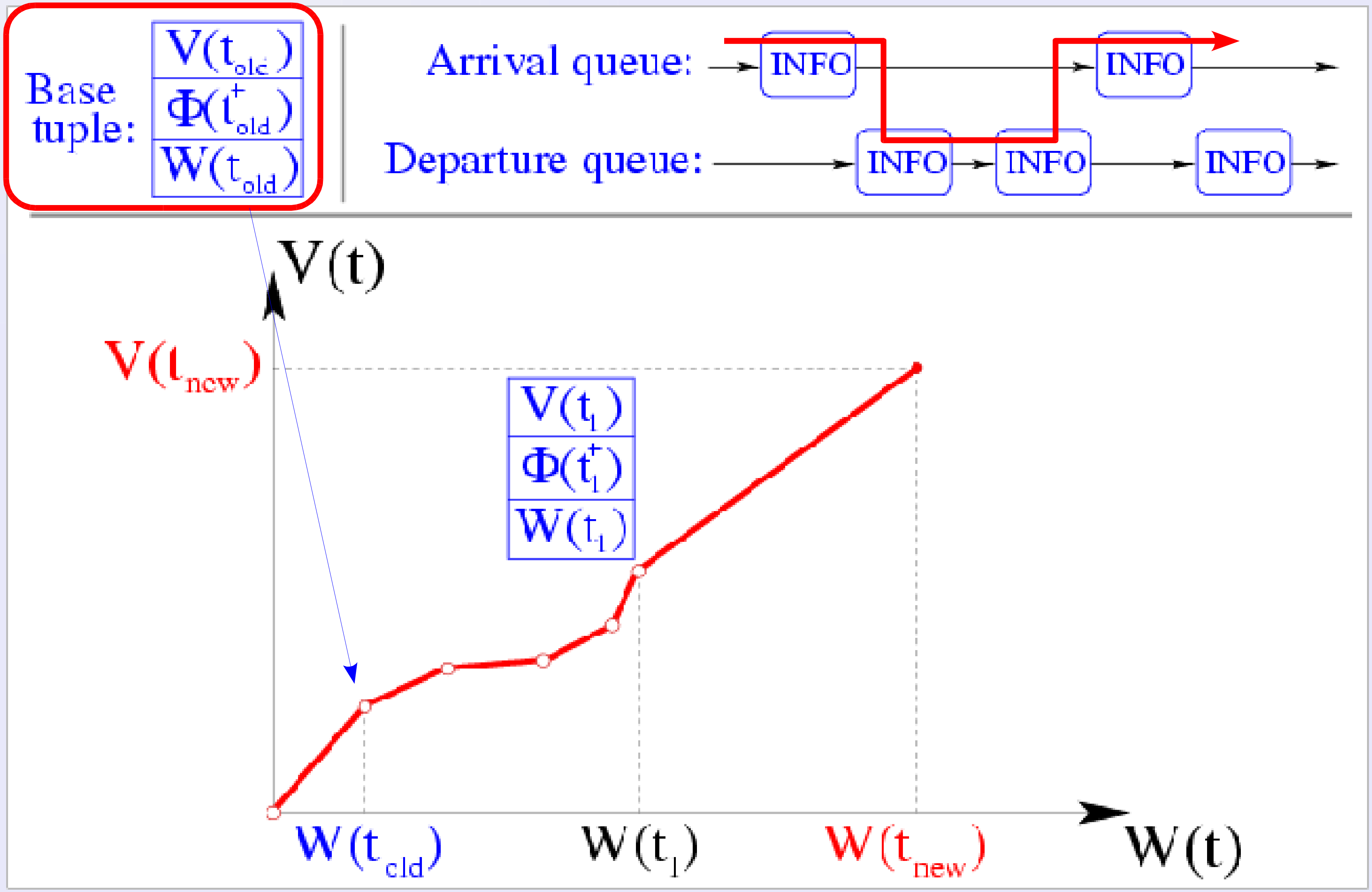
- Let t_l be the smallest time instant such that $\Phi(t)$ does not change in $(t_l, t_{new}]$ …



Greenberg et al. algorithm 1/2

- Variant
  - Store the state in a *base tuple*
  - Do not update the base tuple each time $\Phi(t)$ changes
    - Update it only on packet arrivals

Greenberg algorithm 2/2



Both algorithms are *O(N)*

- *O(N)* departures can occur in an arbitrarily short time interval, e.g. minimum packet transmission time

- Both algorithms make one step per event (arrival/departure), hence they have *O(N)* complexity per packet transmission time

Summary

- Background on GPS and WF^2Q
- State of the Art
- L-GPS: simulating GPS at *O(logN)* cost
- L-WF^2Q: implementing WF^2Q at *O(logN)* cost

Main idea 1/3

- The state changes *O(N)* times in an arbitrarily short time interval...
- ... but a practical scheduler does not need to know $V(t_{new})$ so often
- We can use a solution similar to the Greenberg and Madras algorithm previously shown
  - Store the state in a *base tuple*
  - Do not update the base tuple each time $\Phi(t)$ changes, but only on packet arrivals
- When $V(t_{new})$ is to be computed the base tuple contains the state corresponding to $t_{old} \leq t_l$ ...

Main idea 2/3

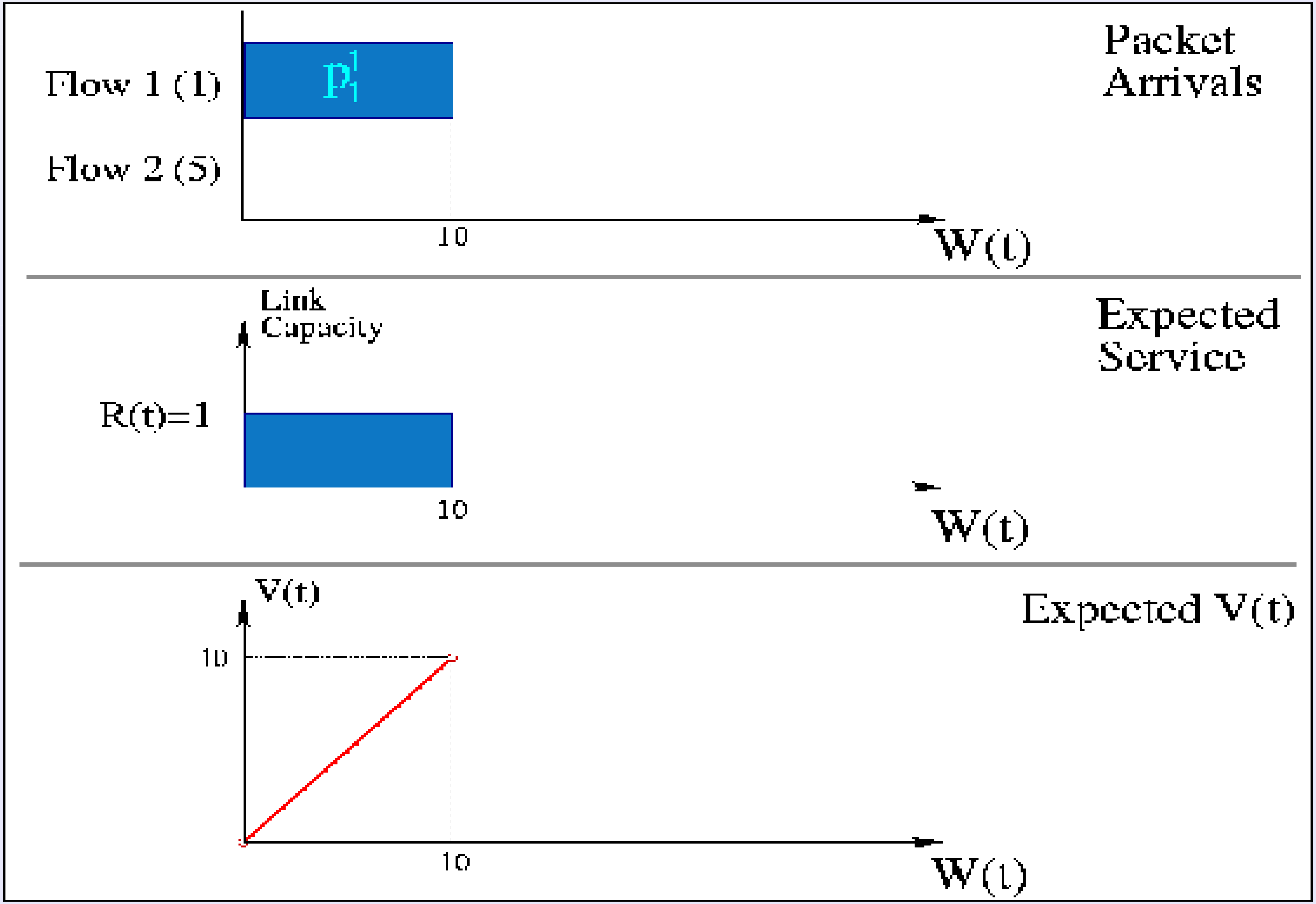
- Reconstruct the evolution of $V(t)$ from t_{old} to t_{new} through a specially augmented balanced binary tree, called U_{tree}

- In the Greenberg algorithm the events, stored in two queues, had to be processed one after the other, whereas L-GPS processes them in groups by navigating the U_{tree}
- L-GPS computes $V(t_{new})$ in $O(d)$ steps, where d is the depth of the U_{tree}

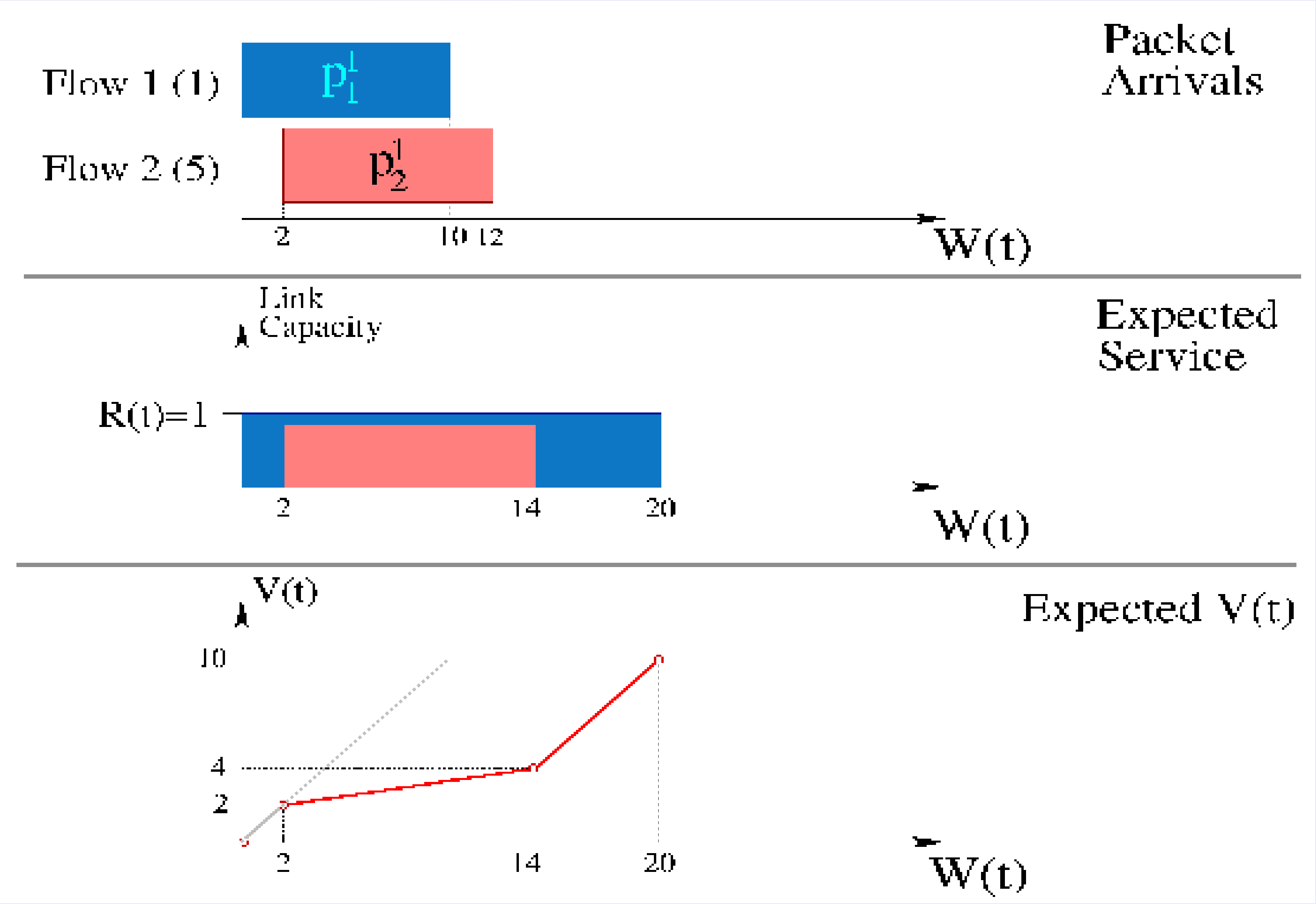
Main idea 3/3

- The idea behind the construction of the U_{tree} is *pre-computing* the *expected* evolution of *V(t)*

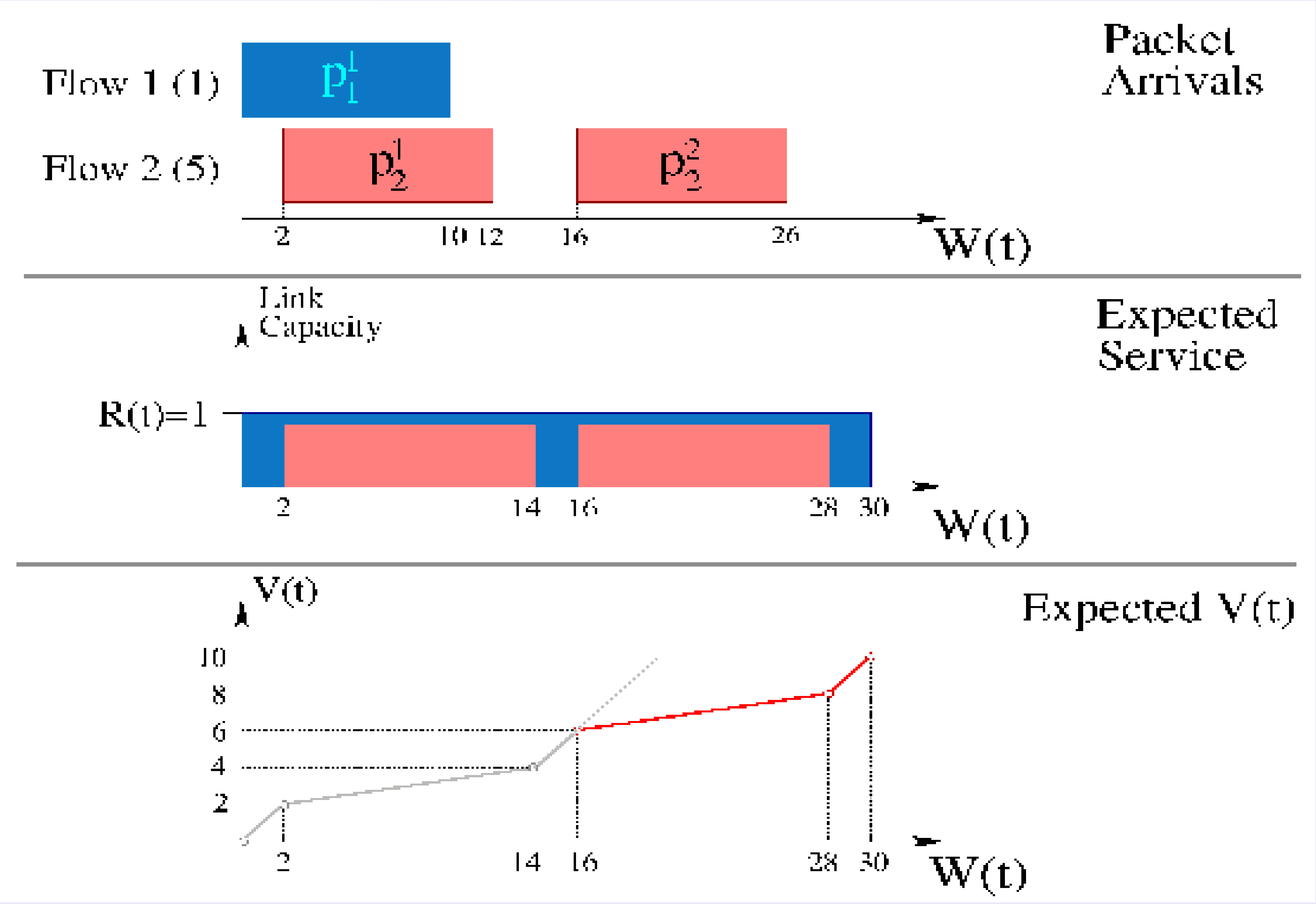
The expected evolution 1/3



The expected evolution 2/3



The expected evolution 3/3



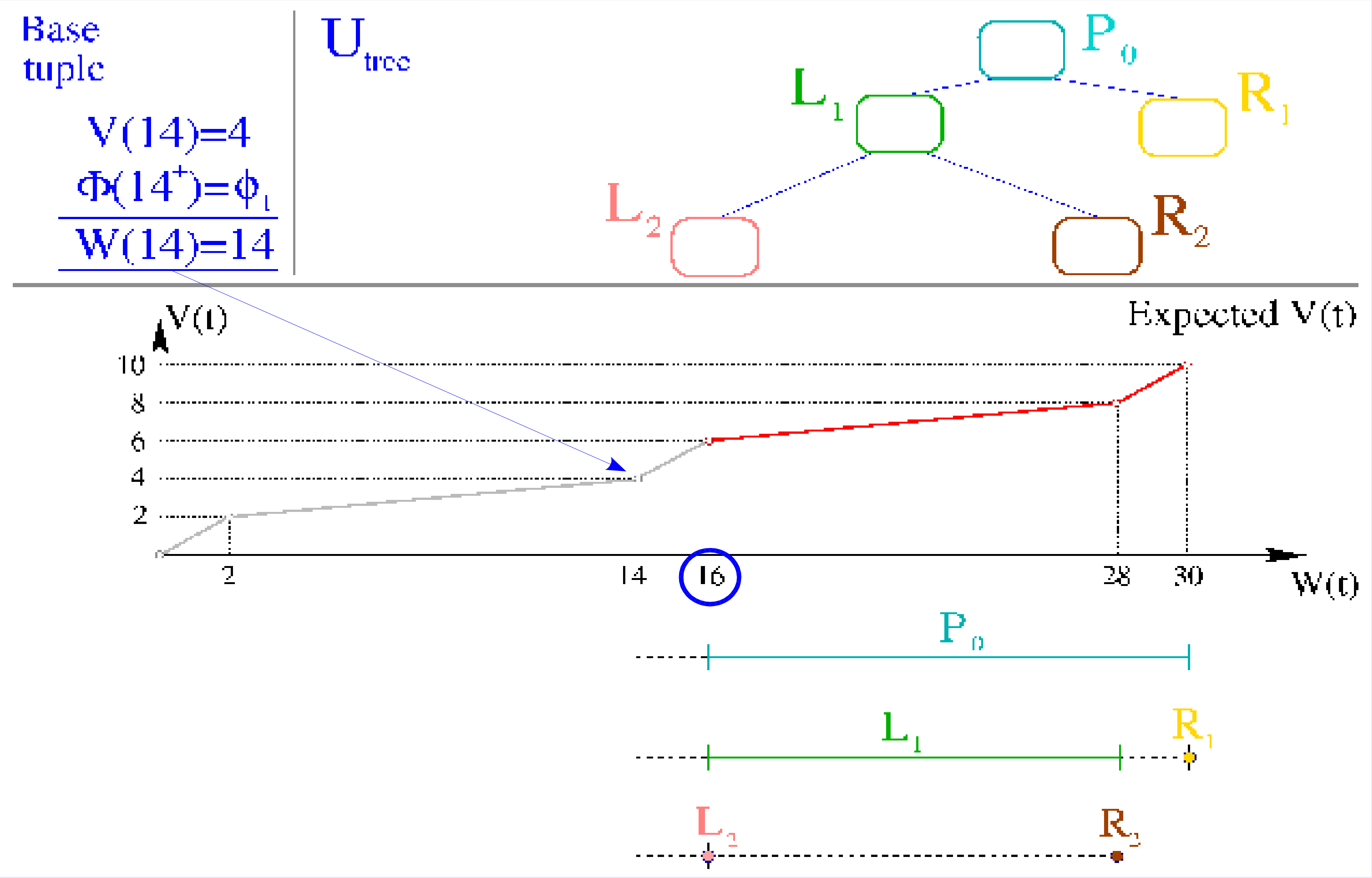
The shape data structure 1/2

- *Pre-computing* the expected evolution of *V(t)* upon each packet arrival is straightforward

- L-GPS stores in the base tuple and the U_{tree} information on the expected evolution of *V(t)*

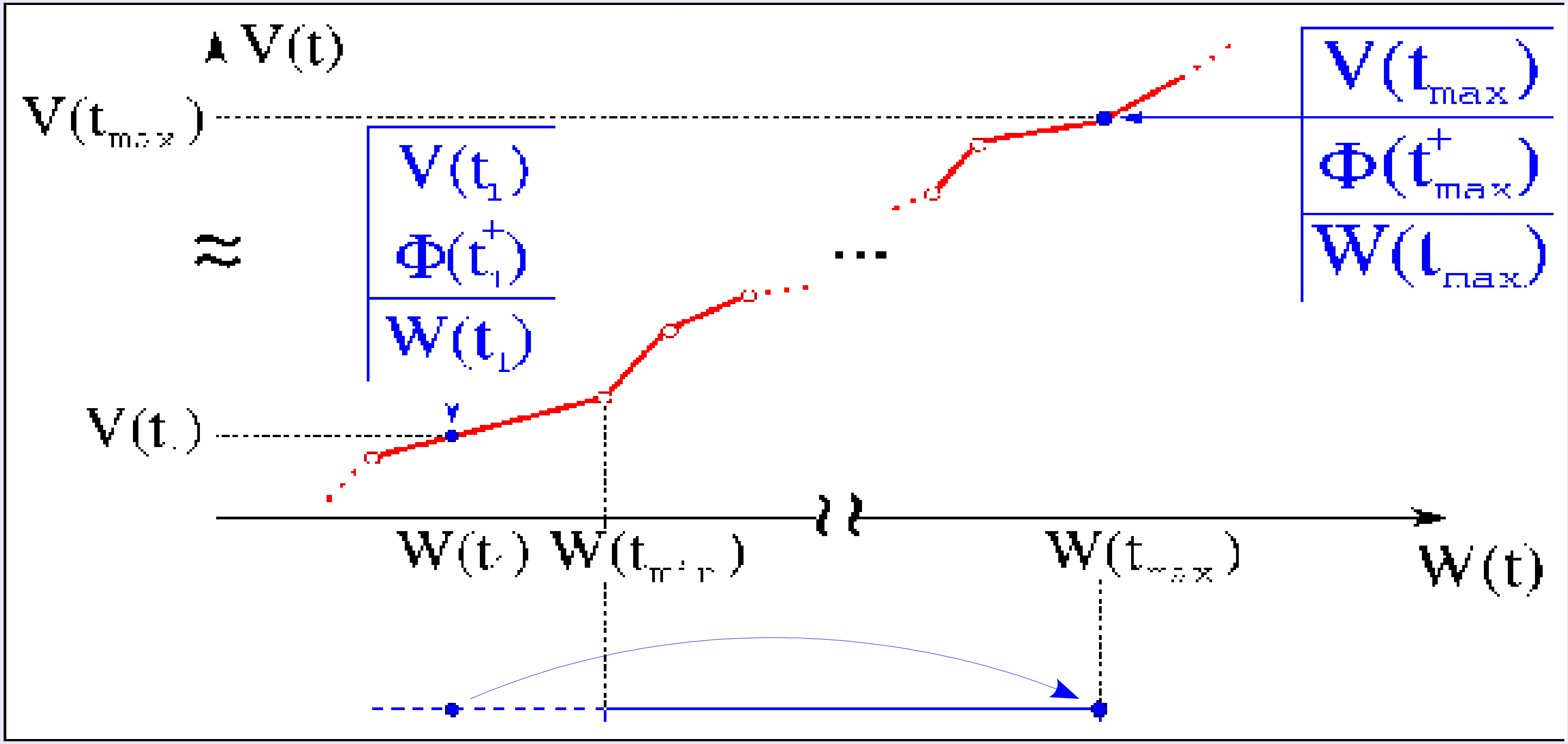
- We call *shape data structure* the union of the base tuple and the U_{tree}

The shape data structure 2/2



Using aggregated information

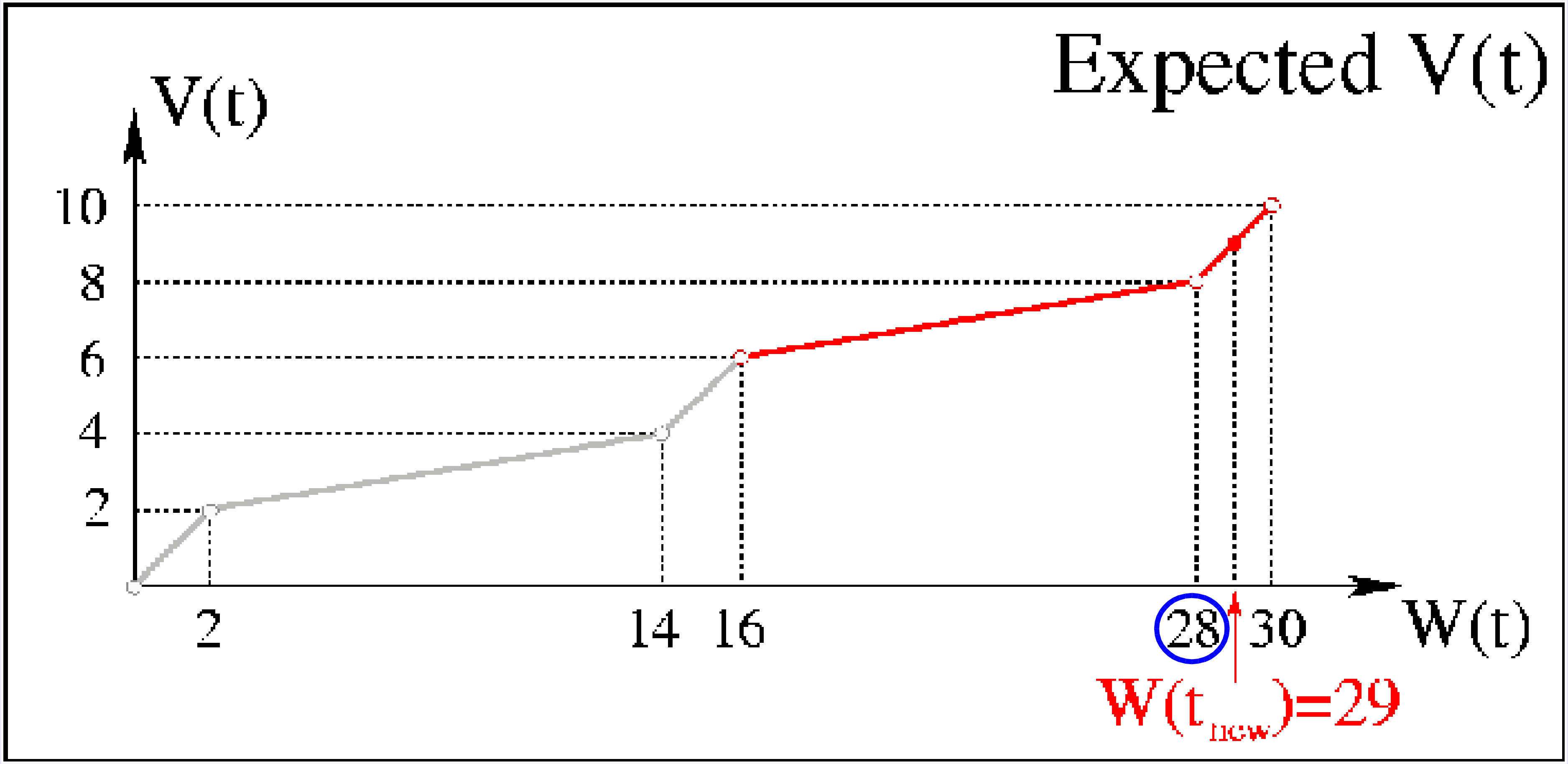
- If the state in t_1 is known, and $\Phi(t)$ does not change during (t_1, t_{min}), the aggregated information in the node allow the state in t_{max} to be computed at $O(1)$ cost



Summary

- L-GPS: simulating GPS at *O(logN)* cost
  - Main Algorithm
    - Shape data structure
    - Computing virtual time
    - Updating the shape data structure
  - Balanced trees

Computing virtual time 1/4



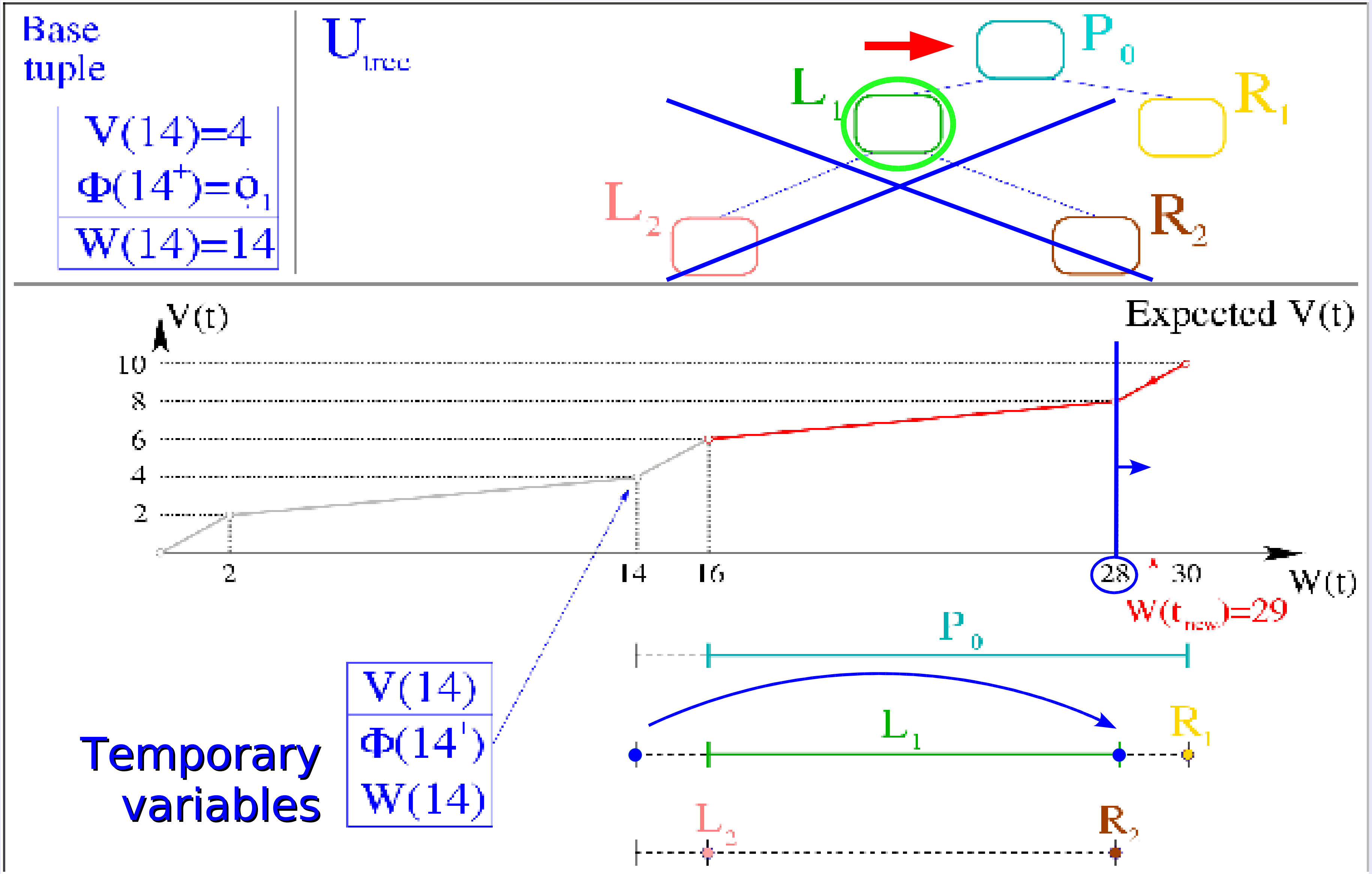
- Defined t_l as the smallest time instant such that $\Phi(t)$ does not change in $(t_l, t_{new}]$ ...

Computing virtual time 2/4

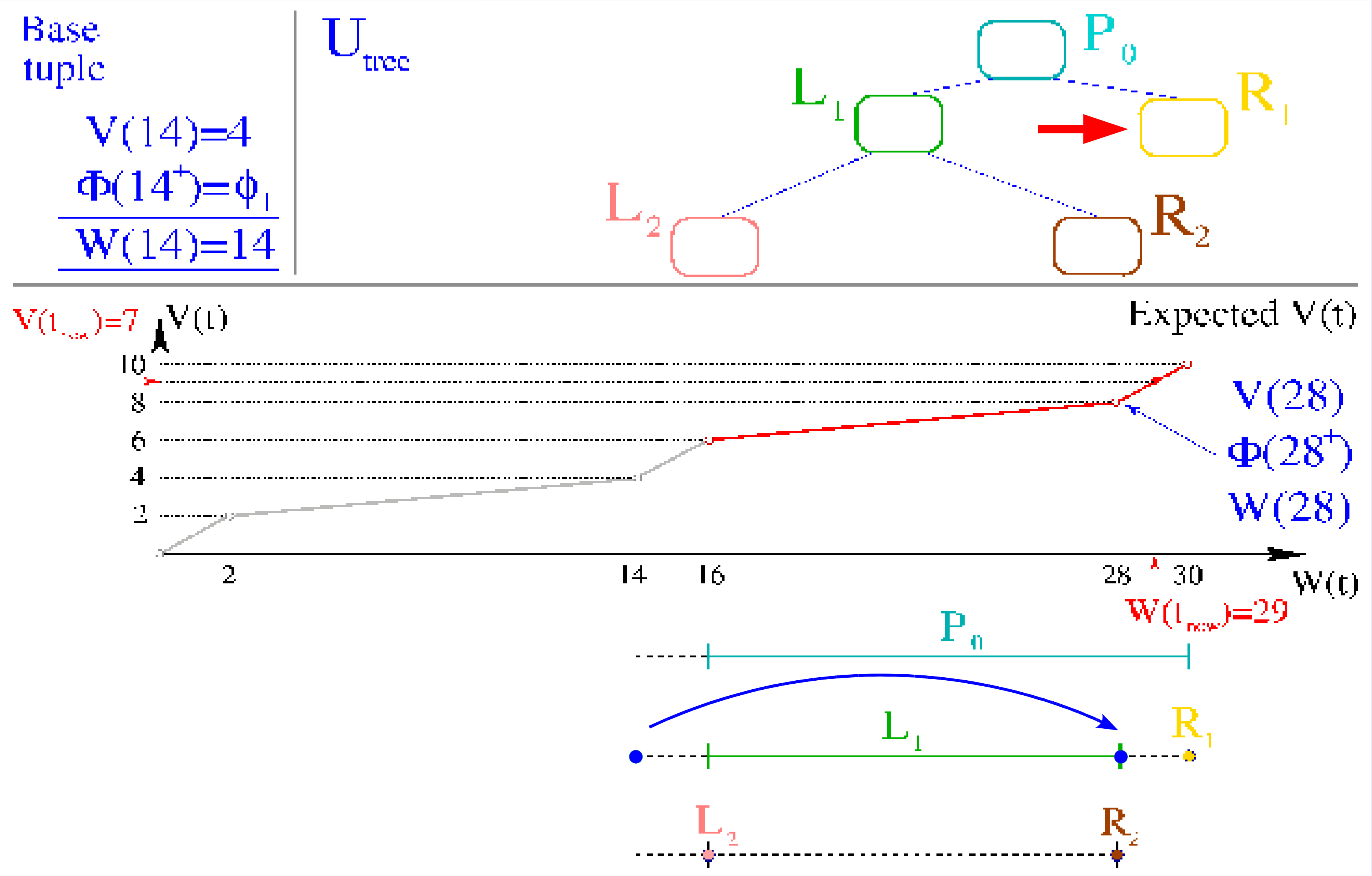
- L-GPS performs a <u>binary search</u> of the leaf representing the time instant t_l,

  and updates <u>three temporary variables</u> at each search step ...

Computing virtual time 3/4



Computing virtual time 4/4

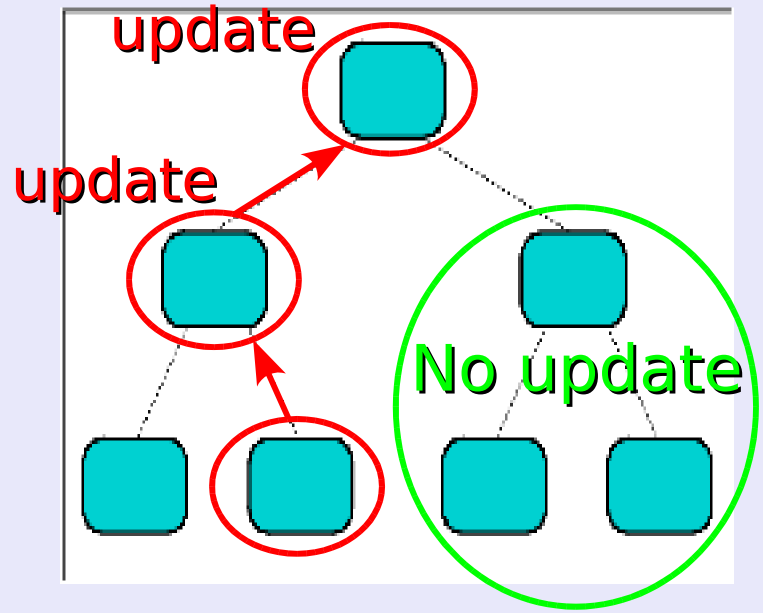


Summary

- L-GPS: simulating GPS at *O(logN)* cost
  - Main Algorithm
    - Computing virtual time
    - Updating the shape data structure
  - Balanced trees

Updating the shape data structure

- It is updated at each packet arrival
  - U_{tree} never contains more than N leaves
  - U_{tree} is balanced, its max depth is $O(\log N)$
- The information stored in each node depend only on its subtree
- Each node is updated at O(1) cost



- The shape data structure is updated at $O(\log N)$ cost on each packet arrival

Summary

- L-GPS: simulating GPS at *O(logN)* cost
  - Main Algorithm
    - Computing virtual time
    - Updating the shape data structure
  - Balanced trees

Balanced Trees 1/2

- The U_{tree} can be implemented by augmenting existing balanced trees
- Patricia Trees
  - *O(logN) average* depth with any practical distribution of the values stored in the tree
  - *O(M) worst-case* depth, where *M* is the number of digits used to represent values
- Red-black Trees
  - *O(logN)* worst-case depth

Balanced Trees 2/2

- Patricia Trees provide a weaker theoretical bound on the depth with respect to Red-black trees ...
- ... but, in practice, Patricia Trees
  - Have a simpler structure
  - Do not need re-balancing after insertions/extractions
  - Allow *entire subtrees* to be removed at *O(1)* cost during the computation of *V(t)*
- We tested the actual performance of Patricia Trees through simulations

Summary

- Background on GPS and WF^2Q
- State of the Art
- L-GPS: simulating GPS at *O(logN)* cost
- L-WF^2Q: implementing WF^2Q at *O(logN)* cost

L-WF^{2}Q

- L-WF^{2}Q uses L-GPS to compute *V(t)* ...
- ... with an additional improvement on L-GPS
- WF^{2}Q meets the Globally Bounded Timestamp (GBT) property, which bounds the maximum va-lue that the virtual time can assume at time t_{new}
- As such, it allows us to know *a priori* if a certain breakpoint will be met or not when $V(t_{new})$ is computed
- L-WF^{2}Q filters the breakpoints to be inserted in the U_{tree}

Conclusions

- The upper bound complexity for simulating a GPS server has been reduced from *O(N)* to *O(logN)*
- The upper bound complexity to provide the minimum deviation from the GPS service has been reduced from *O(N)* to *O(logN)*
    - It has been proven [Xu and Lipton, 2002] that *Ω(logN)* is the lower bound complexity to provide *O(1)* deviation from the GPS service
    - L-WF^2Q achieves the *optimum complexity*
- L-WF^2Q provides an efficient implementation of WF^2Q

Any questions ?

WF²Q+ unfairness/burstiness