

The P-STARA Protocol as an example of load-sensitive routing

Nataliya Skrypnyuk

Supervisor: Nils Kammenhuber

Principles of load-sensitive routing

- flexibly and timely react to unforeseeable events (link breakdown, traffic bursts)
- reduce overprovisioning through more effective use of network resources
- optimize some (or several) metric depending on traffic conditions – thus load-sensitive
- metric examples: packet delays, maximal link loads etc.

Centralized load-sensitive routing

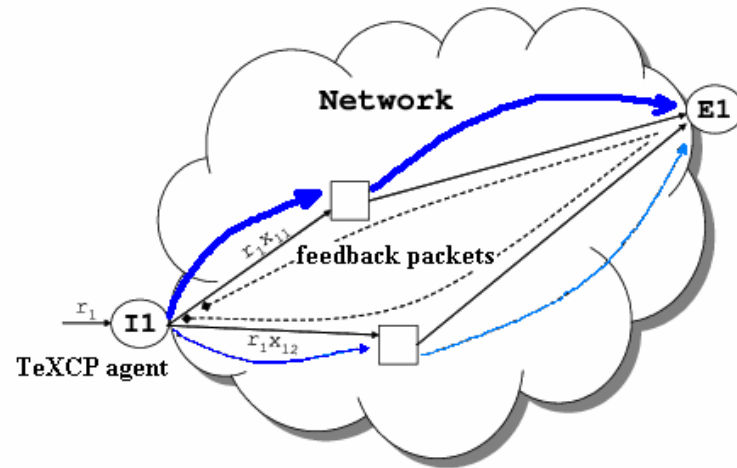
- Centralized load-sensitive routing:
 - + easier to implement
 - one point of failure
 - scalability problems
 - cannot react on short-term scale (needs to summarize information)
- Example :
periodical heuristical OSPF weights recomputation by Fortz and Thorup

Distributed load-sensitive routing

- Distributed algorithms
 - + does not suffer from all the issues with centralized algorithms
 - potential overreaction to changing network conditions
 - potential permanent oscillation effects
- Example: ARPANET
 - firstly delays as metric (disseminated across a network)
later capacity-based for heavy loads
 - route “flapping”

Load-sensitive extensions of OSPF or MPLS

- dynamic OSPF weights or OSPF splitting ratios (in OSPF-AMP through “backpressure” messages)
- traffic balancing between MPLS channels: MATE and TeXCP



- new approaches:
 - swarm intelligence (AntNet)
 - selfish routing

Selfish routing

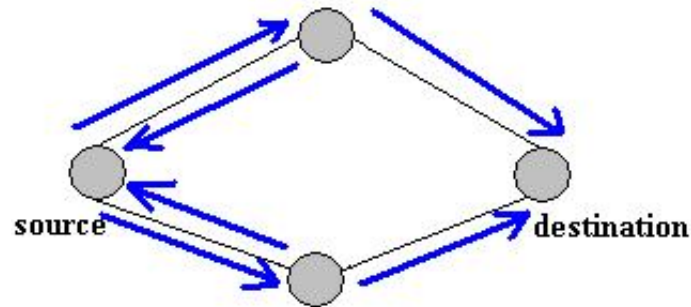
- Requires completely new routing protocol
- Based on game theory:
each packet finds a best route for itself
- Aim: achieve a steady state, i.e. a Wardrop equilibrium in a network
- Under certain realistic conditions, Wardrop equilibrium always exists and is essentially unique

Selfish routing: pro and contra

- “price of anarchy”: difference to the global optimum
- convergence, especially with stale information, is not guaranteed or very slow
- + easier to achieve than global optimum
- + potentially stable as no node has an incentive to change its behaviour
- + no global information distribution
- + “interests” of traffic sources are strongly considered.

Original STARA Algorithm

- Firstly proposed by P. Gupta and P. R. Kumar in 1997
- aimed at the deployment in wireless quasi-static ad-hoc networks
- Uses ALL possible paths from source to destination, explored through “neighbourhood probe packets”



- Core idea: equalize delays (reach Wardrop equilibrium) due to destinations sending back ACK + time stamp
- Probing packets on unused (long delay) paths

Original STARA Algorithm

- Estimate delays on all attached links. Use exponentially sliding moving average, similar to the RTT estimation in TCP
- Use information from neighbour y on its average delay to destination x . Compute $\text{delay_to_x_through_y} = \text{link_delay_to_y} + y\text{'s_average_delay_to_x}$
- Compute the average delay to each destination as $\text{average_delay_to_x} = \sum_y \text{path_to_x_through_y} * \text{probability_to_go_to_x_through_y}$
- Distribute new average delays to all neighbours
- Compute new probabilities for using each path as $\text{new_probability_to_go_to_x_through_y} = \text{old_probability_to_go_to_x_through_y} + (\text{average_delay_to_x} - \text{delay_to_x_through_y}) * \alpha$
- Adjustable learning step α : large enough to be responsive to network dynamics and small enough to be not too sensitive to noise
- Normalize new probabilities

STARA Algorithm drawbacks

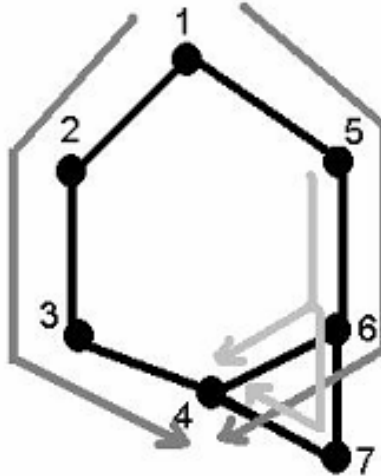
- Routing loops (!!)
- Packet reordering because of probabilistical path choice (!)
- Slow convergence
- Probing overhead
- Slow reaction to traffic changes

=> Result: expect massive service degradation with STARA

P-STARA description

Controlling paths length

- Proposed in 2004 by V. Raghunathan and P. R. Kumar.
- Uses auxiliary hop-distance protocol
- Alternating „odd“ or „even“ packet states (e.g. TTL):
 - „odd“ state: forward through shortest path neighbours
 - „even“ state: use shortest and shortest path+1 neighbours



=> Result: no loops, the length of any path is no more than twice that of shortest path

P-STARA description

Distributed delay estimation

- Periodical link delay measurements
- Periodical (less frequent) recalculation and exchange of
 - average delays to all destinations
 - estimated delays to all destinations through neighbours on POSSIBLE paths
- Difference to STARA:
 - two path delay estimations, for „odd“ and „even“ packets
 - different possible paths for “odd” and “even” packets
 - $\text{odd_delay_to_x_through_y} = \text{link_delay_to_y} + \text{y's_even_average_delay_to_x}$

P-STARA description

Probability based routing tables

- p_j^{ik} =probability_to_go_to_k_through_j on the router i

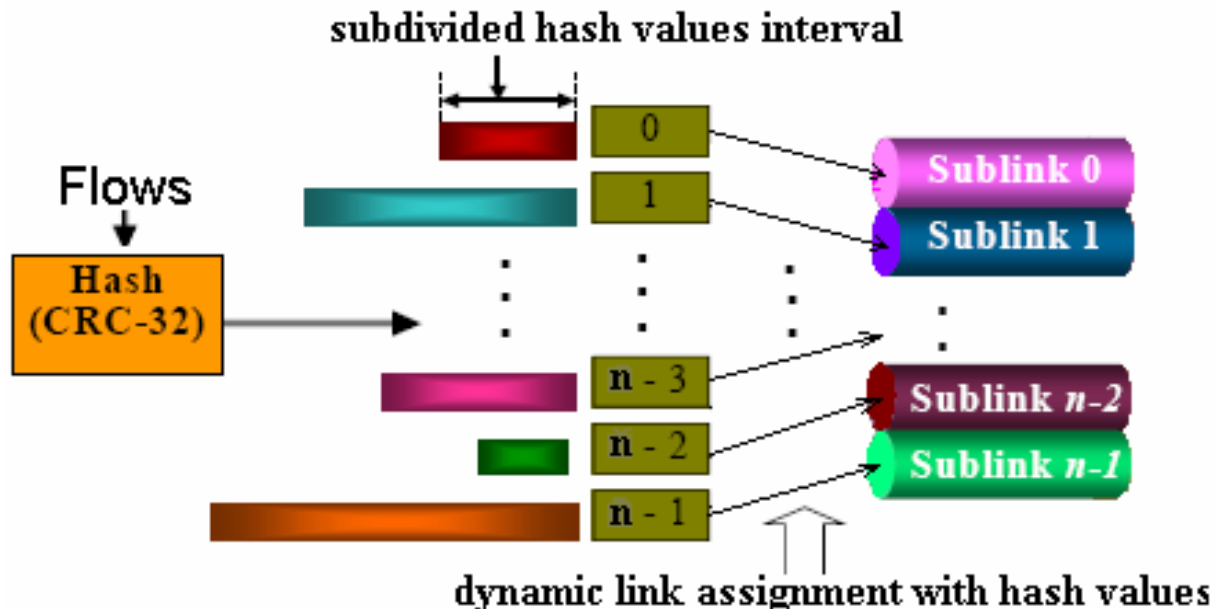
			<i>neigh. j</i>		
⋮					
<i>dest. k</i>			p_j^{ik}		
⋮					

- Two probability routing tables, for „odd“ and „even“ packets
- Probabilities recalculated based on delays, then normalized to 1
- Small “probing” probabilities on unused paths through all (allowed for forwarding) neighbours

P-STARA description

Flow approach

- Randomized path choices for each packet not acceptable because of packets reordering
- Solution: application-level flows
- Flows assignment on available paths:
 - hash value= hash function (some packet header fields) modulo N
 - N is partitioned between the paths according to probability routing tables



P-STARA implementation in ns-2

- Implementation of STARA and its extensions in ns-2 network simulator by V. Raghunathan

Deficiencies of ns-2 implementation:

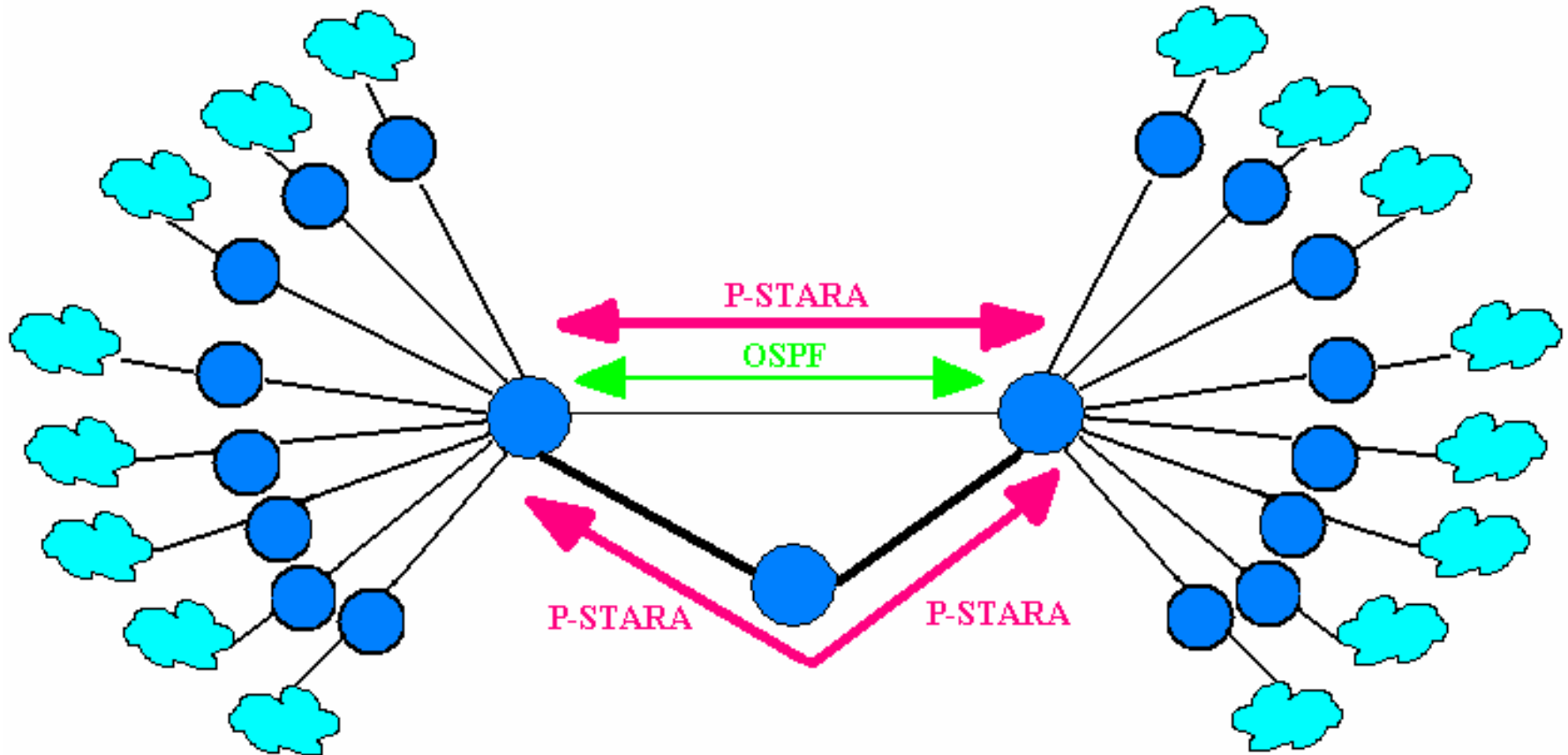
- P-STARA not publicly available
- Implementations for wired networks in test versions, only wireless versions stable
- routing loops (because of periodical reconvergence of hop-count protocol) and packet reordering

⇒ Result: necessity to implement P-STARA anew, in SSFNet

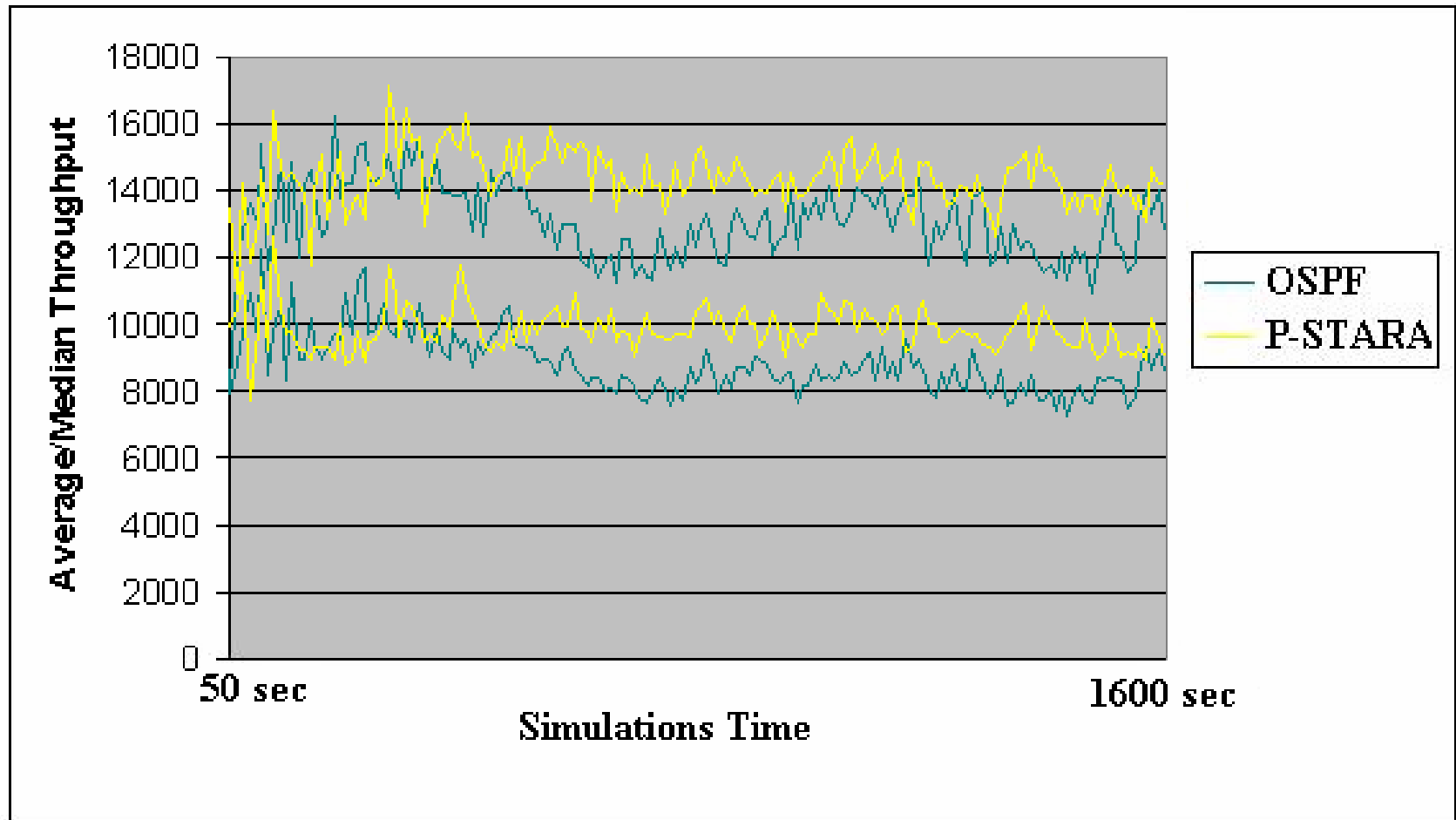
P-STARA implementation in SSFNet

- DroptailQueueMIB provides information on the number of dropped packets exponentially averaged with the tunable parameter
- AlternatingDistanceVectorProtocol plays a role of an auxiliary distance-vector protocol in P-STARA
- WeightedRoutingTable and PstaraRoutingTable are “responsive” for routing respectively packets with odd and even TTLs
- WeightedRoutingTable uses only a subset of paths from PstaraRoutingTable
- OddTTLPstaraAgent distributes traffic on the passes from WeightedRoutingTable, and EvenTTLPstaraAgent on the passes from PstaraRoutingTable through changing path weights in the respective routing tables

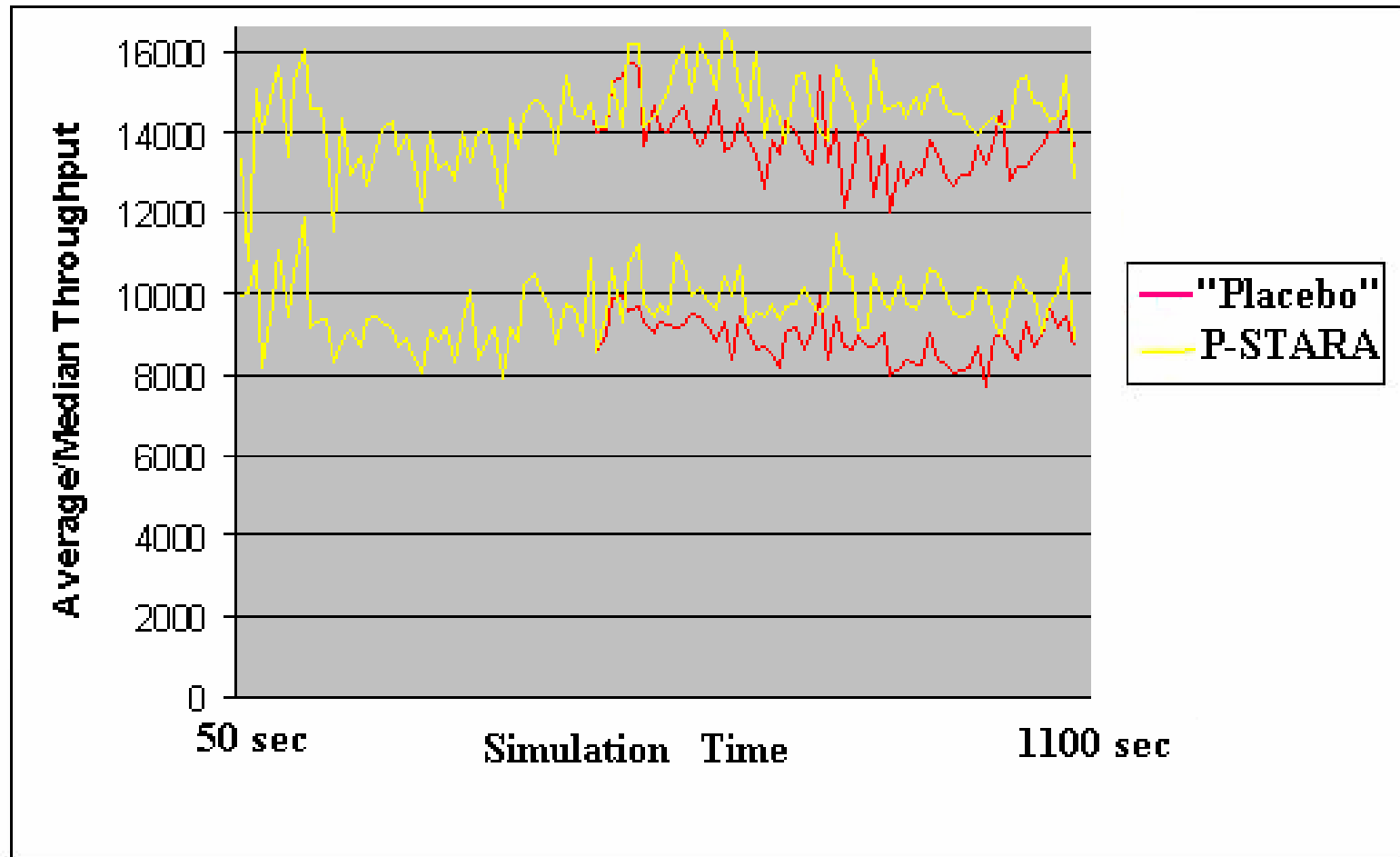
P-STARA and OSPF on a simple topology



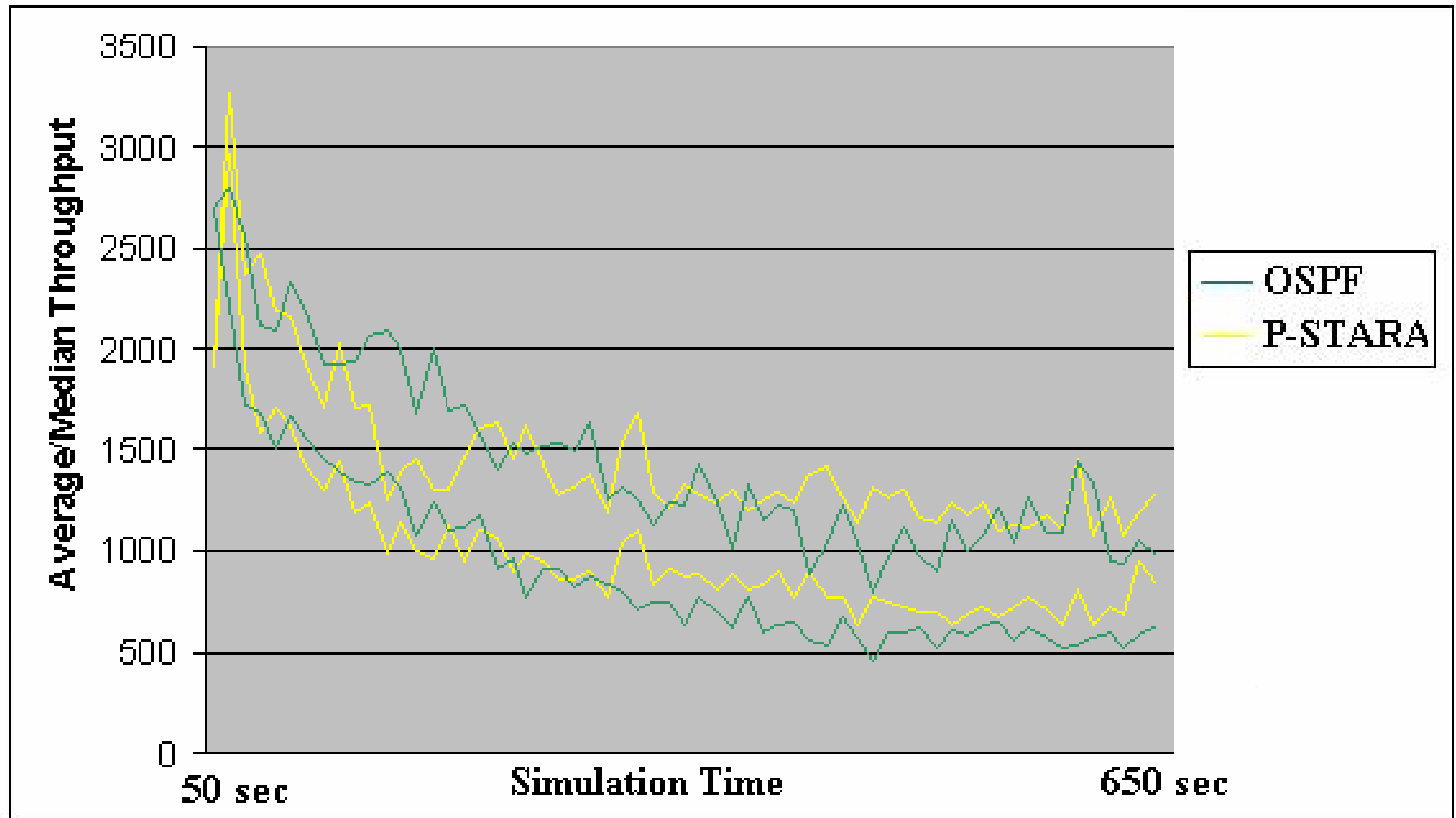
P-STARA and OSPF on a simple topology



P-STARA and "Placebo" on a simple topology



P-STARA and OSPF on a Rocketfuel topology



Conclusions

- Beneficial effect because of more paths in choice and because of traffic shedding out of overloaded links
- No strong oscillations
- For special simple topologies can be strongly beneficial
- For large, well connected topologies – the effect not so pronounced
- In a special case of frequent link failures load-sensitive distribution of traffic can become highly beneficial