Distributed Simulation: An Effective Modeling Tool for Large-Scale PCS Networks

Christopher D. Carothers
and
Richard M. Fujimoto †
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332

† **Contact Author:** e-mail `fujimoto@cc.gatech.edu`, office (404) 894-5615, fax (404) 894-9442.

Abstract

Over the past few years there has been rapid growth in the demand for mobile communications that has led to intensive research and development of complex PCS (personal communication service) networks. Capacity planning and performance modeling are necessary to maintain a high quality of service to the PCS subscriber while minimizing costs. The analysis of these complex networks requires flexible, yet time efficient modeling tools. To this end, we demonstrate that distributed simulation is one such modeling tool. In this study, we develop a model for PCS simulations that is suitable for execution on distributed computing platforms. This model is then validated using previously derived analytical results. We show that this validated distributed simulation using 8 workstations can reduce execution time from 20 hours to 3.5 hours. Next, as an example of distributed simulation's flexibility, the model is extended to allow for different call hold time distributions and simultaneous call arrivals. Using this extended model, we examine the effect of different call hold time distributions and bursty call arrivals on the blocking characteristics of PCS networks.

Key words: distributed simulation, Time Warp, PCS network, performance modeling, statistical interferencing, linear modeling.

1 Introduction

A personal communication service (PCS) network [1] is a wireless network that provides communication services for *PCS subscribers*. The service area of a PCS network is partitioned into several sub-areas or *cells*. Each cell is covered by a radio port. A radio port may serve one or more cells. For demonstration purposes, we assume that a port serves exactly one cell. The port locates a subscriber or *portable*, and delivers calls to and from the portable by means of paging within the cell it serves. A *registration area* (RA) consists of an aggregation of cells, forming a contiguous geographical region. To connect a phone call to a portable, it is necessary to identify the portable's RA. The strategies commonly proposed are two-level hierarchies that maintain a system of home and visited databases (*Home Location Registers* or HLR, and *Visitor Location Registers* or VLR). To order PCS services, a PCS subscriber must "enroll" with a particular PCS provider. When enrolling, the PCS subscriber gives the PCS provider the necessary information, such as credit, service type, and current location, to set up the PCS account. This PCS account information is stored in the HLR of the PCS provider. When the PCS subscriber roams to another RA, which is likely to be owned by another PCS provider, the PCS subscriber becomes a "visitor" of that RA. The VLR of this RA is used to store the visiting PCS subscriber's information. Upon registering with the VLR, the VLR notifies the HLR of the visiting PCS subscriber that "your subscriber is at my place".

General models are needed to understand different aspects of large-scale PCS networks, e.g. user location strategies [2, 3, 4], registration strategies [5, 6, 7], hand-off or automatic link transfer strategies [8, 9, 10, 11, 12], and channel allocation strategies [13, 14, 15, 16, 17], so that the network will provide a high quality of service to mobile subscribers while minimizing the resource cost incurred by the PCS provider. Two widely studied approaches to modeling are simulation and analytic techniques. The primary advantage of simulation based techniques is that systems can be modeled in greater detail, giving the system designer a great amount of flexibility. However, this flexibility comes at the price that the simulation model may be too time consuming to use, particularly for models of large-scale PCS networks. On the other hand, analytic techniques offer an alternative solution, but require the development of models that are both accurate (simplifying assumptions regarding the behavior of PCS subscribers are required) and sufficiently simple to be solved. In practice, network designers must have both simulation and analytic models at their disposal to properly evaluate and design PCS networks.

Distributed simulation overcomes the handicap of sequential simulations and can be used in situations where analytic techniques do not suffice. Distributed simulation reduces the execution time by employing multiple cooperating computers to complete a simulation run while maintaining the flexibility and generality of sequential simulation. In this article we demonstrate the effectiveness of distributed simulation as a tool for modeling and analyzing large-scale PCS networks. First, a PCS simulation model is presented that is suitable for execution on a distributed computing platform. Next, this simulation model is validated using previously derived analytical results and the simulation's performance is discussed. As a demonstration of the distributed simulation model's flexibility, we extend this model to use different call hold time distributions as well as allow for simultaneous call arrivals (a.k.a.. bursty call arrivals). Using this extended model, we examine the effect of different call hold time distributions and bursty call arrivals on PCS network performance: a question that is both challenging and of interest to the PCS community.

2 A Distributed PCS Network Simulation Model

Because of limited computing capacity of sequential simulation techniques, simulation of PCS networks are typically limited to small-scale networks containing fewer than tens of cells [15, 17, 18], and output statistics of the cells at the edge of the grid are usually discarded to avoid boundary effects. Lin and Mak [19] showed that this approach may lead to biased output statistics and suggest using a large network with a wrap-around topology to achieve reliable simulation results. A large network is still needed when using a wrap-around topology to avoid wrapping effects. *Wrapping effects* occur when the network of cells is small, resulting in an event at a cell rippling back to the cell through chains of adjacent cells, which can lead to distorted statistical performance measurements.

To illustrate the impact of network size, consider the following PCS simulation.

PLACE FIGURE 1 ABOUT HERE.

The service area is partitioned into S square cells. The expected number of portables in a cell is N. A portable resides in a cell for a period of time t (we assume t to be exponentially distributed with mean $\frac{1}{Mob}$, as proposed in [20]), then move to one of the four neighboring cells with equal probability. The call arrivals to a portable form a Poisson process with the arrival rate λ. The call holding times have an exponential distribution with mean $\frac{1}{\mu}$. Figure 1 (a) and (b) illustrate the impact of the network size S (where the simulation covers $T = 5 \times 10^4$ seconds of simulated time, the mean call holding time is $\frac{1}{\mu} = 180$ seconds, $N = 50$ and $Mob = 0.04\mu$) as a function of total call incompletion probability, p_{nc} [1].

The figures suggest that $S > 256$ cells are required to avoid the wrapping effects. Figure 1 (c) suggests that the simulation begins to converge on the steady state blocking probability at 5×10^4 (simulated)seconds, but complete convergence is not realized until 2.1×10^5 (simulated) seconds. For $S > 256$ and $T > 5 \times 10^4$ seconds, the number of events executed is on the order of 10^7. We have observed between 10^3 and 10^4 events per second in commercial discrete event simulation packages, such as ModSim [21] on a high performance engineering workstation, so a simulation of this magnitude will require many hours of CPU time. Moreover, execution times will increase as network complexity and the functionality of the simulator increase.

Here, we examine the use of a distributed computing platform to reduce model execution time. The remainder of this section focuses on our model for simulating large-scale PCS networks. We discuss the synchronization problem that arises in executing discrete event simulations on a collection of networked computers, and a distributed simulation protocol called *Time Warp* which is used to address this problem. We then present performance measurements in using Time Warp to simulate a large-scale PCS network.

[1] p_{nc} includes new calls as well as hand-off calls that are blocked due to a lack of available channels at the destination cell.

2.1 PCS Network Simulation Model

The PCS simulation model that is used here is organized around two key objects: *cells* and *portables*. The cell represents a service area covered by a port which has some number of channels allocated to it. The portable represents a mobile phone unit that resides within the Cell for a period of time and then moves to one of the four neighboring Cells. The behavior of a portable can be modeled by events such as portable move, call arrival, and call completion.

The PCS model views a call arrival as a call which can potentially consume a channel in a port. The call is either originated at or destined for a portable in the cell. For demonstration purposes, we assume that the other party of the call is not a portable. Whether calls are originations or terminations make no difference to this PCS simulation model since the model does not simulate the signaling networks. If the PCS model is extended to include end to end signaling, the modifications to the current PCS model are minor. Since call origination and termination is treated the same, only the additional signaling model needs to be verified. As shown in Figure 2, when a new call arrives at a cell, the cell first determines the channel's availability. If all channels are busy, this call is counted as a *blocked call*. If a channel is available, it is allocated for the destination portable's use and the call is allowed to connect. This model assumes that a portable is available to accept an arriving call and is not *busy*. *Busy-lines* reduce the total call incompletion probability in a PCS network since they do not count as blocked call attempts. Accordingly, our model predicts the worst case performance of a PCS network for a particular call load.

While a call is in progress, the portable tracks its location. When the portable determines it is moving out of the current cell's signal range, it drops the currently used channel, and requests a channel from the nearest neighbor cell into which the portable is moving. If all channels from the neighboring destination cell are busy, this call is *forced terminated*. If a channel is available, the call reconnects and continues without interruption.

PLACE FIGURE 2 ABOUT HERE.

2.2 Distributed Simulation Technique

The PCS distributed simulation consists of a collection of *objects* , each modeling a distinct cell or portable in the PCS network. The distributed, Time Warp Simulator consists of a collection of *logical process (LP) objects* that communicate by exchanging *message objects*. Each cell in the PCS simulation is mapped to a distinct LP object, and each portable to a message object. Each LP consists of a state vector that stores the private data, and a set of methods that define the allowable operations that can be performed on that data. Each method corresponds to a type of event. For instance, in the PCS simulation, a cell LP will include methods invoked by receipt of a portable message object from another cell LP to enter or leave that cell. Here cell events (methods) denote portable departures, call arrivals, or call completions. Each event contains a timestamp that represents the simulation time when the event occurs and a message object (portable) that determines the type of the event. For example, if a portable moves from cell A to cell B at time 20, then the LP modeling cell A invokes the portable departure method with timestamp 20 at LP B. Since the LPs may simulate the corresponding cells at the different speeds, a synchronization mechanism is required to ensure that each LP processes events in timestamp order to prevent events in the future from affecting those in the past. For example, suppose that a portable moves from cell A to cell B at time 30, and a portable moves from cell C to B at time 40. It is possible that LP C invokes the portable departure method earlier than LP A does. The synchronization mechanism ensures that these two messages (which are arrival events to LP B) are executed by LP B in increasing timestamp order.

A well-known synchronization mechanism is the Time Warp protocol which uses a detection-and-recovery protocol to synchronize the computation [22]. Any time an LP determines that it has processed events out of timestamp order, it rolls back those events, and reexecutes them. Rolling back an incorrect event computation requires one to (1) undo any changes the incorrect computation made to the state of the LP, and (2) "un-send" any messages that it sent. The first task is accomplished by periodically saving the state of the LP, and restoring it when a rollback occurs. Un-sending previously sent messages is accomplished by sending an *anti-message* for each message that is to be cancelled. When received, the anti-message deletes (annihilates) the prior message. If that message had already been processed, the LP is first rolled back (possibly generating additional anti-messages) prior to annihilation. Recursively applying this "roll back and send anti-message" cycle will eventually erase all effects of the "unsent" message.

In order to reclaim memory (e.g., processed messages and snapshots of the LP's state), and to allow operations that cannot be rolled back (e.g., I/O), *global virtual time (GVT)* is defined. GVT is a lower bound of the timestamp of any rollback that might later occur, and is defined as the smallest timestamp of any unprocessed or partially processed message or anti-message. Irrevocable operations occurring at simulated times older than GVT can be performed, and storage carrying a timestamp older than GVT can be reclaimed.

A version of Time Warp has been developed that executes on a collection of DEC 5000 workstations, Sun Sparc workstations, or a mixture of these machines [23]. The Time Warp system is written in C++. A principal objective of this implementation is to enable efficient simulation of thousands of "light weight" simulator objects (LPs and message objects) that contain a small amount of state and perform little computation in each event in an object-oriented environment on networked, heterogeneous workstations. Here we will briefly describe some of the features of the distributed Time Warp system. It is anticipated that most simulations will contain far more LPs than processors, so each processor will typically contain hundreds or thousands of LPs. A priority queue data structure called the calendar queue [24] is used in each processor to efficiently select the next LP to execute. The processor's scheduler always selects the LP containing the smallest timestamped event as the next one for execution. Each LP includes a linear list to hold the unprocessed events (method invocations) scheduled for that LP. It is anticipated that the number of unprocessed events for each LP will remain small (often only a few events) for the PCS simulation.

2.3 Validation of the Simulation

The simulation model is validated using an analytical model that makes the following assumptions and uses the following notation.

- The call arrivals to each portable are a Poisson process.
- The portable residual times in a coverage area are exponentially distributed with the mean $1/Mob$ as suggested in [20].
- The call holding time is exponentially distributed with mean $1/\mu$.
- p_f is the forced termination probability (the probability that a hand-off call is blocked).
- p_o is the probability that a new call is blocked.
- p_{nc} is the call incompletion probability. This probability includes both new call blocks and hand-off calls that are forced to terminate. An incomplete call is either blocked as a new call or it may make several successful hand-offs before it is forced terminated.

The call incompletion probability p_{nc} for general portable residual time distribution was derived in [9]. p_{nc} for the exponential residual time distribution is as follows.

$$p_{nc} = \frac{\mu p_o + Mob * p_f}{\mu + Mob * p_f} \tag{1}$$

PLACE FIGURE 3 ABOUT HERE.

We validated the simulation by comparing p_{nc} produced by the simulation to p_{nc} produced using Equation 1 for varying call loads and mobilities. The results for this comparison is shown in Figure 3. In all cases, we see that the simulation tracks the values produced by Equation 1.

2.4 Performance Results

PLACE FIGURE 4 ABOUT HERE.

PLACE FIGURE 5 ABOUT HERE.

We conducted a series of PCS simulation experiments to measure speedup where the number of portables per cell, mobility and call inter-arrival times were varied. Speedup is defined as the execution time of the sequential simulation divided by the execution time of the distributed simulation. All performance results presented here were performed on DEC 5000 machines interconnected by an Ethernet. In the following experiments, the simulated call holding times, call inter-arrival times, and portable residual times were exponentially distributed, and the number of portables per cell, N, was 2.5 times of the number of channels per cell, c. All experiments were conducted on 8 workstations, each containing 128 cells, for a total of 1024 cells, and ran for 2.5×10^5 simulated seconds. For the same input parameters, speedup was calculated as the the best sequential execution time divided the best distributed execution time.

To give a fair measure of speedup, we used a sequential simulator that has the same calendar queue data structure for managing the pending set of events, but does not have the state saving, rollback and fossil collection overheads associated with the distributed simulator.

Speedup results using 8 processors are shown by Figure 4. Each data points represents the average of six runs. Depending on the parameters of the simulation, speedups range from 5.8 to 6.5. It is observed that speedup does not vary when either mobility or the call inter-arrival time is changed. These results are attributed to the number of remote messages sent between processors not varying across the different PCS parameters. As shown in Figure 5, the number of remote messages is below 1% for all PCS parameters and the variance is sufficiently small across different mobility and call inter-arrival times that the execution time of the simulation does not vary. We also observed a slight increase in speedup as the call inter-arrival time is decreased. As the call inter-arrival time is deceased, the amount of local work available to a processor is increased. Because calls are generated by the locally occupied cell, they result in messages whose sender LP and receiver LP are the same. By decreasing the call inter-arrival time, calls occur more frequently in simulated time, which results in more local work available to a processor. As the processor becomes more heavily loaded with work, the rate of progress through simulated time slows. By slowing the simulation's rate of progress, messages sent between processors have a better chance of not arriving late and causing the destination LP to rollback, thus reducing the obtainable speedup. For a detailed performance analysis of PCS models executing on a Time Warp simulator, we refer the reader to [25].

This study demonstrates that distributed simulation techniques can effectively speed up the execution of PCS simulation models. With 8 workstations, the execution times are reduced from 20 hours to less than 3.5 hours in many cases. Because of the high degree of locality exhibited by the PCS simulation model, we anticipate that performance will scale with the addition of more workstations.

3 Application of PCS Model

In this section we show the effectiveness of distributed simulation as an analysis tool by examining the effect of different call hold time distributions and bursty call arrivals on PCS performance. This problem was chosen because (i) PCS networks are likely to be exposed to these types of traffic patterns and it is currently not known what effect they will have on PCS network performance based on traffic patterns already observed in line based networks [26], (ii) this problem is extremely difficult to study using analytic techniques, and (iii) the computing requirements are quite large, because of the large size of the network. Accordingly, this problem is an excellent candidate to attack using the distributed simulation techniques discussed in this article.

To perform this study, it was necessary to make minor modifications to the PCS simulation which was previously described. First, we had to allow for the generation of multiple calls that would arrive at the same time for a given cell. In terms of code changes, this required adding a simple for-loop in the call generation event. The only other modification was making the simulation allow for different call hold time distributions. These changes took less than a half-hour including compile time and the simulation maintained its validity.

3.1 Results

To determine the effect of different call hold time distributions and bursty call arrivals on PCS performance, a series of experiments was performed where the (i) offered load λ, (ii) mobility Mob, (iii) burst level BL, and (iv) call hold time distribution were varied. The call hold time distributions were: exponential, Pareto, and uniform. These distributions were chosen because their characteristics represent a wide range statistical distributions. In particular, the Pareto has a *heavy tail* [2], and the exponential and uniform distributions have relatively short tails. In each experiment, the number of channels per cell is fixed at 8, the number of cells is fixed at 1024, the call hold time mean was fixed at 180 seconds (3 minutes), and call arrival and mobility rates are exponentially distributed. Each experiment executes for 1.0×10^5 simulated seconds. For the initial Pareto experiments, α was set to 100.0, since this value resulted in the worst case PCS blocking statistics. α's effect on PCS blocking statistics will be discussed later. Bursty call arrivals are modeled using a Poisson process where each burst contains K call arrivals and the arrival rate for this process is $1/K$ times the rate of the non-bursty Poisson call arrival process, making the total number of call arrivals the same for the bursty and non-bursty call arrival processes.

PLACE FIGURE 6 ABOUT HERE.

In Figure 6, Mob is fixed at 0.04μ and BL is fixed at 1 arrival per burst (so effectively there are no bursty arrivals). As shown in Figures 6(a),(b) and (c), changes in the call hold time distribution do not cause a change in p_f, p_o, or p_{nc}. We believe this behavior has to do with the relationship between the length of a distribution's tail and its mode. As the tail becomes longer, the mode is smaller. Consequently, only a few channels are occupied at any one time because call arrivals are sufficiently far apart, allowing incoming calls to connect, resulting in identical call blocking characteristics across all the different call hold-time distributions.

PLACE FIGURE 7 ABOUT HERE.

PLACE FIGURE 8 ABOUT HERE.

[2] A probability distribution function F is called *heavy tailed* if $1 - F(x)$ decays like $x^{-\alpha}$ as $x \to \infty (\alpha > 0)$ [26].

However, when bursty call arrivals are introduced into the network, a different picture beings to emerge. In Figure 7(a), there is 300% increase in p_f when the call hold time distribution is varied from exponential to Pareto ($BL = 8$). We initially thought that this behavior is caused by differences in the tails of these two distributions in the face of bursty arrivals. The Pareto has a much heavier tail than the exponential making some calls last many times longer than the mean resulting in calls that can be handed-off multiple times. Accordingly, the probably of a call being forced terminated is much more likely in the Pareto than in the exponential, especially during periods of bursty call arrivals where channel availability is decreased. However, based on that argument, one would expect that the exponential to yield higher blocking statistics than the uniform, since the exponential has a heavier tail than the uniform. This does not appear to be the case. It is observed that the uniform distribution results in higher p_f values than the exponential distribution. We attribute these findings to the variance in the call hold-time distributions. As shown in Figure 8(a), the exponential distribution exhibits a large amount of variance resulting in 63% of it's values *below* the mean. Because such a large percentage of the distributions values are below the mean, the probability that a call will make multiple hand-offs is reduced, which reduces the probability a call will be force terminated. We see that the uniform distribution (Figure 8(c)) has even less variance and that 50% of it's values are below the mean, which indicates that a call will be more likely to be force terminated since it has an increased probability of experiencing multiple hand-offs. The Pareto distribution, shown in Figure 8(c), has almost zero variance ($\alpha = 100.0$) and all values fall extremely close to the mean of the distribution. Accordingly, a larger percentage of the calls can experience multiple hand-offs, which results in the higher p_f statistics for the Pareto distribution.

For each call hold-time distribution, p_f exhibits nonlinear behavior, especially in the Pareto case. These spikes are attributed to the bursty call arrivals reducing the number of connected calls that will handoff at certain BLs which in turn reduces p_f. This phenomenon is still under investigation.

Figures 7(b) and (c) show p_o and p_{nc} as a function of the burst level for each of the different call hold-time distributions. As the burst level is increased from 1 to 8 calls per arrival, p_o increases from 2.2% to 31.3% and p_{nc} increases from 2.6% to 33.3%, an almost 14 fold increase in both cases. Also, it is observed that p_o and p_{nc} vary across the different call hold-time distribution for BLs in the range of 1 to 7, but then converge for BL equal to 8. Figure 9 is a "zoom-in" view of the previous figure for burst levels in the range of 1 to 4. In (b) and (c) of this figure, it is observed that the Pareto call hold-time distribution results in p_o and p_{nc} values that are up to 60% higher than those for the exponential call hold-time distribution when BL is 3 and the uniform is also higher than the exponential. We believe this behavior is due to the variance characteristics of the call hold-time distributions. With the Pareto distribution, all the calls last 180 seconds (see Figure 8(b)). Now, the inter-arrival time between successive bursts is exponentially distributed, making it more likely that the time between two bursts is well below the mean, which increases the possibility that the first set of call arrivals have not completed before the next set arrives, thus increasing p_o and ultimately p_{nc}. However, with exponential and to some degree the uniform call hold-times, it is possible that some subset of calls in a burst will have a call hold-time far below the 180 second mean, allowing more calls in the next bursty arrival set to connect, thus lowering both the probability that new calls will block (p_o), and the total network blocking probability (p_{nc}).

PLACE FIGURE 9 ABOUT HERE.

The convergence is due to the number of channels per cell being fixed at 8, which is the BL point where the different call hold-time distribution's values for p_o and p_{nc} converge. This suggests that when BL is equal to the number of channels, the influence of the call hold-time distribution's variance on p_o and p_{nc} is negated by BL.

PLACE FIGURE 10 ABOUT HERE.

PLACE FIGURE 11 ABOUT HERE.

Figure 10 shows p_f for different call hold-time distributions as a function of mobility rate, Mob when BL is 1. It is observed that changes in the call hold-time distribution do not impact p_f. However, when BL is 3, p_f is significantly affected by the call hold-time distribution across different values of Mob. This change in p_f's behavior as the call hold-time distribution is varied is most likely attributed to the different variance characteristics of the call hold time distributions. The Pareto's small variance for α equal to 100.0 causes an increase in the multiple hand-offs made by a single call, which when combined with bursty call arrivals, leads to higher p_f values when compared to the exponential and uniform cases.

PLACE FIGURE 12 ABOUT HERE.

PLACE FIGURE 13 ABOUT HERE.

Figure 12 shows the effect of the Pareto distribution's α parameter on call blocking statistics. We observe that as α is increased from 1.6 to 20.0, p_f, p_o and p_{nc} increase sharply, but then tapper off for α values greater than 80.0. We believe the reason for this behavior, lies in the variance of the distribution. As shown in Figure 13, for α equal to 1.6, the variance in the distribution is infinite. However, as α is increased to 100.0 (Figure 13(c)), we observe that the variance reduces to almost zero and the distribution becomes almost deterministic. Again, because the variance in the call hold times is almost zero ($\alpha = 100.0$), all calls last as long as the mean of the distribution which can keep many of a cell's channels occupied until the next burst of calls arrives and allow more calls to experience multiple hand-offs, increasing p_f, p_o and p_{nc}.

PLACE TABLE 1 ABOUT HERE.

Because distributed simulation reduces the execution time of the simulation model, a wide range of experiments that cover many model parameters can be completed in a reasonable amount of time. However, these experiments may produce a deluge of data from which the modeler must sift through. One way to make sense of this data is through statistical inferencing. Using the collected simulation data, we derive linear models for each blocking statistic (p_f, p_o and p_{nc}) across the different call hold time distributions (exponential, Pareto, and uniform) for a fixed offer load. All models were obtained using the Splus statistical analysis package [27]. Table 1 shows a summary of the different models. All models, except Pareto p_f and uniform p_f, have an R^2 statistic above 90%. The R^2 statistic determines how much of the data's variation is accounted for by the model.

For a complete guide to the design of experiments and development of regression models, we refer the reader to [28]. In many cases, the linear model accounts for over 99% of the data's variation, making them very good predictors of the blocking statistics. Moreover, we find that the burst level, BL, is the largest contributing factor. For exponential p_f, BL explains 65% of the p_f's variation and for exponential p_{nc}, BL explains 98% of the p_f's variation. Adequate simple linear models for Pareto p_f and uniform p_f could not be found due to their nonlinear behavior. To account for this nonlinearity, we attempted fitting *generalized linear models* (GLMs) to the experimental data. In both cases (Pareto p_f and uniform p_f), the use of GLMs did not improve the statistical accuracy when compared with simple linear models. Consequently, the results presented here for Pareto and uniform p_f are based on simple linear models which account for only 63% and 77% of the data's variance respectively. For a comprehensive text about GLMs, we refer the reader to [29].

To determine each models validity, we conducted a series of visual tests. The results of these tests are shown in Figures 14 - 22. Each figure contains three panels. The *top left panel* plots the actual blocking statistic versus the predictor variables using a linear model. In all cases except Pareto p_f and uniform p_f, the linear models appears to be fairly accurate in predicting the blocking statistics. The *top right panel* plots the residual error from the model as a function of the model's predicted values. These plots should have a random pattern of points, implying that the residuals do not have a systematic trend. In all cases these plots appear to be random. One of the underlying assumptions of linear regression models is that the residual errors are independent and in the simplest case normally distributed. The *bottom center panel* tests for this attribute by showing results for the normal quantile-quantile plot of the residual error. These plots should have an linearly increasing line of dots, which for the most part is true of the linear models described here.

PLACE FIGURE 14 ABOUT HERE.

PLACE FIGURE 15 ABOUT HERE.

PLACE FIGURE 16 ABOUT HERE.

PLACE FIGURE 17 ABOUT HERE.

PLACE FIGURE 18 ABOUT HERE.

PLACE FIGURE 19 ABOUT HERE.

PLACE FIGURE 20 ABOUT HERE.

PLACE FIGURE 21 ABOUT HERE.

PLACE FIGURE 22 ABOUT HERE.

4 Summary

This article describes an effective simulation approach for developing practical models for large-scale PCS networks. First we introduced a distributed simulation technique for simulating PCS networks. This approach is able to simulate realistically sized networks in a short amount of time, giving better insight into the behavior of these networks. With a simple PCS model, our experiments indicate that distributed simulations using 8 workstations reduce execution time from 20 hours to 3.5 hours or less in many cases.

After validating the simulation using previously derived analytical results, we exploit the simulation's flexibility and examine the effect of call hold-time distributions and bursty call arrivals on PCS blocking statistics - a problem that is not easily addressed using an analytical approach. The primary results from this study are:

- For PCS networks where the call arrivals form a Poisson process, the distribution of call hold-times does not seem to effect network performance.
- Bursty Poisson call arrival processes significantly degrades PCS network performance, even for small burst sizes (2 or 3). Also, the call hold-time distribution does impact network performance in the presence of Poisson bursty call arrivals.
- In the presence of bursty Poisson call arrivals, small variances in the call hold-time distribution result in degraded PCS network performance. Thus, *heavy tailed* call hold-time distributions,

such as Pareto with $\alpha < 2.0$ and lognormal may yield lower blocking statistics than the exponential distribution.

- Many of the PCS blocking statistics appear to be adequately modeled by simple or generalized linear regression for the range of data in this study. These statistical methods may aide in the analysis of large data sets produced by the distributed simulator.

We view this study as an example of the many different aspects of PCS networks that can be modeled using distributed simulation. Issues such as hand-off strategies, dynamic channel assignment, and wireless packet switching can be investigated by extending the simulation model presented here.

Acknowledgments

The authors would like to thank Yi-Bing Lin and Amarnath Mukherjee for their invaluable support in the development of this work.

References

[1] D.C. Cox. Personal Communications – A Viewpoint. *IEEE Communications Magazine*, 128(11), 1990.

[2] R. Jain, Y.-B. Lin, C.N. Lo and S. Mohan. A Caching Strategy to Reduce Network Impacts of PCS. To appear in*IEEE Journal on Selected Areas in Communications*, 1994.

[3] Y.-B. Lin. Determining the User Locations for Personal Communications Networks. To appear in *IEEE Transactions on Vehicular Technology*, 1994.

[4] S. Mohan and R. Jain. Two User Location Strategies for Personal Communications Services (PCS): A Tutorial. To appear in *IEEE PCS Magazine*, 1994.

[5] Bellcore. Generic Criteria for Version 0.1 Wireless Access Communications Systems (WACS) and Supplement. Technical Report TR-INS-001313, Issue 1, Bellcore, October 1993.

[6] Y.-B. Lin and A. Noerpel. Implicit Deregistration in a PCS Network. To appear in *IEEE Transactions on Vehicular Technology*, 1994.

[7] P. Porter, D. Harasty, M. Beller, A. Noerpel and V. Varma. The Terminal Registration/Deregistration Protocol for Personal Communication Systems. *Wireless 93 Conference on Wireless Communications*, July 1993.

[8] D. Hong and S.S. Rappaport. Traffic Model and Performance Analysis for Cellular Mobile Radio Telephone Systems with Prioritized and No-Protection Handoff Procedure. *IEEE Transactions on Vehicular Technology*, VT-35(3), August 1986.

[9] Y.-B. Lin, S. Mohan and A. Noerpel. Modeling the Queueing Channel Assignment Strategies for Handoff and Initial Access for a PCS Network. Submitted to *IEEE Transactions on Vehicular Technology*, 1994.

[10] S. Tekinary and B. Jabbari. Handover Policies and Channel Assignment Strategies in Mobile Cellular Networks. *IEEE Communications Magazine*, 29(11), 1991.

[11] S. Tekinary and B. Jabbari. A Measurement Based Prioritization Scheme for Handovers in Cellular and Microcellular Networks. *IEEE J. Sel. Areas Comm.*, October 1992.

[12] C.H. Yoon and K. Un. Performance of Personal Portable Radio Telephone Systems with and without Guard Channels. *IEEE Journal on Selected Areas in Communications*, 11(6):911–917, August 1993.

[13] S.M. Elnoubi, R. Singh and S.C. Gupta. A New Frequency Channel Assignment Algorithm in High Capacity Mobile Communication Systems. *IEEE Transactions on Vehicular Technology*, VT-31(3):125–131, 1982.

[14] T. J. Kahwa and N. D. Georganas. A Hybrid Channel Assignment Scheme in Large-Scale Cellular-Structure Mobile Communication Systems. *IEEE Transactions on Communications*, COM-26(4):432–438, April 1978.

[15] S.S. Kuek and W.C. Wong. Ordered Dynamic Channel Assignment Scheme with Reassignment in Highway Microcells. *IEEE Trans. Veh. Technol.*, 41(3):271–277, 1992.

[16] Y.-B. Lin and W. Chen. Call Request Buffering in a PCS Network. *IEEE INFOCOM*, 1994.

[17] M. Zhang and T.-S. Yum. Comparisons of Channel-Assignment Strategies in Cellular Mobile Telephone Systems. *IEEE Trans. Veh. Technol.*, 38(4):211–215, 1989.

[18] G.J. Foschini, B. Gopinath and Z. Miljanic. Channel Cost of Mobility. *IEEE Transactions on Vehicular Techology*, 42(4):414–424, November 1993.

[19] Y.-B. Lin and V.K. Mak. On Simulating a Large-Scale Personal Communications Services Network. To appear in ACM Transactions on Modeling and Computer Simulation, 1993.

[20] W.C. Wong. Packet Reservation Multiple Access in a Metropolitan Microcellular Radio Environment. *IEEE J. Sel. Areas Comm.*, 11(6):918–925, 1993.

[21] O.F. Bryan. Modsim II - An Object-Oriented Simulation Language for Sequential and Parallel Processors. In *1989 Winter Simulation Conference Proceedings*, pages 122–127, 1989.

[22] D.R. Jefferson. Virtual Time. *ACM TOPLAS*, 7(3):404–425, July 1985.

[23] C. Carothers, R.M. Fujimoto, Y.-B. Lin and P. England. Distributed Simulation of PCS Networks Using Time Warp. *Proc. International Workshop on Modeling Analysis and Simulation of Computer and Telecommunication Systems*, pages 2–7, 1994.

[24] R. Brown. Calendar Queues: A Fast O(1) Priority Queue Implementation for the Simulation Event Set Problem. *Communications of the ACM*, 31(10):1220–1227, Oct. 1988.

[25] C. Carothers, R.M. Fujimoto and Y.-B. Lin. A Case Study in Simulating PCS Networks Using Time Warp. *Proceedings of the* 9^{th} *Workshop on Parallel and Distributed Simulation*, pages 87–94, 1995.

[26] D. Duffy, A. McIntosh, M. Rosenstein and W. Willinger. Statistical Analysis of CCSN/SS7 Traffic Data from Working CCS Subnetworks. *IEEE Journal on Selected Areas in Communications*, 12(3):544–551, April 1994.

[27] J.M. Chambers and T.J. Hastie. *Statistical Models in S.* Chapman and Hall, London, 1993.

[28] R. Jain. *The Art of Computer Systems Performance Analysis.* John Wiley and Sons, New York, 1991.

[29] P. McCullagh and J.A. Nelder. *Generalized Linear Models.* Chapman and Hall, London, 1989.

Author Biographies

Christopher Carothers is a Ph.D. candidate in the College of Computing at the Georgia Institute of Technology. He received a BS degree in Computer Science (1991) from the Georgia Institute of Technology in Atlanta, Georgia. His research interests include parallel and distributed simulation and the performance modeling of personal communications services networks. He is currently a guest-editor for INTERNATIONAL JOURNAL IN COMPUTER SIMULATION.

Richard Fujimoto is a professor in the College of Computing at the Georgia Institute of Technology. He received BS degrees in Computer Science (1977) and Computer Engineering (1978) from the University of Illinois in Urbana-Champaign, and MS (1980) and Ph.D. (1983) degrees from the University of California in Berkeley. He has been actively engaged in research in parallel and distributed simulation since 1986. He is currently an area-editor for ACM TRANSACTIONS ON MODELING AND COMPUTER SIMULATION, and chairs the steering committee for the annual WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION (PADS).

Figure-Table Caption List

CAPTION FOR FIGURE 1: Estimating the problem size (T is measured in simulated seconds, the mean call holding time is $1/\mu = 180$ seconds. The expected number of portables in a cell is $N = 50$.)

CAPTION FOR FIGURE 2: Flowchart for each PCS call.

CAPTION FOR FIGURE 3: Simulated and predicted p_{nc} for (a) Mobility = 1/(75 minutes), (b) Mobility = 1/(15 minutes), (c) 1/(3 minutes) as a function of offered load.

CAPTION FOR FIGURE 4: Speedup versus number of portables per cell (N), mobility and call inter-arrival times. Call holding times are exponentially distributed with a 3 minute mean. Experiments performed using 8 processors. $\bullet$: Mobility = 1/(15 minutes), $\diamond$: Mobility = 1/(45 minutes), $\circ$: Mobility = 1/(75 minutes).

CAPTION FOR FIGURE 5: Percent remote communications versus number of portables per cell (N), mobility and call inter-arrival times. Call holding times are exponentially distributed with a 3 minute mean. Experiments performed using 8 processors. $\bullet$: Mobility mean = 1/(15 minutes), $\diamond$: Mobility mean = 1/(45 minutes), $\circ$: Mobility mean = 1/(75 minutes).

CAPTION FOR FIGURE 6: Effect of exponential, Pareto, and uniform call hold-time distributions on blocking probabilities as a function of λ. $Mob = 0.04\mu$, $BL = 1$. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

CAPTION FOR FIGURE 7: Effect of exponential, Pareto, and uniform call hold-time distributions on blocking probabilities as a function of BL. $Mob = 0.04\mu$, $\lambda = 1.25$. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

CAPTION FOR FIGURE 8: Inverse cumulative distribution functions as a function of $x : range[0, 1]$, mean = 180.0. (a) exponential, (b) Pareto, and (c) uniform.

CAPTION FOR FIGURE 9: Effect of exponential, Pareto, and uniform call hold-time distributions on blocking probabilities as a function of BL in the range of 1 to 4 arrivals per burst.

$Mob = 0.04\mu$, $\lambda = 1.25$. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

CAPTION FOR FIGURE 10: Effect of exponential, Pareto, and uniform call hold-time distributions on p_f as a function of Mob. $BL = 1$, $\lambda = 3.75$.

CAPTION FOR FIGURE 11: Effect of exponential, Pareto, and uniform call hold-time distributions p_f as a function of Mob. $BL = 3$, $\lambda = 3.75$.

CAPTION FOR FIGURE 12: The effect of Parteo's α parameter on (a) p_f, (b) p_o, and (c) p_{nc}.

CAPTION FOR FIGURE 13: Inverse Pareto function with mean = 180.0 seconds for different values of α: (a) $\alpha = 1.6$, (b) $\alpha = 5.0$, (c) $\alpha = 100.0$.

CAPTION FOR TABLE 1: Summary of linear models for p_f, p_o, and p_{nc} for the different call duration distributions: exponential, Pareto, uniform. Mob is mobility rate. BL is the number of calls contained in a single burst. λ is fixed at 3.75 Erlangs.

CAPTION FOR FIGURE 14: Linear model for predicting p_f with exponential call durations. *Top left panel*: p_f vs. fitted p_f. *Top right panel*: residual error in model vs. fitted p_f. *Bottom center panel*: quantile-quantile plot of residual errors.

CAPTION FOR FIGURE 15: Linear model for predicting p_o with exponential call durations. *Top left panel*: p_o vs. fitted p_o. *Top right panel*: residual error in model vs. fitted p_o. *Bottom center panel*: quantile-quantile plot of residual errors.

CAPTION FOR FIGURE 16: Linear model for predicting p_{nc} with exponential call durations. *Top left panel*: p_{nc} vs. fitted p_{nc}. *Top right panel*: residual error in model vs. fitted p_{nc}. *Bottom center panel*: quantile-quantile plot of residual errors.

CAPTION FOR FIGURE 17: Linear model for predicting p_f with Pareto call durations. *Top left panel*: p_f vs. fitted p_f. *Top right panel*: residual error in model vs. fitted p_f. *Bottom center panel*: quantile-quantile plot of residual errors.

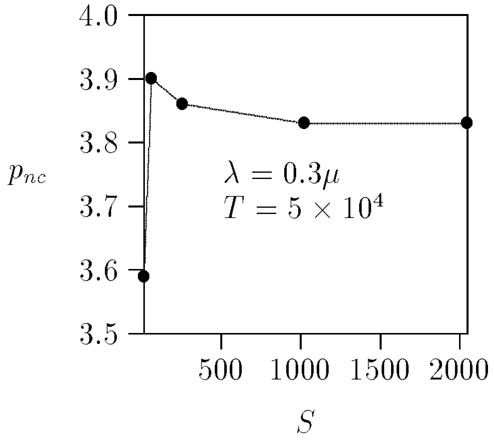
CAPTION FOR FIGURE 18: Linear model for predicting p_o with Pareto call durations. *Top left panel*: p_o vs. fitted p_o. *Top right panel*: residual error in model vs. fitted p_o. *Bottom center* *panel*: quantile-quantile plot of residual errors.

CAPTION FOR FIGURE 19: Linear model for predicting p_{nc} with Pareto call durations. *Top left panel*: p_{nc} vs. fitted p_{nc}. *Top right panel*: residual error in model vs. fitted p_{nc}. *Bottom center panel*: quantile-quantile plot of residual errors.

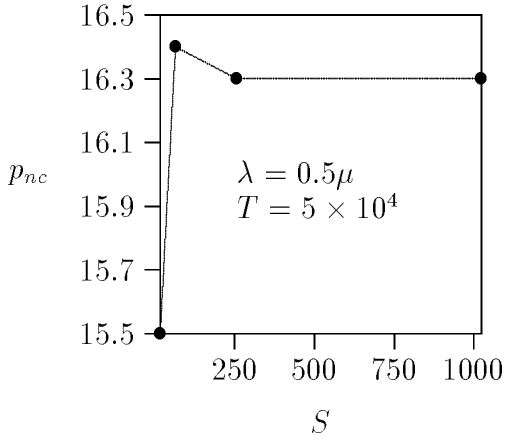
CAPTION FOR FIGURE 20: Linear model for predicting p_f with uniform call durations. *Top left panel*: p_f vs. fitted p_f. *Top right panel*: residual error in model vs. fitted p_f. *Bottom center panel*: quantile-quantile plot of residual errors.

CAPTION FOR FIGURE 21: Linear model for predicting p_o with uniform call durations. *Top left panel*: p_o vs. fitted p_o. *Top right panel*: residual error in model vs. fitted p_o. *Bottom center panel*: quantile-quantile plot of residual errors.

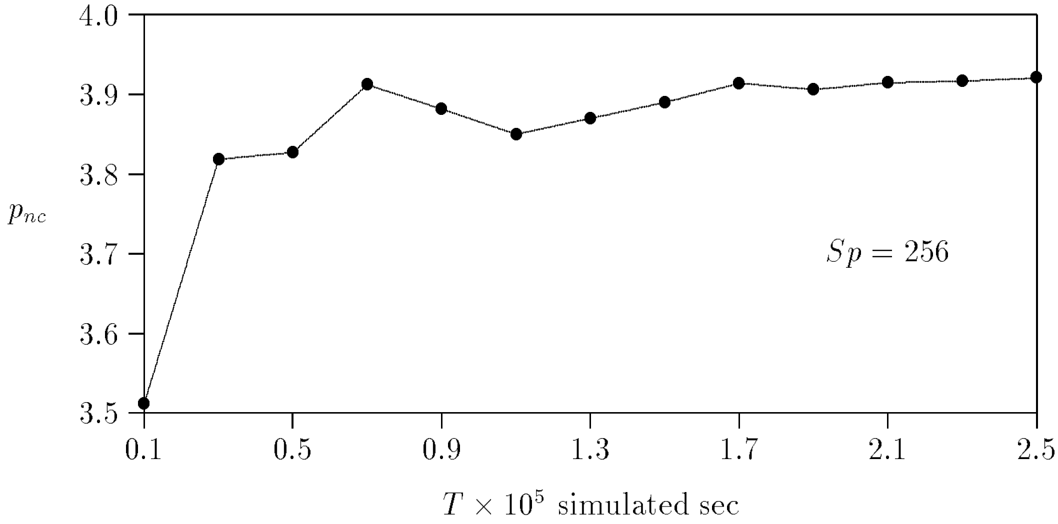
CAPTION FOR FIGURE 22: Linear model for predicting p_{nc} with uniform call durations. *Top left panel*: p_{nc} vs. fitted p_{nc}. *Top right panel*: residual error in model vs. fitted p_{nc}. *Bottom center panel*: quantile-quantile plot of residual errors.



(a) The impact of network size



(b) The impact of network size



(c) The impact of the termination time

Figure 1: Estimating the problem size (T is measured in simulated seconds, the mean call holding time is $1/\mu = 180$ seconds. The expected number of portables in a cell is $N = 50$.)

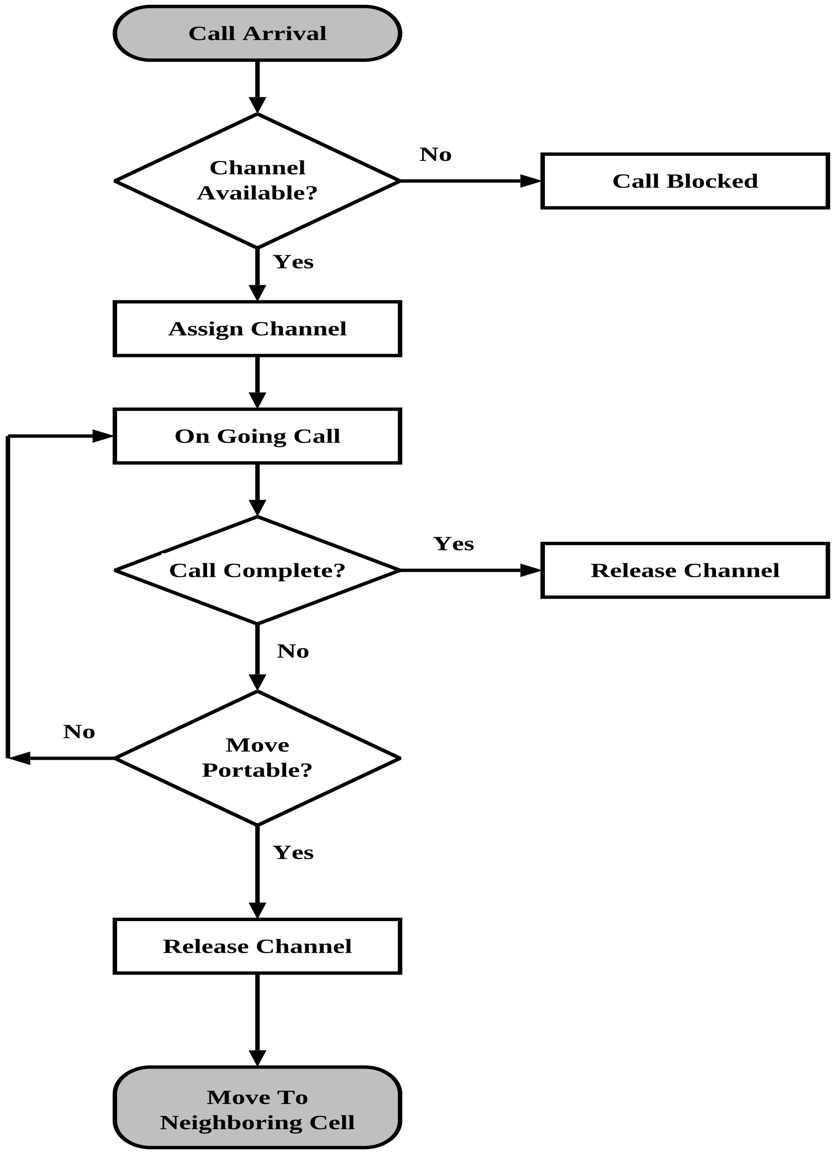


Figure 2: Flowchart for each PCS call.

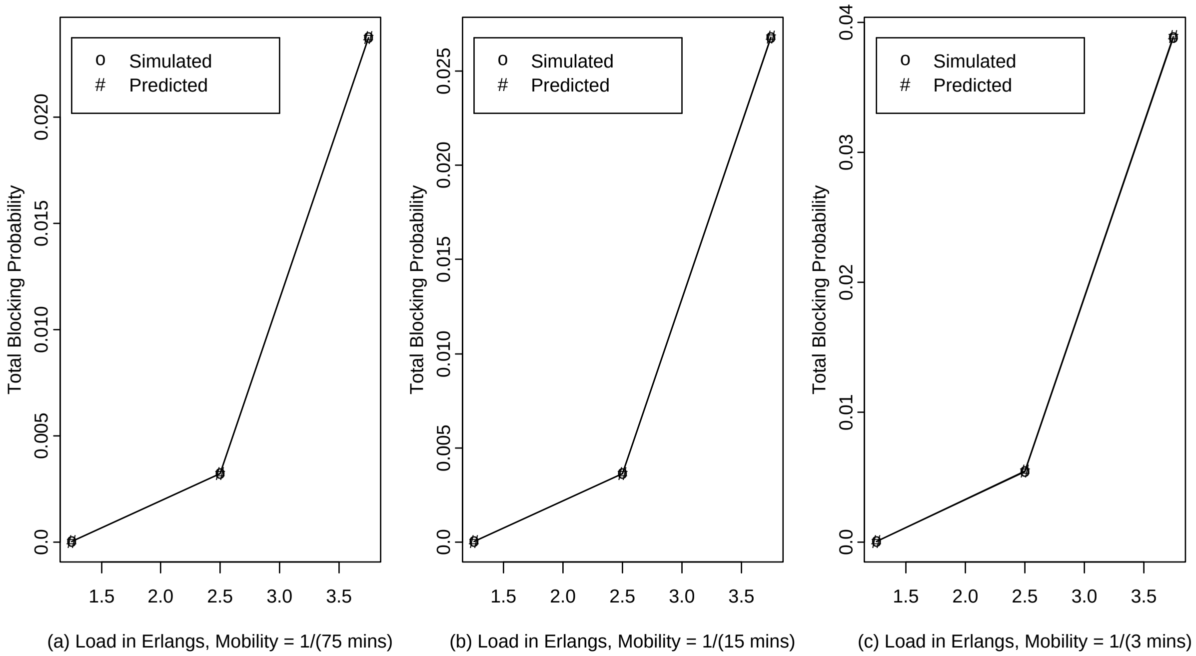


Figure 3: Simulated and predicted p_{nc} for (a) Mobility = 1/(75 minutes), (b) Mobility = 1/(15 minutes), (c) 1/(3 minutes) as a function of offered load.

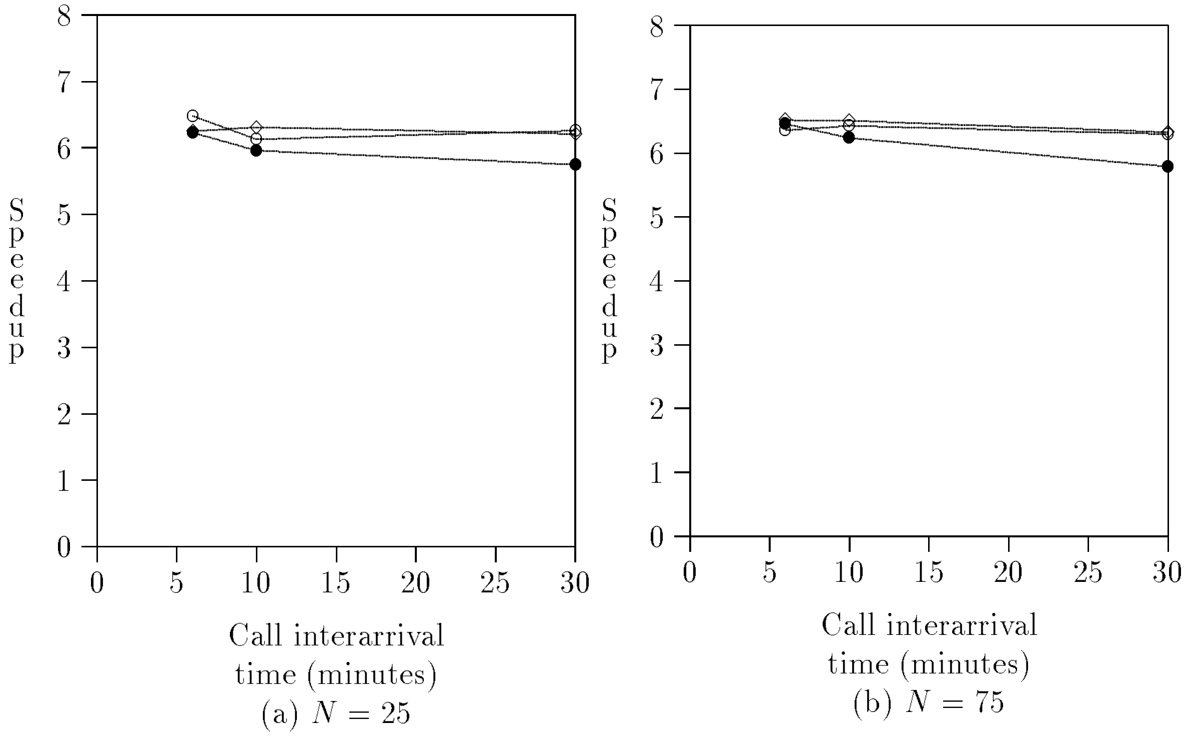


Figure 4: Speedup versus number of portables per cell (N), mobility and call inter-arrival times. Call holding times are exponentially distributed with a 3 minute mean. Experiments performed using 8 processors. $\bullet$: Mobility = 1/(15 minutes), $\diamond$: Mobility = 1/(45 minutes), $\circ$: Mobility = 1/(75 minutes).

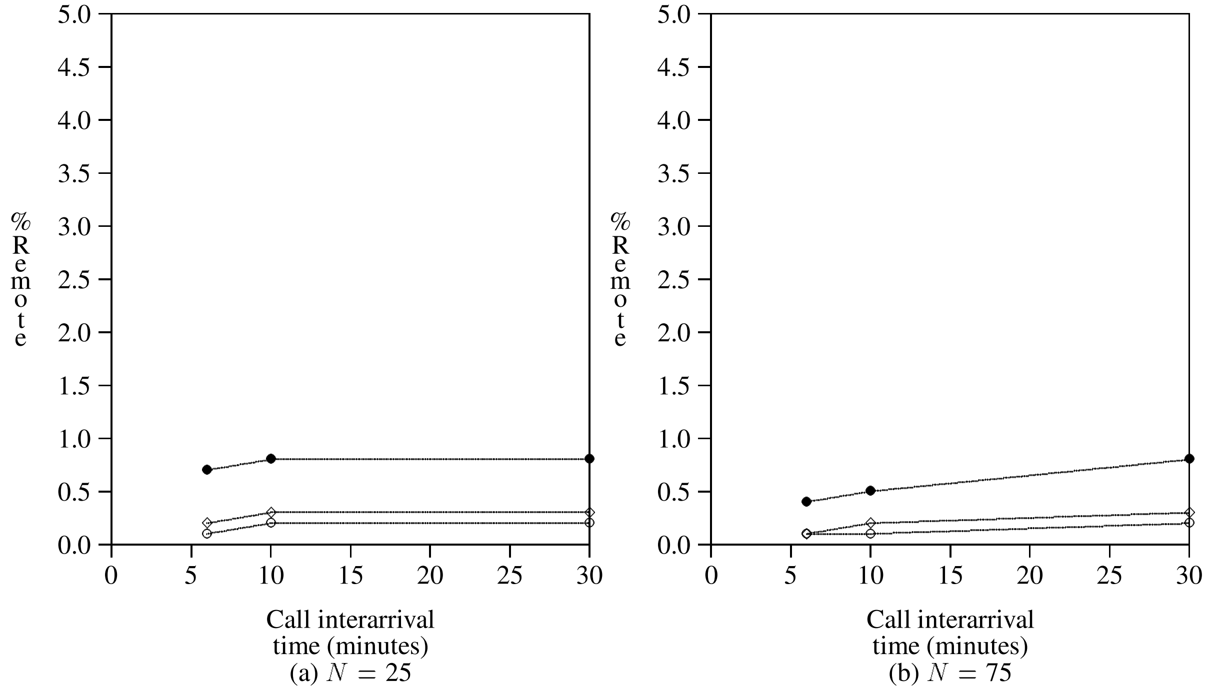


Figure 5: Percent remote communications versus number of portables per cell (N), mobility and call inter-arrival times. Call holding times are exponentially distributed with a 3 minute mean. Experiments performed using 8 processors. $\bullet$: Mobility mean = 1/(15 minutes), $\diamond$: Mobility mean = 1/(45 minutes), $\circ$: Mobility mean = 1/(75 minutes).

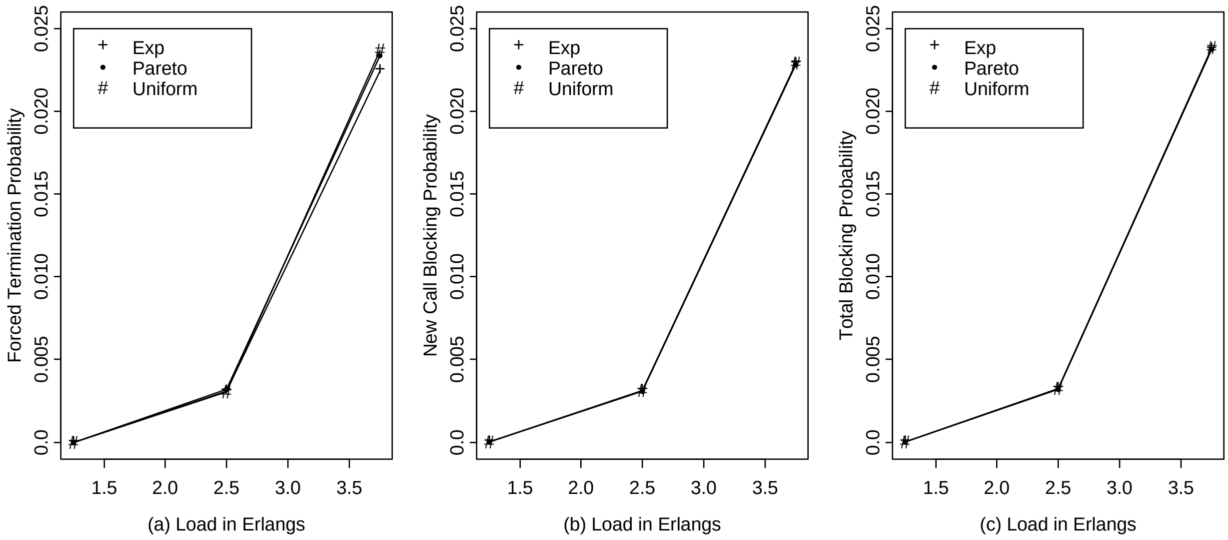


Figure 6: Effect of exponential, Pareto, and uniform call hold-time distributions on blocking probabilities as a function of λ. $Mob = 0.04\mu$, $BL = 1$. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

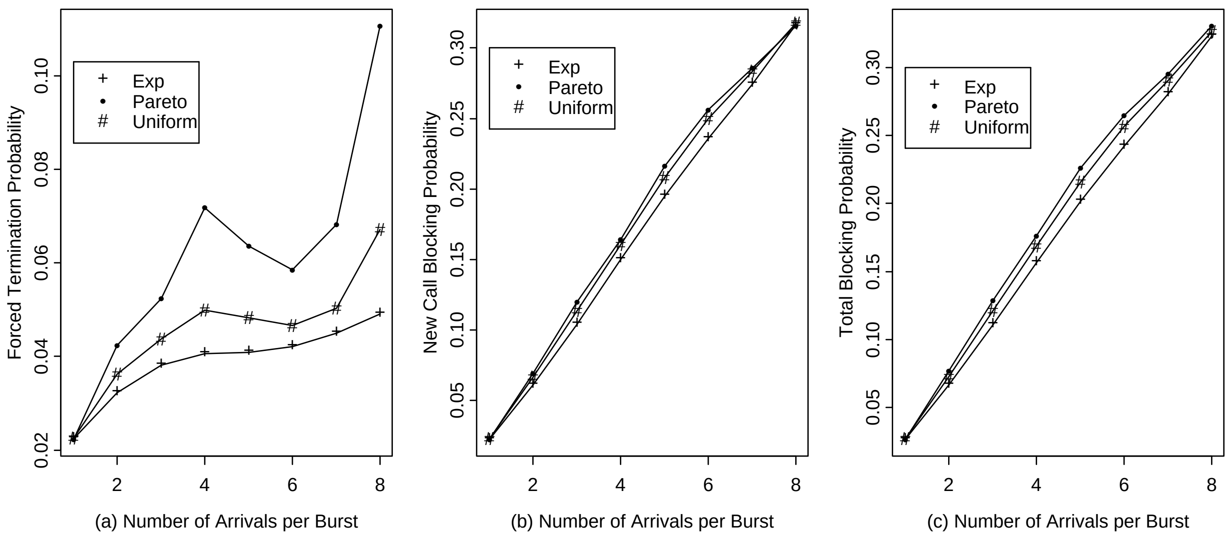


Figure 7: Effect of exponential, Pareto, and uniform call hold-time distributions on blocking probabilities as a function of BL. $Mob = 0.04\mu$, $\lambda = 1.25$. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

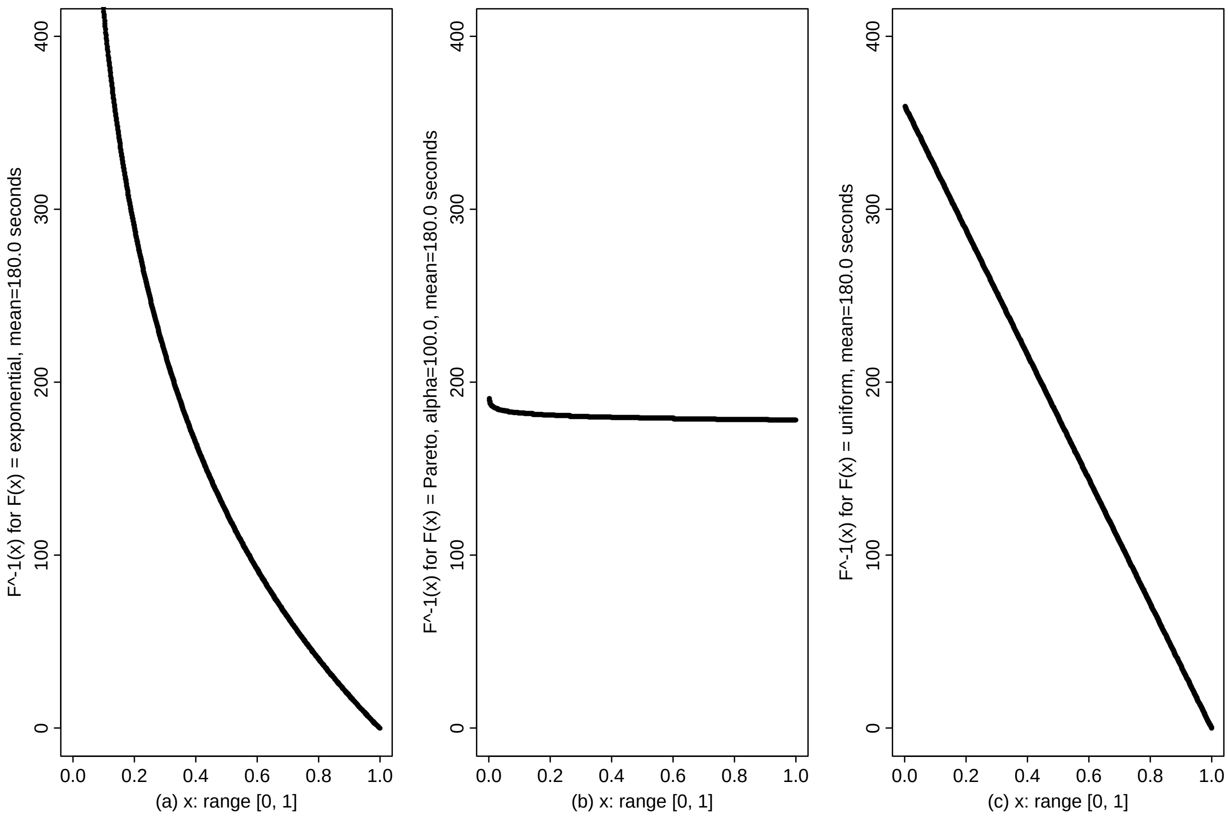


Figure 8: Inverse cumulative distribution functions as a function of $x : range[0, 1]$, mean = 180.0. (a) exponential, (b) Pareto, and (c) uniform.

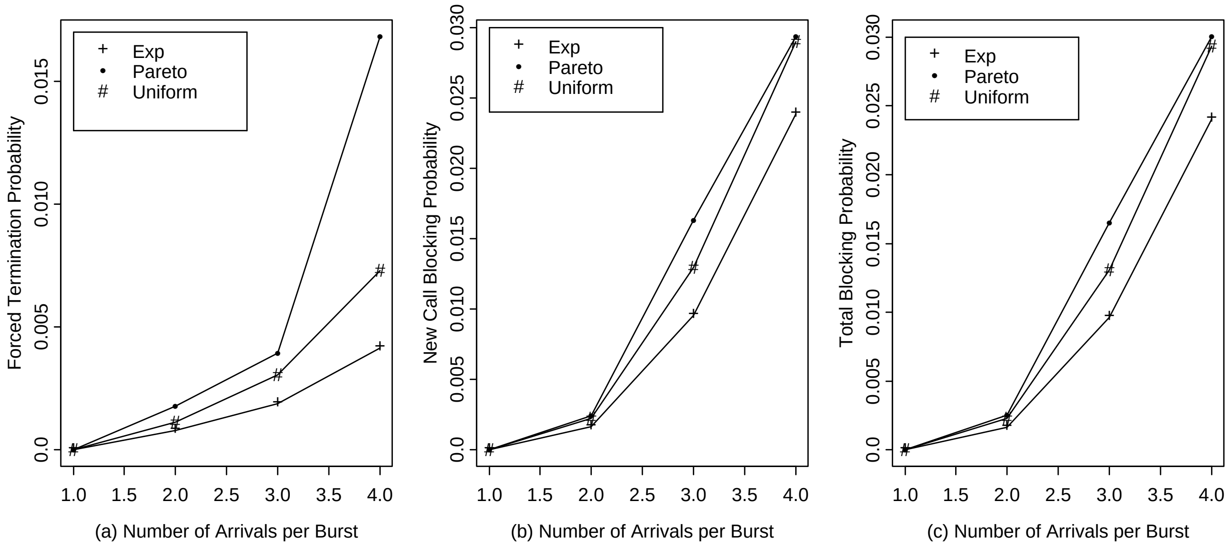


Figure 9: Effect of exponential, Pareto, and uniform call hold-time distributions on blocking probabilities as a function of BL in the range of 1 to 4 arrivals per burst. $Mob = 0.04\mu$, $\lambda = 1.25$. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

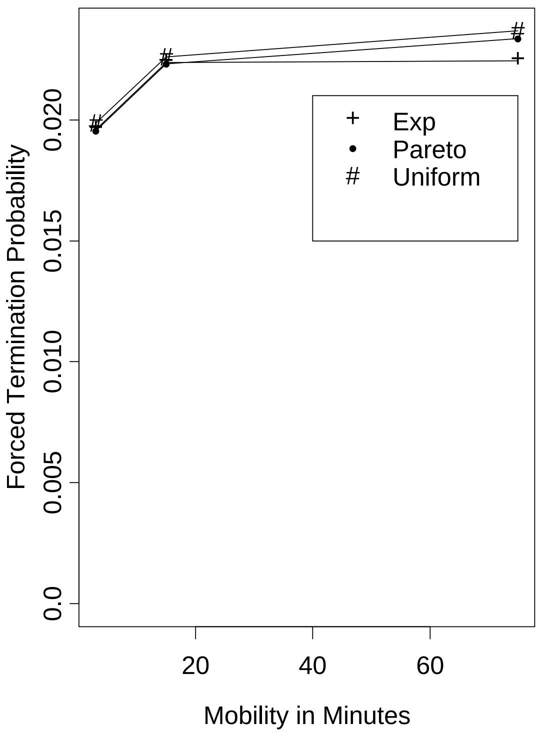


Figure 10: Effect of exponential, Pareto, and uniform call hold-time distributions on p_f as a function of Mob. $BL = 1$, $\lambda = 3.75$.

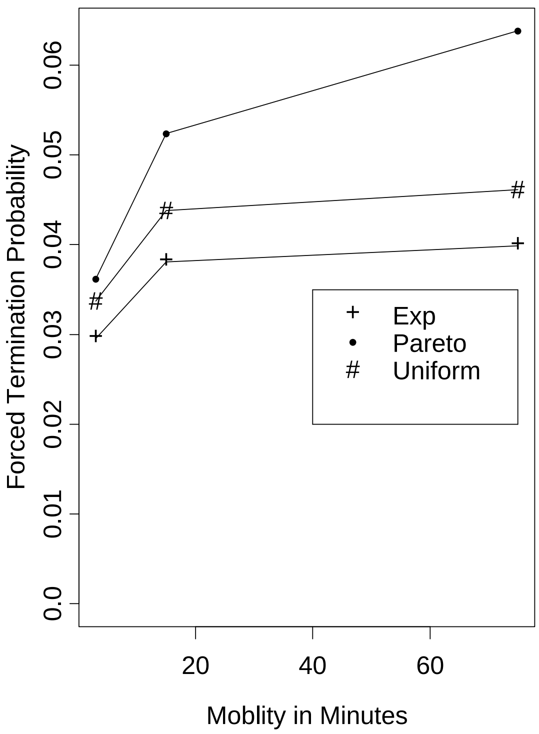


Figure 11: Effect of exponential, Pareto, and uniform call hold-time distributions p_f as a function of Mob. $BL = 3$, $\lambda = 3.75$.

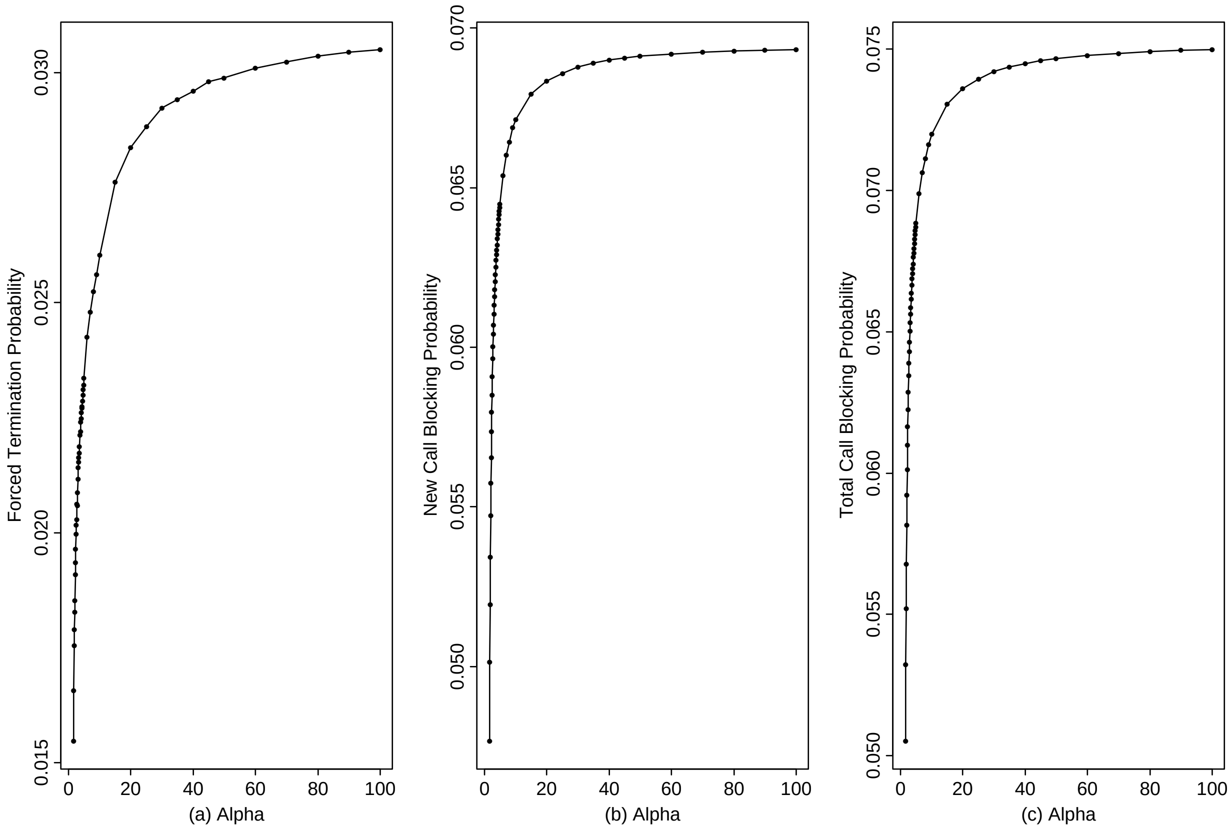


Figure 12: The effect of Parteo's α parameter on (a) p_f, (b) p_o, and (c) p_{nc}.

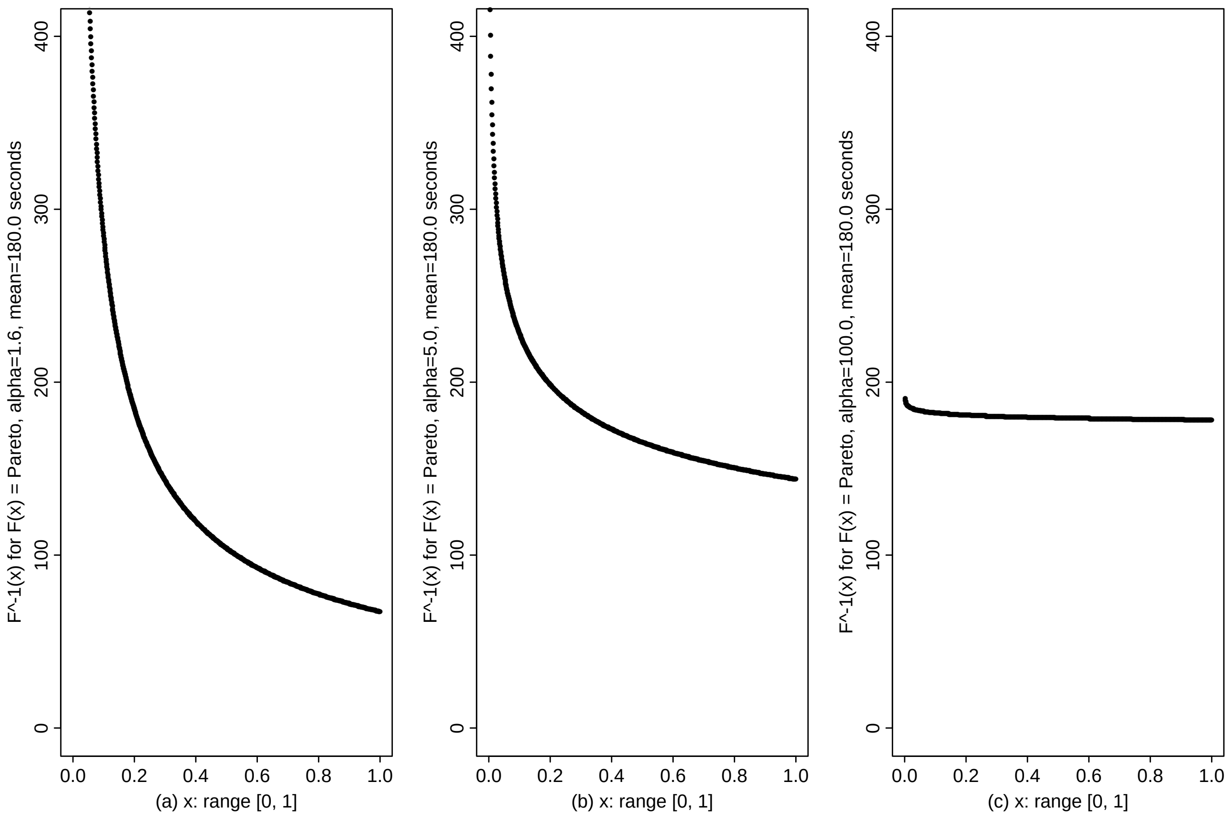


Figure 13: Inverse Pareto function with mean = 180.0 seconds for different values of α: (a) $\alpha = 1.6$, (b) $\alpha = 5.0$, (c) $\alpha = 100.0$.

Distribution	PCS Statistic	Model Equation	R^2 Statistic
exponential	p_f	$p_f = a_1 * Mob + a_2 * BL^{0.006} + \epsilon$	92.8%
exponential	p_o	$p_o = a_1 * Mob + a_2 * BL + \epsilon$	99.8%
exponential	p_{nc}	$p_{nc} = a_1 * Mob + a_2 * BL + \epsilon$	99.9%
Pareto	p_f	$p_f = a_1 * Mob + a_2 * BL + \epsilon$	63%
Pareto	p_o	$p_o = a_1 * Mob + a_2 * BL + \epsilon$	99.8%
Pareto	p_{nc}	$p_{nc} = a_1 * Mob + a_2 * BL + \epsilon$	99.9%
uniform	p_f	$p_f = a_1 * Mob + a_2 * BL + \epsilon$	76.3%
uniform	p_o	$p_o = a_1 * Mob + a_2 * BL + \epsilon$	99.6%
uniform	p_{nc}	$p_{nc} = a_1 * Mob + a_2 * BL + \epsilon$	99.6%

Table 1: Summary of linear models for p_f, p_o, and p_{nc} for the different call duration distributions: exponential, Pareto, uniform. Mob is mobility rate. BL is the number of calls contained in a single burst. λ is fixed at 3.75 Erlangs.

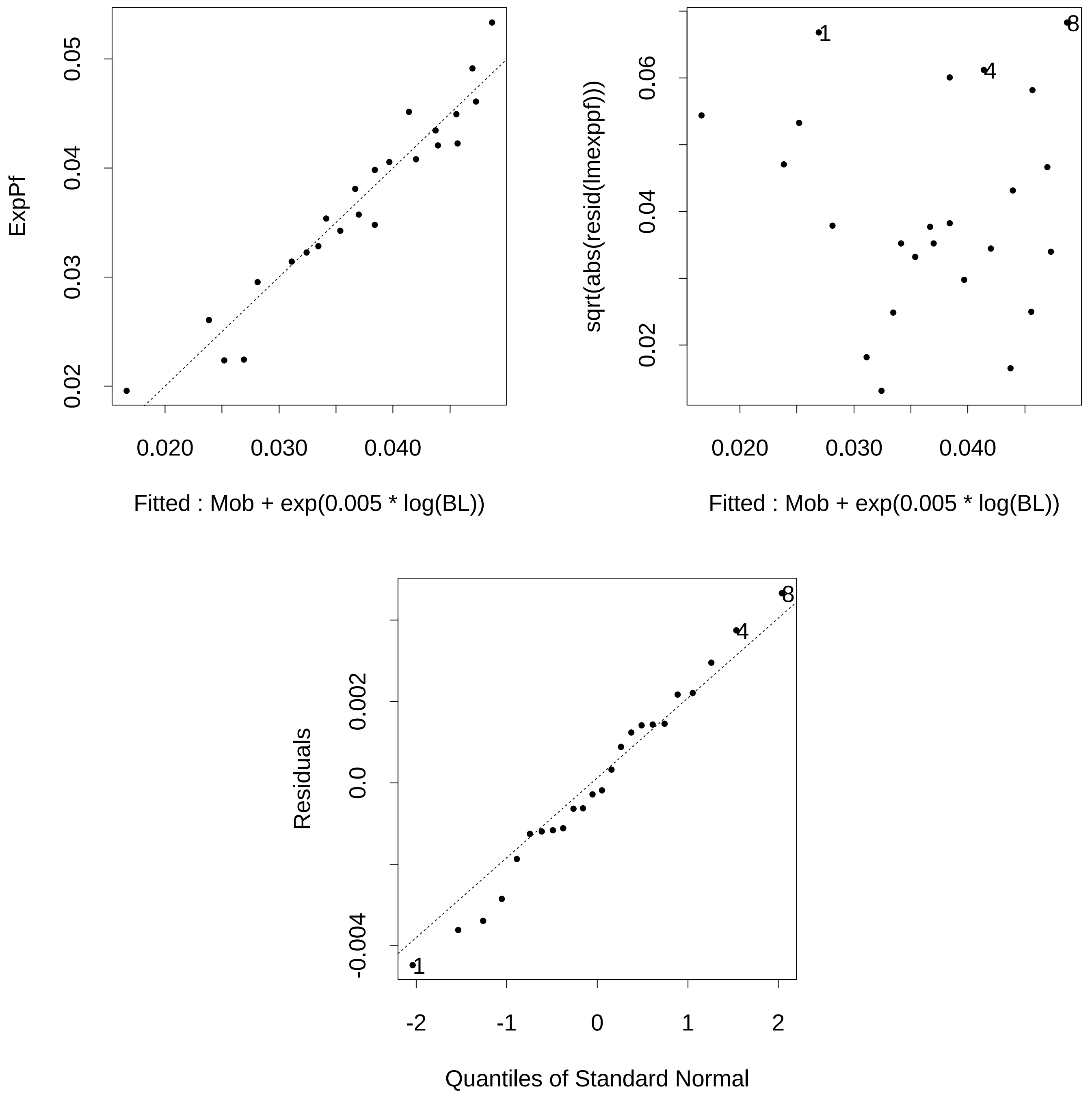


Figure 14: Linear model for predicting p_f with exponential call durations. *Top left panel*: p_f vs. fitted p_f. *Top right panel*: residual error in model vs. fitted p_f. *Bottom center panel*: quantile-quantile plot of residual errors.

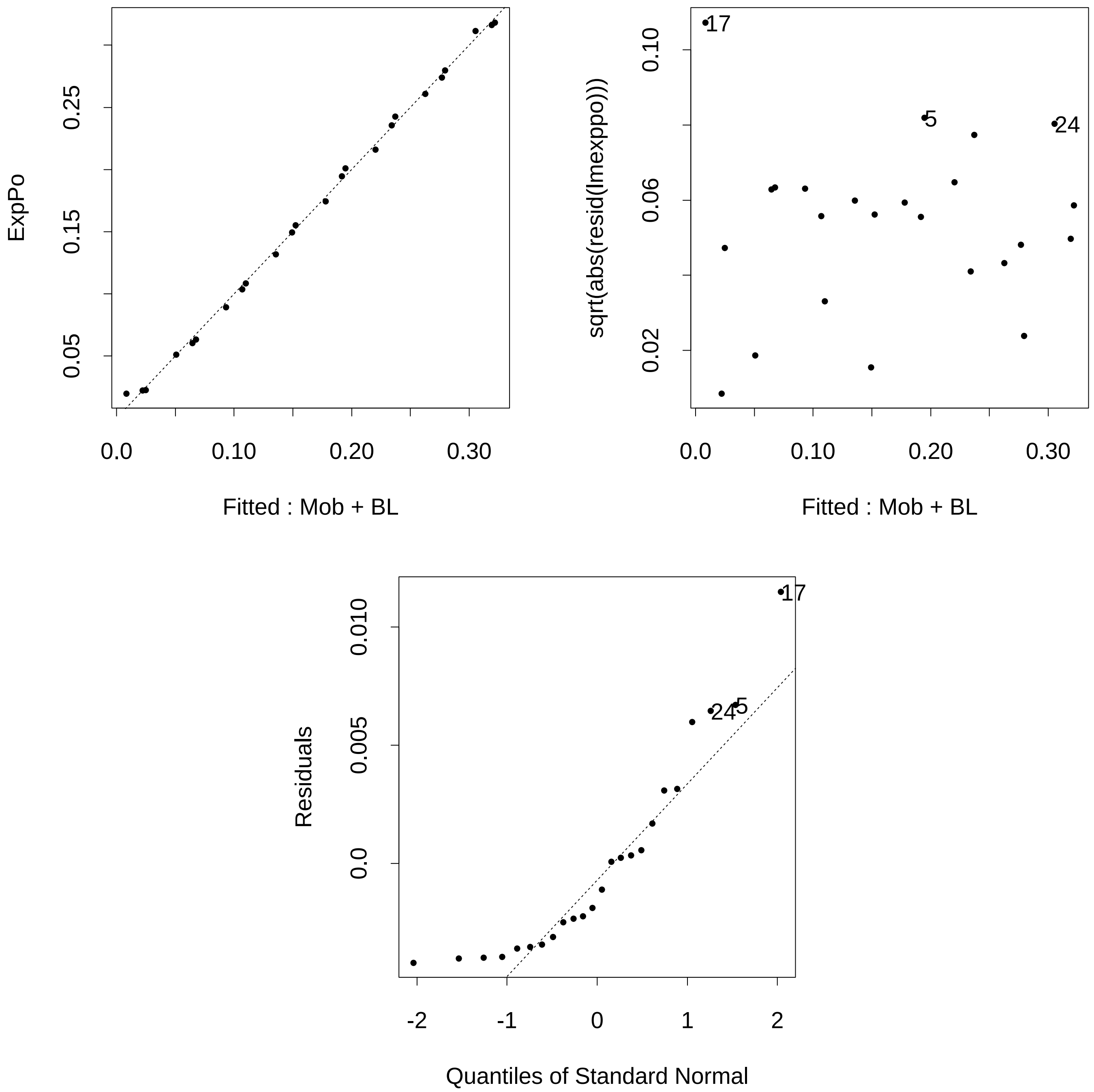


Figure 15: Linear model for predicting p_o with exponential call durations. *Top left panel*: p_o vs. fitted p_o. *Top right panel*: residual error in model vs. fitted p_o. *Bottom center panel*: quantile-quantile plot of residual errors.

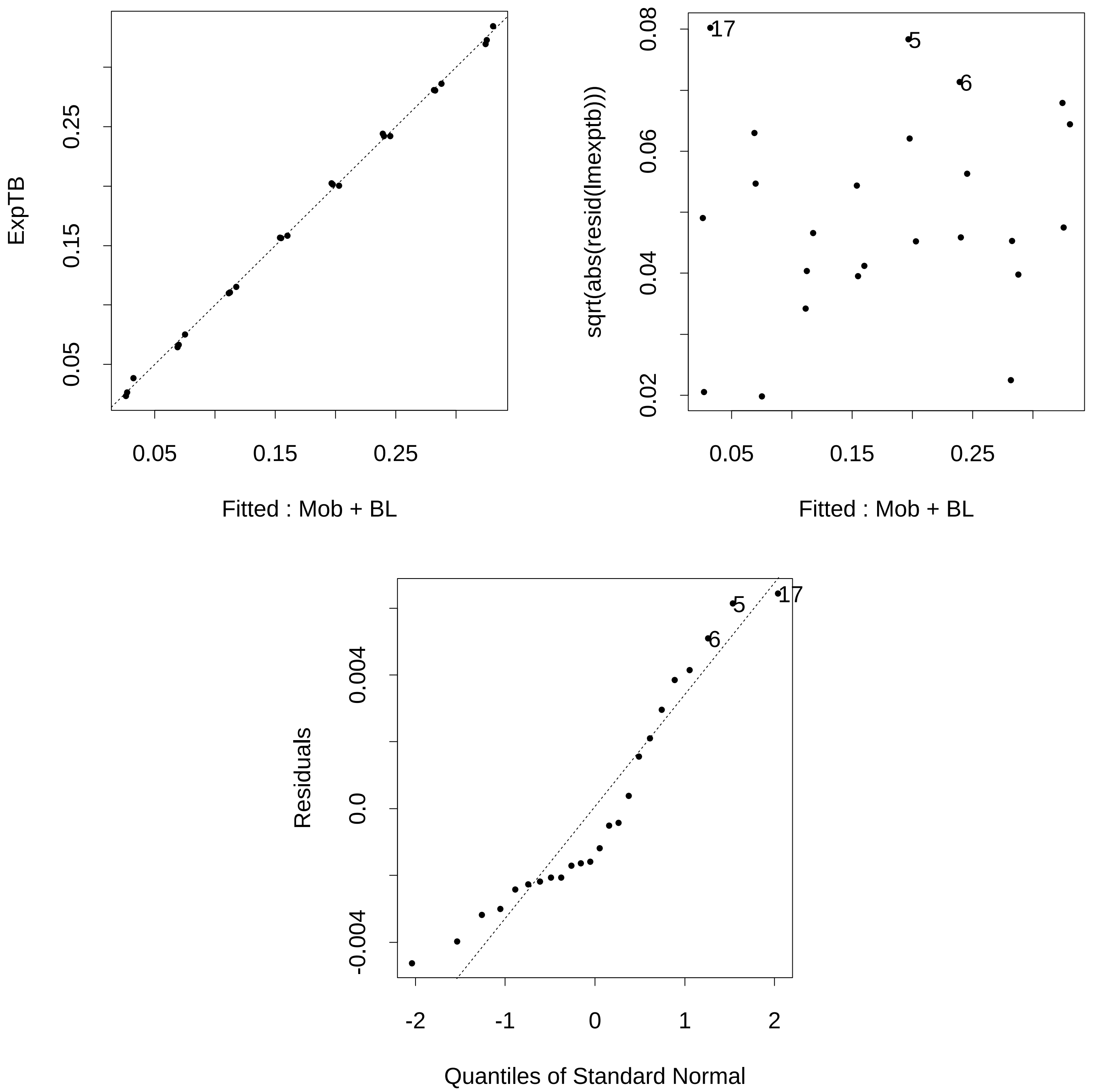


Figure 16: Linear model for predicting p_{nc} with exponential call durations. *Top left panel*: p_{nc} vs. fitted p_{nc}. *Top right panel*: residual error in model vs. fitted p_{nc}. *Bottom center panel*: quantile-quantile plot of residual errors.

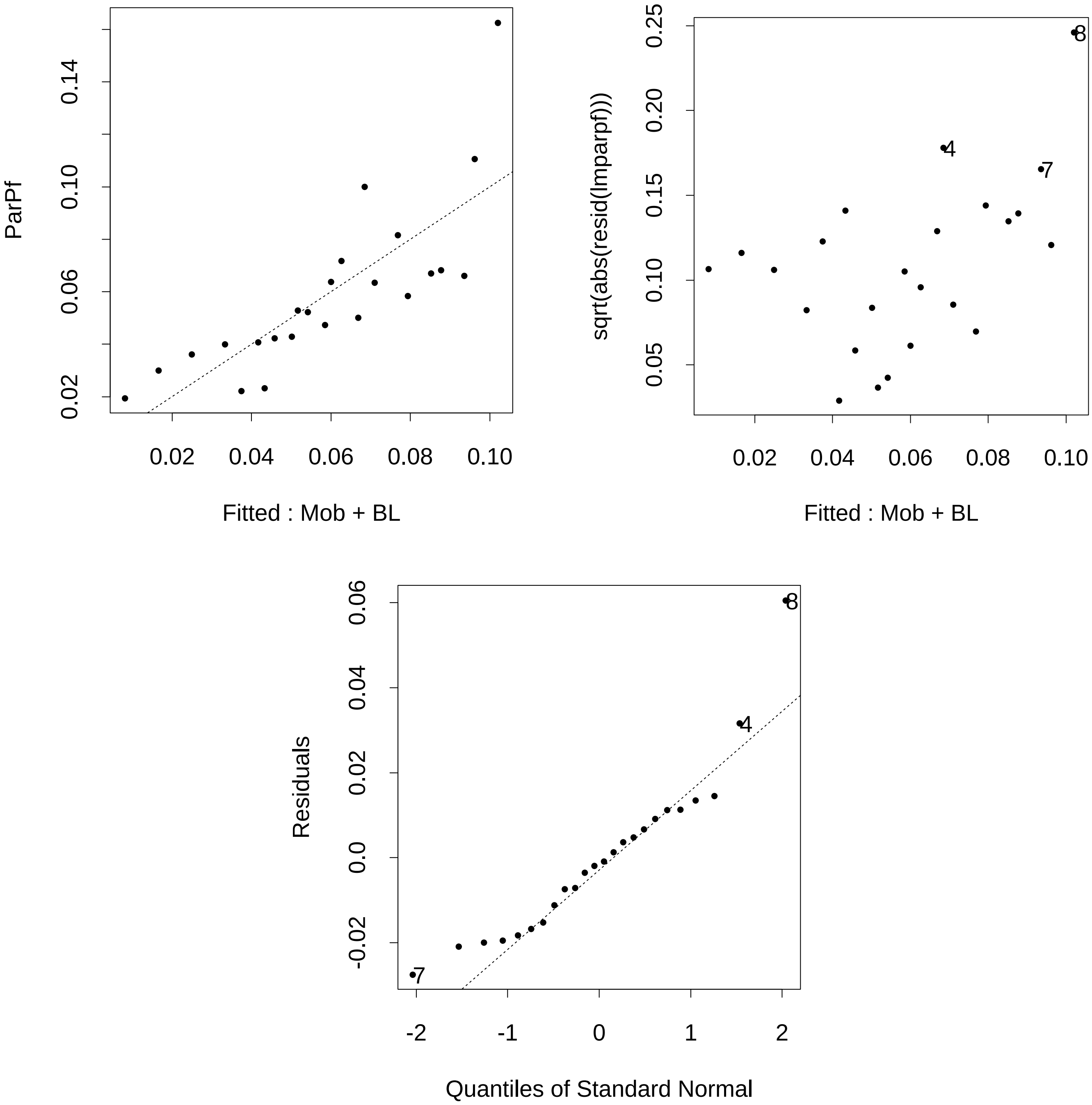


Figure 17: Linear model for predicting p_f with Pareto call durations. *Top left panel*: p_f vs. fitted p_f. *Top right panel*: residual error in model vs. fitted p_f. *Bottom center panel*: quantile-quantile plot of residual errors.

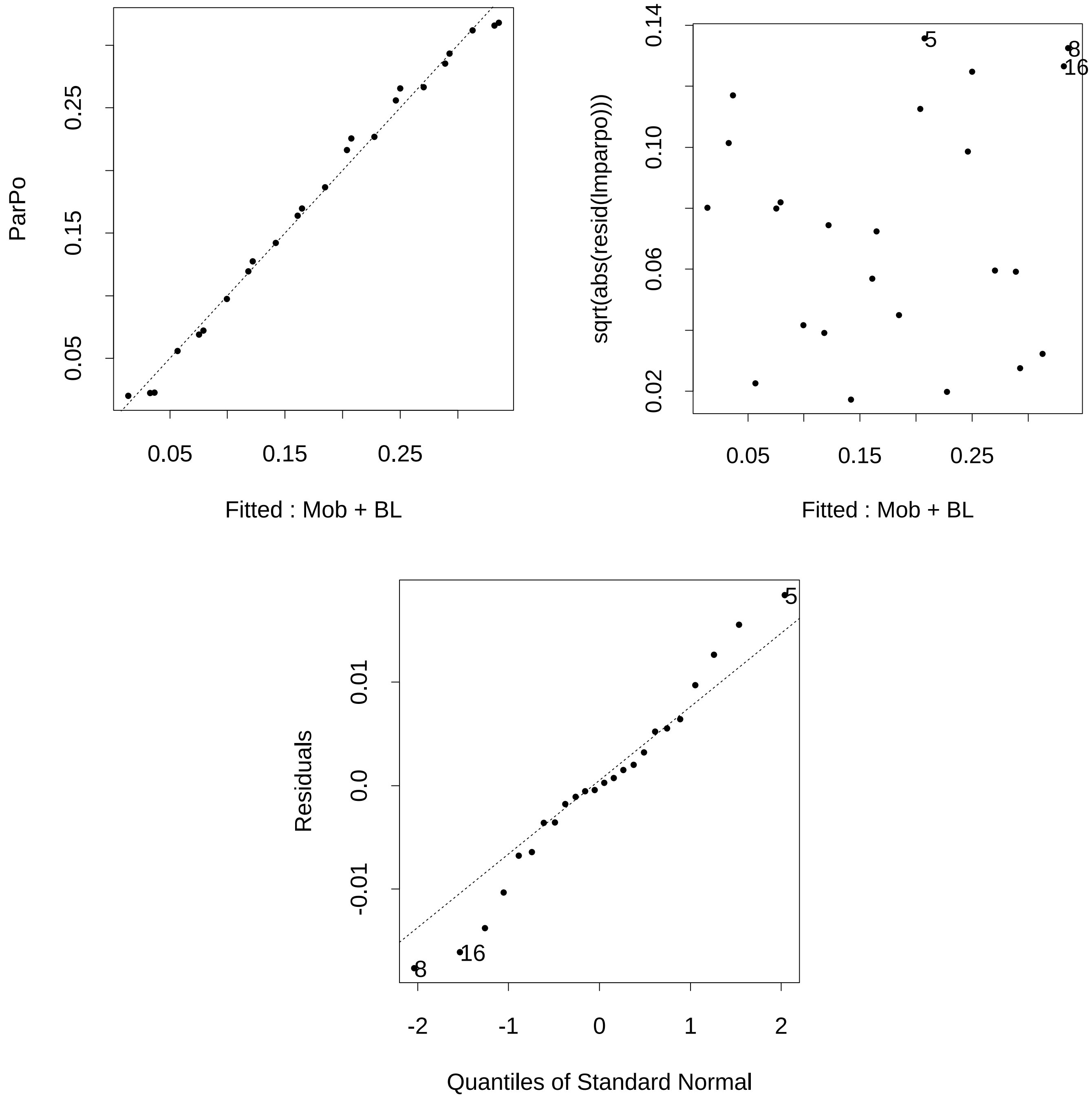


Figure 18: Linear model for predicting p_o with Pareto call durations. *Top left panel*: p_o vs. fitted p_o. *Top right panel*: residual error in model vs. fitted p_o. *Bottom center panel*: quantile-quantile plot of residual errors.

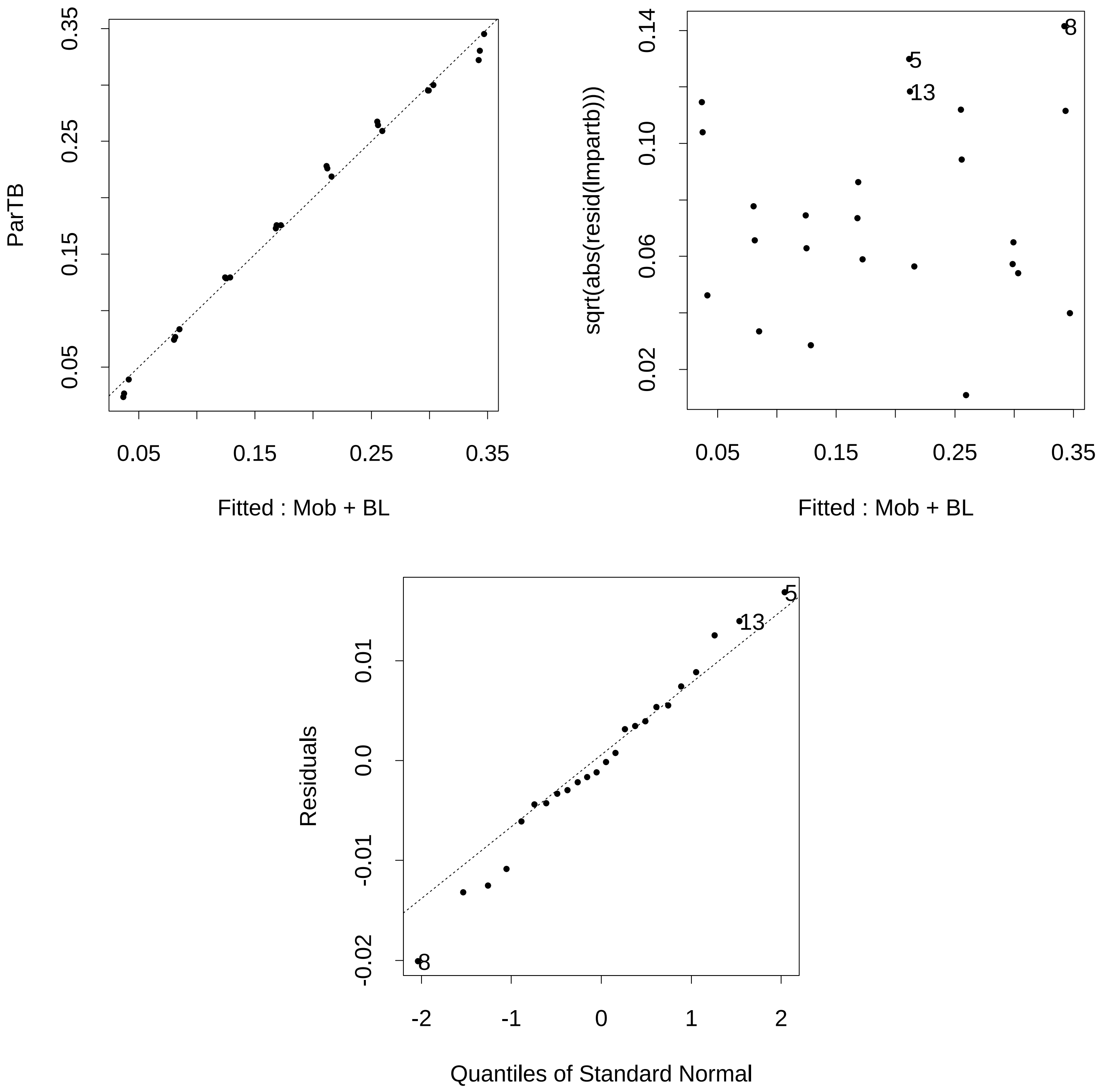


Figure 19: Linear model for predicting p_{nc} with Pareto call durations. *Top left panel*: p_{nc} vs. fitted p_{nc}. *Top right panel*: residual error in model vs. fitted p_{nc}. *Bottom center panel*: quantile-quantile plot of residual errors.

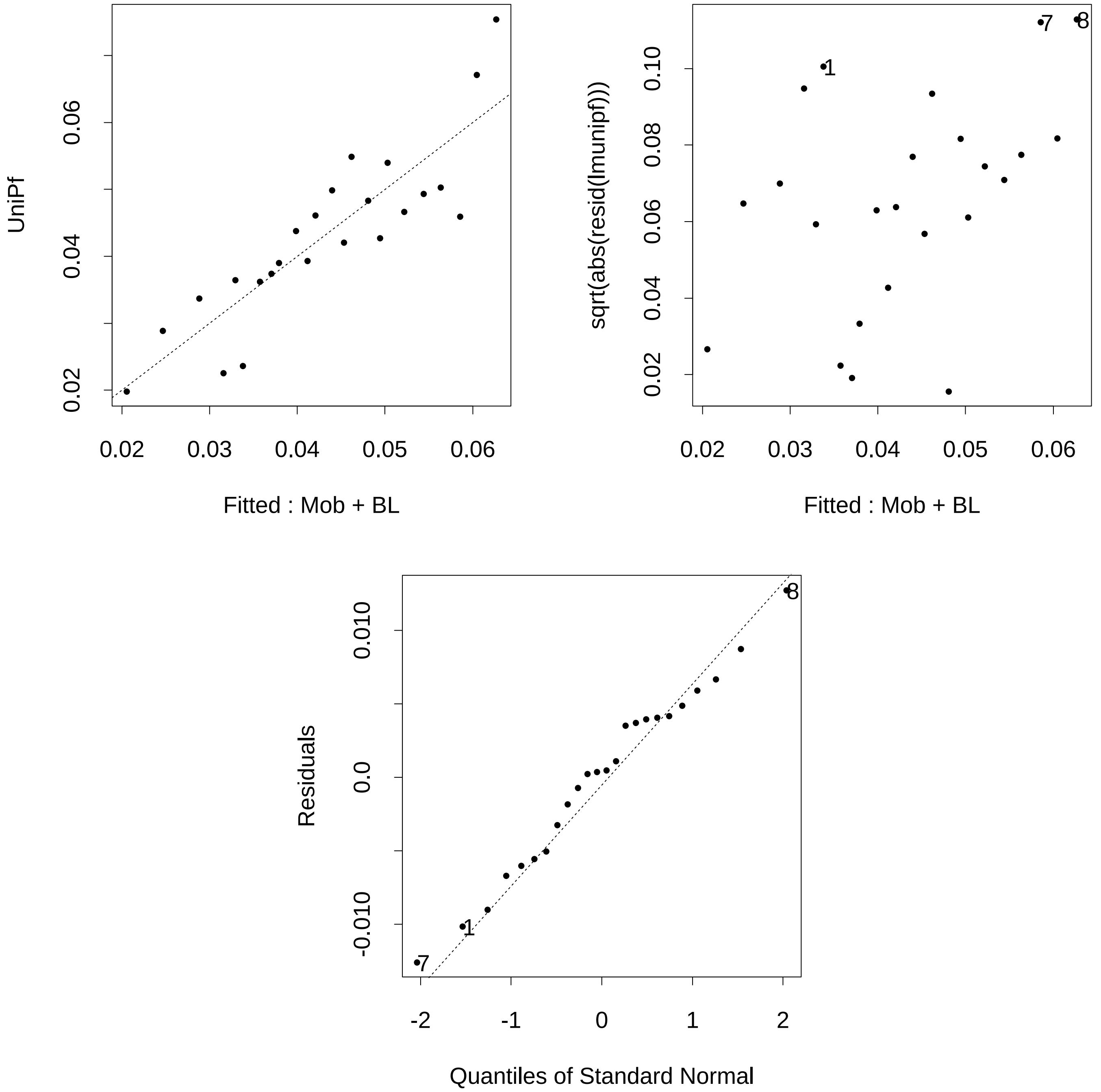


Figure 20: Linear model for predicting p_f with uniform call durations. *Top left panel*: p_f vs. fitted p_f. *Top right panel*: residual error in model vs. fitted p_f. *Bottom center panel*: quantile-quantile plot of residual errors.

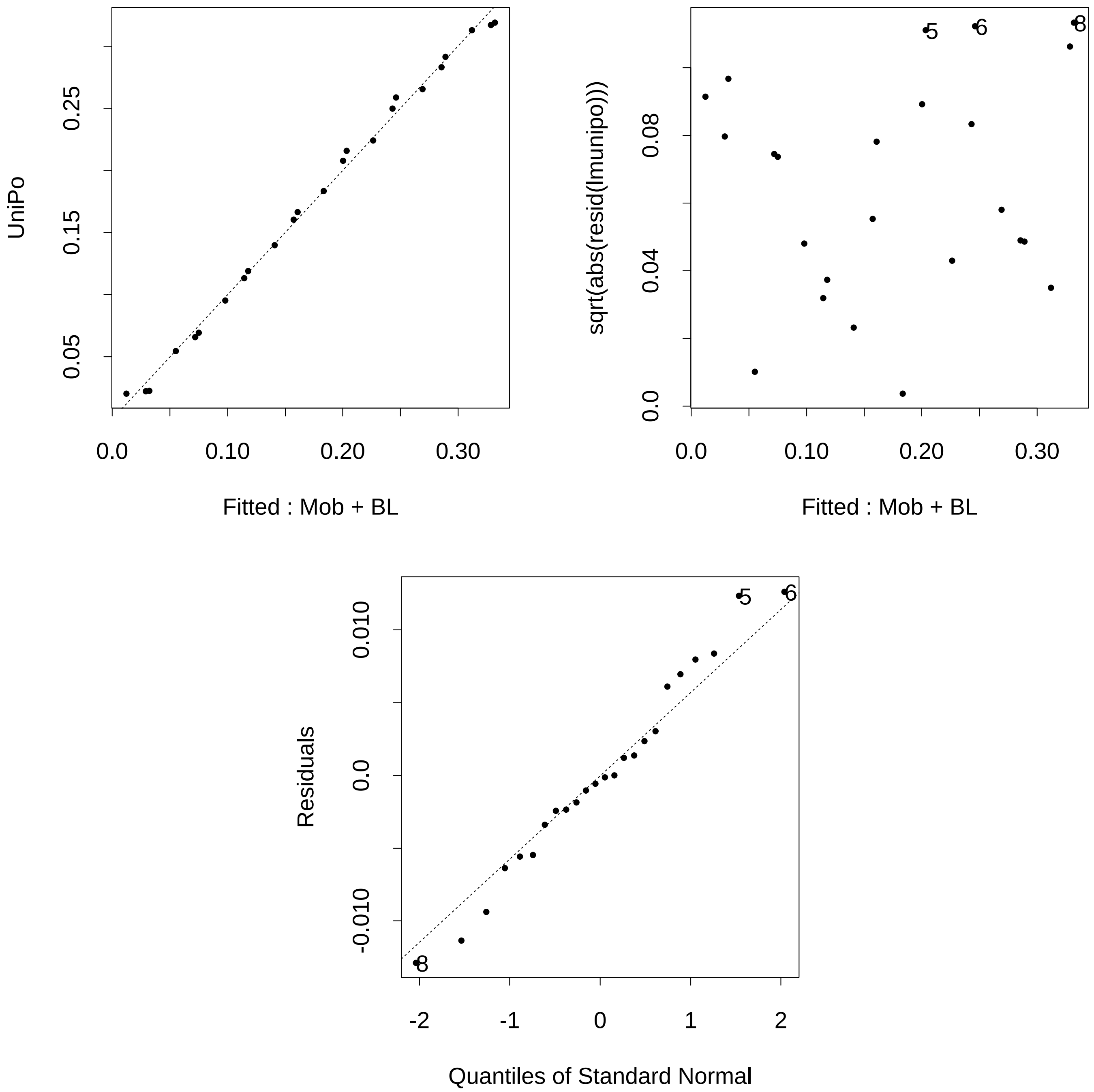


Figure 21: Linear model for predicting p_o with uniform call durations. *Top left panel*: p_o vs. fitted p_o. *Top right panel*: residual error in model vs. fitted p_o. *Bottom center panel*: quantile-quantile plot of residual errors.

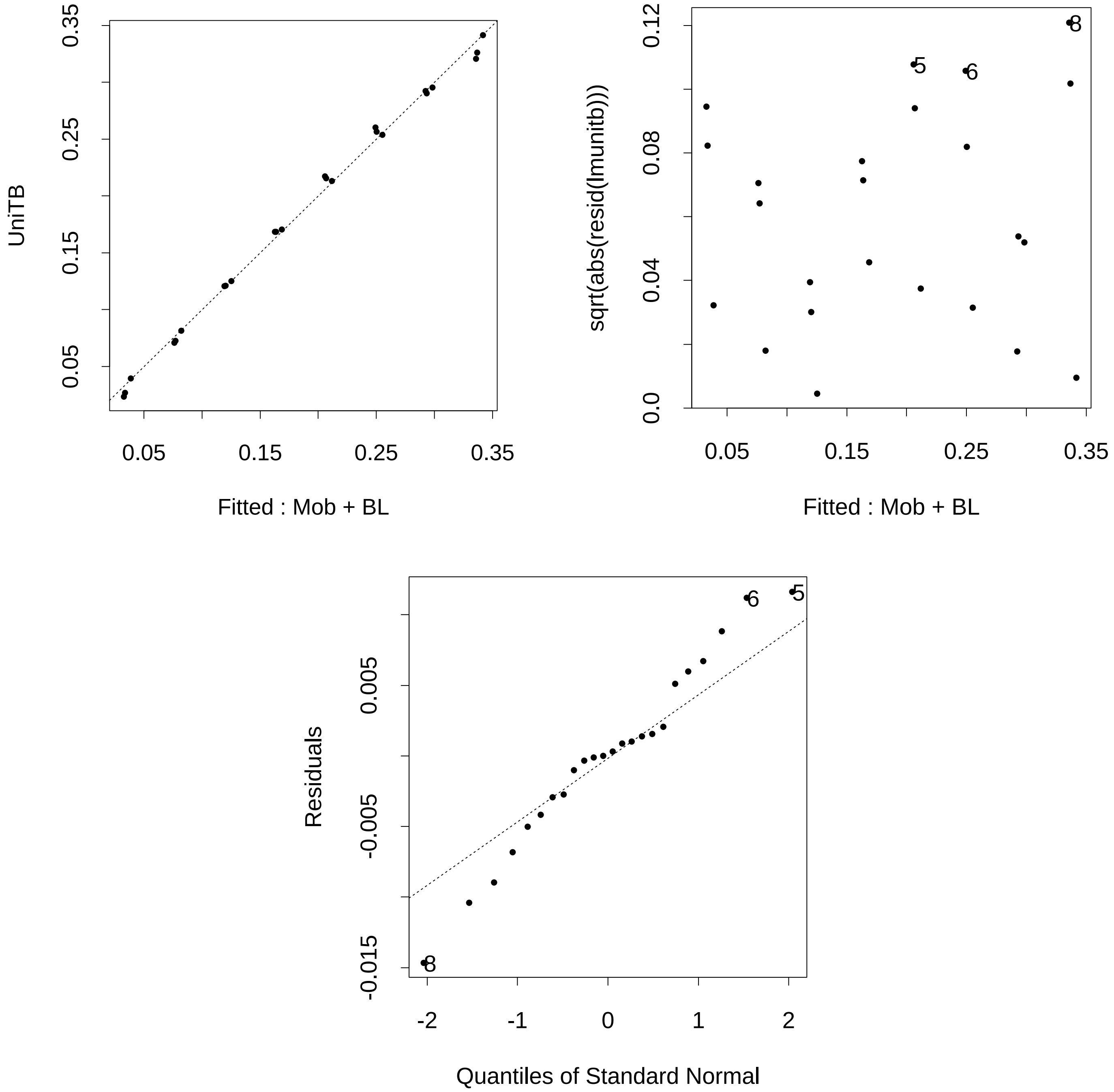


Figure 22: Linear model for predicting p_{nc} with uniform call durations. *Top left panel*: p_{nc} vs. fitted p_{nc}. *Top right panel*: residual error in model vs. fitted p_{nc}. *Bottom center panel*: quantile-quantile plot of residual errors.