Timing is everything: the impact of wakeup schedule distribution on asynchronous power save protocols

Laura Marie Feeney, SICS
Christian Rohner, Uppsala University
Bengt Ahlgren, SICS

SICS Technical Report T2005:15
ISSN 1100-3154
ISRN:SICS-T–2005/15-SE

Abstract

Asynchronous power save protocols have been proposed for use in ad hoc networks. In many protocols, nodes independently follow a common periodic wakeup schedule, each with some unknown offset relative to its neighbors. The schedule is defined to ensure deterministic intervals of overlap between nodes, regardless of the distribution of the nodes' wakeup schedules.

This paper studies the sensitivity of a simple asynchronous power save protocol to the actual distribution of the nodes' wakeup schedules. In practical terms: For given topology and traffic load, are there particularly "good" or "bad" distributions?

We define a simplified model of network operation that allows us to study this question in simulation. The results show that the performance variation has a narrow probability distribution, but with long tails. The variation is shown to derive largely from timing dependencies rather than overall capacity of the system.

The result suggests the feasibility of manipulating the wakeup schedule distribution to improve performance. Although the best wakeup distributions often mitigate the performance penalty imposed by the power save protocol, their relative rarity implies that randomized strategies will not be sufficient to obtain maximum advantage.

1 Background

Ad hoc networks are intended to operate in a self-organizing manner. For the lowest layer communication protocols, this implies that there is no naturally centralized synchronization or scheduling of channel access. At higher layers, nodes must cooperatively forward traffic for each other to maintain network connectivity.

These constraints mean that nodes do not, *a priori*, know when they might receive messages and must be prepared to receive and forward traffic at any time. Unfortunately, listening to the wireless channel consumes significant energy, requiring the development of solutions that allow the network interface to spend as much time as possible in a low energy consumption sleep state.

In the ad hoc environment, nodes therefore must cooperatively buffer traffic for their sleeping neighbors and arrange appropriate rendezvous times to exchange traffic. Each node must determine its wakeup schedule in a decentralized manner, seeking an appropriate tradeoff between duty cycle (wake time) and network performance.

A variety of power save protocols have been proposed for ad hoc networks. Protocols include solutions based on clustering (e.g. [4]). This approach often involves some form of synchronization. A number of asynchronous protocols, both probabilistic (e.g. [10, 3]) and deterministic (e.g. [7, 18, 9]), have also been proposed.

It is this latter category of deterministic, asynchronous protocols that is of interest for this paper. In general, these protocols are based on a common periodic wakeup schedule, which is known to all nodes. The wakeup schedule is defined such that there are deterministic periods of overlap between the wake intervals of each pair of neighbors. A neighbor discovery protocol allows nodes to use these overlapping wake intervals to determine the offset between themselves and their neighbors. The offset information is used schedule data transmission during nodes' shared wake intervals.

Because the nodes operate asynchronously[1], each node follows the common wakeup schedule with an unknown offset relative to the others. We refer to this distribution of offsets as the "wakeup schedule distribution".

The goal of this paper is to investigate the impact of the wakeup schedule distribution on the performance of these asynchronous power save protocols. Here, we study a simplified version of a protocol defined in [9, 7].

This power save protocol is based on the simple observation that, if every node is awake slightly (ϵ) more than half of each period, its awake interval will overlap with that of each of its neighbors, regardless of the distribution of their wakeup schedules. This minimum overlap is guaranteed to include

[1]In practical terms, the clock drift is assumed to be small relative to the length of the period.

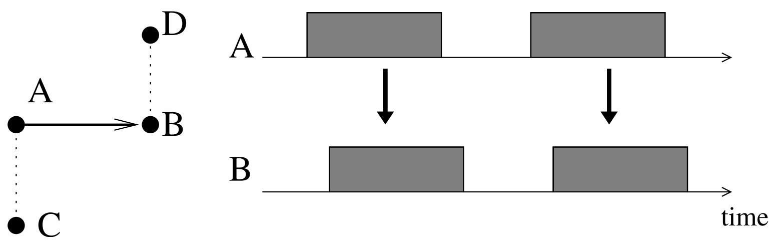


Figure 1: Nodes alternate between wake(dark) and sleep (light) states: the period is identical, but the offset is arbitrary.

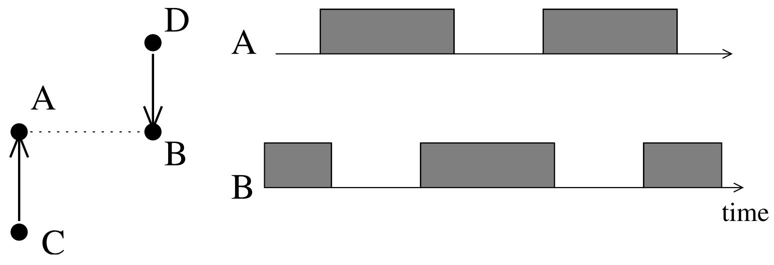


Figure 2: Interfering transmissions benefit from schedules with minimal overlap.

the first and last sub-intervals ϵ of the awake interval, which can be used by the power save protocol for management traffic. The total amount of overlap available for exchanging data traffic between neighbors depends on the distribution of their wakeup intervals.

Intuitively, it seems clear that there will be more and less friendly situations induced by the distribution of wakeup schedules. For example, if nodes A and B exchange traffic (Figure 1), the optimal situation is maximum overlap between their wakeup schedules. But if nodes A and B are receiving traffic from nodes C and D, minimal overlap between their wakeup schedules minimizes contention, as in Figure 2. Similarly, in the case of a flow A-B-D, intra-flow contention and latency are minimized when the wakeup schedules of the three nodes are staggered, as in Figure 3.

This simple example also makes clear the complexity of determining a good wakeup distribution for the case of multiple flows, especially when the situation is further complicated by interference, transmission error and mobility. The problem is roughly analogous to various TDMA scheduling problems, which are difficult both computationally [13] and in practice [8].

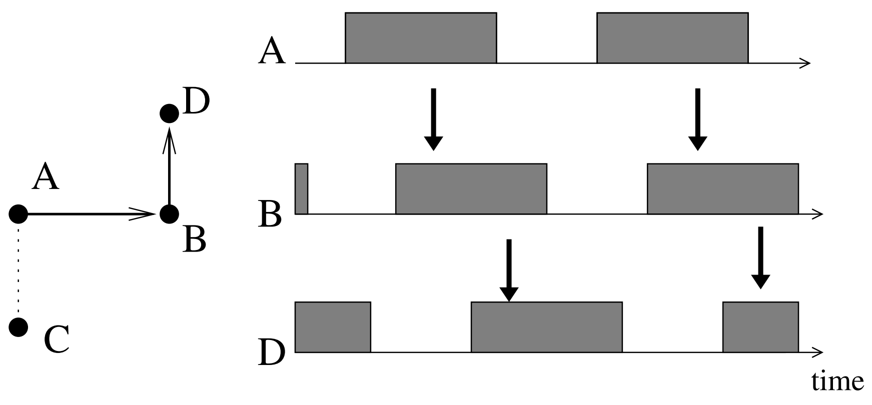


Figure 3: Delay is minimized by a staggered sequence of wakeup times.

2 Contribution

The developers of all the various power save protocols mentioned above have evaluated their performance in simulation. Metrics such as throughput, latency, and delivery ratio are evaluated for a variety of node topologies, traffic loads, and mobility patterns.

To the best of our knowledge, no work has investigated the question of the sensitivity of any asynchronous protocol to the distribution of wakeup schedules. The primary contribution of this paper is to answer this question for one such protocol.

In practical terms: For a given topology and traffic scenario, are there particularly "good" or "bad" distributions of the nodes' wakeup schedules? What is the likelihood of obtaining these distributions? Posed this way, the questions are of more than academic interest. The answer provides insight to not only the internal behavior of these protocols, but also the feasibility of manipulating the wakeup schedule distribution to obtain better energy/performance tradeoff.

Consider a thought-experiment where we measure the 'performance' of a network many times, keeping the topology and traffic load constant, and varying only the (random) distribution of the nodes wakeup schedules. The performance measurements will define some probability distribution.

The shape of this distribution suggests answers to the questions above. A very narrow distribution implies that the network performance is not too much affected by the wakeup schedule distribution: What is good for some flows is bad for others and it is difficult to obtain some overall advantage. If the performance distribution has a long tail, this implies that there do exist "good" distributions, but that they may be hard to realize. Conversely, a flatter performance distribution means that randomized changes of wakeup schedule are likely to noticeably affect the network performance.

In this paper, we define a simplified model of network operation that allows us to perform the above experiment. Our chosen performance metric is the flow capacity (number of feasible CBR flows) of the network. The feasibility of each flow depends on the overlap between the awake intervals of each transmitter-receiver pair in the flow and on the interfering transmissions.

The results show that the flow capacity is more or less normally distributed. The parameters of the distribution are fairly narrow, with long tails. The median 50% of measurements account for about a variation of only 5% about the median flow capacity (about 20 the total variation), while the extremes vary 20-30% about the median. In more than 70% of topologies, the best wakeup distributions obtain over 70% of the flow capacity obtained without the power save protocol operating. These results hold quite broadly over a range of network scenarios.

Examining the various internal behaviors of the system confirms that the wakeup schedule, rather than underlying wireless capacity constraints, is significant in determining the variation.

To the best of our knowledge, there are no power save mechanisms that manipulate the distribution of the wakeup schedule to improve performance or adapt to changing load. Some existing protocols simply increase the nodes' duty cycle in response to traffic or estimated neighbor density (e.g. [18]). Our observations suggest it may be possible to improve protocol performance by developing techniques for managing the wakeup schedule distribution, without needing to increase the duty cycle. The development of specific techniques, we leave for future work.

3 Network Model

Most popular simulation environments for evaluating wireless protocols (.e.g *ns-2*[1]) are discrete event simulations that model network operation in some detail, including wireless communication, IEEE 802.11 MAC and various higher layer protocols. With these tools, metrics such as throughput and latency can be effectively evaluated for a large number of topologies and traffic and mobility scenarios.

The simpler model used in this paper is intended to highlight the impact of wakeup schedule distribution, as well as to make it feasible to simulate many wakeup schedule distributions for each of a statistically large number of topologies.

We avoid the complexity of modeling the time evolution of the system. The model addresses the only "steady state" operation of the network over a single period of the wakeup schedule. Transient effects and those exhibiting random variation in time are either excluded or approximated as predictable. As a result, the MAC layer is assumed to provide consistently scheduled access, avoiding highly variable backoff situations. In addition, operational and background traffic such as the routing protocol are not modeled.

Our chosen performance metric is flow capacity. All flows are assumed to be periodic (CBR) flows, with a period equal to that of the power save protocol. Because time varying elements are ignored, the feasibility of a flow can be determined by calculating whether the relevant transmissions are feasible in a single period of the power save protocol.

This approach is not intended to suggest that such precise resource scheduling is plausible in CSMA/CA-based ad hoc networks. However, the model roughly corresponds to a scenario in which some user applications initiate CBR flows. A subset of users obtain the specified throughput, while the remaining flows are excluded from the network. (Examples of admission control strategies for "soft" QoS in IEEE 802.11-based ad hoc networks include [2, 5].) This assumption of soft admission control also implies that the network is less likely to be in a highly congested state that would result in large backoffs.

Clearly, this simplified model is not suitable for any absolute estimate of performance. But the model has the advantage of being independent of any particular technology choices, thus highlighting the structural impact of distribution of wakeup schedules on network performance. In practical terms, the model also allows us to simulate a large number of wakeup schedule distributions for each of a large number of topologies in a more reasonable compute time.

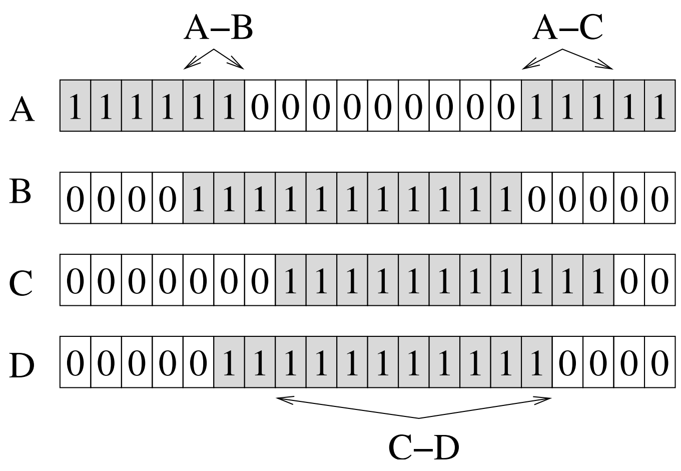
3.1 Model description

Timing model Because the wakeup schedule and traffic are periodic and no transient effects or timing variability are introduced by the lower layers, it is only necessary to model a single period of the wakeup schedule.

The period is divided into *resolution* time slots and all events are defined in terms of slots. The *resolution* determines the granularity of the simulation; it is not intended to imply slotted or synchronous operation of the MAC layer. (Indeed, slot synchronization in TDMA-based ad hoc networks is a well-known problem [15]).

Power save protocol The power save protocol used in this paper is a simple one in which timing overlap between nodes is guaranteed by a wakeup schedule that requires nodes to be awake for slightly more than half the time. If nodes are awake over half of the time $(0.5 + \epsilon)$, any two nodes will both for either the first or last ϵ portion of their wakeup interval. Nodes use this interval to broadcast neighbor discovery or HELLO messages to determine neighbor offsets . This structure is used in power save mechanisms described in [7, 9].

The distribution of wakeup schedules is assumed to be uniformly, randomly distributed over the period. This assumption is reasonable, given


A–B
A–C
A 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1
B 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
C 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0
D 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
C–D
Figure 4: Representation of the wakeup schedule of four nodes (*resolution* = 20, duty cycle = 55%).

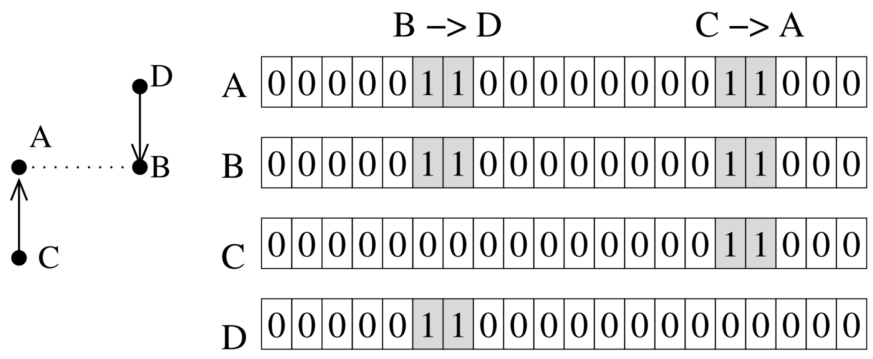
that the wakeup schedule is usually on the order of 100's ms, while the initialization and deployment of nodes takes place over several seconds or minutes, possibly even days.

Communication model Nodes are deployed on a rectangular field, with x- and y-coordinates uniformly distributed random variables over the size of the field. Nodes are stationary.

Each node has a circular transmit radius *xmit* and interference radius $intf \geq xmit$. Nodes are assumed to communicate without error to any node within their transmit radius. Nodes within the interference radius will sense the channel as busy.

The MAC protocol is assumed to operate without error to schedule channel access so as to prevent conflicting transmissions. For each node, the state of the channel is represented as a sequence of *resolution* values indicating the channel state in that slot (we emphasize again that the MAC itself is not slotted). A node can transmit in a slot if the channel is free at both the transmitter and receiver. Each transmission occupies the channel at all nodes within interference range of the transmitter or within communication range of the receiver. This rule roughly models the RTS/CTS operation of IEEE 802.11. The time that the transmission occupies the channel is intended to reflect the complete channel access process, with an implicit assumption that the variation in channel access time (e.g. random backoff) is small relative to the allocated time.

A transmission is feasible if the transmitter and receiver are both awake and have a free channel for a sequence of contiguous time slots corresponding to the packet transmission time. The transmission is assigned to the first set of slots that meets these criteria. This assignment is fixed and is not



B -> D
C -> A
A
B
C
D
A 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0
B 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0
C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
D 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 5: Representation of the channel occupancy for transmissions $C \rightarrow A$ and $D \rightarrow B$. Transmissions from C are assumed not to result in interference at D and vice verse.

adapted to the optimize the channel utilization. (Such optimization would be extremely complex in practice: The analogous optimal multi-hop TDMA slot assignment is problem is hard computationally and in practice [8].)

Because the model is based on analysis of a single period, it implicitly assumes that the MAC protocol will obtain consistent scheduling. That is, node s is assumed to transmit to node r at the same time in each period. This assumption is true in many TDMA MACs. In CSMA/CA MACs, the channel access time is inherently variable. In effect, we assume that the backoff time is small and its variation is fairly small relative to the total channel access time.

In addition to its computational efficiency, one advantage of the model is that it is straightforward to deal with hypothetical questions about channel availability (e.g. What proportion of the time is the channel still available at some node?).

Traffic model All flows are assumed to be CBR flows with the same period as the power save protocol. No other traffic (e.g. overhead) is modeled.

A flow is feasible if each transmission along its route is feasible. That is, the source and each forwarding node must be able to transmit once each each period, so that one packet enters and one packet leaves the network in each period. The model does not distinguish between a packet that is forwarded in the same period it was received and one that is not forwarded until the following period, so latency is not considered.

The operation of the routing protocol is not included in the model. Routes are fixed, shortest path routes, computed once for each topology. More significantly, the routing is not adaptive, so it is not possible for flows to dynamically“discover” a longer, but still feasible route[2]. Such adapta-

[2]Some throughput models [11], as well as many proposed routing protocols, indicate

large square	nodes	connected pairs (%)	mean path length
total area = 1.0 6.3 x 6.3 hops	50 75 100 125	39% (11-92) 79% (27-100) 94% (49-100) 98% (78-100)	3.9 (1.2) 5.5 (1.1) 5.3 (0.6) 5.0 (0.3)
small square	nodes	connected pairs (%)	mean path length
total area = 1.0 2.6 x 2.6 hops	15 20 30 40	89% (19-100) 96% (47-100) 99% (63-100) 99% (81-100)	2.1 (0.4) 2.1 (0.3) 2.0 (0.2) 2.0 (0.1)
large rectangle	nodes	connected pairs (%)	mean path length
total area = 1.0 3.2 x 13 hops	75 100 125 150	58% (21-100) 86% (33-100) 95% (39-100) 99% (45-100)	5.2 (1.6) 6.3 (1.2) 6.5 (0.8) 6.3 (0.4)
small rectangle	nodes	connected pairs (%)	mean path length
total area = 1.0 1.3 x 5.2 hops	20 35 50 65	82% (31-100) 98% (48-100) 99% (49-100) 100% (100)	2.4 (0.6) 2.6 (0.3) 2.6 (0.2) 2.5 (0.1)

Table 1: Connectivity (min-max) and mean path length (68% confidence). Average over 100 scenarios.

tion is not, however, directly supported in standards-oriented protocols (e.g. [12]).

3.2 Performance metric

The performance metric used in these experiments is CBR flow capacity. The flow capacity is the number of feasible flows in a set of candidate flows. The candidate flows are an ordered set of randomly chosen connected source-destination pairs.

Flows in the candidate set are evaluated one at a time. If a flow is feasible, it occupies the relevant slots. If a flow is not feasible, no transmissions associated with the flow occupy the channel and the next flow in the candidate set is evaluated. In practical terms, the candidate set reflects the offered load (user application flows), some of which are blocked by an admission control mechanism as infeasible.

that adaptive routing can be an effective strategy.

The candidate set is used to determine the "baseline" flow capacity of the network [3]. The baseline capacity is the flow capacity of the network when the power save protocol is not applied. Because the nodes are always wake, there is no wakeup schedule distribution.

If the baseline set is smaller than the original candidate set, we say that the network is channel-limited, because the channel constraint alone makes some of the candidate flows infeasible. Again, we note that channel-limited does not mean that no additional flows can be admitted to the network. Additional flows might succeed using different routings, or ordering the flows differently, or enlarging the candidate set. The experimental scenarios are, however, defined such that the baseline scenario exhibits mild congestion.

3.3 Implementation note

The model is computationally simple because connectivity, interference, and channel state are represented as binary or integer matrices. Testing and setting the channel occupancy can be done using simple matrix operations. The simulation is implemented in Matlab and the authors expect to be able to make the code available to interested researchers.

4 Simulation Experiments

In this section, we complete the model outlined above by defining the simulation parameters used in the experiments. To give a sense of the raw data generated by the experiments, we explore the output from one configuration in some detail. Finally, we present high-level results integrated over multiple experimental runs.

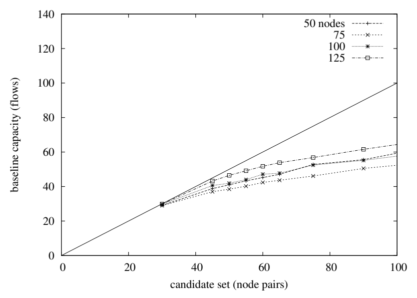
The simulation experiment itself closely resembles the thought experiment outlined in section 2. We randomly generate a topology and set of candidate flows, and evaluate the feasibility of each candidate flow, in the absence of a power save protocol. We then assign wakeup schedules to each node such that the offset between the nodes' schedules is randomly uniformly distributed over the period. The feasibility of each flow in the baseline set (i.e. each flow that is known to be feasible when the power save protocol is not used) is evaluated

The process is repeated for a large number of wakeup distributions to obtain the probability distribution for the variation in network performance.

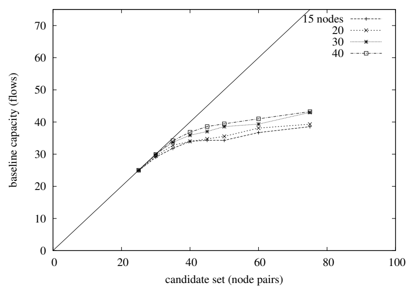
4.1 Experimental parameters

Timing parameters All of the experiments use a *resolution* of 500. For a wakeup schedule period of 100ms, each slot corresponds to 200μs.

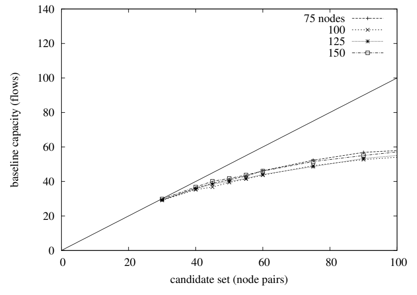
[3]Note that this baseline set is not necessarily the largest feasible subset of candidate flows due to the ordering requirement.



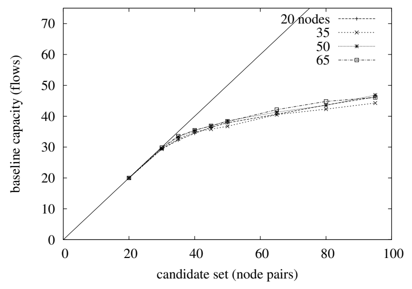
large square (candidate set = 50)



small square (candidate set = 40)



large rectangle (candidate set = 45)



small rectangle (candidate set = 40)

Figure 6: Choosing the candidate set size such that the network is loaded, but not significantly overloaded.

The power save protocol uses 55% duty cycle, so that each node is awake just over half of the time, as required to obtain the predictable overlap described above. In our example, this would correspond to a wakeup schedule of 55ms on and 45ms off.

Scenario parameters We consider four deployment scenarios: small square; large square; small rectangle; and large rectangle, each with a variety of node densities. As shown in table 1, the scenarios represent moderately- to fully-connected networks, with topologies ranging from compact to nearly linear.

Communication parameters The transmission range is defined as in the scenarios above. The interference range is assumed to be 40% larger than the transmission range. The packet transmission time is 2.2 ms This corresponds roughly to the total transmission time for a short (ca 137-byte) IEEE 802.11b frame transmitted at 11Mbps or a longer frame transmitted on a newer IEEE 802.11 interface.

Since each source is assumed to transmit once in each period, a wakeup schedule period of 100ms implies CBR flows with a rate of 10 packets per second.

Traffic parameters As mentioned above, the choice of candidate set size is fairly complex.

We use the candidate set to determine the baseline capacity of the network when there is no power save protocol and thus no wakeup schedule distribution. Flows in the candidate set are evaluated one at a time, until no more flow can be added to the channel or there are no more flows in the candidate set.

The baseline set of flows fills the channel to capacity, in the sense that no other flows from the candidate set can be admitted to the network. However, the actual total channel utilization depends on the size of baseline – and thus on the size of the candidate – set.

If the candidate set is small, all of the flows may be included in the baseline set, leaving the network well under-capacity (and unlikely to exhibit any interesting variation). If candidate set is very large, then the baseline set will reflect a network that is highly congested because we search exhaustively for potentially feasible flows. However, such candidate set implies that the offered load contains many infeasible flows; the system is, in effect, hunting for users that it can satisfy. In reality, this would imply a network that is significantly under-dimensioned and basically dysfunctional, since most user flows would be rejected.

We would therefore like the size of the candidate set to reflect an offered load that is realistic in the context of the scenario and to result in a baseline set in which the network is likely to be near or slightly above capacity. We defined such a candidate set size experimentally.

In the experiment, we simply compute the baseline (no power save protocol) capacity for a variety of candidate set sizes. At first, the size of the baseline set increases with the size of the candidate set. As the size of the candidate set increases, the network becomes congested and some flows are infeasible, so the size of the baseline set begins to increase more slowly than the size of the candidate set. Eventually, it becomes very unlikely that an additional candidate flow is feasible and the size of the baseline set approaches the capacity of the network.

Figure 6 show the experimental results for each scenario, showing the baseline flow averaged over 100 topologies. Since we are interested in mildly congested conditions, we choose the candidate set size to be slightly above the point at which the baseline capacity no longer increases. Note that there is little dependency on the node density, because the wireless transmissions affect all of the nodes in a fixed area, regardless of their number. (The similarity between the large and small scenarios is accidental - the latter has less area, but much shorter path-length flows.)

The values chosen for the simulation are also shown in table 6. For most of these configurations, the size of the candidate set implies between 0.25 and 2 flows per node (or user), which is a reasonable magnitude of offered load.

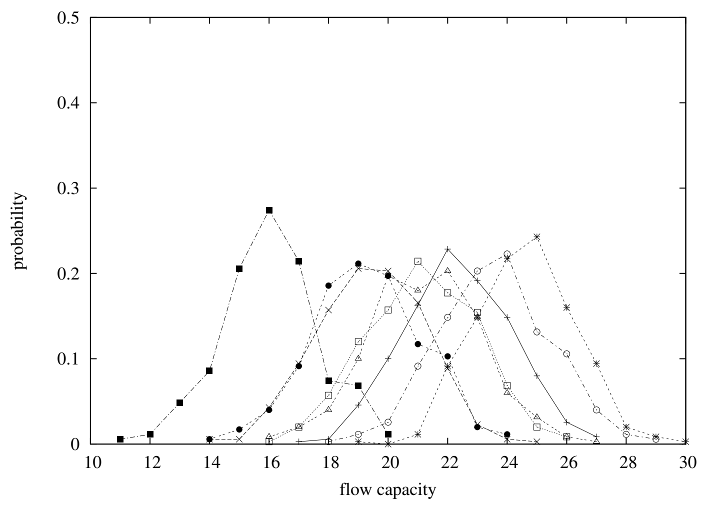


Figure 7: Raw probability distribution (large rectangle, 100 nodes)

Note that these values are averages and there can be considerable variation between topologies generated with the same configuration parameters. Therefore the actually conditions associated with the baseline capacity for each topology will vary somewhat.

4.2 Sampling the data

For each of the scenarios described above, we generate 50 random topologies and for each topology, we evaluate the flow capacity for 350 wakeup schedule distributions. The flow capacity values exhibit some probability distribution, as in figure 7. Each curve represents the variation in flow capacity observed in a given topology. (Although the full experiment included 50 topologies, for clarity only eight (randomly selected) are shown.)

The result appears to have a roughly symmetric normal distribution. This is not surprising, given that the underlying random variable – namely, the overlap between nodes' wakeup schedules – is (almost) uniformly distributed. Note the considerable variation in absolute flow capacity among the various topologies. This variation persists, even when each result is normalized to its baseline topology, as in figure 8.

Plotting the quartile distribution loses some of the detail present in the full distribution, but makes it possible to present all 50 topologies in a single plot, as in figure 9. In this configuration, nearly all of the topologies are capacity limited ($baseline < 45$). By inspection, we note that the two middle quartiles account for only a small part (about one-fifth) of the total variation. In the capacity-limited case, the best schedule distributions generally obtain about 75% of baseline capacity.

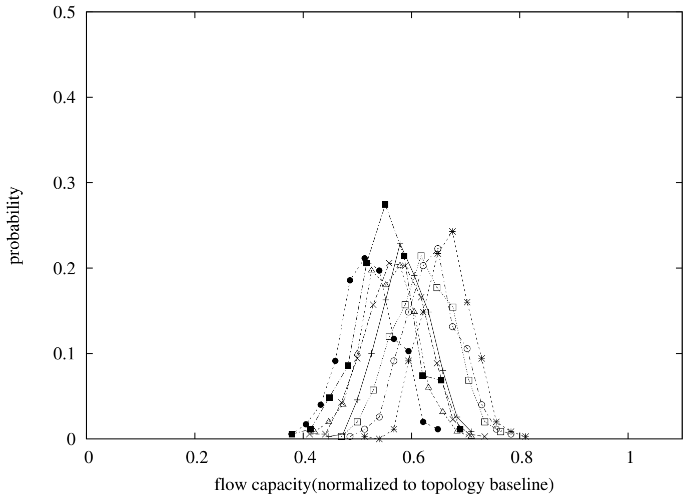


Figure 8: Normalized probability distribution(large rectangle, 100 nodes)

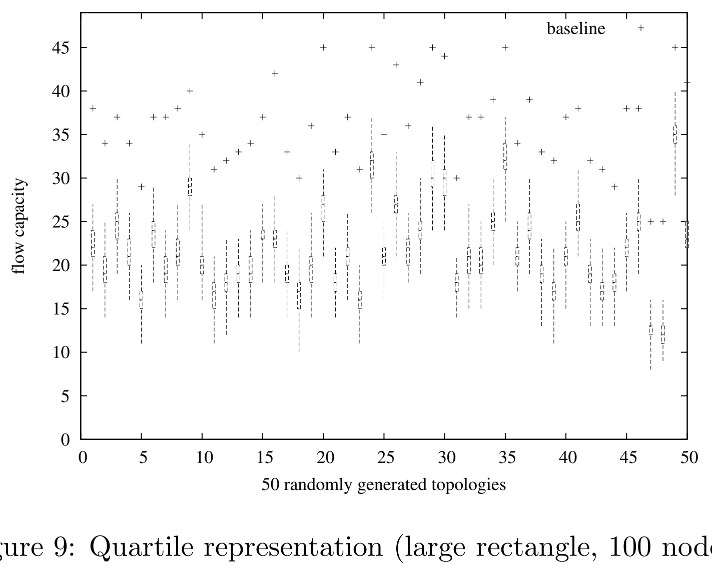


Figure 9: Quartile representation (large rectangle, 100 nodes)

nodes	median	inner quartile		min - max	
small square					
15	20.2 (3.6)	-0.06 (0.02)	+0.07 (0.02)	-0.31 (0.05)	+0.25 (0.05)
20	20.8 (2.7)	-0.06 (0.02)	+0.06 (0.02)	-0.28 (0.04)	+0.23 (0.04)
30	23.3 (2.7)	-0.05 (0.02)	+0.05 (0.02)	-0.25 (0.05)	+0.21 (0.04)
40	24.7 (2.8)	-0.05 (0.02)	+0.05 (0.01)	-0.22 (0.04)	+0.19 (0.03)
small rectangle					
20	22.3 (5.7)	-0.06 (0.02)	+0.06 (0.02)	-0.27 (0.07)	+0.22 (0.05)
50	22.7 (3.2)	-0.05 (0.02)	+0.05 (0.02)	-0.22 (0.05)	+0.19 (0.05)
35	22.1 (3.6)	-0.06 (0.02)	+0.05 (0.02)	-0.25 (0.05)	+0.21 (0.04)
65	24.1 (3.2)	-0.04 (0.01)	+0.04 (0.01)	-0.21 (0.04)	+0.17 (0.03)
large rectangle					
50	24.2 (7.3)	-0.05 (0.02)	+0.06 (0.02)	-0.24 (0.05)	+0.24 (0.07)
75	23.4 (5.1)	-0.05 (0.02)	+0.06 (0.02)	-0.25 (0.05)	+0.23 (0.05)
100	23.6 (3.2)	-0.06 (0.02)	+0.05 (0.02)	-0.23 (0.04)	+0.23 (0.05)
125	26.6 (2.7)	-0.05 (0.02)	+0.05 (0.02)	-0.22 (0.03)	+0.20 (0.04)
large rectangle					
75	22.5 (6.2)	-0.06 (0.02)	+0.06 (0.02)	-0.25 (0.06)	+0.24 (0.07)
100	21.7 (4.8)	-0.06 (0.02)	+0.06 (0.02)	-0.25 (0.05)	+0.23 (0.05)
125	21.3 (3.4)	-0.06 (0.02)	+0.06 (0.02)	-0.25 (0.05)	+0.23 (0.05)
150	22.4 (3.1)	-0.05 (0.02)	+0.05 (0.02)	-0.24 (0.04)	+0.21 (0.04)

Table 2: Quartiles: Median flow capacity: mean of the distance of each quartile from the median, relative to the median (computed for each topology). Each mean is computed over 50 topologies, stddev included in parentheses.

4.3 Summarizing the data

The previous section provides a sense of the raw data accumulated in the simulation experiments. The large number of configurations makes it impractical to present data for all the configurations and would, in any case, obscure the more fundamental results in a mass of detail. Therefore, we first present the combined quartile data for each scenario, then we compare the baseline performance with that obtained by the "best" wakeup schedule distribution.

Quartile analysis We use a simple relative quartile to integrate the per-topology quartile data (as in figure 9). For each topology, we compute the distance from the median flow capacity to each quartile, computed relative to the median (for that topology) to eliminate the inherent variation in capacity between topologies. This relative inter-quartile distance is averaged over the 50 topologies simulated in each scenario. The standard deviation is also given. (Note that the stddev reflects the variation between topologies and not the wakeup schedule dependent variation within a topology.)

Table 2 shows the inter-quartile distances for each scenario. The median values themselves are also given, allowing us to translate this variation into absolute terms. We see that the inner quartiles lie within a range about $\pm 5\%$ of the median. In absolute terms, this means that 50% of wakeup schedule distributions will be in a tight range of only one flow above or below the median flow capacity of twenty-some flows. The max-min excursions are much larger: the “best” wakeup schedule distributions support some 20-25% more flows than the median. This represents a further four or five flows beyond the central cluster – a level of improvement that seems worth striving for. The “worst” wakeup schedule distributions exhibit a slightly larger excusrion, suggesting that it is especially worthwhile to escape such distributions.

Maximum capacity The results above address the relative distribution of flow capacities and show that the best wakeup schedules achieve significantly higher capacities than the median. A further practical question concerns the difference between the capacity of the “best” wakeup schedule distribution and the capacity with no power save protocol (i.e. no wakeup schedule).

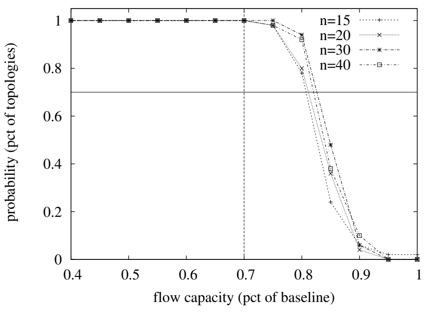
Figure 10 shows, for each scenario, the proportion of topologies (y-axis) in which the best wakeup schedule distribution obtains a flow capacity that exceeds some percentage (x-axis) of the baseline capacity for that topology.

Observe that for all topologies, the ratio between the best flow capacity and baseline capacity is significantly larger than the 55% duty cycle. In even the most heavily loaded configurations, over half of the topologies obtain a maximum flow capacity of at least 75% of the baseline capacity, while over 70% obtain at least 70% of the baseline capacity obtained in the absence of the power save protocol..

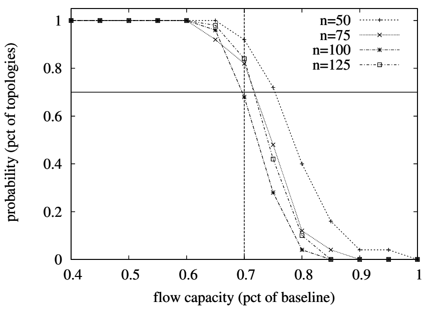
5 Discussion

In the section above, we showed that there is considerable variation in flow capacity, but we did not show that the variation in wakeup distribution is substantively responsible. One possible explanation for our result is that variation in the number of feasible flows simply reflects variation in the path-length of the admitted flows (recall that the order in which of candidate flows are evaluated is the same for all wakeup schedule distributions).

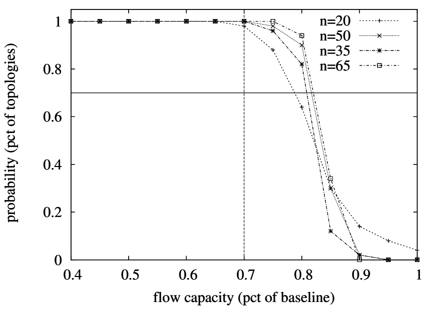
We call this model the total “total transmissions hypothesis”. If a long path-length flow happens to be feasible in some wakeup schedule distribution, the flow capacity only increases by one, even though many transmissions are required. Conversely, if the long path-length flow is not feasible, then two (or more) short path-length flows may be able to use its transmission slots, increasing the flow capacity by two (or more).



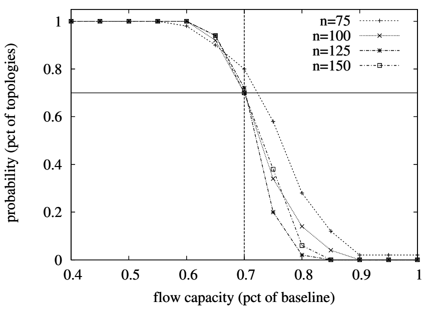
small square



large square



small rectangle



large rectangle

Figure 10: Maximum flow capacity (relative to baseline)

Consider the trivial example of a flow A-B-C-D-E. If this flow is feasible, the flow capacity increases by one. But obviously, the flows A-B, B-C, etc are also feasible and would increase the flow capacity by four.

Alternatively, consider the example of a set of candidate flows of path lengths 2, 1, 1, 4, 2, 1, 1 .... Assume that the first three flows (with a total of four transmissions) are feasible in all wakeup distributions, but in some wakeup schedule distributions the 4-hop flow is feasible and in others it is not. In the cases where the flow is feasible, there will be four flows with eight total transmissions in each period, so that there are no transmission slots for the remaining flows. In the case where the flow is rejected, there will be three flows, but there are only four total transmissions in the network. This makes it more likely that the following three flows (with a total of four transmissions) will be feasible, resulting in six flows, again with eight total transmissions.

The model of flows of varying path length being introduced into the network in some arbitrary order is perfectly reasonable. But, if this hypothesis were true, then there would be no reason to try to adapt the wakeup schedule distribution to increase flow capacity - it would simply tune the network to favor short path-length flows.

If this hypothesis is true, we expect that the total number of transmis-

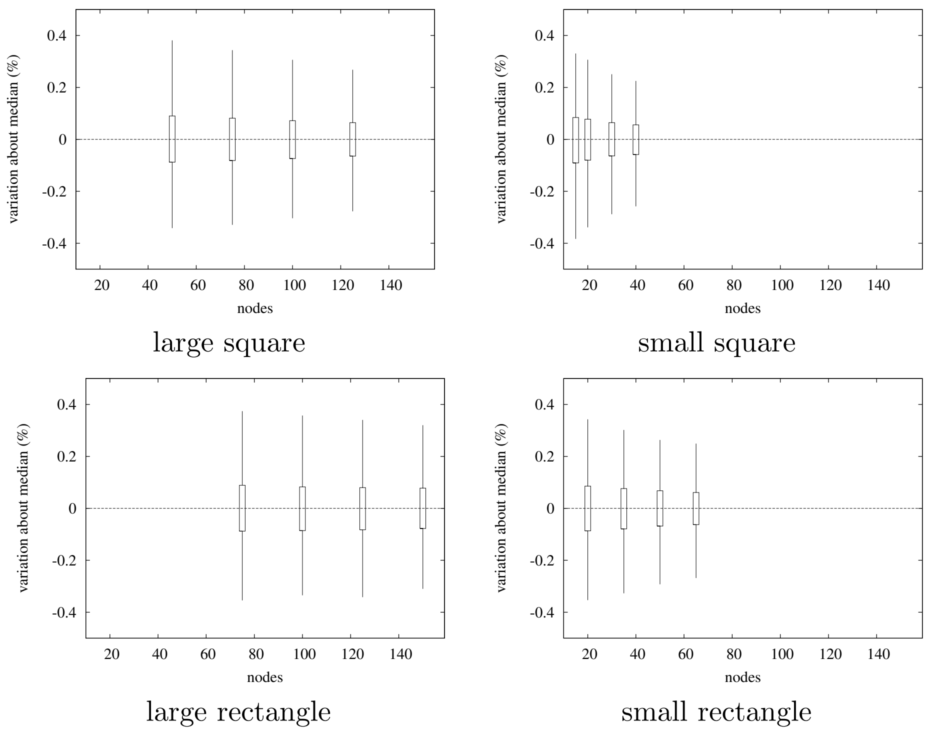


Figure 11: Variation in the total number of transmissions. The variation in the total number of transmissions is large, suggesting that flow capacity does not simply reflect the variation in path length among the candidate flows.

sions in the network – the number of feasible flows times their mean path length – should be roughly constant for each wakeup schedule distribution.

Figure 11 shows the mean distance of each quartile from the median, calculated as described in the previous section. Generally speaking, the inner quartiles (50% of the observed wakeup schedules) vary $\pm 10\%$ from the median, while the extremes vary $\pm 30 - 40\%$, which is even larger than the variation in the flow capacity. In short, there does not seem to be any evidence for a pattern of small variation around the natural total capacity of each topology – and thus little support for the "total transmission hypothesis".

To investigate further, we construct the artificial case where there is no variation in the path lengths of the candidate flows. Each subset of feasible flows therefore has the same number of total transmissions. Specifically, the candidate set includes only those connected node pairs with a path-length equal to the mean path-length of the configuration. In figure 3, we show the same analysis as in table 2.

nodes	median	inner quartile		min - max	
small square, small packet (1.5ms)					
14	22.4 (4.9)	-0.10 (0.05)	+0.08 (0.03)	-0.38 (0.11)	+0.30 (0.12)
22	27.6 (3.1)	-0.07 (0.02)	+0.07 (0.02)	-0.29 (0.07)	+0.23 (0.05)
32	30.5 (1.8)	-0.05 (0.02)	+0.06 (0.02)	-0.24 (0.04)	+0.19 (0.03)
small sqaure, large packet (4ms)					
8	9.3 (3.7)	-0.12 (0.07)	+0.10 (0.09)	-0.50 (0.17)	+0.30 (0.22)
14	8.2 (1.6)	-0.12 (0.04)	+0.11 (0.05)	-0.48 (0.11)	+0.39 (0.10)
22	8.8 (1.4)	-0.10 (0.05)	+0.09 (0.06)	-0.39 (0.07)	+0.34 (0.11)
32	9.8 (0.9)	-0.08 (0.04)	+0.09 (0.03)	-0.35 (0.06)	+0.30 (0.07)
small rectangle, small packet (1.5ms)					
10	17.5 (5.7)	-0.03 (0.05)	+0.02 (0.04)	-0.24 (0.19)	+0.03 (0.08)
25	22.8 (5.5)	-0.09 (0.03)	+0.09 (0.04)	-0.40 (0.09)	+0.33 (0.10)
40	24.4 (4.1)	-0.08 (0.04)	+0.07 (0.03)	-0.34 (0.08)	+0.27 (0.08)
55	25.8 (1.8)	-0.07 (0.02)	+0.07 (0.02)	-0.29 (0.05)	+0.24 (0.05)
small rectangle, large packet (4ms)					
10	9.6 (3.9)	-0.13 (0.10)	+0.11 (0.07)	-0.50 (0.19)	+0.32 (0.24)
25	7.1 (2.6)	-0.11 (0.09)	+0.12 (0.09)	-0.51 (0.16)	+0.45 (0.18)
40	7.4 (1.6)	-0.10 (0.07)	+0.11 (0.08)	-0.43 (0.10)	+0.41 (0.14)
55	7.8 (1.1)	-0.10 (0.06)	+0.09 (0.06)	-0.42 (0.08)	+0.37 (0.10)
large square, small packet (1.5ms)					
45	25.6 (7.9)	-0.08 (0.03)	+0.09 (0.05)	-0.35 (0.11)	+0.29 (0.16)
60	22.3 (6.1)	-0.09 (0.04)	+0.09 (0.03)	-0.37 (0.12)	+0.35 (0.13)
75	21.2 (4.7)	-0.08 (0.03)	+0.08 (0.03)	-0.36 (0.08)	+0.31 (0.09)
30	28.6 (9.9)	-0.07 (0.04)	+0.06 (0.04)	-0.29 (0.14)	+0.18 (0.15)
large square, large packet (4ms)					
45	8.3 (2.9)	-0.10 (0.06)	+0.12 (0.08)	-0.45 (0.11)	+0.42 (0.19)
60	6.9 (2.4)	-0.11 (0.07)	+0.13 (0.08)	-0.48 (0.12)	+0.48 (0.19)
75	6.3 (1.9)	-0.13 (0.08)	+0.13 (0.10)	-0.50 (0.12)	+0.48 (0.16)
30	12.1 (3.6)	-0.09 (0.04)	+0.09 (0.05)	-0.41 (0.11)	+0.34 (0.15)
large rectangle, small packet (1.5ms)					
25	24.9 (8.5)	-0.05 (0.04)	+0.03 (0.04)	-0.25 (0.16)	+0.08 (0.10)
75	24.0 (8.8)	-0.09 (0.04)	+0.09 (0.04)	-0.34 (0.11)	+0.33 (0.17)
100	20.4 (6.7)	-0.09 (0.03)	+0.10 (0.04)	-0.37 (0.10)	+0.36 (0.13)
50	32.3 (8.4)	-0.07 (0.03)	+0.07 (0.04)	-0.29 (0.11)	+0.24 (0.15)
large rectangle, large packet (4ms)					
75	7.6 (2.5)	-0.12 (0.07)	+0.12 (0.09)	-0.45 (0.12)	+0.46 (0.21)
100	6.1 (2.2)	-0.11 (0.09)	+0.14 (0.10)	-0.48 (0.11)	+0.53 (0.17)
25	13.4 (4.8)	-0.09 (0.06)	+0.09 (0.06)	-0.41 (0.18)	+0.29 (0.22)
50	11.7 (3.2)	-0.10 (0.04)	+0.09 (0.06)	-0.40 (0.12)	+0.33 (0.12)

Table 3: The experiment is similar to that of table 2, but the flows in the candidate set all have the same path length (the mean path length for the topology). We consider here two different packet sizes, 1.5 ms and 4 ms, which are slightly larger and smaller than the previous experiment. The variation continues to be substantial.

The relative variation in flow capacity observed in the case of a fixed path-length is at least as large as (and in some cases larger than) the variation in the previous case. This further confirms that path-length is not the primary source of variation in flow capacity. (Note that the median flow capacity is also somewhat reduced compared to the previous experiment, since we use the mean path length of all connected pairs, rather than that of the baseline flows. The absolute variation is, however, comparable between the two experiments.)

6 Related work

Below, we outline some power save protocols have been proposed for use in ad hoc networks. To the best of our knowledge, none of these protocols have been subjected to analysis presented above.

Some of the most popular strategies [4, 16] use clustering to mimic the power save operation of an infrastructure network. In these techniques, nodes dynamically elect a topologically covering subset of nodes act as cluster-heads, remaining awake and buffering traffic for their sleeping neighbors. The cluster-head role rotates among the nodes in order to equalize energy consumption across the network. Such protocols often require network-wide synchronization for nodes to perform topology discovery and generate efficient clustering structures. Although synchronization mechanisms have been developed[19, 6, 14], these solutions do incur non-negligible overhead and complexity, particularly in the case of networks subject to partition and merge.

To address these concerns, researchers have proposed a number of asynchronous power save mechanisms.

A few protocols are based on completely probabilistic techniques[10, 17]. No attempt is made to predict nodes' wakeup schedules: messages are transmitted sufficiently often to ensure that they are (with high probability) received and forwarded. This approach is also popular in sensor networks due to their high node density and specialized traffic pattern in which traffic is always directed to or from a centralized "gateway".

A larger class of protocols [7, 3, 18, 9] are based on a common periodic wakeup schedule, which is known to all nodes. Because the nodes operate asynchronously, each node follows the wakeup schedule with an unknown offset relative to the others. This paper investigated a simple instance of such a protocol, in which length of the wakeup schedule is fixed. Other variants of these protocols

[4]. In this paper, this distribution of offsets is referred to as the "wakeup schedule distribution". We investigate a simple instance of such a protocol.

[4]In practical terms, the clock drift is assumed to be small relative to the length of the period.

7 Conclusions and future work

In this paper, we investigate the sensitivity of a simple asynchronous power save protocol to the random variation in the distribution of the nodes' wakeup schedules.

We define a simplified model of network operation that is efficient for studying this problem in simulation. The model is also amenable to combinatoric analysis and we are currently pursuing work further understanding both the power and limitations of this model.

In the simulation results, we do observe considerable variation in flow capacity. This variation is more or less normally distributed, with the inner quartiles varying only about $\pm 5\%$ from the median, while the maximum and minimum flow capacities vary some 20% from the median. Further analysis confirms that this variation is related to the variation in wakeup schedule distribution.

As a matter of practical significance, we see that the "best" wakeup schedule distributions obtain significantly higher than the median capacity, often approaching 75% of the capacity obtained in the absence of a power save protocol.

This result suggests that developing techniques for managing the wakeup schedule distribution may improve the performance of such protocols. Because asynchronous power save protocols are intended to function correctly for any wakeup schedule distribution, it should be possible for nodes to (randomly or deterministically) seek out more favorable distributions. The relative rarity of particularly good distributions suggests, however, that simple randomized strategies may be unlikely to be highly effective. The development of effective strategies for managing wakeup schedules is the focus of future work.

References

[1] http://www.isi.edu/nsnam.

[2] Ahn, G.-S., Campbell, A. T., Veres, A., and Sun, L.-H. Supporting service differentiation for real-time and best-effort traffic in stateless wireless ad hoc networks (swan). *IEEE Trans. Mob. Comput. 1*, 3 (2002), 192–207.

[3] Braun, T., and Feeney, L. M. Power saving in wireless ad hoc networks without synchronization. In *Scandinavian Workshop on Wireless Ad Hoc Networks ADHOC'05* (2005).

[4] Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks 8*, 5 (2002), 481–494.

[5] Chen, L., and Heinzelman, W. R. Network architecture to support qos in mobile ad hoc networks. In *ICME* (2004), IEEE, pp. 1715–1718.

[6] Elson, J., Girod, L., and Estrin, D. Fine-grained network time synchronization using reference broadcasts. In *OSDI* (2002).

[7] Feeney, L. M. A QoS aware power save protocol for wireless ad hoc networks. In *Proceedings of the First Mediterranean Workshop on Ad Hoc Networks (MedHocNet 2002)* (2002).

[8] Grönkvist, J. Distributed scheduling for mobile ad hoc networks - a novel approach. In *Proceeedings of the 15th IEEE Int'l Symposium on Personal, Indoor and Mobile Radio Communications PIMRC* (2004).

[9] Jiang, J.-R., Tseng, Y.-C., Hsu, C.-S., and Lai, T.-H. Quorum-based asynchronous power-saving protocols for ieee 802.11 ad hoc networks. *MONET 10*, 1-2 (2005), 169–181.

[10] McGlynn, M. J., and Borbash, S. A. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *MobiHoc* (2001), ACM, pp. 137–145.

[11] Németh, G., Turányi, Z., and Valkó, A. G. Throughput of ideally routed wireless ad hoc networks. *Mobile Computing and Communications Review 5*, 4 (2001), 40–46.

[12] Perkins, C., Belding-Royer, E., and Das, S. Ad hoc on-demand distance vector (AODV) routing. In *RFC 3561* (2003).

[13] Ramanathan, S. A unified framework and algorithm for channel assignment in wireless networks. *Wirel. Netw. 5*, 2 (1999), 81–94.

[14] Römer, K. Time synchronization in ad hoc networks. In *Proc. of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) 2001* (Oct. 2001), pp. 173–182.

[15] Salonidis, T., and Tassiulas, L. Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks. In *MobiHoc '05: Proceedings of the 6th ACM Int'l symposium on mobile ad hoc networking and computing* (2005), pp. 145–156.

[16] Wu, J., Gao, M., and Stojmenovic, I. On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. In *IEEE International Conference on Parallel Processing* (Sept. 2001), pp. 346–353.

[17] Xu, Y., Heidemann, J., and Estrin, D. Adaptive energy-conserving routing for multihop ad hoc networks. Research Report 527, USC/Information Sciences Institute, October 2000.

[18] Zheng, R., Hou, J. C., and Sha, L. Asynchronous wakeup for ad hoc networks. In *MobiHoc* (2003), ACM, pp. 35–45.

[19] Zhou, D., and Lai, T.-H. A compatible and scalable clock synchronization protocol in ieee 802.11 ad hoc networks. In *ICPP* (2005), IEEE Computer Society, pp. 295–302.

Appendix

Additional detail plots are included below. These plots include a series of experiments examining the effect of various frame transmission times and investigation of the relationship between the flow capacity and mean path length. **Note:** The results below are combined from a number of experimental configurations which do not necessarily correspond exactly to those presented in the paper. In particular, there is significantly greater variation in the range of packet sizes and of the candidate pool.

7.1 Histograms and normalized histograms

This section plots the histogram and the histogram normalized to the baseline capacity for each of three frame transmission times (each row).

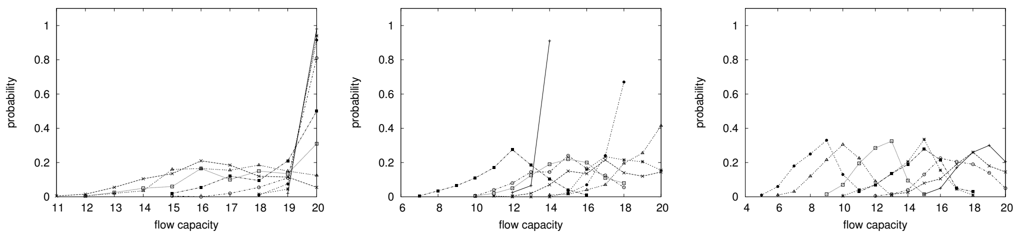


Figure 12: field = 6x6 hops, nodes = 20, flow pool = 20, xmit = 1.5/2.5/4% of period

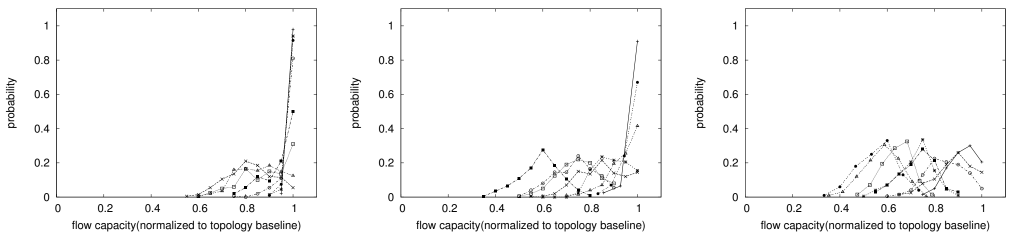


Figure 13: field = 6x6 hops, nodes = 20, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

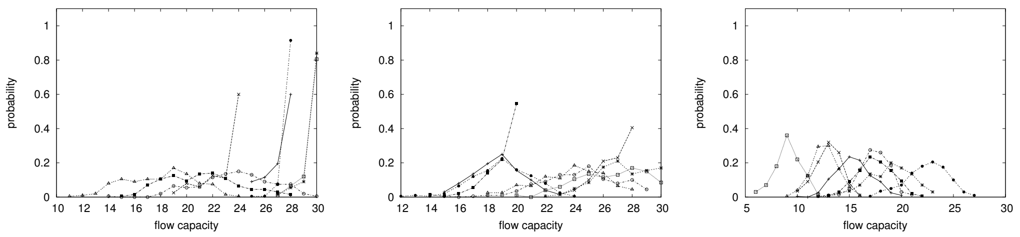


Figure 14: field = 6x6 hops, nodes = 20, flow pool = 30, xmit = 1.5/2.5/4% of period

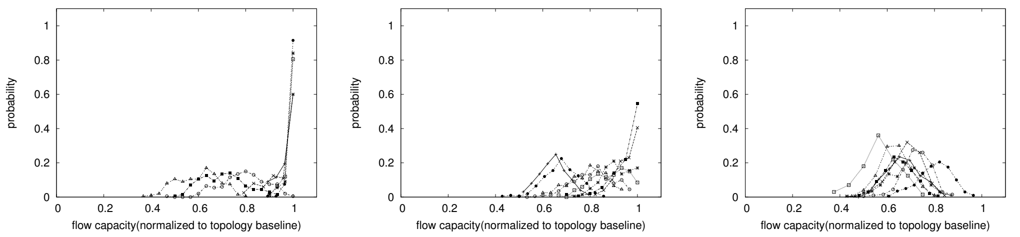


Figure 15: field = 6x6 hops, nodes = 20, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

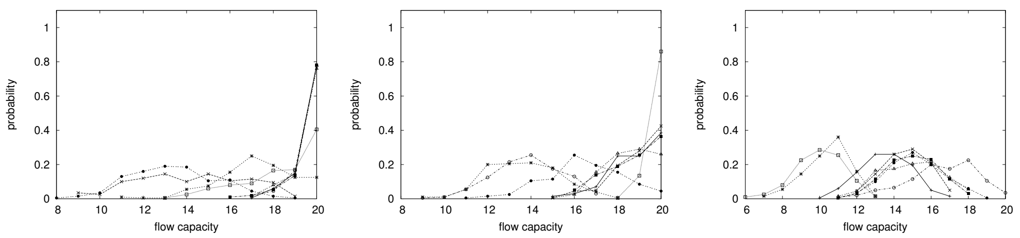


Figure 16: field = 6x6 hops, nodes = 30, flow pool = 20, xmit = 1.5/2.5/4% of period

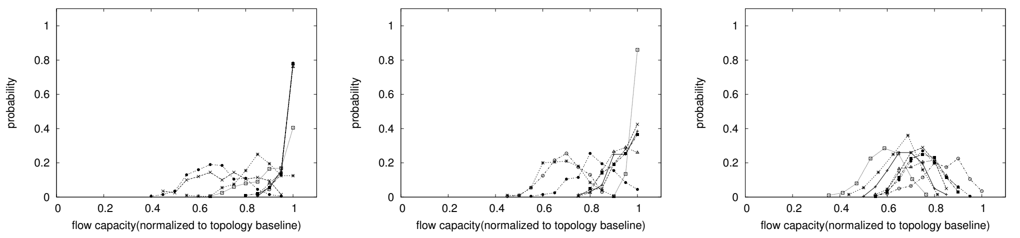


Figure 17: field = 6x6 hops, nodes = 30, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

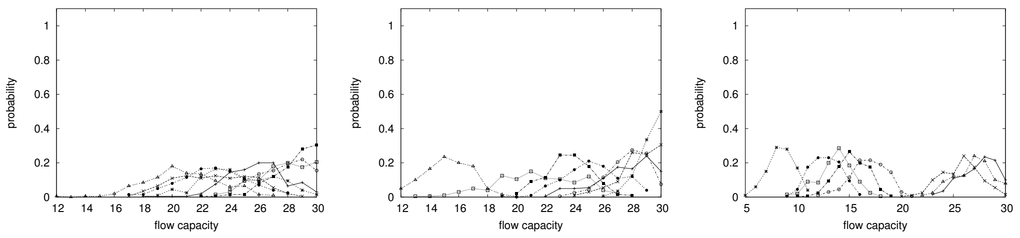


Figure 18: field = 6x6 hops, nodes = 30, flow pool = 30, xmit = 1.5/2.5/4% of period

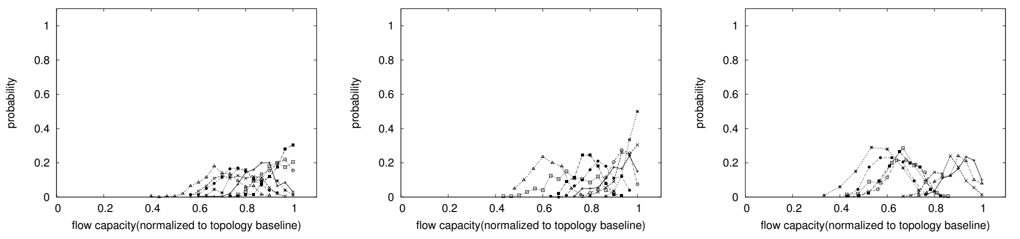


Figure 19: field = 6x6 hops, nodes = 30, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

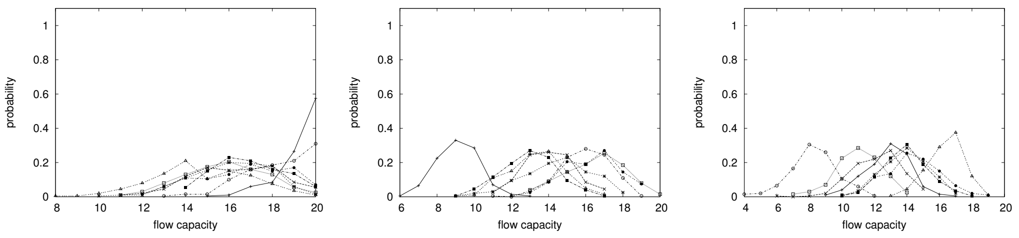


Figure 20: field = 6x6 hops, nodes = 40, flow pool = 20, xmit = 1.5/2.5/4% of period

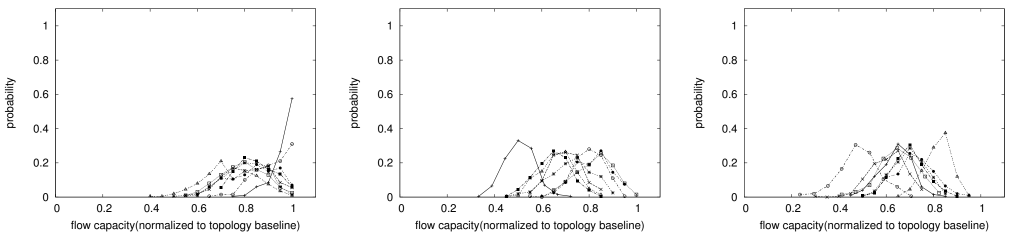


Figure 21: field = 6x6 hops, nodes = 40, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

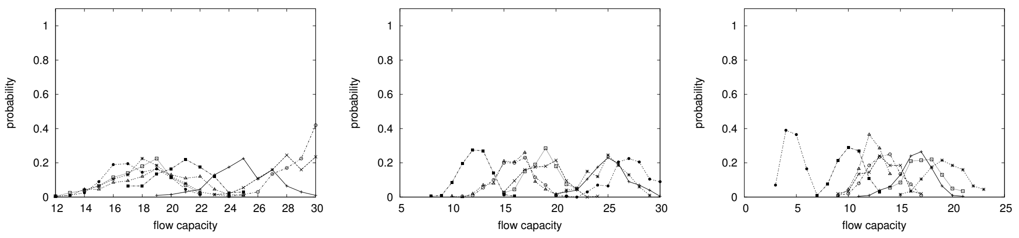


Figure 22: field = 6x6 hops, nodes = 40, flow pool = 30, xmit = 1.5/2.5/4% of period

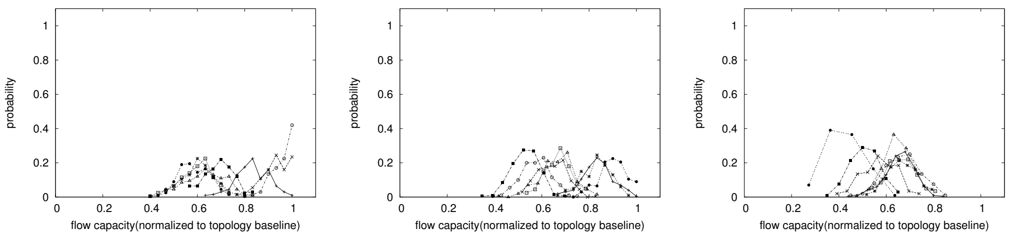


Figure 23: field = 6x6 hops, nodes = 40, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

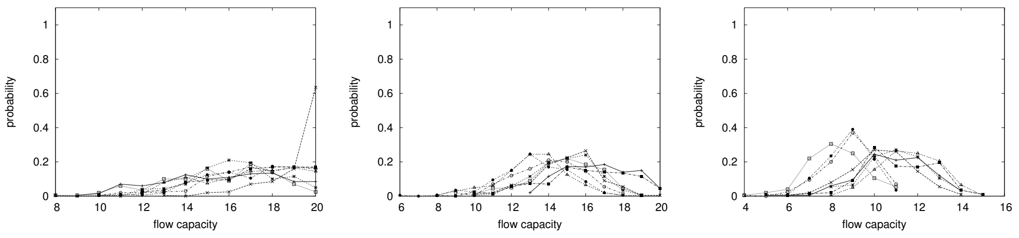


Figure 24: field = 2.5x2.5 hops, nodes = 10, flow pool = 20, xmit = 1.5/2.5/4% of period

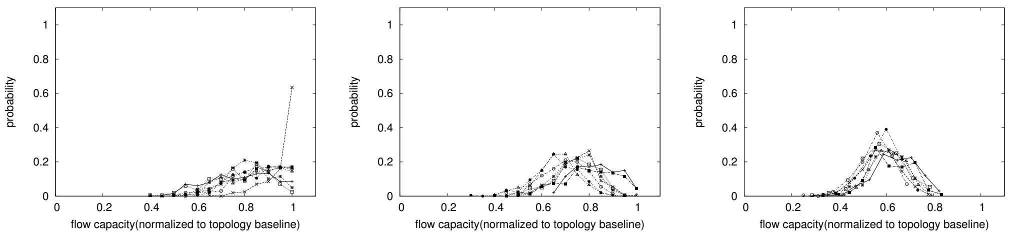


Figure 25: field = 2.5x2.5 hops, nodes = 10, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

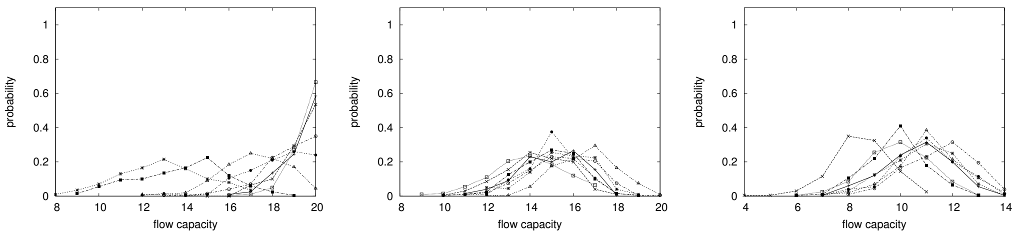


Figure 26: field = 2.5x2.5 hops, nodes = 20, flow pool = 20, xmit = 1.5/2.5/4% of period

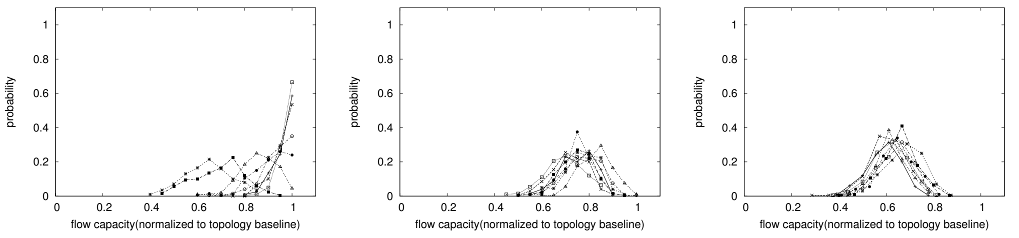


Figure 27: field = 2.5x2.5 hops, nodes = 20, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

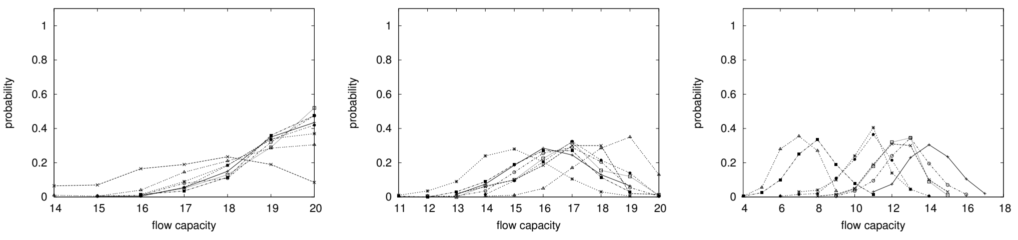


Figure 28: field = 2.5x2.5 hops, nodes = 30, flow pool = 20, xmit = 1.5/2.5/4% of period

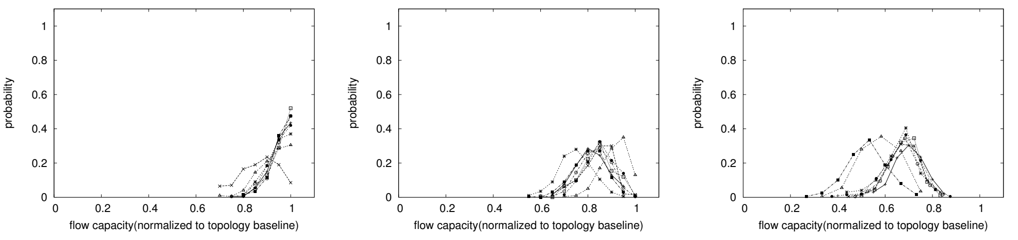


Figure 29: field = 2.5x2.5 hops, nodes = 30, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

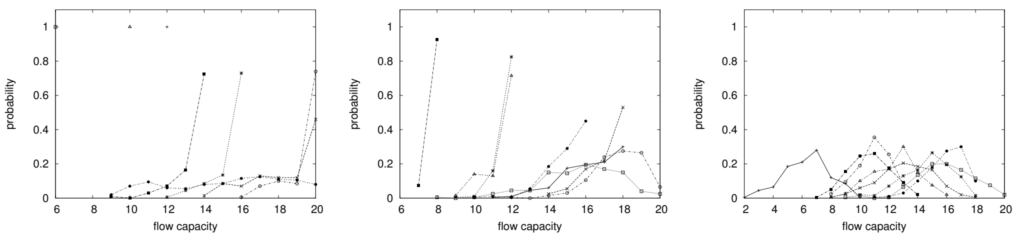


Figure 30: field = 0.7x10 hops, nodes = 10, flow pool = 20, xmit = 1.5/2.5/4% of period

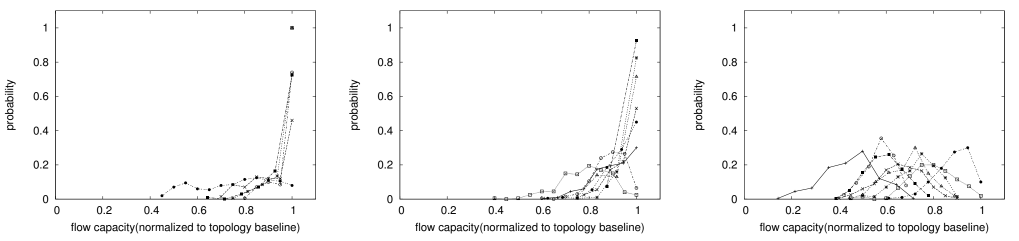


Figure 31: field = 0.7x10 hops, nodes = 10, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

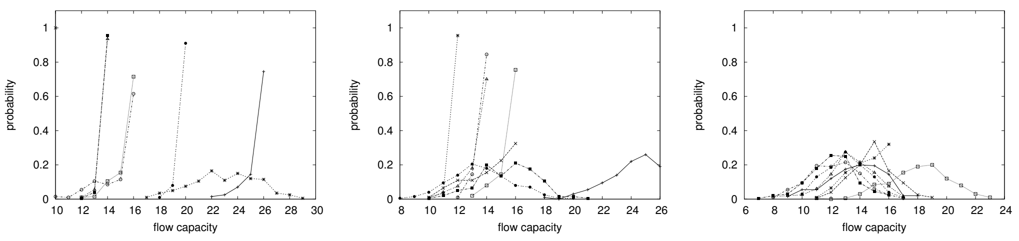


Figure 32: field = 0.7x10 hops, nodes = 10, flow pool = 30, xmit = 1.5/2.5/4% of period

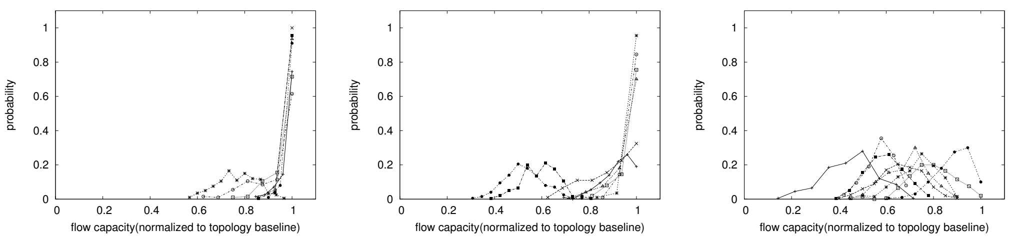


Figure 33: field = 0.7x10 hops, nodes = 10, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

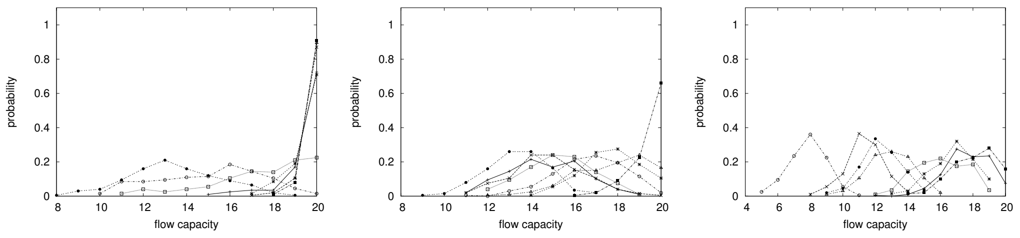


Figure 34: field = 0.7x10 hops, nodes = 20, flow pool = 20, xmit = 1.5/2.5/4% of period

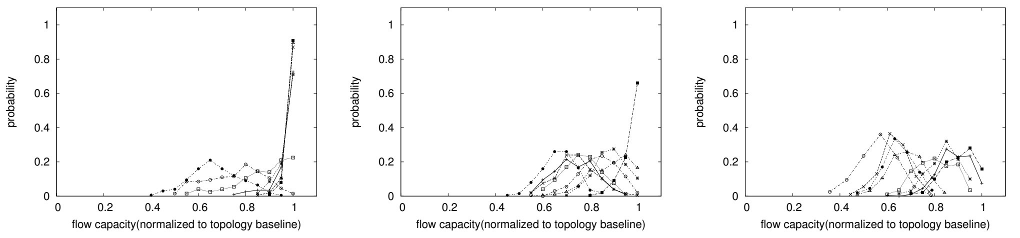


Figure 35: field = 0.7x10 hops, nodes = 20, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

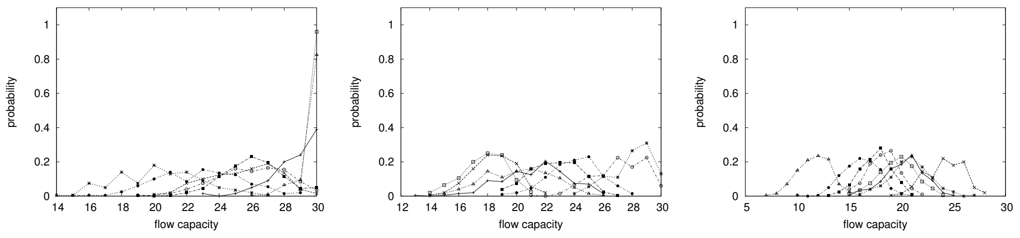


Figure 36: field = 0.7x10 hops, nodes = 20, flow pool = 30, xmit = 1.5/2.5/4% of period

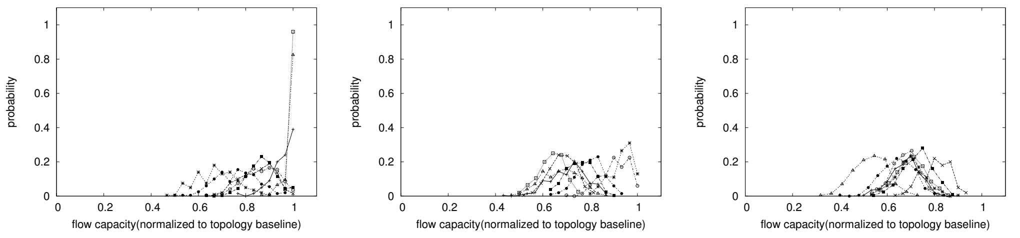


Figure 37: field = 0.7x10 hops, nodes = 20, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

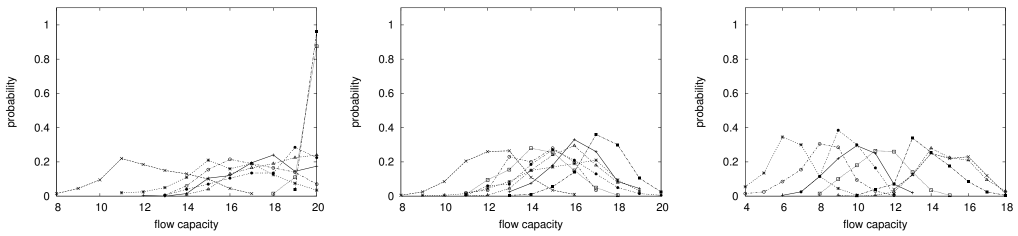


Figure 38: field = 0.7x10 hops, nodes = 30, flow pool = 20, xmit = 1.5/2.5/4% of period

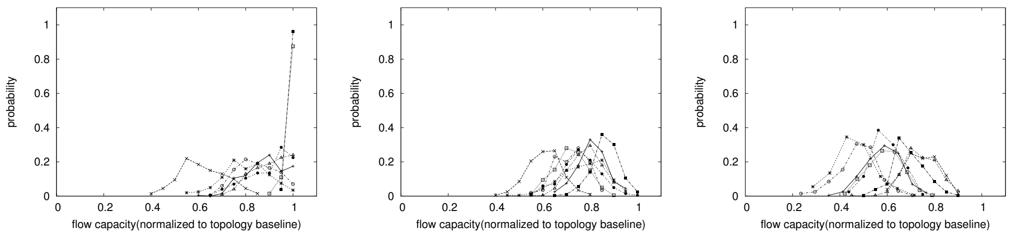


Figure 39: field = 0.7x10 hops, nodes = 30, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

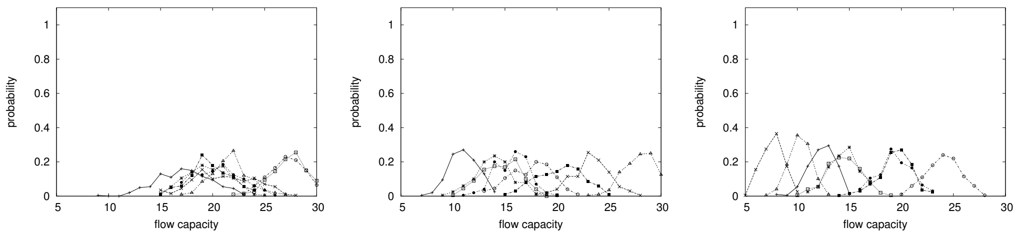


Figure 40: field = 0.7x10 hops, nodes = 30, flow pool = 30, xmit = 1.5/2.5/4% of period

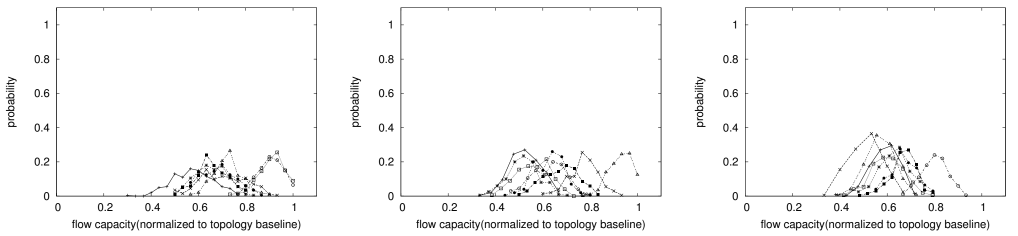


Figure 41: field = 0.7x10 hops, nodes = 30, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

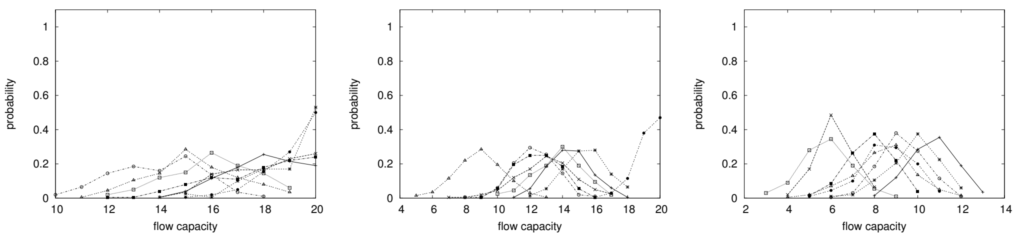


Figure 42: field = 0.7x10 hops, nodes = 40, flow pool = 20, xmit = 1.5/2.5/4% of period

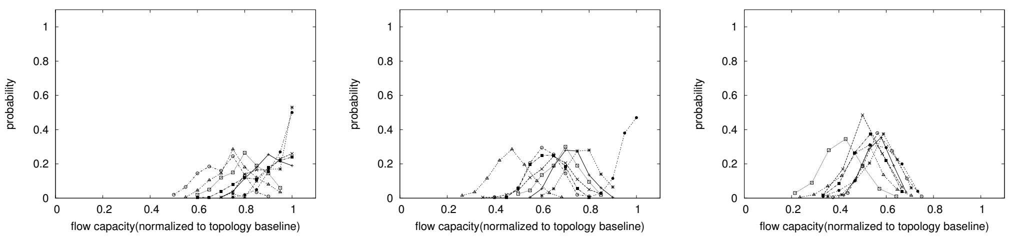


Figure 43: field = 0.7x10 hops, nodes = 40, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

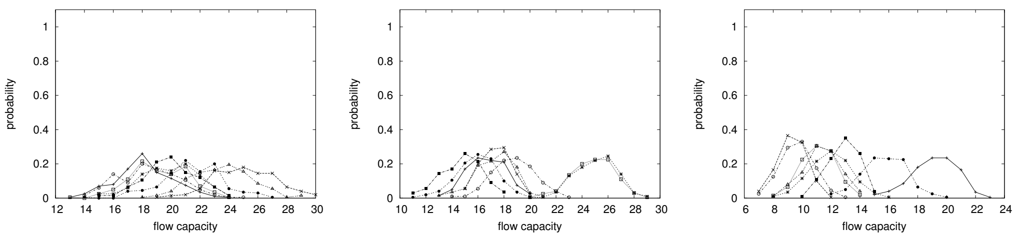


Figure 44: field = 0.7x10 hops, nodes = 40, flow pool = 30, xmit = 1.5/2.5/4% of period

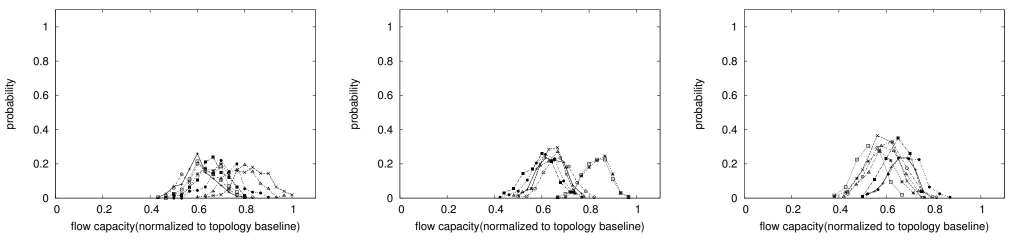


Figure 45: field = 0.7x10 hops, nodes = 40, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

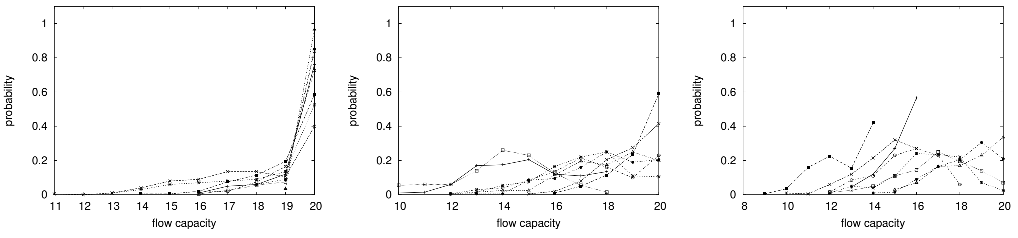


Figure 46: field = 1.5x25 hops, nodes = 20, flow pool = 20, xmit = 1.5/2.5/4% of period

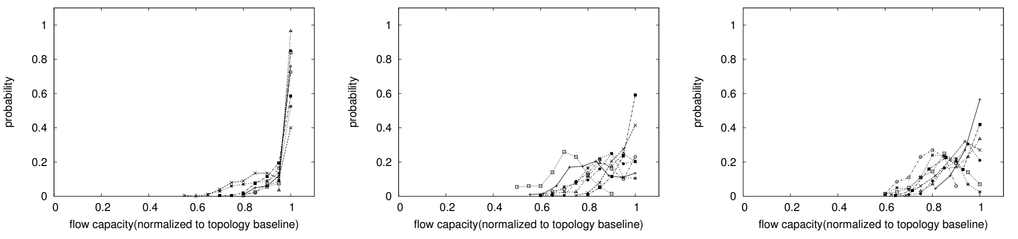


Figure 47: field = 1.5x25 hops, nodes = 20, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

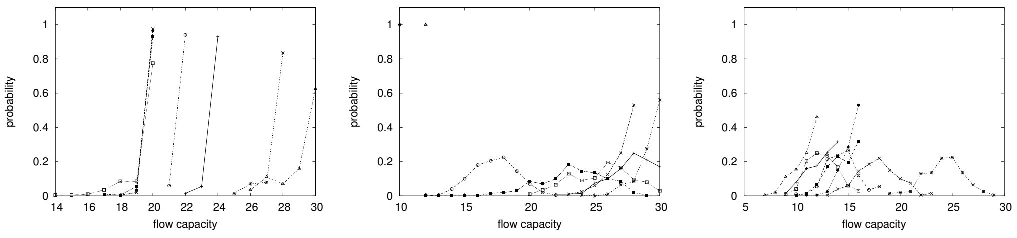


Figure 48: field = 1.5x25 hops, nodes = 20, flow pool = 30, xmit = 1.5/2.5/4% of period

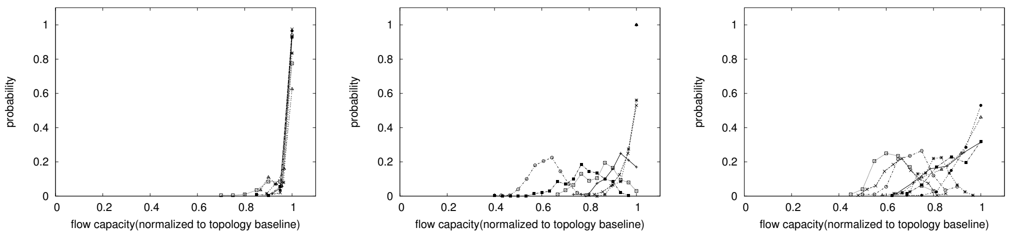


Figure 49: field = 1.5x25 hops, nodes = 20, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

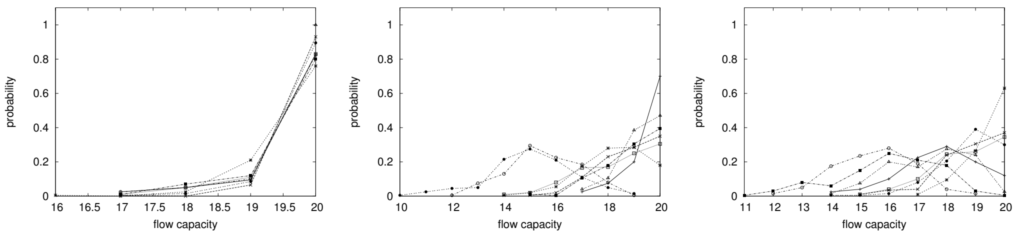


Figure 50: field = 1.5x25 hops, nodes = 30, flow pool = 20, xmit = 1.5/2.5/4% of period

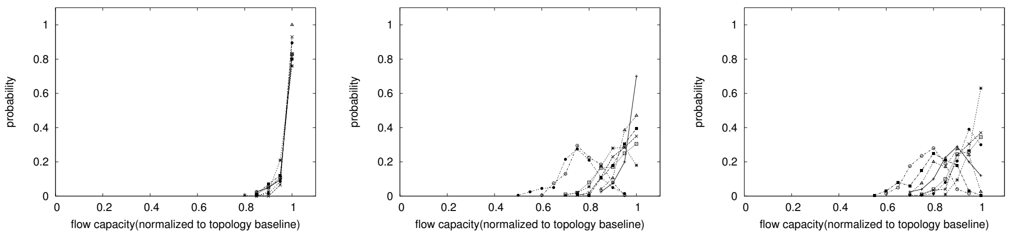


Figure 51: field = 1.5x25 hops, nodes = 30, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

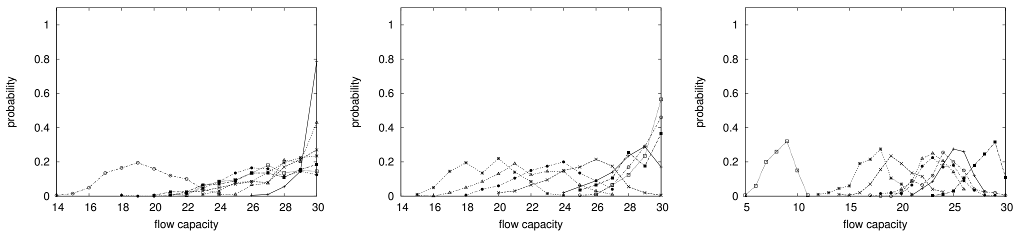


Figure 52: field = 1.5x25 hops, nodes = 30, flow pool = 30, xmit = 1.5/2.5/4% of period

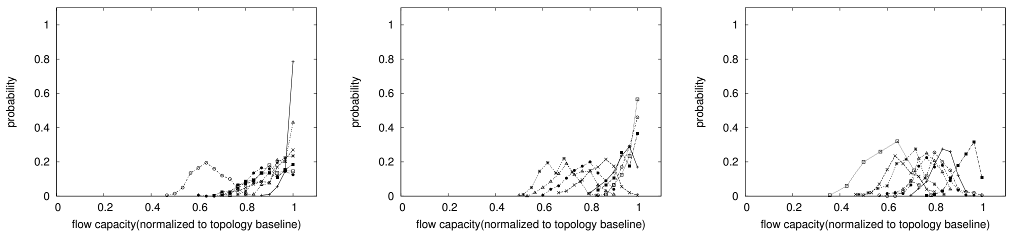


Figure 53: field = 1.5x25 hops, nodes = 30, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

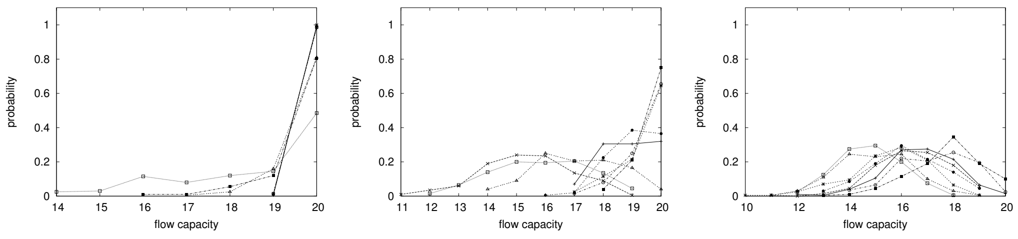


Figure 54: field = 1.5x25 hops, nodes = 40, flow pool = 20, xmit = 1.5/2.5/4% of period

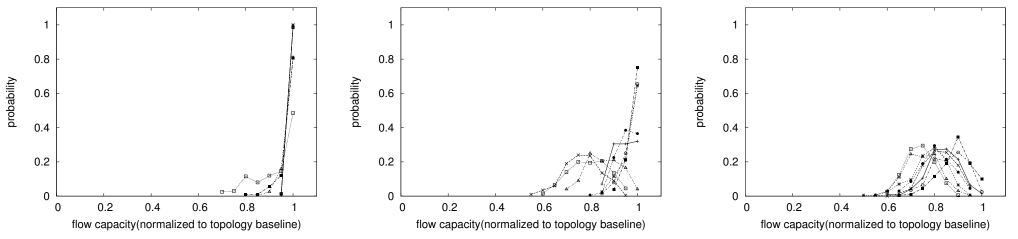


Figure 55: field = 1.5x25 hops, nodes = 40, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

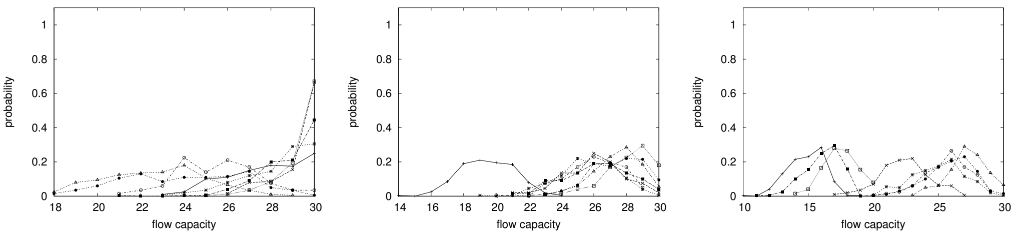


Figure 56: field = 1.5x25 hops, nodes = 40, flow pool = 30, xmit = 1.5/2.5/4% of period

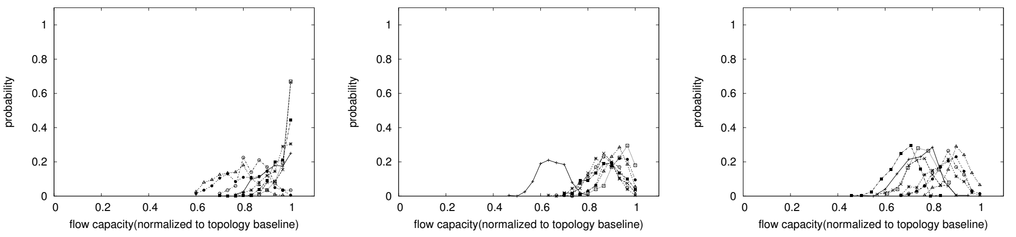


Figure 57: field = 1.5x25 hops, nodes = 40, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

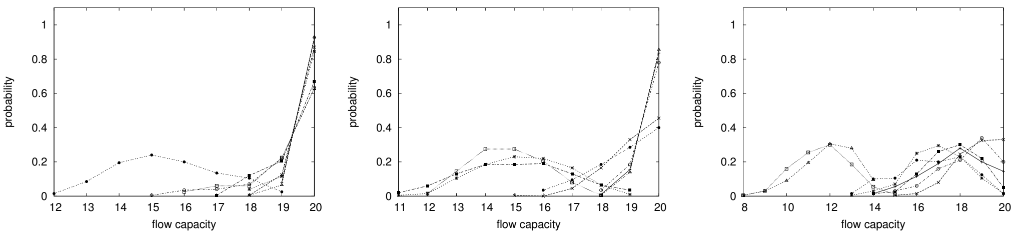


Figure 58: field = 1.5x25 hops, nodes = 50, flow pool = 20, xmit = 1.5/2.5/4% of period

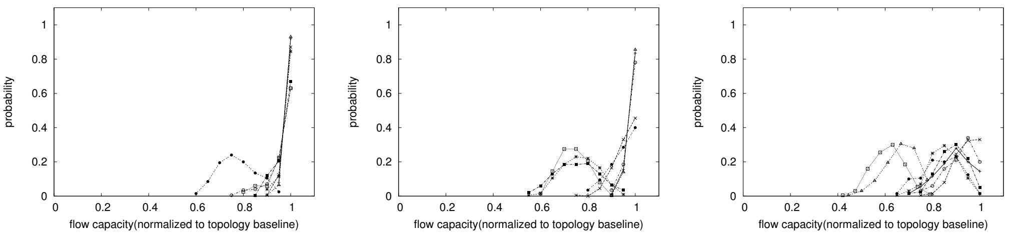


Figure 59: field = 1.5x25 hops, nodes = 50, flow pool = 20, xmit = 1.5/2.5/4% of period, normalized

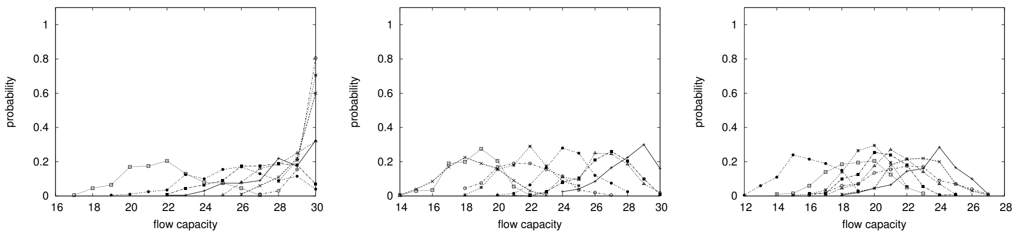


Figure 60: field = 1.5x25 hops, nodes = 50, flow pool = 30, xmit = 1.5/2.5/4% of period

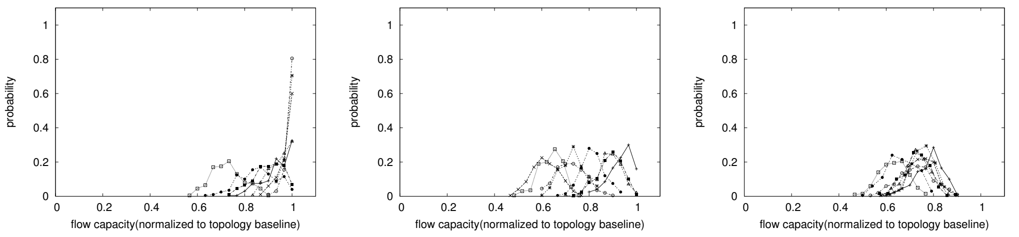


Figure 61: field = 1.5x25 hops, nodes = 50, flow pool = 30, xmit = 1.5/2.5/4% of period, normalized

7.2 Histograms and CDF

This subsection plots the histogram, normalized histogram and cdf of the normalized histogram for various scenarios

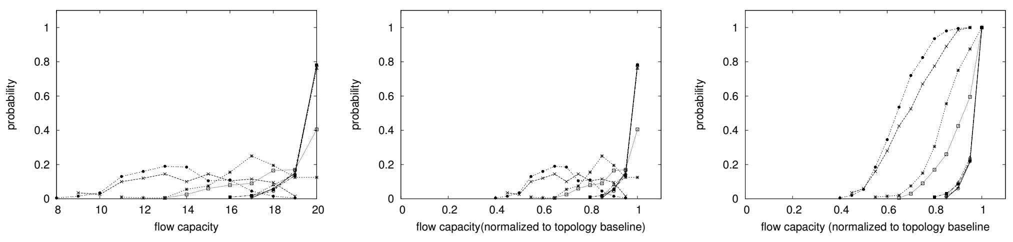


Figure 62: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 30, flow pool = 20, xmit = 1.5% of period

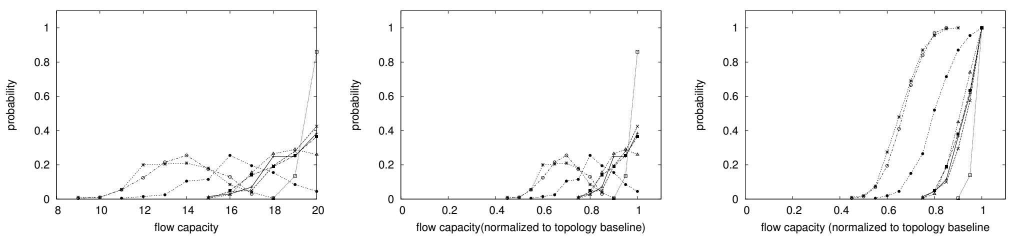


Figure 63: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 30, flow pool = 20, xmit = 2.5% of period

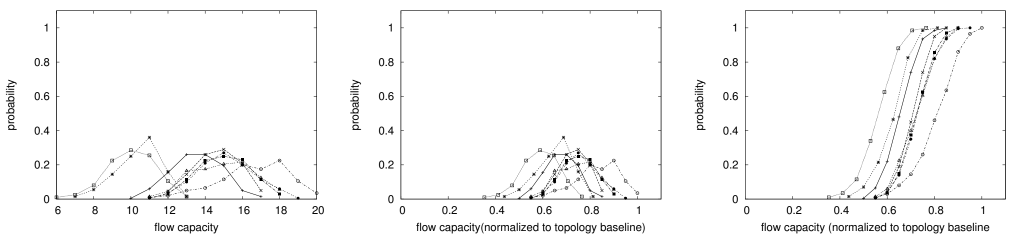


Figure 64: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 30, flow pool = 20, xmit = 4% of period

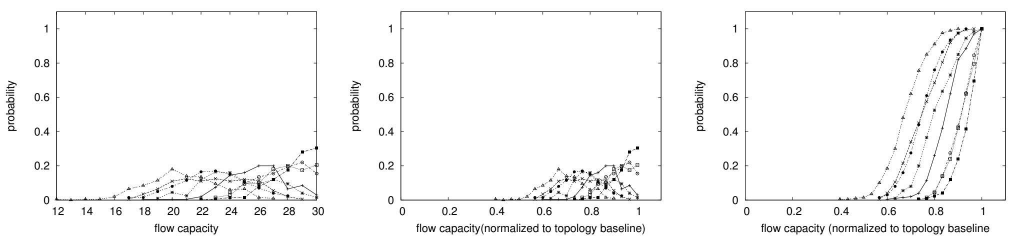


Figure 65: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 30, flow pool = 30, xmit = 1.5% of period

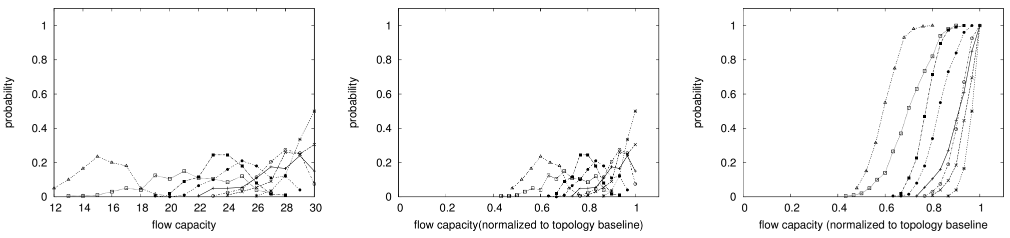


Figure 66: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 30, flow pool = 30, xmit = 2.5% of period

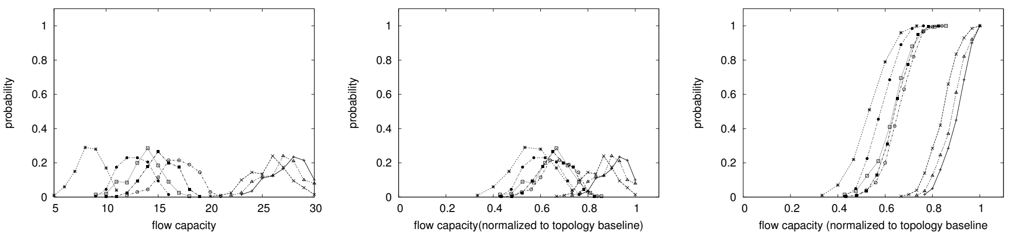


Figure 67: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 30, flow pool = 30, xmit = 4% of period

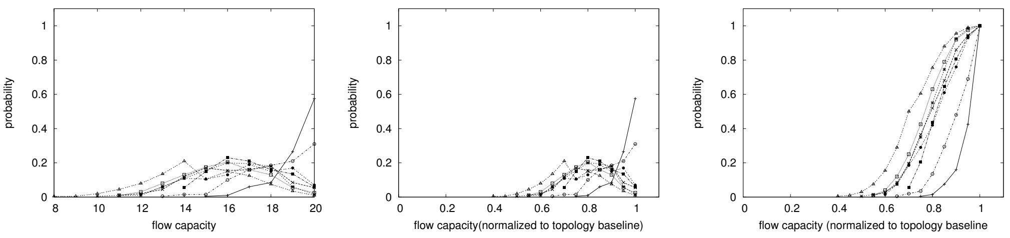


Figure 68: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 40, flow pool = 20, xmit = 1.5% of period

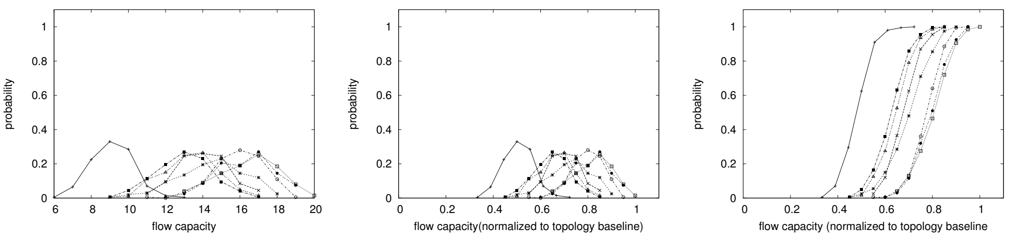


Figure 69: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 40, flow pool = 20, xmit = 2.5% of period

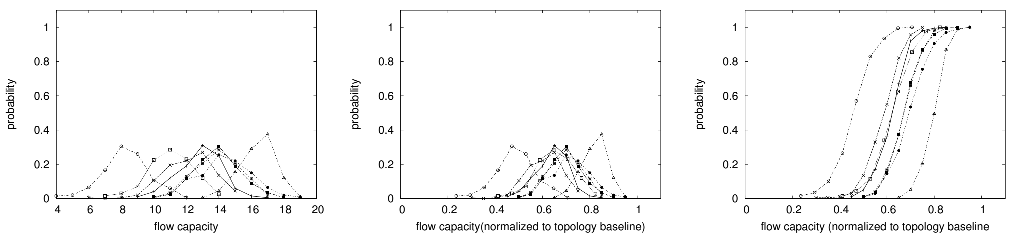


Figure 70: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 40, flow pool = 20, xmit = 4% of period

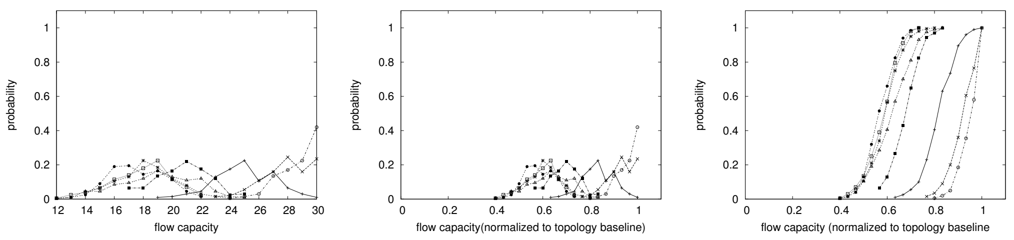


Figure 71: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 40, flow pool = 30, xmit = 1.5% of period

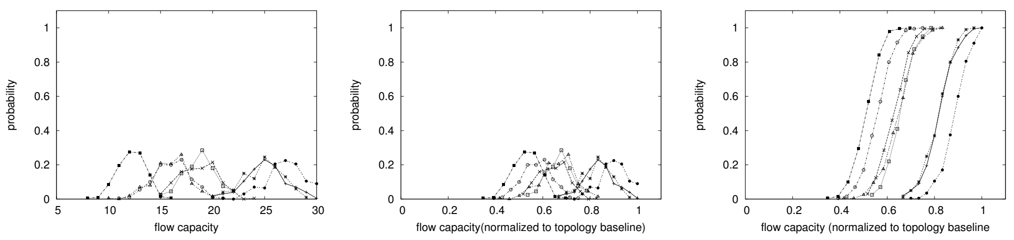


Figure 72: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 40, flow pool = 30, xmit = 2.5% of period

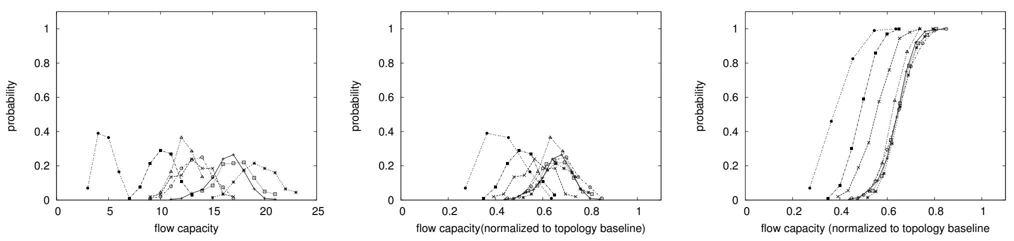


Figure 73: histogram, normalized histogram, normalized cdf; field = 6x6 hops, nodes = 40, flow pool = 30, xmit = 4% of period

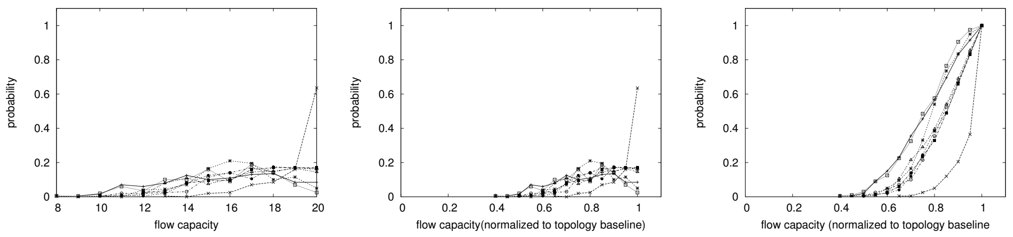


Figure 74: histogram, normalized histogram, normalized cdf; field = 2.5x2.5 hops, nodes = 10, flow pool = 20, xmit = 1.5% of period

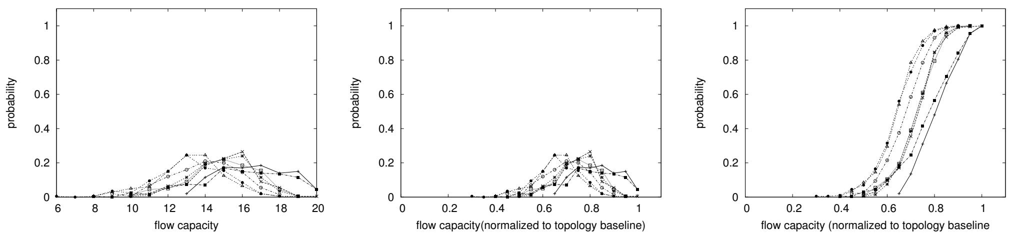


Figure 75: histogram, normalized histogram, normalized cdf; field = 2.5x2.5 hops, nodes = 10, flow pool = 20, xmit = 2.5% of period

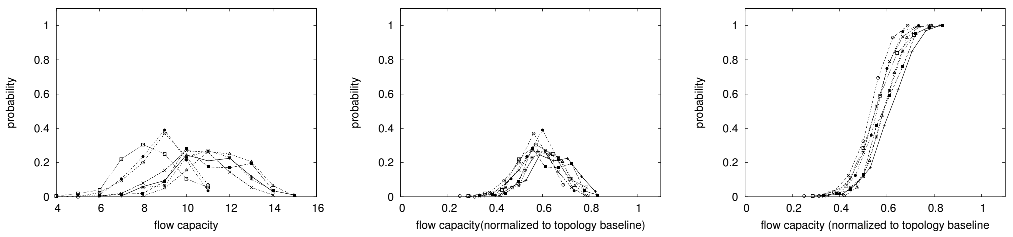


Figure 76: histogram, normalized histogram, normalized cdf; field = 2.5x2.5 hops, nodes = 10, flow pool = 20, xmit = 4% of period

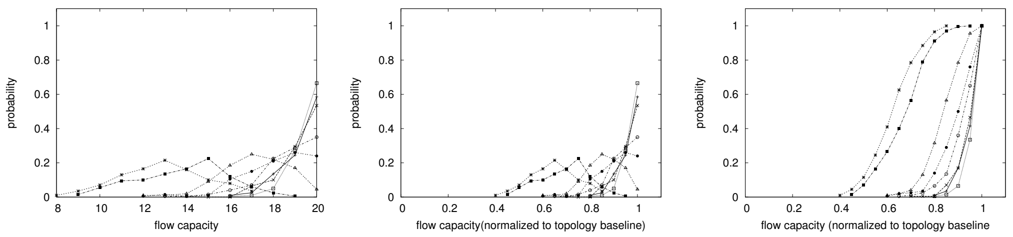


Figure 77: histogram, normalized histogram, normalized cdf; field = 2.5x2.5 hops, nodes = 20, flow pool = 20, xmit = 1.5% of period

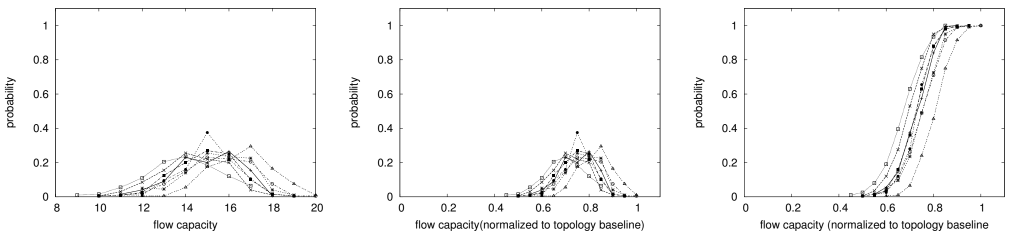


Figure 78: histogram, normalized histogram, normalized cdf; field = 2.5x2.5 hops, nodes = 20, flow pool = 20, xmit = 2.5% of period

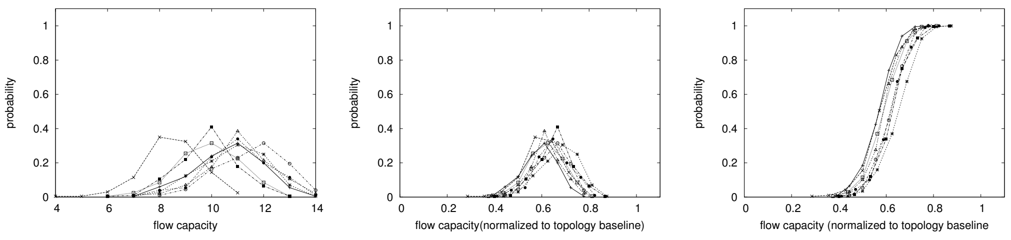


Figure 79: histogram, normalized histogram, normalized cdf; field = 2.5x2.5 hops, nodes = 20, flow pool = 20, xmit = 4% of period

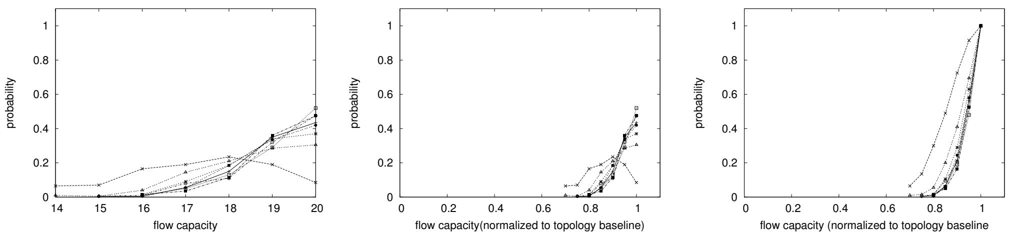


Figure 80: histogram, normalized histogram, normalized cdf; field = 2.5x2.5 hops, nodes = 30, flow pool = 20, xmit = 1.5% of period

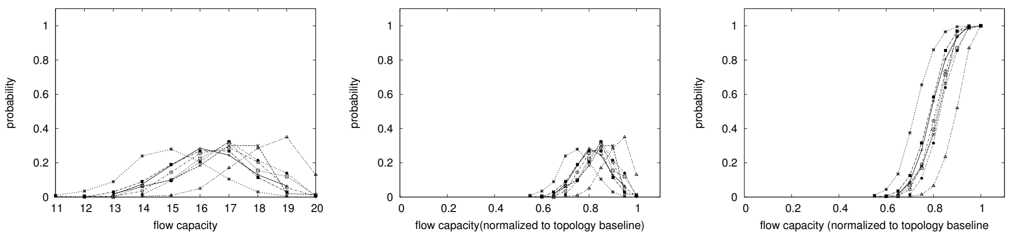


Figure 81: histogram, normalized histogram, normalized cdf; field = 2.5x2.5 hops, nodes = 30, flow pool = 20, xmit = 2.5% of period

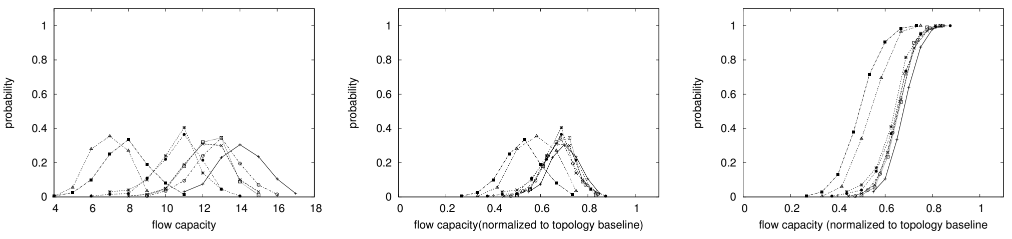


Figure 82: histogram, normalized histogram, normalized cdf; field = 2.5x2.5 hops, nodes = 30, flow pool = 20, xmit = 4% of period

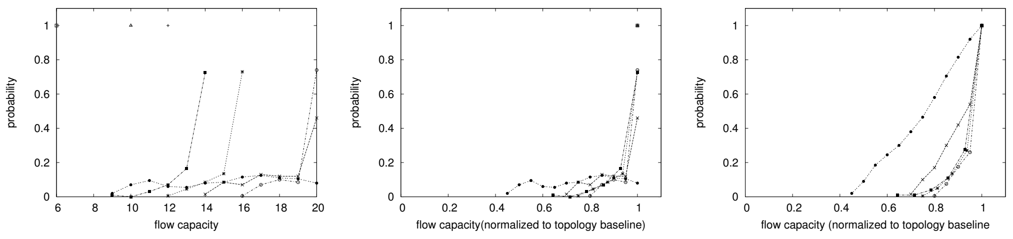


Figure 83: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 10, flow pool = 20, xmit = 1.5% of period

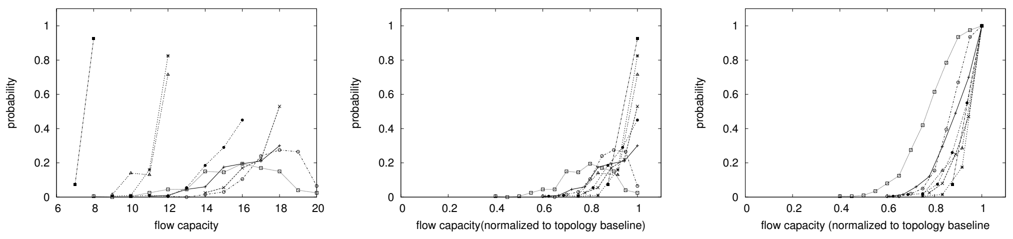


Figure 84: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 10, flow pool = 20, xmit = 2.5% of period

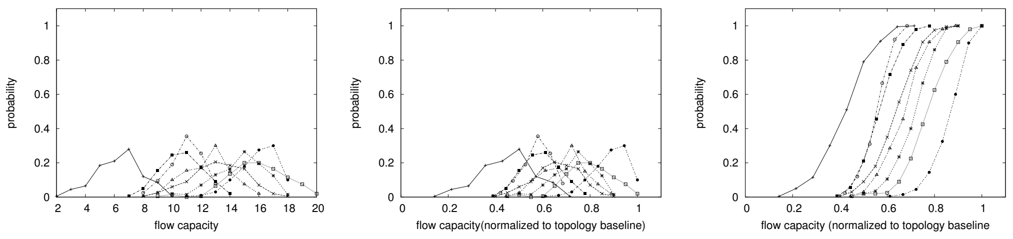


Figure 85: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 10, flow pool = 20, xmit = 4% of period

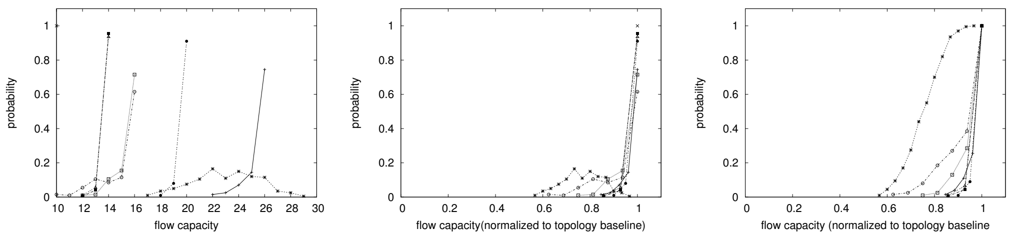


Figure 86: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 10, flow pool = 30, xmit = 1.5% of period

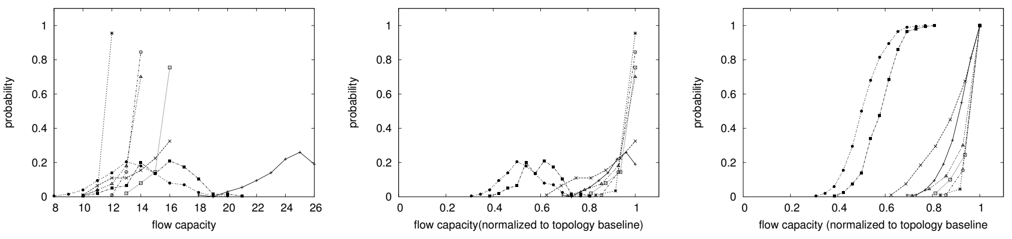


Figure 87: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 10, flow pool = 30, xmit = 2.5% of period

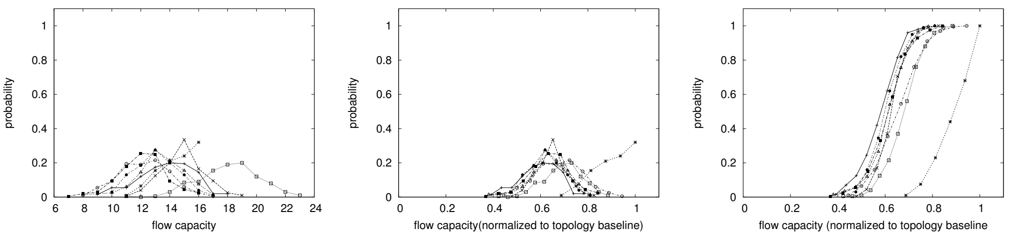


Figure 88: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 10, flow pool = 30, xmit = 4% of period

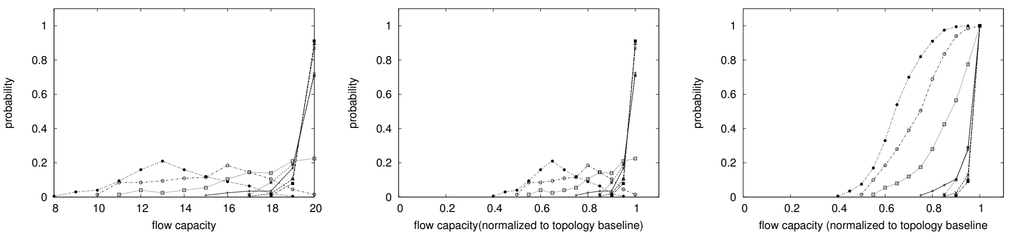


Figure 89: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 20, flow pool = 20, xmit = 1.5% of period

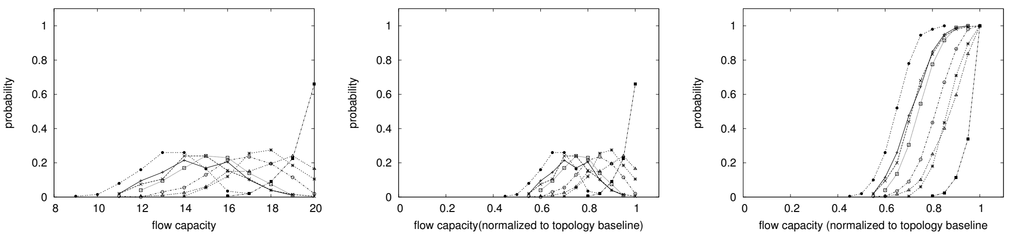


Figure 90: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 20, flow pool = 20, xmit = 2.5% of period

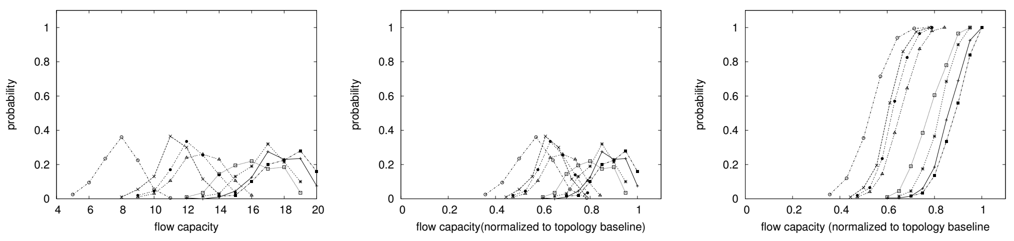


Figure 91: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 20, flow pool = 20, xmit = 4% of period

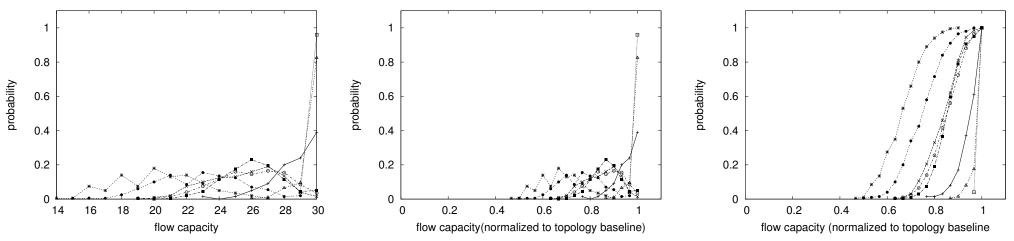


Figure 92: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 20, flow pool = 30, xmit = 1.5% of period

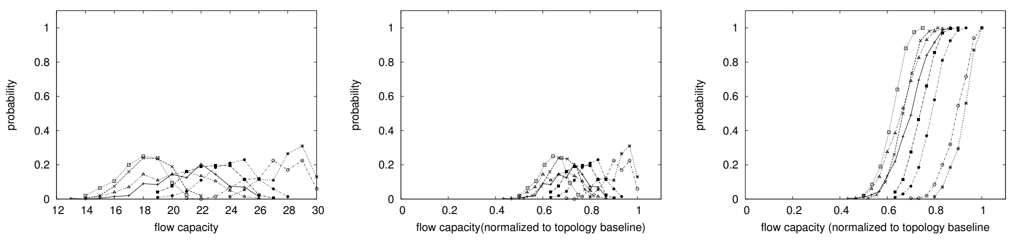


Figure 93: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 20, flow pool = 30, xmit = 2.5% of period

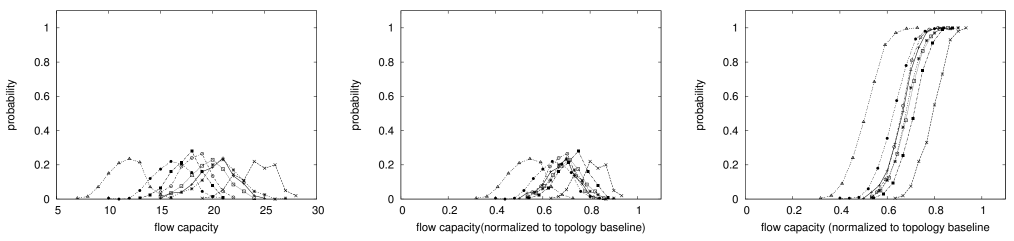


Figure 94: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 20, flow pool = 30, xmit = 4% of period

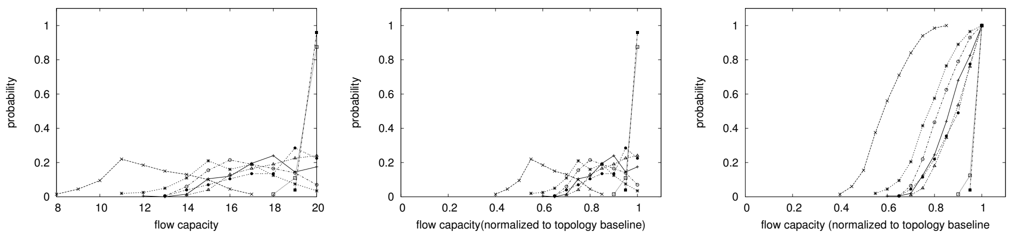


Figure 95: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 30, flow pool = 20, xmit = 1.5% of period

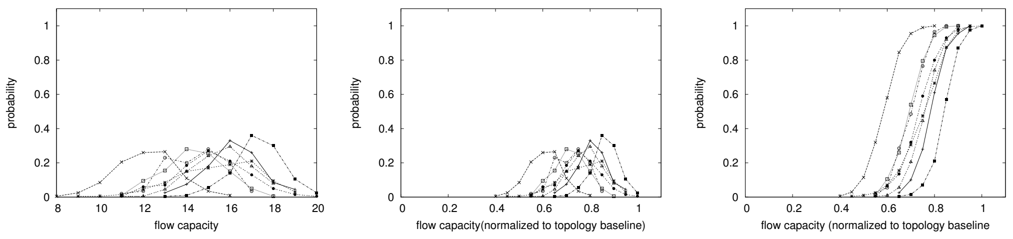


Figure 96: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 30, flow pool = 20, xmit = 2.5% of period

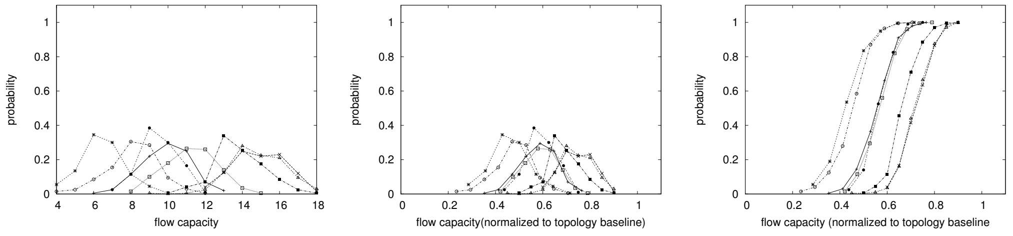


Figure 97: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 30, flow pool = 20, xmit = 4% of period

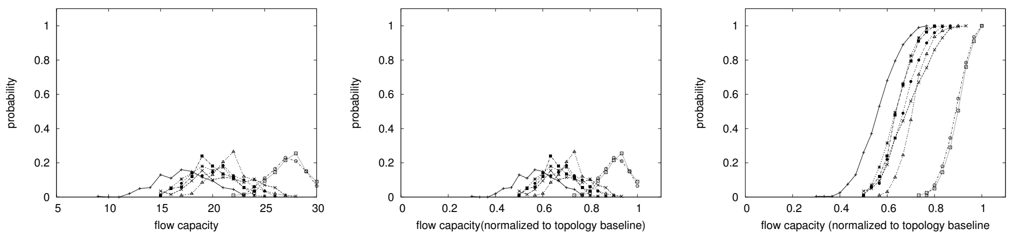


Figure 98: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 30, flow pool = 30, xmit = 1.5% of period

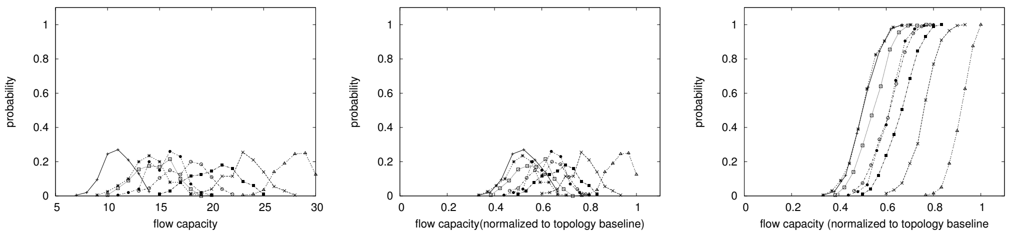


Figure 99: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 30, flow pool = 30, xmit = 2.5% of period

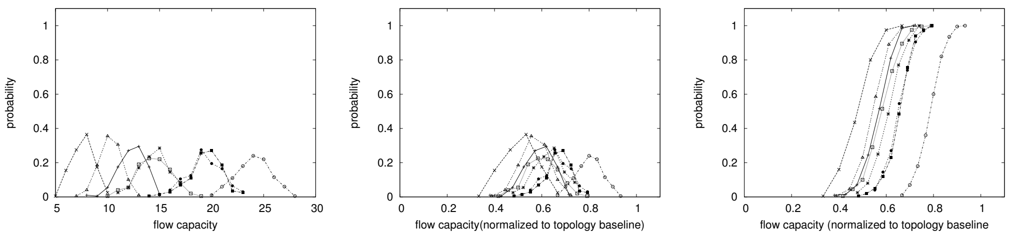


Figure 100: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 30, flow pool = 30, xmit = 4% of period

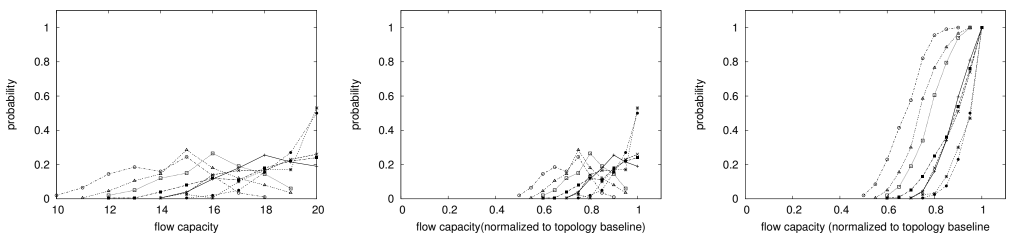


Figure 101: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 40, flow pool = 20, xmit = 1.5% of period

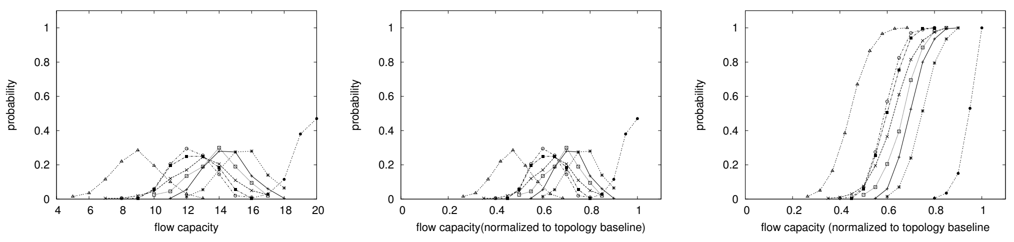


Figure 102: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 40, flow pool = 20, xmit = 2.5% of period

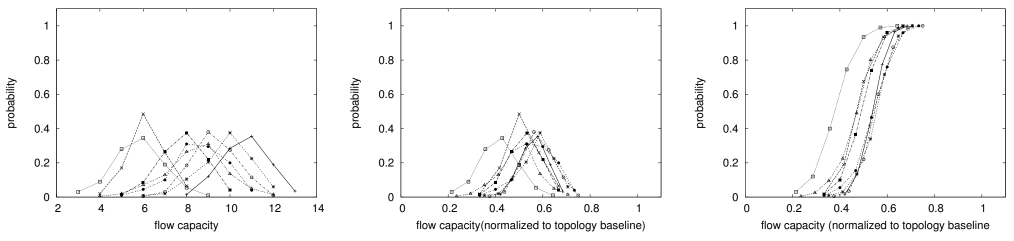


Figure 103: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 40, flow pool = 20, xmit = 4% of period

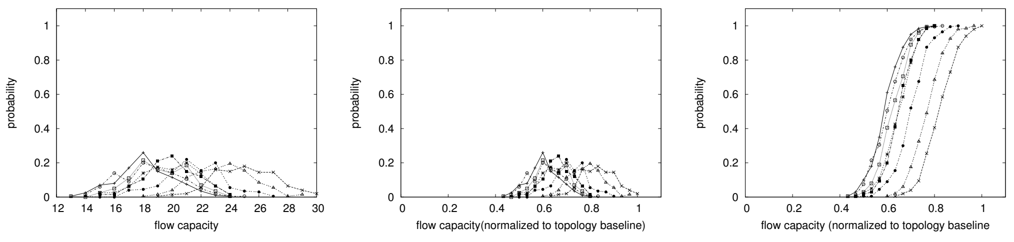


Figure 104: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 40, flow pool = 30, xmit = 1.5% of period

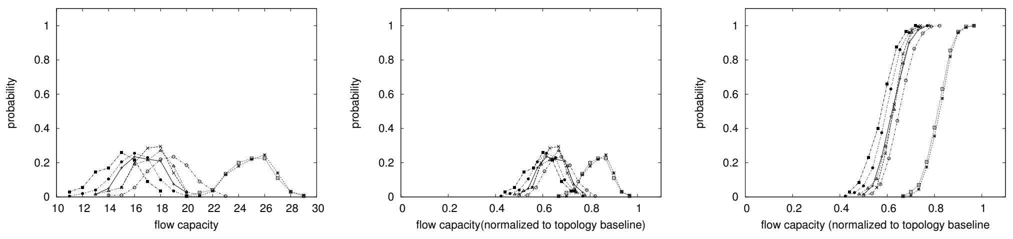


Figure 105: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 40, flow pool = 30, xmit = 2.5% of period

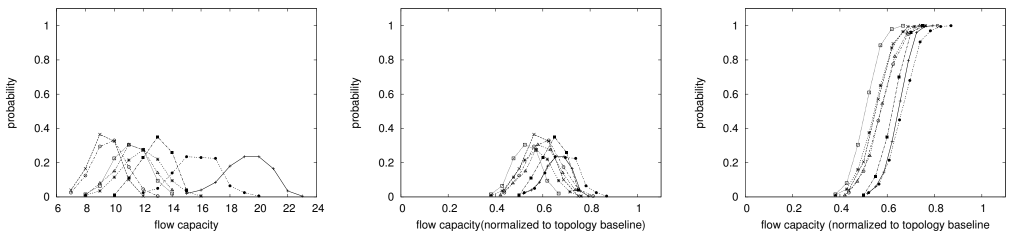


Figure 106: histogram, normalized histogram, normalized cdf; field = .7x10 hops, nodes = 40, flow pool = 30, xmit = 4% of period

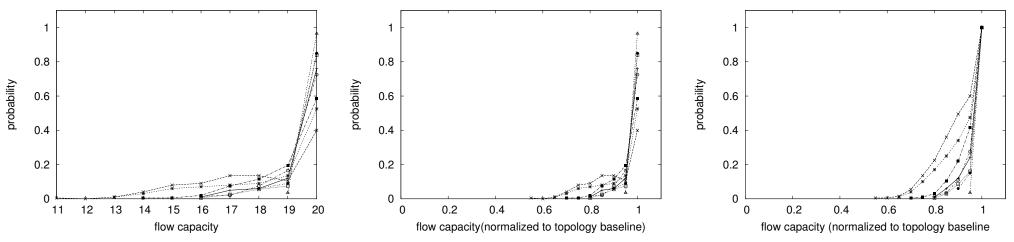


Figure 107: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 20, flow pool = 20, xmit = 1.5% of period

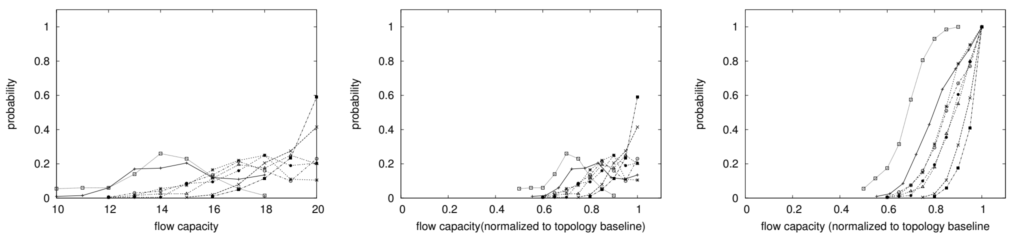


Figure 108: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 20, flow pool = 20, xmit = 2.5% of period

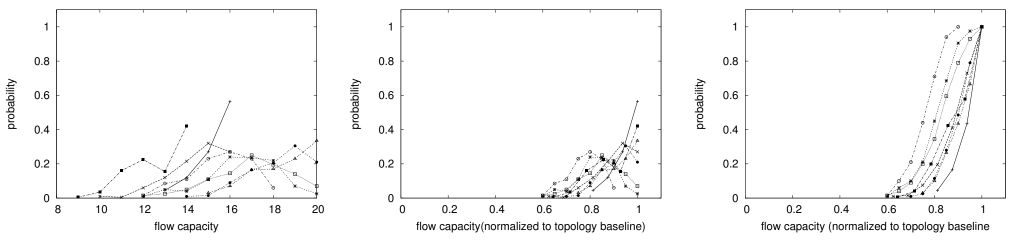


Figure 109: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 20, flow pool = 20, xmit = 4% of period

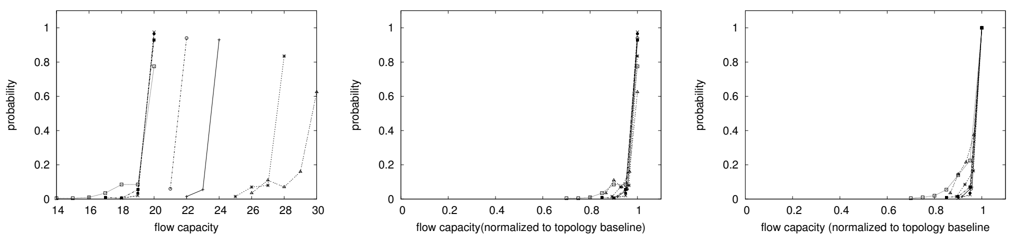


Figure 110: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 20, flow pool = 30, xmit = 1.5% of period

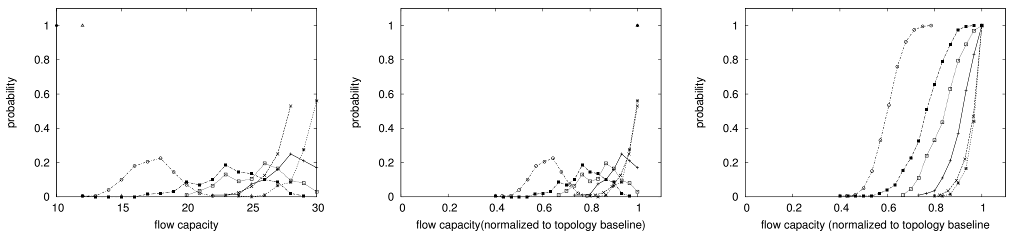


Figure 111: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 20, flow pool = 30, xmit = 2.5% of period

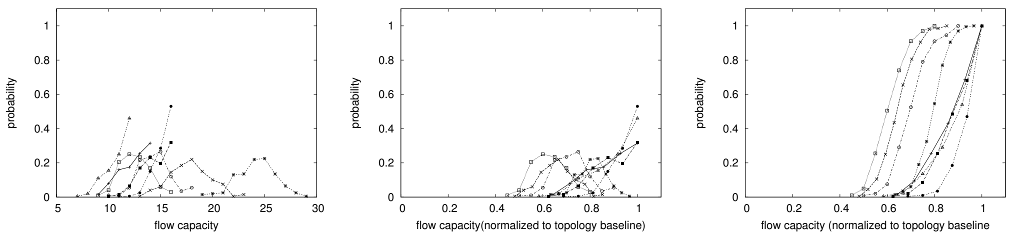


Figure 112: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 20, flow pool = 30, xmit = 4% of period

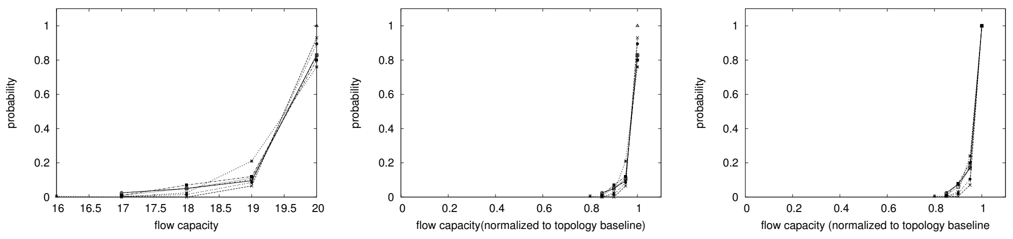


Figure 113: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 30, flow pool = 20, xmit = 1.5% of period

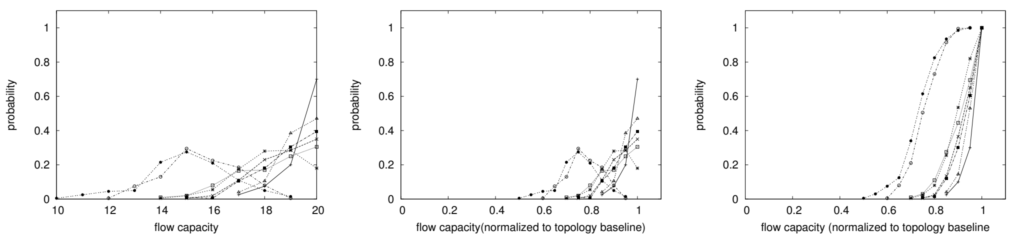


Figure 114: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 30, flow pool = 20, xmit = 2.5% of period

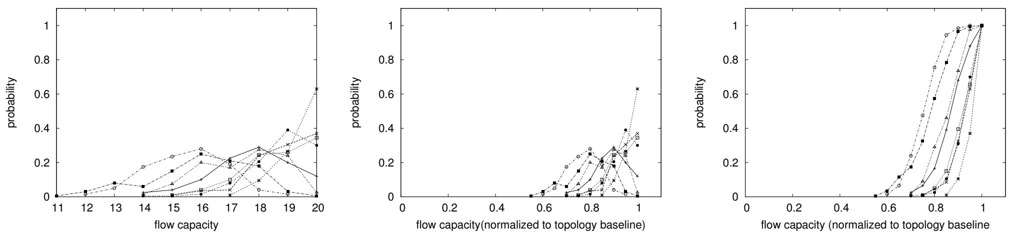


Figure 115: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 30, flow pool = 20, xmit = 4% of period

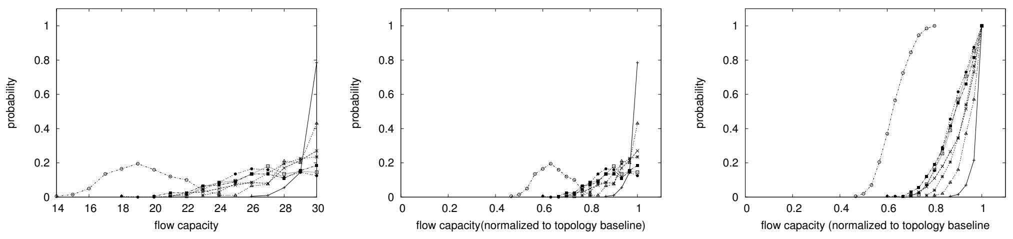


Figure 116: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 30, flow pool = 30, xmit = 1.5% of period

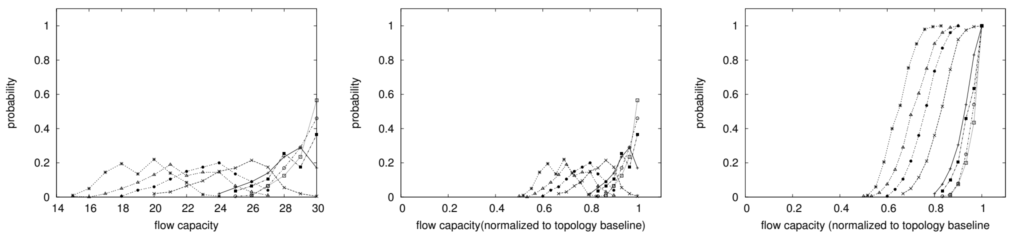


Figure 117: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 30, flow pool = 30, xmit = 2.5% of period

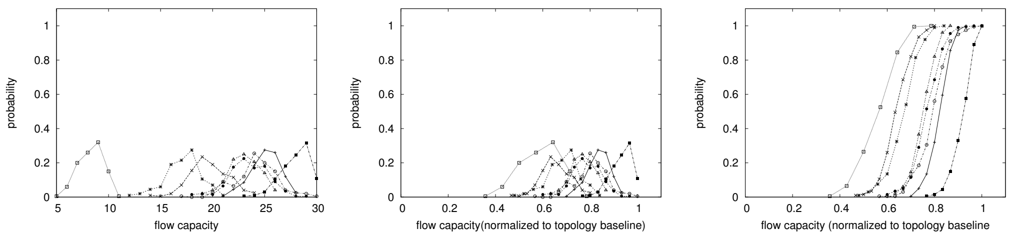


Figure 118: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 30, flow pool = 30, xmit = 4% of period

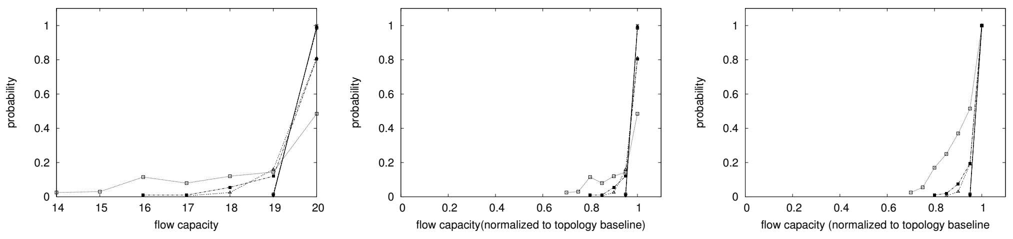


Figure 119: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 40, flow pool = 20, xmit = 1.5 of period

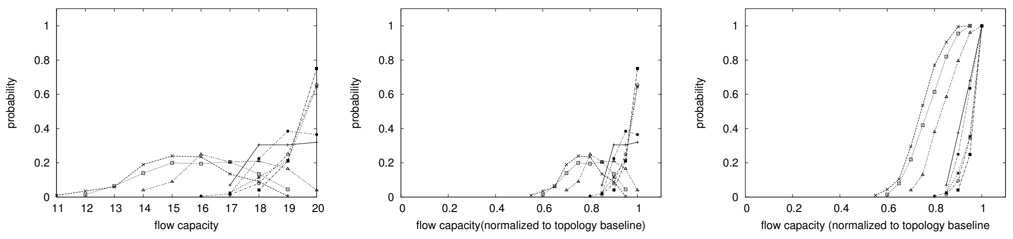


Figure 120: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 40, flow pool = 20, xmit = 2.5% of period

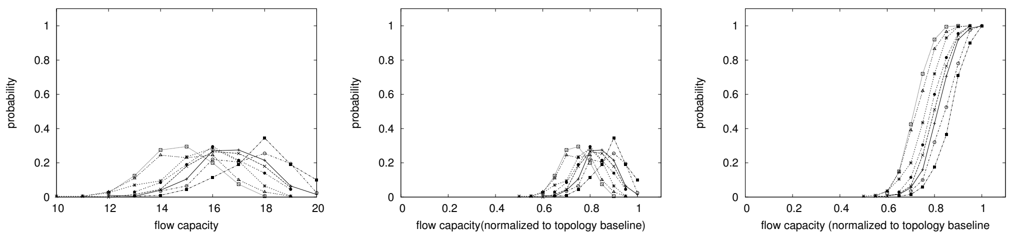


Figure 121: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 40, flow pool = 20, xmit = 4% of period

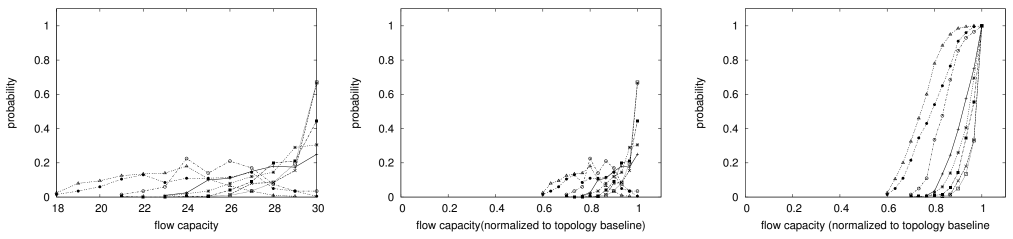


Figure 122: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 40, flow pool = 30, xmit = 1.5 of period

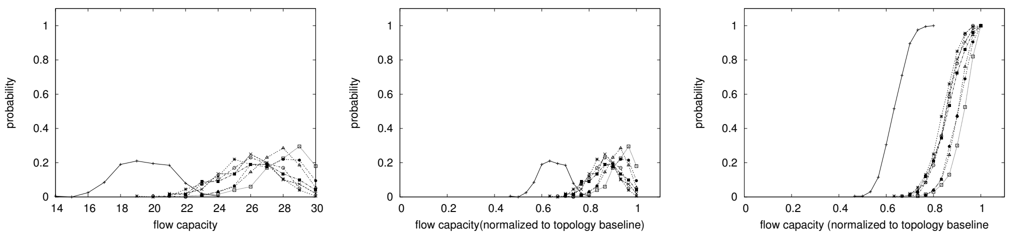


Figure 123: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 40, flow pool = 30, xmit = 2.5% of period

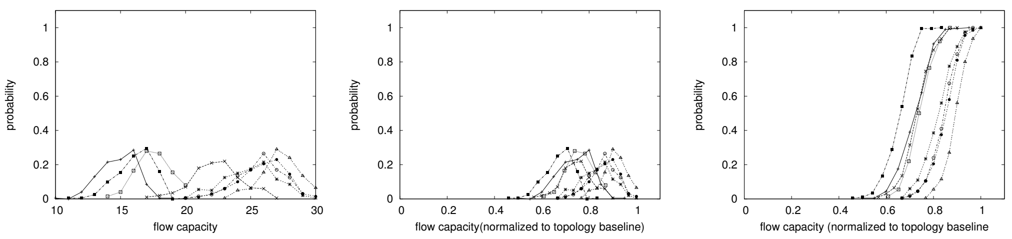


Figure 124: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 40, flow pool = 30, xmit = 4% of period

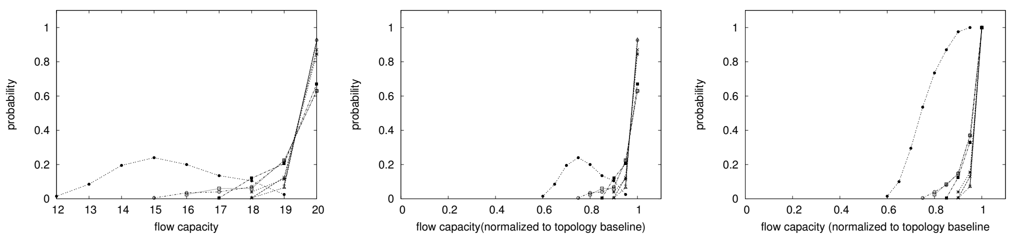


Figure 125: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 50, flow pool = 20, xmit = 1.5% of period

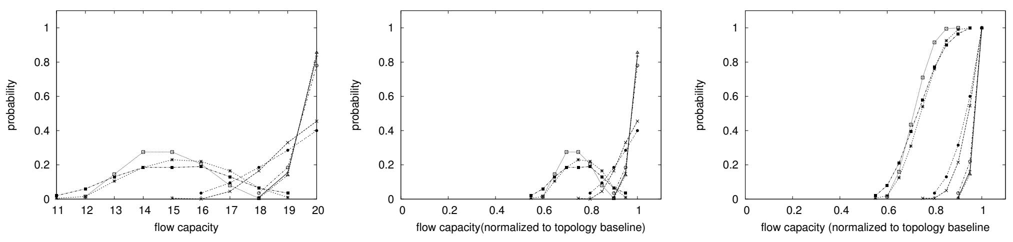


Figure 126: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 50, flow pool = 20, xmit = 2.5% of period

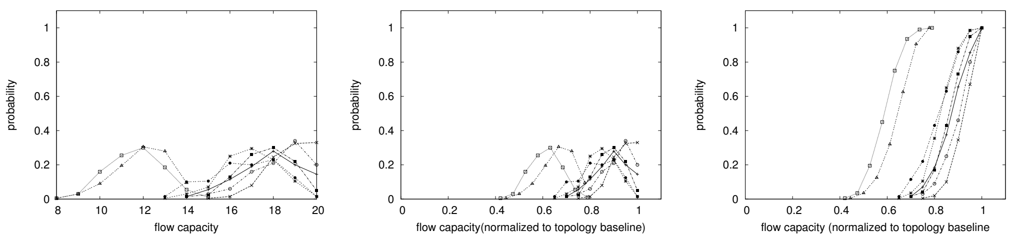


Figure 127: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 50, flow pool = 20, xmit = 4% of period

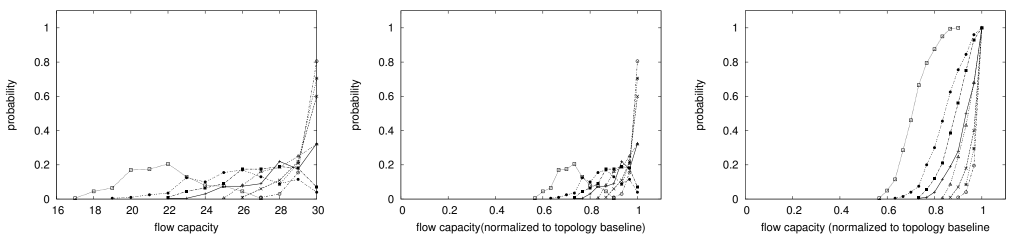


Figure 128: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 50, flow pool = 30, xmit = 1.5% of period

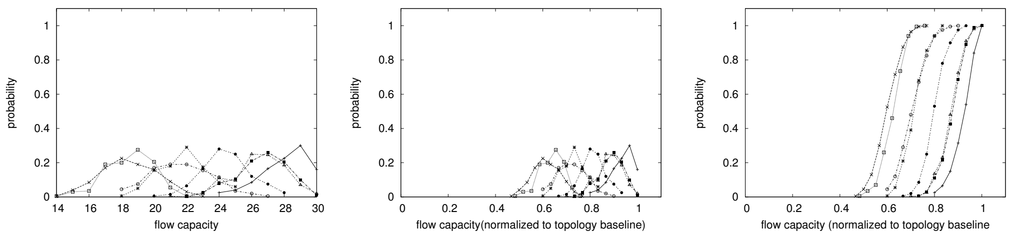


Figure 129: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 50, flow pool = 30, xmit = 2.5% of period

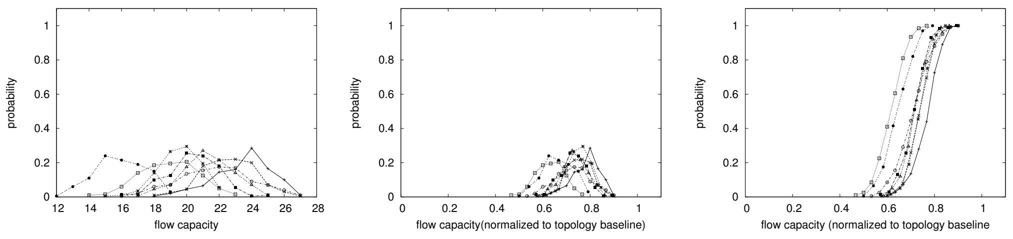


Figure 130: histogram, normalized histogram, normalized cdf; field = 1.5x25 hops, nodes = 50, flow pool = 30, xmit = 4% of period

7.3 Correlation between capacity and mean path length

This section demonstrated the rather suprising phenomena of positive correlation between the flow capacity (number of admitted flows) and their mean path length.

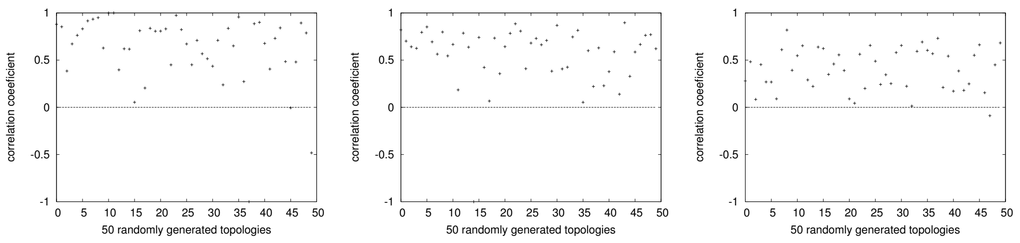


Figure 131: field = 6x6 hops, nodes = 30, flow pool = 20, xmit = 1.5/2.5/4% of period

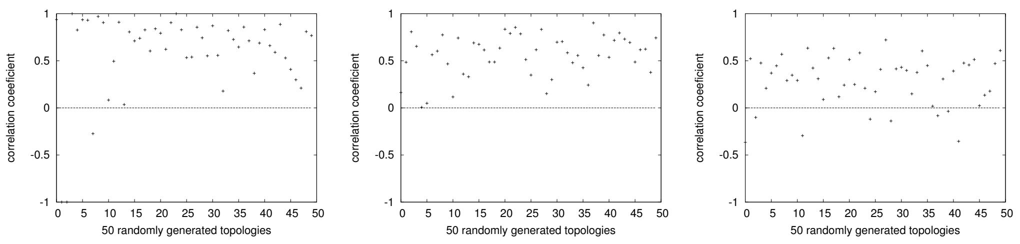


Figure 132: field = 6x6 hops, nodes = 40, flow pool = 20, xmit = 1.5/2.5/4% of period

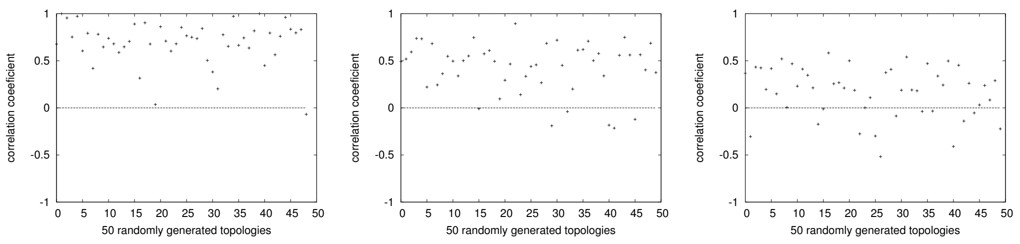


Figure 133: field = 6x6 hops, nodes = 50, flow pool = 20, xmit = 1.5/2.5/4% of period

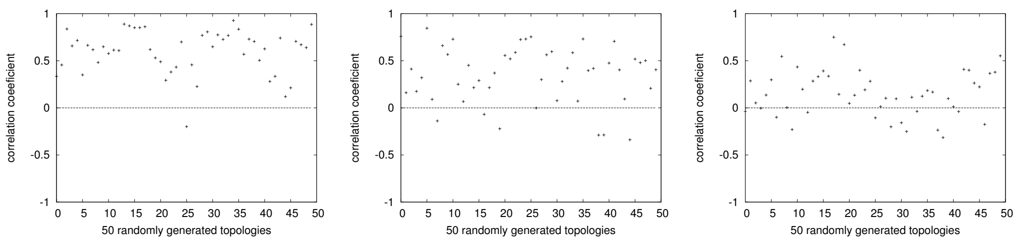


Figure 134: field = 6x6 hops, nodes = 60, flow pool = 20, xmit = 1.5/2.5/4% of period

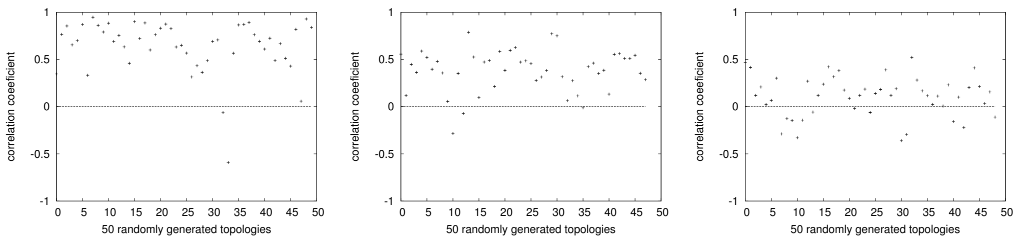


Figure 135: field = 2.5x2.5 hops, nodes = 10, flow pool = 20, xmit = 1.5/2.5/4% of period

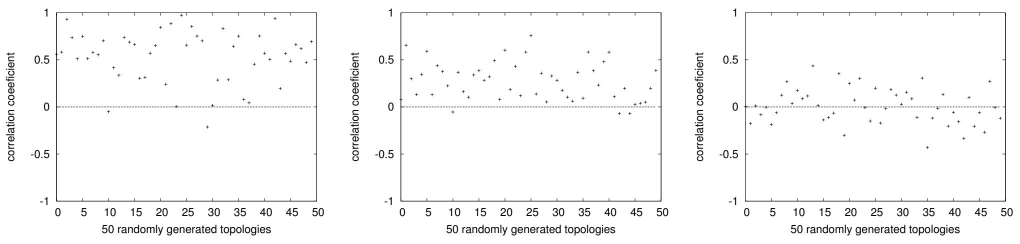


Figure 136: field = 2.5x2.5 hops, nodes = 30, flow pool = 20, xmit = 1.5/2.5/4% of period

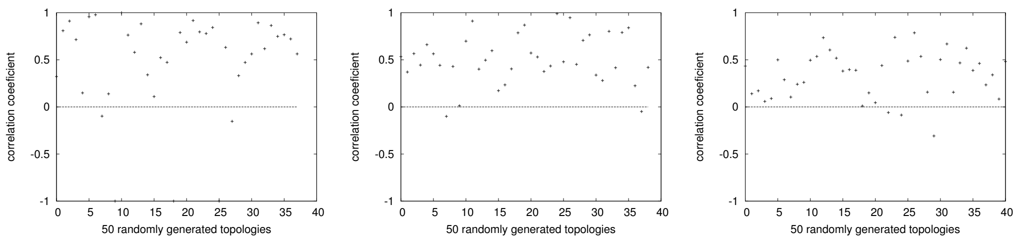


Figure 137: field = 0.7x10 hops, nodes = 10, flow pool = 30, xmit = 1.5/2.5/4% of period

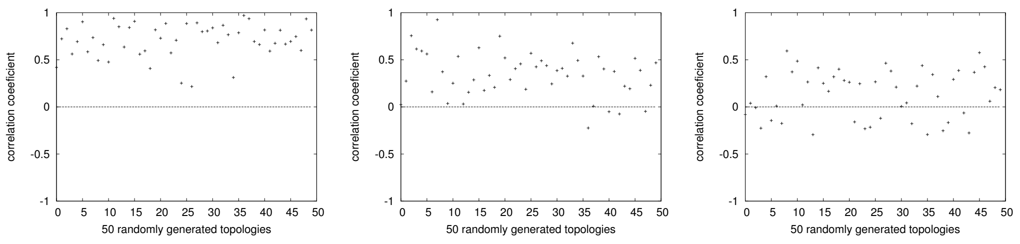


Figure 138: field = 0.7x10 hops, nodes = 30, flow pool = 30, xmit = 1.5/2.5/4% of period

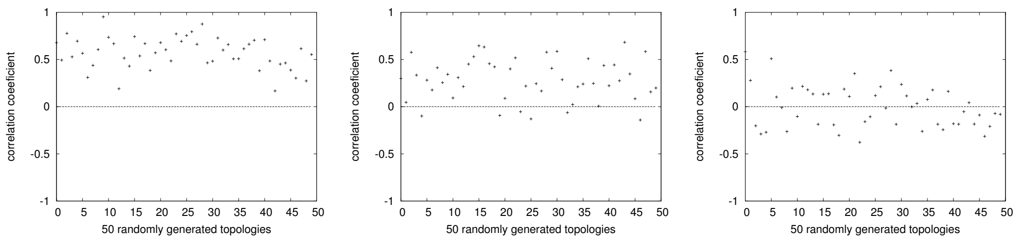


Figure 139: field = 0.7x10 hops, nodes = 50, flow pool = 30, xmit = 1.5/2.5/4% of period

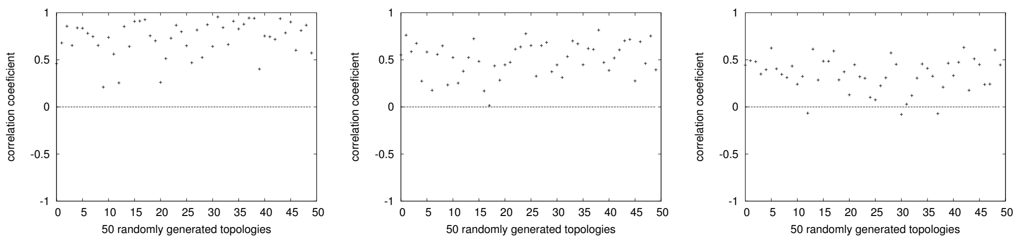


Figure 140: field = 1.5x25 hops, nodes = 40, flow pool = 40, xmit = 1.5/2.5/4% of period

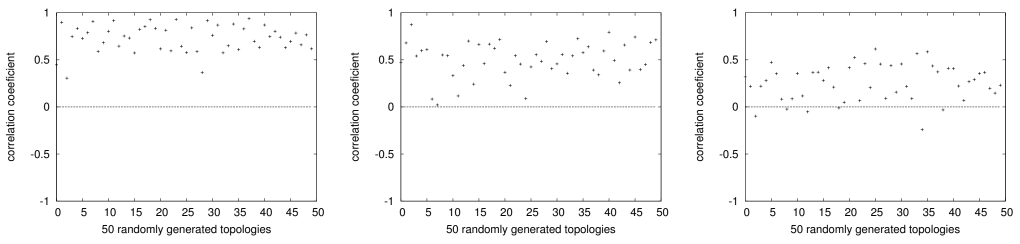


Figure 141: field = 1.5x25 hops, nodes = 60, flow pool = 40, xmit = 1.5/2.5/4% of period

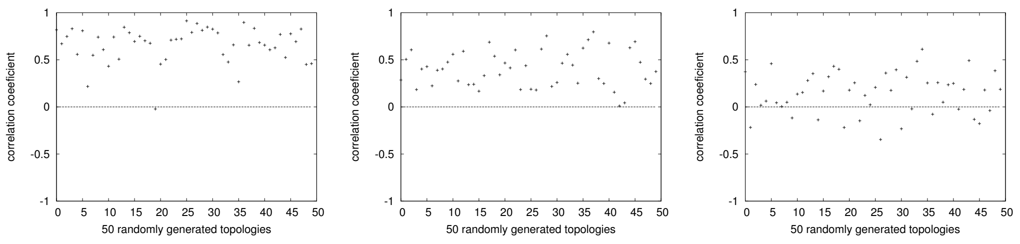


Figure 142: field = 1.5x25 hops, nodes = 80, flow pool = 40, xmit = 1.5/2.5/4% of period

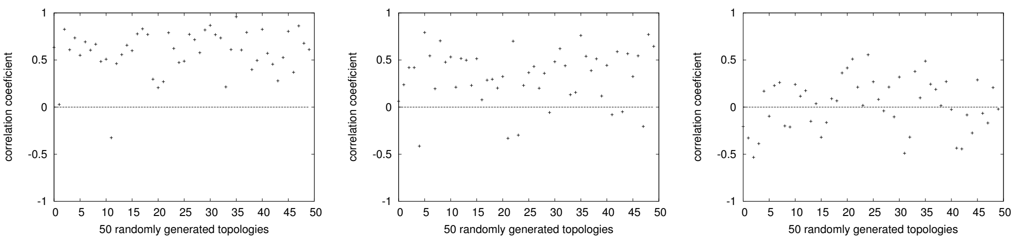


Figure 143: field = 1.5x25 hops, nodes = 100, flow pool = 40, xmit = 1.5/2.5/4% of period

7.4 Quartiles

This section shows the distribution of flow capacity measurements as quartiles. Each quartiles reprsents the variation in flow capacity measurements over one instance of a topology and traffic load. The point refelcts the baseline capacity of the topology (no power save protocol.)

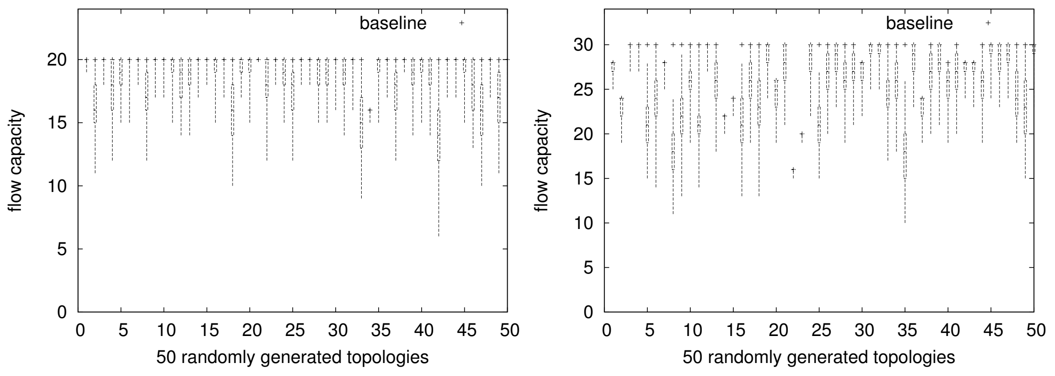


Figure 144: field = 6x6 hops, nodes = 20, flow pool = 20/30 xmit = 1.5% of period

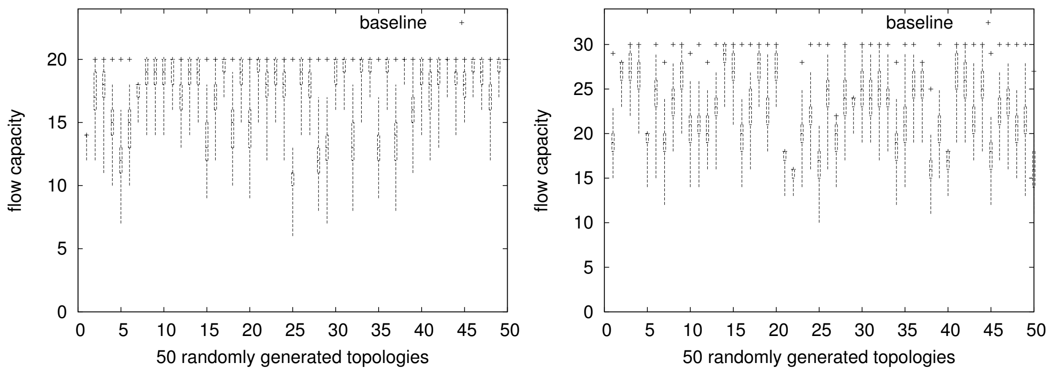


Figure 145: field = 6x6 hops, nodes = 20, flow pool = 20/30 xmit = 2.5% of period

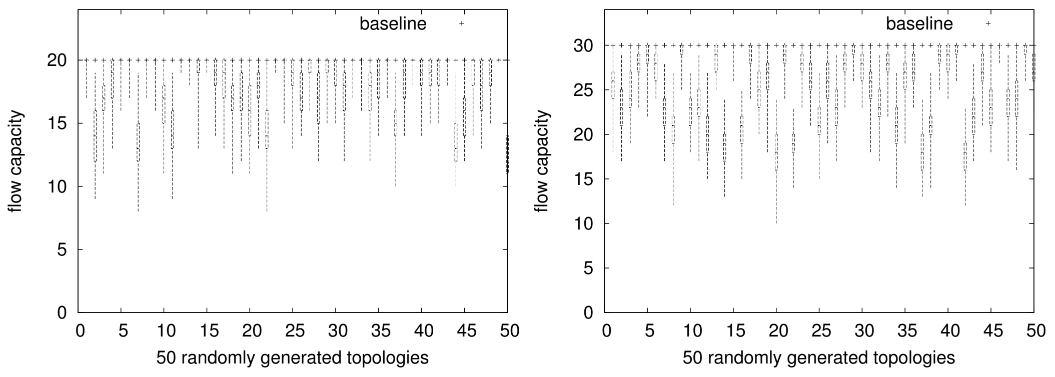


Figure 146: field = 6x6 hops, nodes = 30, flow pool = 20/30 xmit = 1.5% of period

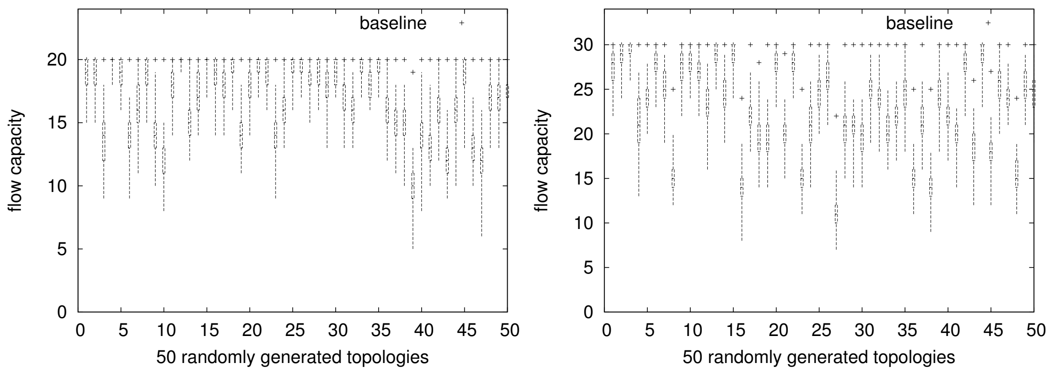


Figure 147: field = 6x6 hops, nodes = 30, flow pool = 20/30 xmit = 2.5% of period

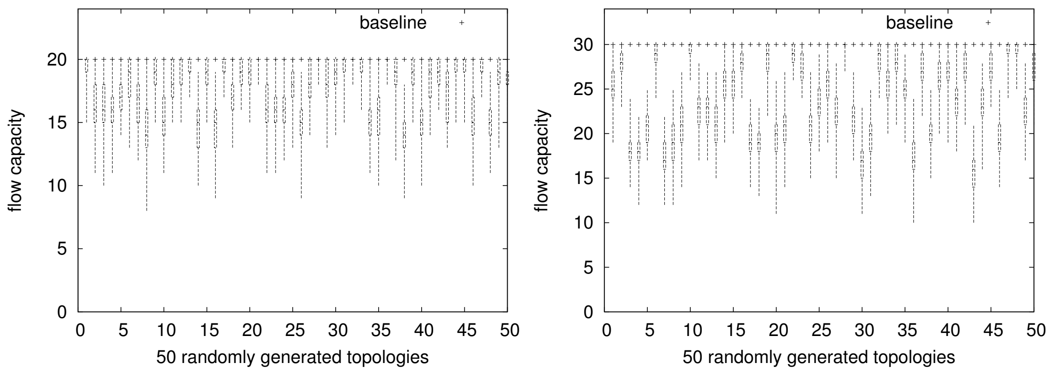


Figure 148: field = 6x6 hops, nodes = 40, flow pool = 20/30 xmit = 1.5% of period

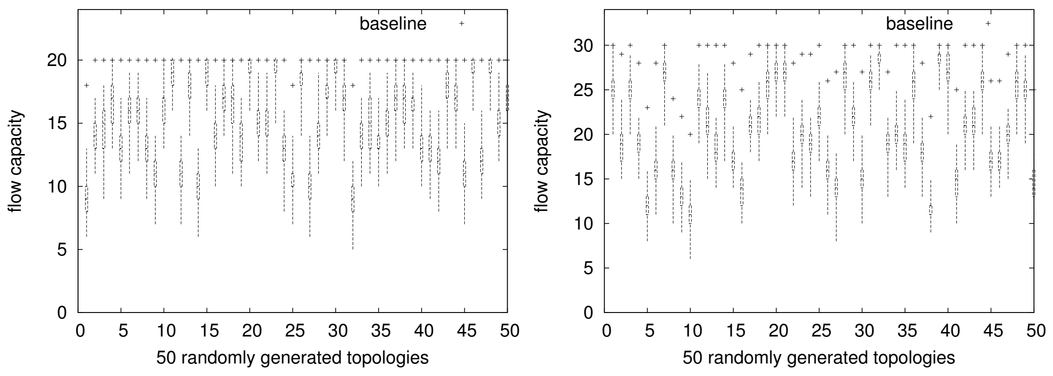


Figure 149: field = 6x6 hops, nodes = 40, flow pool = 20/30 xmit = 2.5% of period

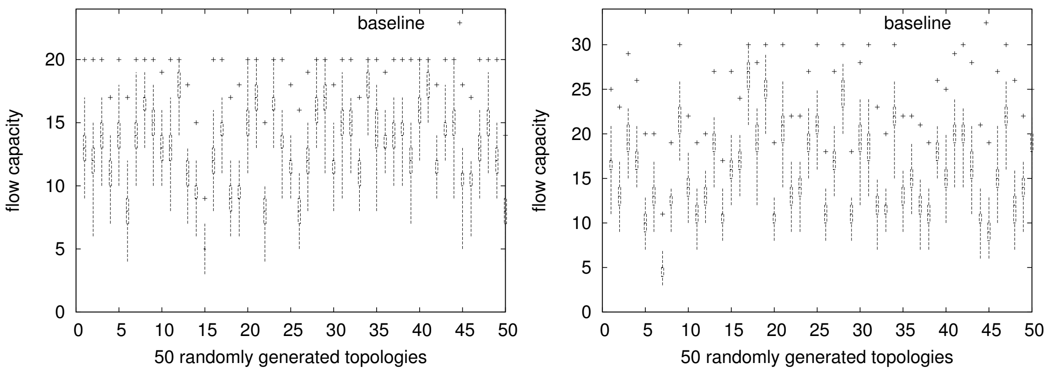


Figure 150: field = 6x6 hops, nodes = 40, flow pool = 20/30 xmit = 4% of period

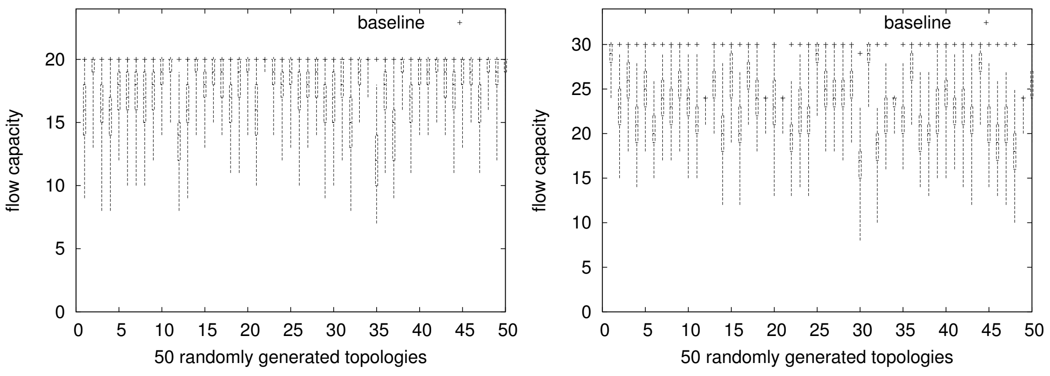


Figure 151: field = 2.5x2.5 hops, nodes = 10, flow pool = 20/30 xmit = 1.5% of period

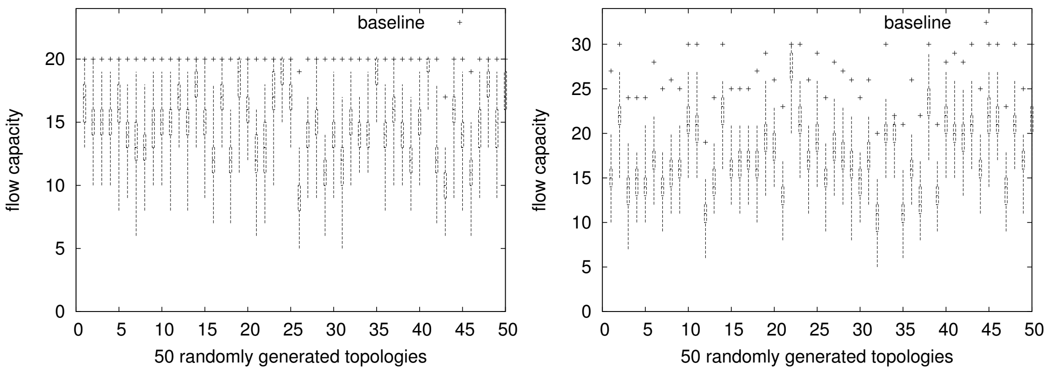
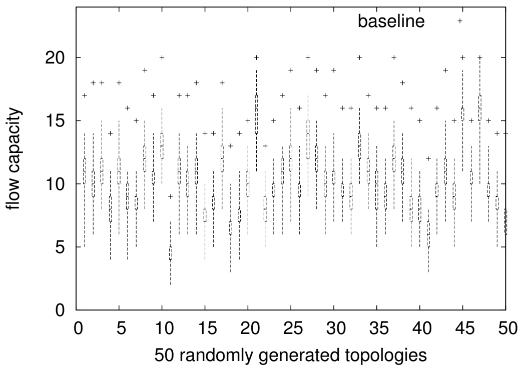


Figure 152: field = 2.5x2.5 hops, nodes = 10, flow pool = 20/30 xmit = 2.5% of period



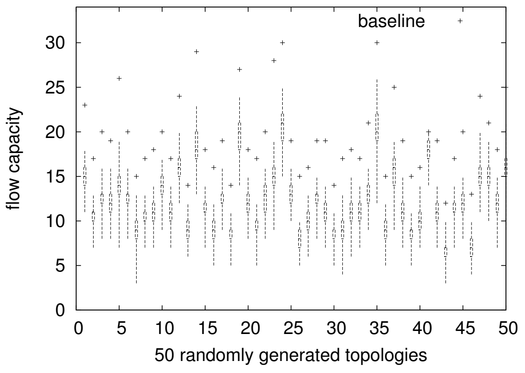
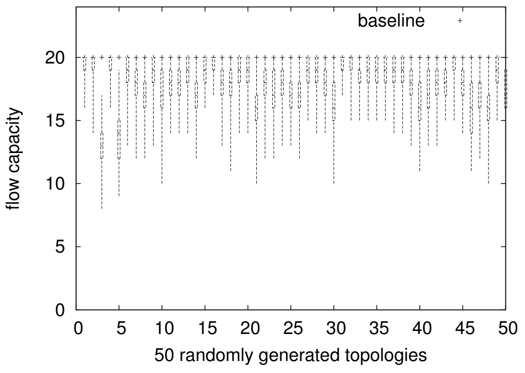


Figure 153: field = 2.5x2.5hops, nodes = 10, flow pool = 20/30 xmit = 4% of period



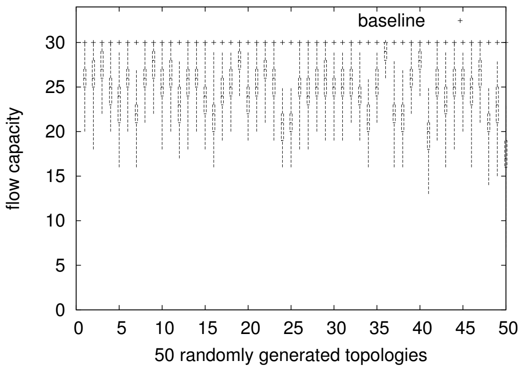
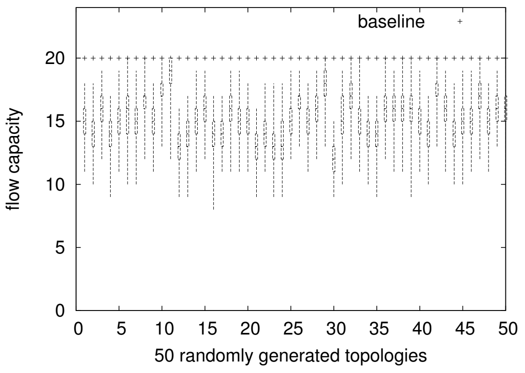


Figure 154: field = 2.5x2.5 hops, nodes = 20, flow pool = 20/30 xmit = 1.5% of period



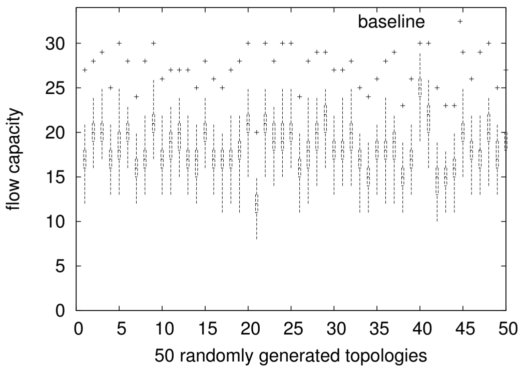


Figure 155: field = 2.5x2.5 hops, nodes = 20, flow pool = 20/30 xmit = 2.5% of period

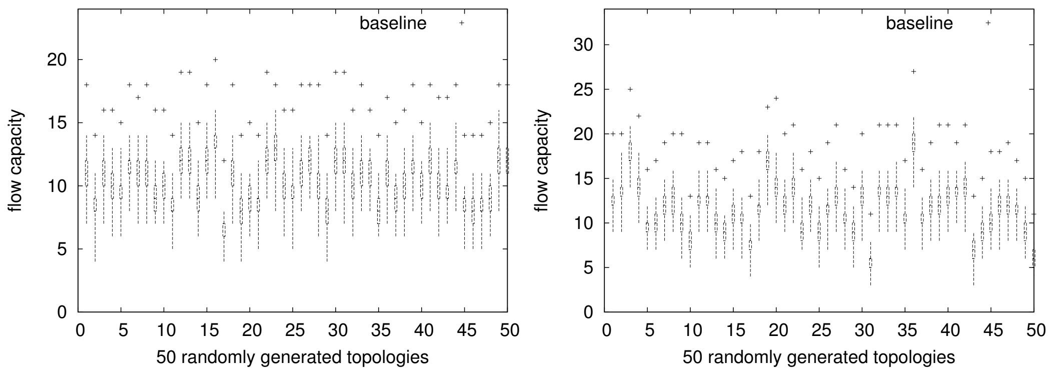


Figure 156: field = 2.5x2.5hops, nodes = 20, flow pool = 20/30 xmit = 4% of period

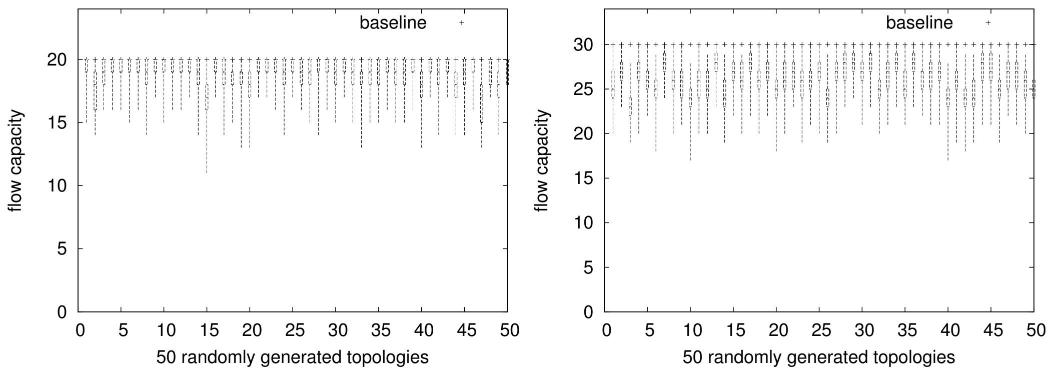


Figure 157: field = 2.5x2.5 hops, nodes = 30, flow pool = 20/30 xmit = 1.5% of period

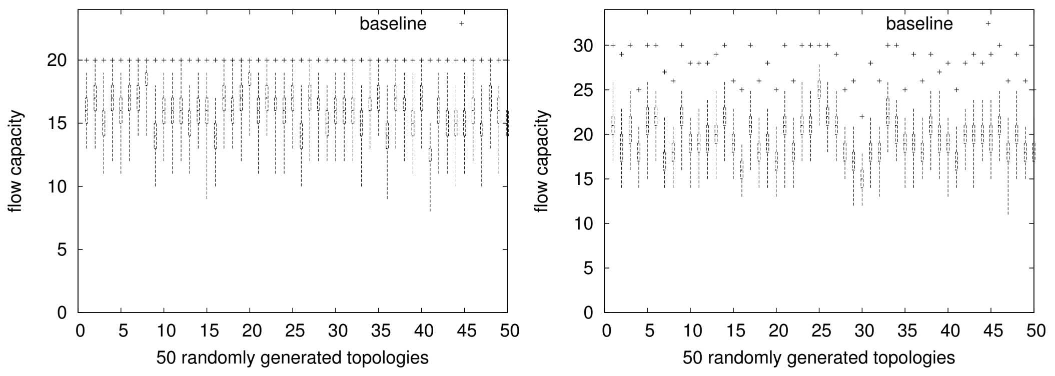


Figure 158: field = 2.5x2.5 hops, nodes = 30, flow pool = 20/30 xmit = 2.5% of period

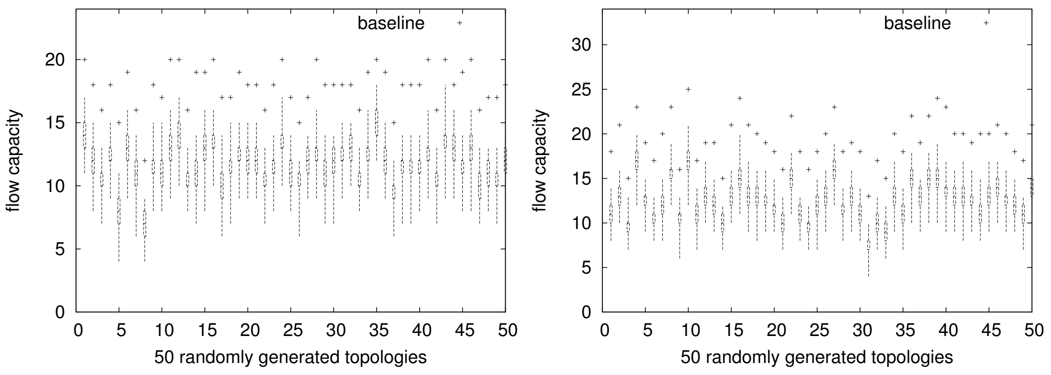


Figure 159: field = 2.5x2.5hops, nodes = 30, flow pool = 20/30 xmit = 4% of period

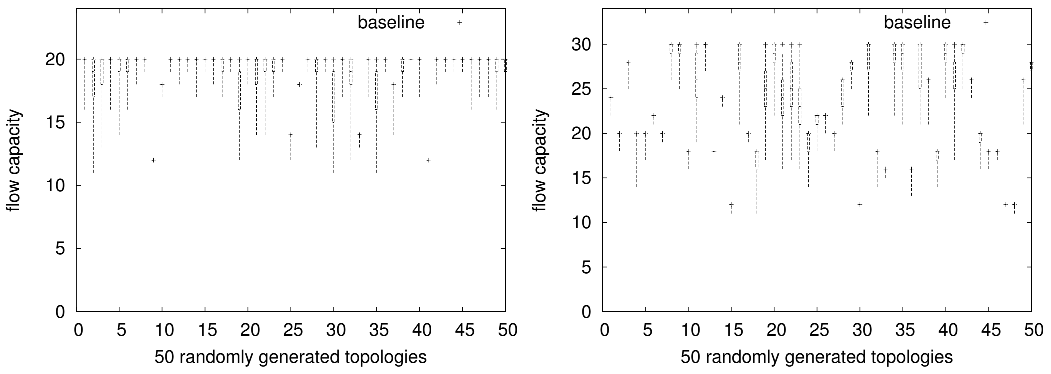


Figure 160: field = 1.5x25 hops, nodes = 20, flow pool = 20/30 xmit = 1.5% of period

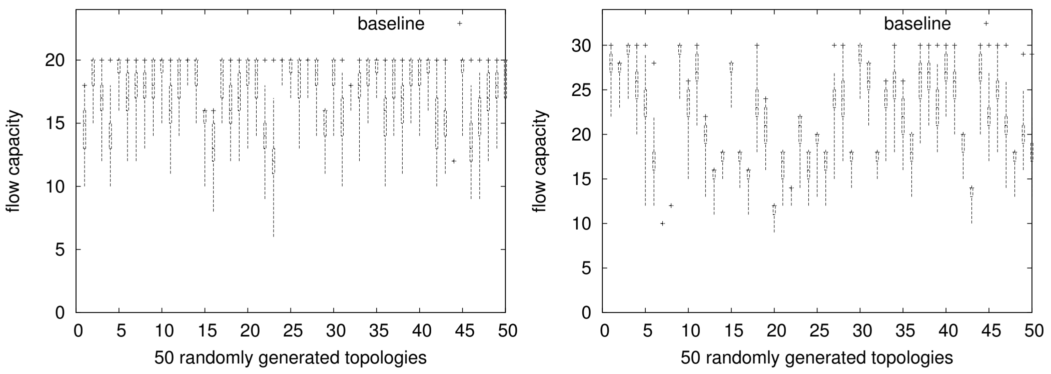


Figure 161: field = 1.5x25 hops, nodes = 20, flow pool = 20/30 xmit = 2.5% of period

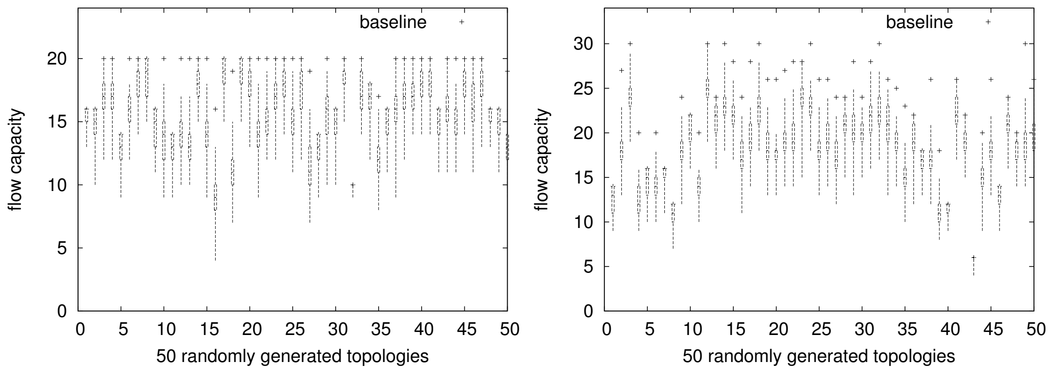


Figure 162: field = 1.5x25hops, nodes = 20, flow pool = 20/30 xmit = 4% of period

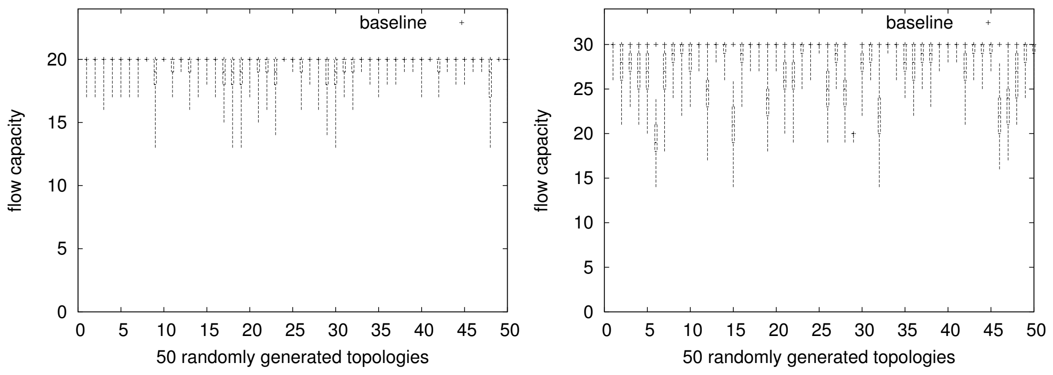


Figure 163: field = 1.5x25 hops, nodes = 30, flow pool = 20/30 xmit = 1.5% of period

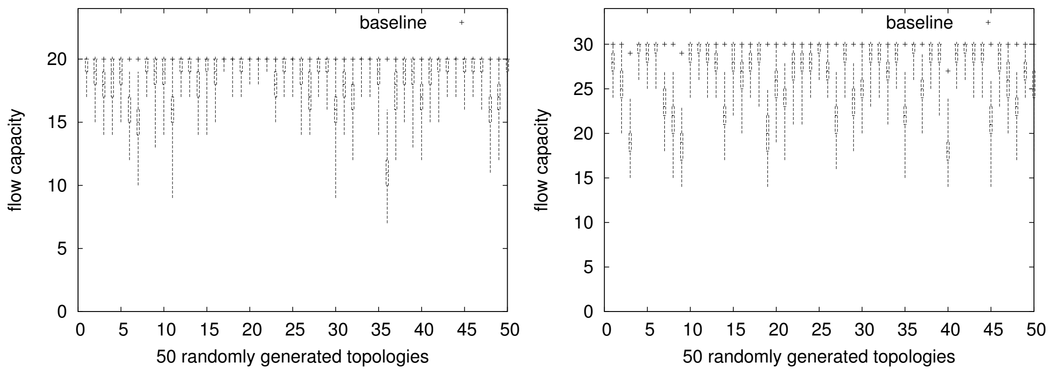


Figure 164: field = 1.5x25 hops, nodes = 30, flow pool = 20/30 xmit = 2.5% of period

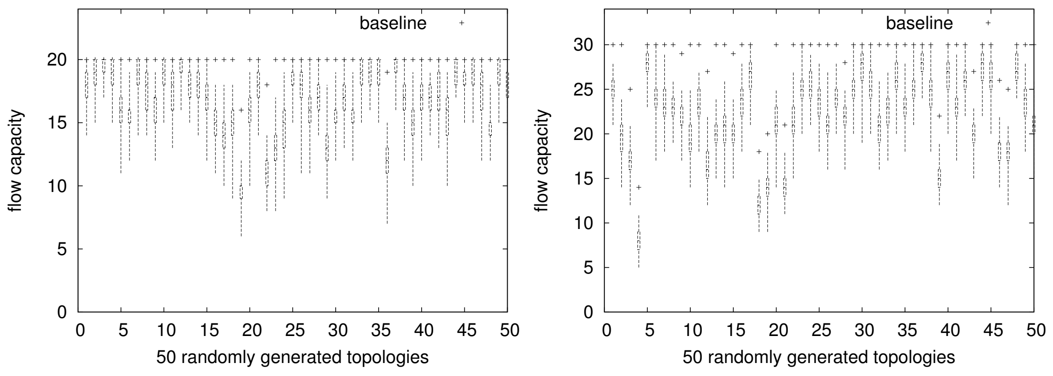


Figure 165: field = 1.5x25hops, nodes = 30, flow pool = 20/30 xmit = 4% of period

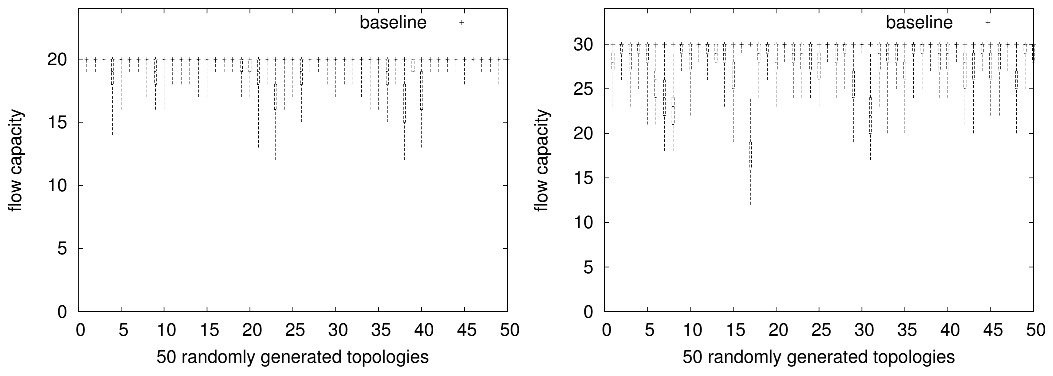


Figure 166: field = 1.5x25 hops, nodes = 40, flow pool = 20/30 xmit = 1.5% of period

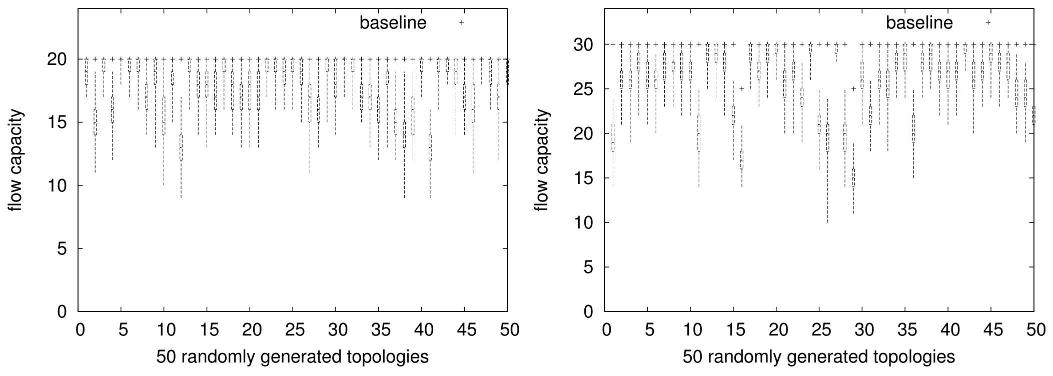


Figure 167: field = 1.5x25 hops, nodes = 40, flow pool = 20/30 xmit = 2.5% of period

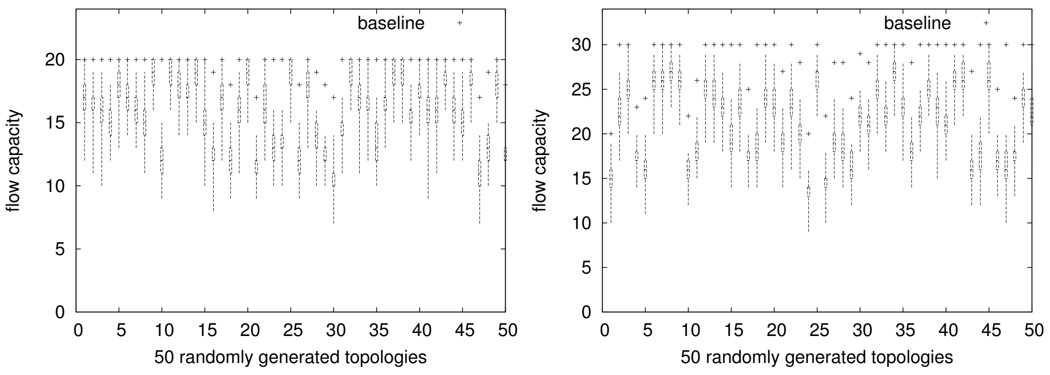


Figure 168: field = 1.5x25 hops, nodes = 40, flow pool = 20/30 xmit = 4% of period

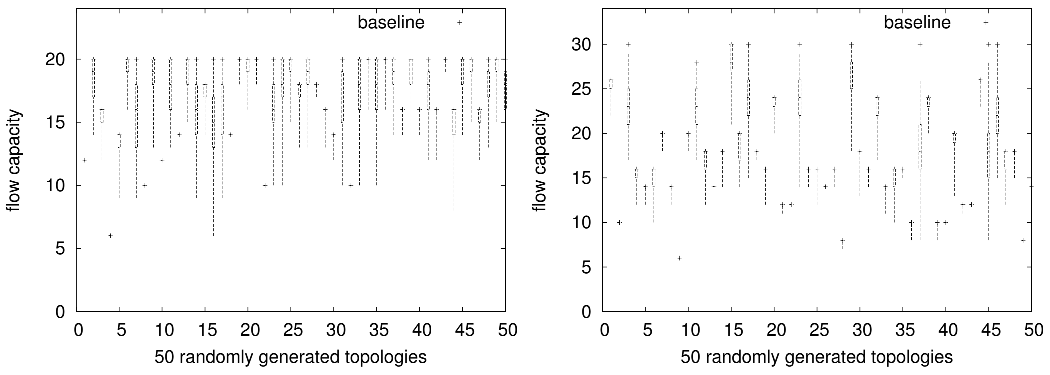


Figure 169: field = .7x10 hops, nodes = 10, flow pool = 20/30 xmit = 1.5% of period

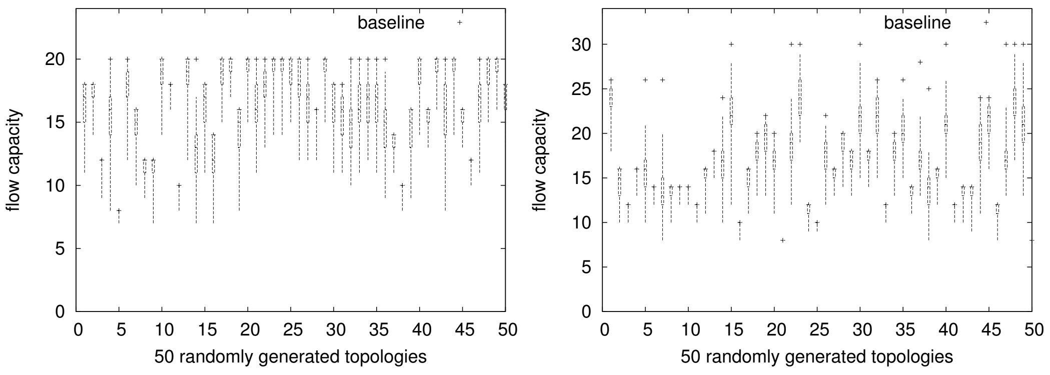


Figure 170: field = .7x10 hops, nodes = 10, flow pool = 20/30 xmit = 2.5% of period

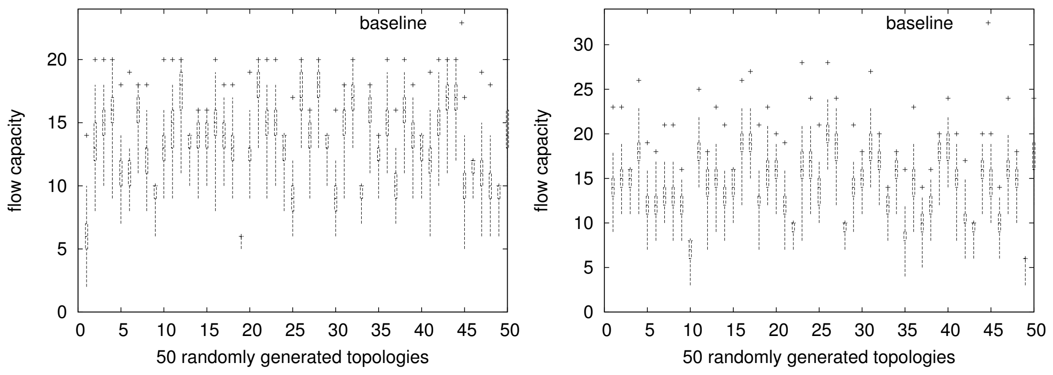


Figure 171: field = .7x10hops, nodes = 10, flow pool = 20/30 xmit = 4% of period

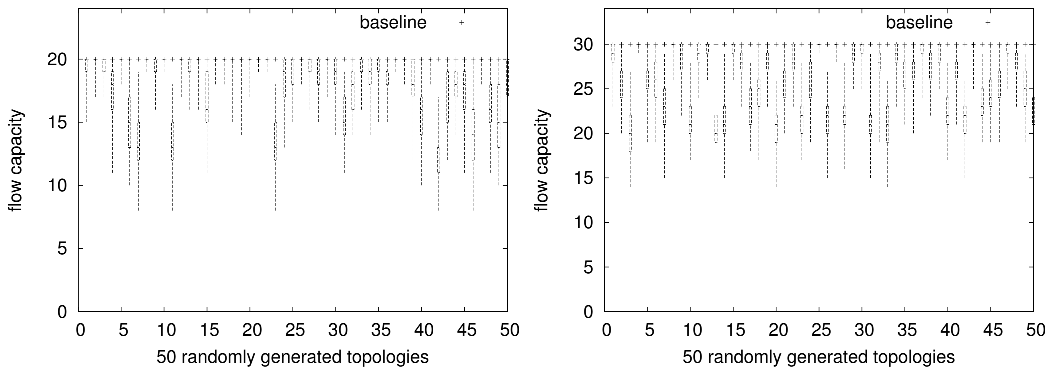


Figure 172: field = .7x10 hops, nodes = 20, flow pool = 20/30 xmit = 1.5% of period

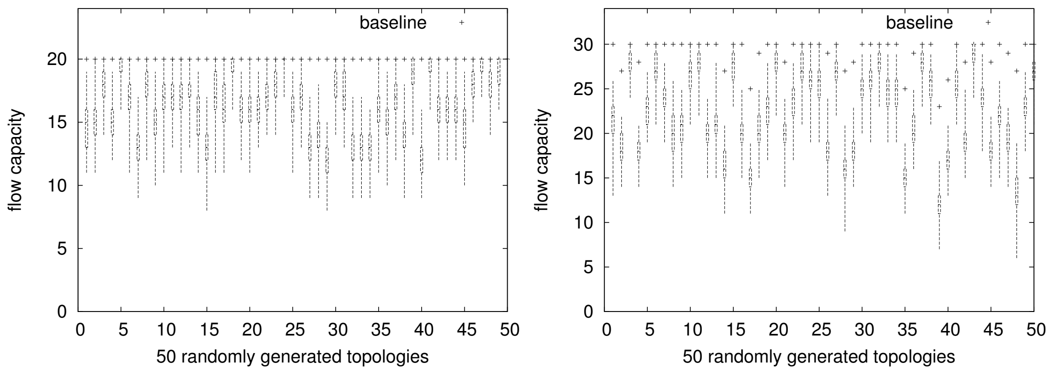


Figure 173: field = .7x10 hops, nodes = 20, flow pool = 20/30 xmit = 2.5% of period

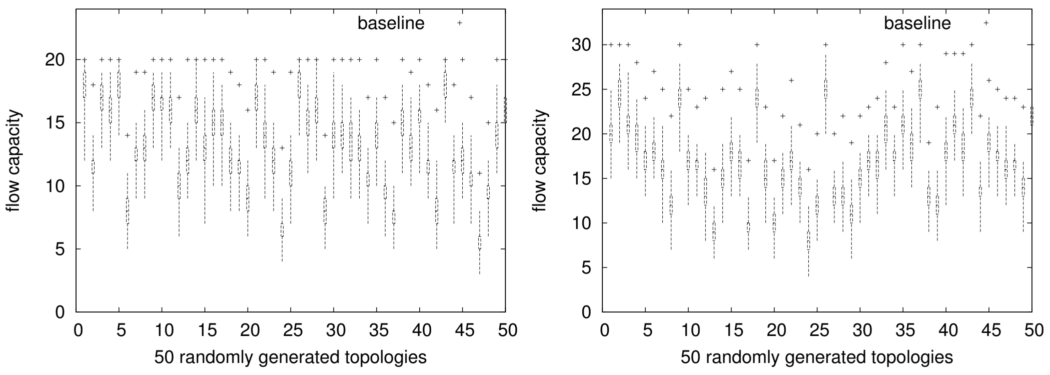


Figure 174: field = .7x10hops, nodes = 20, flow pool = 20/30 xmit = 4% of period

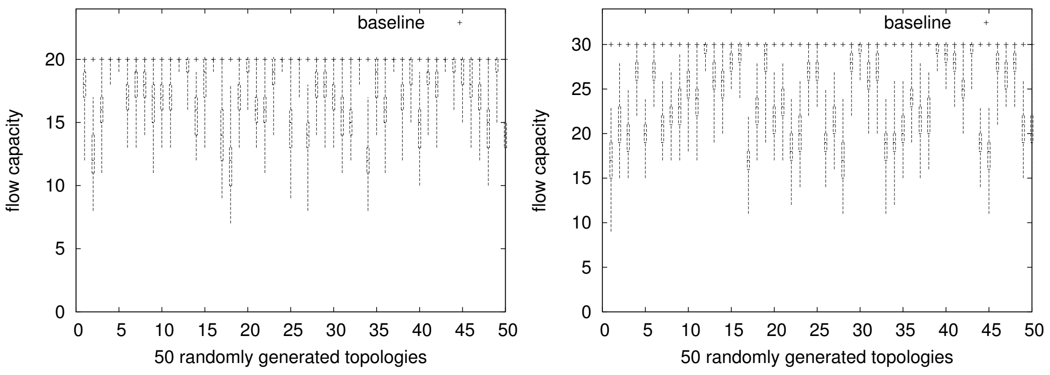


Figure 175: field = .7x10 hops, nodes = 30, flow pool = 20/30 xmit = 1.5% of period

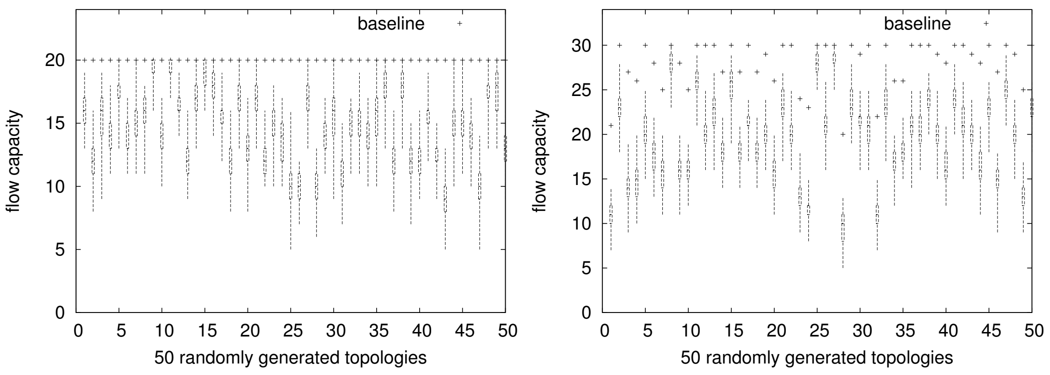


Figure 176: field = .7x10 hops, nodes = 30, flow pool = 20/30 xmit = 2.5% of period

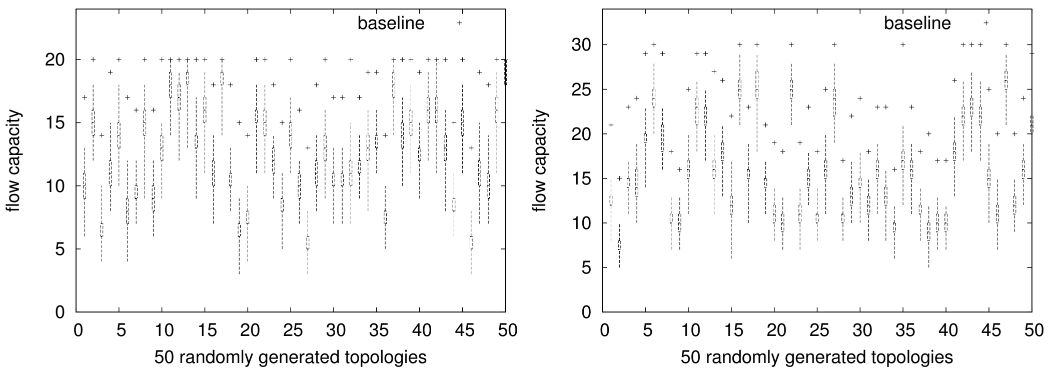


Figure 177: field = .7x10hops, nodes = 30, flow pool = 20/30 xmit = 4% of period

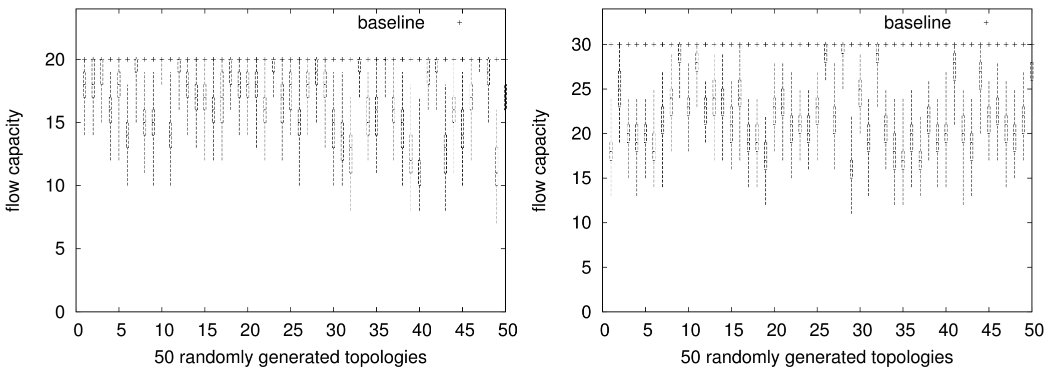


Figure 178: field = .7x10 hops, nodes = 40, flow pool = 20/30 xmit = 1.5% of period

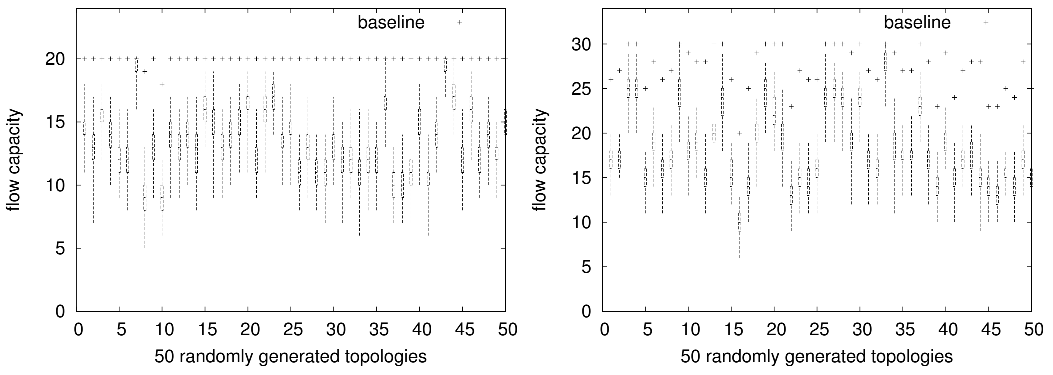
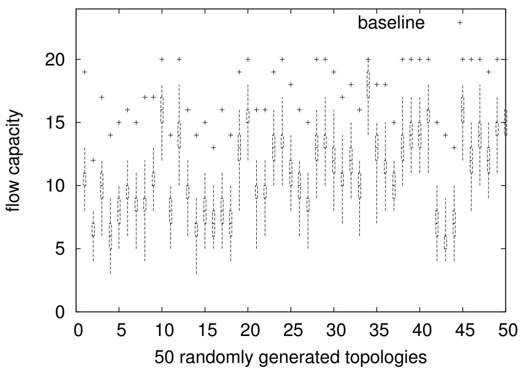


Figure 179: field = .7x10 hops, nodes = 40, flow pool = 20/30 xmit = 2.5% of period



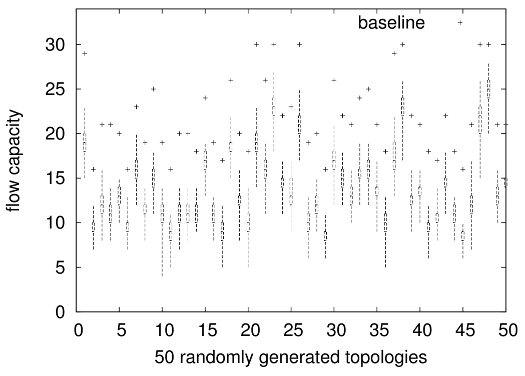


Figure 180: field = .7x10 hops, nodes = 40, flow pool = 20/30 xmit = 4% of period