SAND2017-4550C

Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

A NOVEL APPROACH TO EXPONENTIAL SPEEDUP OF SIMULATION EVENTS IN WIRELESS NETWORKS

Anand Ganti

Department of Advanced Systems
Sandia National Laboratories
Albuquerque, NM 87185, USA

Uzoma Onunkwo

Department of Advanced Systems
Sandia National Laboratories
Albuquerque, NM 87185, USA

Richard Schroeppel

Department of Quantum Information Sciences
Sandia National Laboratories
Albuquerque, NM 87185, USA

Michael Scoggin

Department of Hardware/Software Contract Support
Sandia National Laboratories
Albuquerque, NM 87185, USA

Brian Van Leeuwen

Department of Cyber Security Initiatives
Sandia National Laboratories
Albuquerque, NM 87185, USA

ABSTRACT

We demonstrate a new approach that yields exponential speedup of communication events in discrete event simulations (DES) of mobile wireless networks. With the trending explosive growth of wireless devices including internet-of-things devices, it is important to have the capability to simulate wireless networks at large scale accurately. Unfortunately, current simulation techniques are inadequate for large scale network simulation especially at high rate of total transmission events due to poor performance scaling. This has limited many studies to much smaller sizes than desired. We propose a method for attaining high fidelity DES of mobile wireless networks that leverages (i)path loss of transmissions and (ii)the relatively slower topology changes in a high transmission rate environment. Our approach guarantees a runtime of $k(r)O(\log N)$ per event, where N is the number of simulated agents, r is a factor of maximum transmission range, and $k(r)$ is typically constant obeying $k(r) \ll N$.

1 INTRODUCTION

Wireless networks are ubiquitous. By wireless networks, we refer to not only network of wireless devices but any network with significant number of wireless devices. The de facto, cost-effective method for studying wireless networks is the use of *discrete event simulation* (DES). A DES of wireless networks has two fundamental event classes: (i) mobility and (ii) communication. Mobility events change the motion velocity of simulated agents, while communication events result in data transmissions and receptions for simulated agents. The simulation of these events poses challenges that are generally not applicable to wired network simulations. The shared medium access for wireless networks and topolgy changes, imply that their simulators need to cater for much more complicated channel models and interference from shared

usage of the communication channel as well as node mobility. Despite these complications, DES remains the only feasible cost-effective approach for understanding the performance of protocols and applications in wireless networks.

The standard approach for simulating wireless networks is inadequate, as it takes prohibitively long to simulate more than a few hundred wireless nodes (agents) on a day scale. In a wireless mobile network with N simulated agents, processing a mobility event just involves updating the motion parameter of a station, such as velocity, and this can be executed in time $O(1)$. However, a data transmission event (communication event) has a complexity of $O(N)$ since it involves querying every agent for potential data reception. Some simulations will schedule a reception event at each agent irrespective of its distance to the transmission. This is due to the inherent broadcast nature of wireless mediums. It is the $O(N)$ cost per communication event that limits current wireless simulations to hundreds of nodes over a day scale because each node can generate thousands of packets per second. Even if many computing cores are used to attain some speedup by implementing *parallel* DES (PDES), the scaling of such simulations is meager because the dynamics in wireless environment are not easily amenable to efficient parallel implementation: high inter-processor communication to per-processor computation ratio. In wireless networks, the average rate of motion events is generally much lower than the average rate for communication events. This implies that we can speed up the simulation substantially if we lower the processing cost of communication events even at the expense of a higher cost for motion events.

We propose an approach that exponentially speeds up the runtime of communication events from $O(N)$ to $O(\log N)$ at the cost of increasing the runtime of motion events from $O(1)$ to $O(\log N)$. The speedup allows for high fidelity simulations of much larger network sizes that would otherwise be too prohibitive in wall-clock time to simulate with current approaches. In the rest of this paper, we describe and analyze an approximation algorithm for faster simulation of wireless networks, called **KMsim** (pronounced as *kim-sim*), which relies on two tree structures: a k-d tree and a modified min-heap (MMH). We then describe various simulations that we performed and discuss their results showing the improved runtime performance without significant loss of simulation fidelity. Finally, we conclude and propose other ideas for speeding up (parallel) discrete event simulation of mobile wireless networks.

2 PROBLEM STATEMENT

Consider a wireless network with N stations[1], where the stations' positions are assumed to change continuously. Most wireless communications systems today are RF-based and inherently occur over broadcast medium. Hence, transmission events by stations cause reception events at every station. A data packet is *successfully* received at a station if the signal-to-interference-plus-noise ratio (SINR) is above a certain threshold. The question is:

How can one simulate communication events faster than $O(N)$ in a DES?

In the standard approach, which we call **BFsim** (pronounced *bif-sim*), a mobility event just involves updating the motion parameters of a station, which can be executed in constant time, $O(1)$. A packet transmission event has a runtime complexity of $O(N)$ because it involves scheduling a packet reception event at every single station. When executing a reception event, the simulation decides a packet reception is successful if the SINR is above a fixed threshold.

In practical wireless networks, in order to maximize throughput, stations typically transmit at low power. The transmit power is large enough to lead to successful reception at a short range (for most mobile ad hoc networks (MANETs), this range is usually less than 100 meters) and small enough that it causes very little interference at moderately larger ranges. Furthermore, the total number of simultaneous transmissions in a localized region is small in order to limit interference in the wireless network and not render the channel unusable (Gupta and Kumar 2000). So the BFsim approach of scheduling a reception event at every station leads to unnecessary events with negligible increase in the fidelity of the simulation.

[1]We use 'stations' and 'nodes' interchangeably to denote the simulated wireless agents.

3 ALGORITHM DESCRIPTION

One simple optimization to wireless network simulations is to omit scheduling of reception events to agents beyond a fixed factor of the transmission range. The truncation of reception events, also called "clipping", forms the first step in the KMsim algorithm.

The first challenge in wireless simulations is the explosion of reception events due to the broadcast nature of the wireless medium. A station transmission leads to a reception in every station in the wireless network. Since the received signal power generally decreases with the square of distance (and even higher exponent in the presence of shadowing), there exists a range beyond which the packet reception event can be ignored as long as there are not too many of them. Our DES simulation studies have shown that there is no loss in simulation fidelity if one adopts a range-based model for transmissions. The concept is extensible to general power-based models by mapping distance-measures to SINR-based measures.

3.1 Clipping

So in our algorithm, a transmission event at space-time (p,t) only causes reception events at stations that are within distance r from p at time t, where r is a design parameter. Our simulation studies (see Section 6) suggest that $r = 4r_{TX}$ is sufficient for a maximum transmission range of r_{TX}. Note that this approximation is valid as long as the node density and more precisely the density of simultaneous transmissions does not grow without bound as the network size increases. *A key assumption that we make is that the node density is invariant with the network size*. As we stated previously, prior research has shown that this must be the case in order for the total network throughput to increase with the network size. In order to see why our assumption breaks down when the node (transmission) density scales with the network size, consider a unit disc with N nodes. Any radius $r \leq 1$ has a node population that scales with $O(r^2)$ while the interference per user scales as $O(r^{-2})$ and the total interference contribution from nodes beyond any r is significant. So in this case a clipping approach would drastically underestimate the interference.

The KMsim algorithm utilizes two key tree structures: (a) a k-d tree for spatial indexing and fast retrieval of neighbor lists during communication events and (b) a modified min-heap (MMH) that tracks motion events that do not require an immediate update of the k-d tree. We considered another spatial indexing tree structure, the *R*-tree* variant (Kriegel, Brinkhoff, and Schneider 1993), for retrieval of neighbor lists, but found that the k-d tree open source implementation called the *Spatial C++ Library* (Bougerel 2017) had better performance than the open source implementations of *R*-tree* that we tested.

A broad desciption of the algorithm is as follows. At initialization of a wireless simulation, all stations positions are mapped into the k-d tree. The MMH is then used to track a crossing time, t_{cross}, for each station, defined as the time at which the station crosses an imaginary sphere of radius r_D centered at the k-d tree location when traveling at its current velocity. Hence for all simulation times less than the t_{cross} for a station, we know its location must be within that sphere and when making a query for neighbors within a transmission range, we can query the k-d tree with range equal to $r_D + r$. When a node crosses its range-sphere its position is updated in the k-d tree. The detailed setup, initialization, and running of our algorithm is described in the following subsections.

3.2 Setup

We have a PDES of a wireless network with multiple Logical Processes (LPs). Each LP is in charge of events originating from a fixed geographical region in the simulation world; this fixed region can contain zero or more wireless stations. A transmission within an LP region can only reach a fixed number of neighboring LP regions and each LP can inform the relevant neighboring LPs. Each simulated wireless agent maintains the following states:

- geographical position stored in the k-d tree p_k, which serves as the key in the k-d tree;
- last updated waypoint position p_l, which is not necessarily the same as p_k;

- last update-time t_l corresponding to p_l;
- current velocity v_l, defined as the vector interpolating the current waypoint position and the next waypoint position;
- next time to update waypoint position & velocity, t_n. This is the time at which a station's position and velocity will be updated;
- crossing-time, t_{cross}. This value is the time at which this station crosses a sphere of radius r_D centered at the k-d tree position (p_k) as long as it is less than t_n. If not, then is set to infinity. The crossing-time is also the key in the modified minimum heap (MMH). The MMH has two data structures: (i) a min-heap and (ii) a hash table. Stations whose current $t_{\text{cross}} < \infty$ are in the min-heap and the rest are in the hash table. When a station's t_{cross} changes to infinity it gets moved from the min-heap to the hash table and when t_{cross} changes from infinity to a finite value, the station gets moved from the hash table back into the min-heap; and
- its position in the min-heap (i_m) for efficient deletion: this last piece is what distinguishes our *modified* min-heap from a standard min-heap.

Note that the position and velocity parameters are vectors.

3.3 Initialization

The simulation is initialized for each LP as follows: for each station inside an LP, define an initial set of values: $p_k = p_l = p_0$, $t_l = 0$, and $v_l = v_0$. The motion events that update stations' velocities are scheduled which populates $\{t_n\}$, some of which could be infinity. The initial positions of stations are used to build a k-d tree. The initial velocity allows for the computation of the initial t_{cross}, which is used to build the MMH in time $O(N)$. Stations with $t_{\text{cross}} < \infty$ are used to build a minimum heap. Stations with $t_{\text{cross}} = \infty$ are stored in a hash-table with their station-id as the key. Note that these include both stationary stations as well as stations whose velocity will be updated before they can cross the range-sphere at the current velocity. The packet transmission events for relevant stations are scheduled.

3.4 Running the Simulation

There are essentially four classes of events during the simulation run: (i) transmission events (Tx) and (ii) motion events (Mv), (iii) station additions (Add) and (iv) station deletions (Del).

i) Tx Event: Consider the execution of a transmission event at space-time of (p,t), i.e., t is the current simulation time. The source LP schedules reception events on itself and neighboring LPs after a delay based on the packet size and bandwidth, that are within the transmit range, r_{TX}. The execution of a reception event at an LP consists of several steps. We first update the current position of all the stations in the LP that have crossed their range-spheres. This is done as follows. We observe the MMH repeatedly examining the minimum key $t_m := \min\{t_{\text{cross}}\}$. If $t_m > t$ we stop, else the corresponding station, which is at the root of the MMH, has crossed the range-sphere. We then update the t_{cross} of the root node and bubble-down. This is more efficient than a delete followed by an insert in a min-heap. In the case where the updated $t_{\text{cross}} = \infty$ we delete this station from the min-heap and insert it in the hash table; this can be done efficiently using the station's tracked position in the min-heap (i_m). For each station whose old t_{cross} was less than t we update the k-d tree geographical position parameter: $p_k \mapsto p_l + (t - t_l)v_l$. We then poll the k-d tree for stations within range $(r + r_D)$ from p. This list will include every station that is within r from p: If a station at time t is within distance r from p and its position at t is within distance r_D from the position in the k-d tree, then by the triangle inequality its position in the k-d tree is within distance $r + r_D$ from p. We do not update the k-d tree geographical position parameter, p_k, of the stations returned by the k-d tree. We schedule a reception if their positions at time t would put them within r of point p.

ii) Mv Event: Consider the execution of a move event at time t on a station. Executing this event might lead to an update of the station's position in the MMH. But contrary to intuition, it might not lead to any change in the k-d tree. The following station parameters: $p_l, t_l, v_l, t_{\text{cross}}$ will get updated in order: $p_l \mapsto p_l + (t - t_l)v_l$, $t_l = t$, and v_l from the mobility model. There are two sub-cases. The first involves a move event of a station within the LP (region associated with the LP). That is, the new velocity of the station is such that it will remain inside the LP until its next move event. In this case there is no change to its position inside the k-d tree. Consider the MMH: if $t_{\text{cross}} \leq t$, implying that the station has already crossed the range-sphere then there is nothing to be done. If the old t_{cross} is greater than t then the new velocity could potentially lower its t_{cross}. As an example consider a previously stationary station that is now moving. The new t_{cross} is obtained using the updated p_l and v_l and the equation:

$$|(p_l - p_k) + |v_l|(t_{\text{cross}} - t)| = r_D \tag{1}$$

In this equation we have used a conservative calculation for t_{cross}, since the true t_{cross} is at least this large. For better accuracy one can solve a quadratic expression derived from vector arithmetic (see Figure 1). Next, the position of this station within the min-heap is appropriately moved up. Note that this is not necessarily a root node of the min-heap and hence its position i_m within the min-heap is utilized to perform this operation in $O(\log N)$. Also, note that the min-heap only consists of mobile nodes, since stationary nodes are stored in the hash table. If the number of mobile nodes in the network is a small fraction of the total population N then this runtime could be constant in practice.

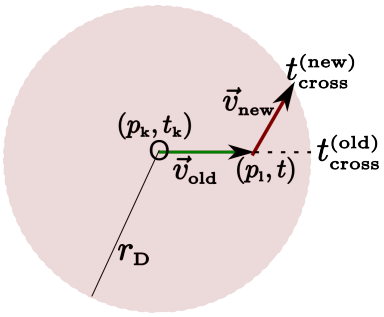


Figure 1: Computing the new crossing time, $t^{(\text{new})}_{\text{cross}}$, involves solving a quadratic from the above geometry.

In the second case, where a station's move event (waypoint change) will result in it exiting the current LP before its next move event, the LP schedules a deletion of the station on itself and an addition on the neighboring LP at the time of the crossover. The t_{cross} update is similar to the case where the station remains within the LP with one exception: if it turns out that the station will cross the current LP before it crosses its range-sphere, then we set t_{cross} to infinity.

iii) Add Event: The LP updates the station's parameters using: $p_l \mapsto p_l + (t - t_l)v_l$, $t_l = t$, $p_k = p_l$, $t_{\text{cross}} = t + r_D/|v_l|$. It then inserts the station in its k-d tree and MMH.

iv) Del Event: The LP deletes the station from the k-d tree and MMH. In the latter it utilizes the station's i_m if t_{cross} is finite.

4 COMPLEXITY ANALYSIS OF KMsim AND BFsim

In this section, we analyze the computational complexity of executing transmission and move events, where move events refer to events that change a node's waypoint. The reception events are clearly $O(1)$ in KMsim as only the single receiving station is served by the simulator.

In the initialization phase, each LP builds a k-d tree and a MMH. For algorithmic efficiency, the k-d tree needs to be close to a balanced tree. A fully balanced k-d tree with N nodes can be built in time $O(N \log^2 N)$

using repeated sort (Wald and Havran 2006, Ooi 1987). Faster constructions in time $O(N \log N)$ exist that give nearly balanced trees that might be sufficient in practice. The MMH is a combination of a min-heap and a hash table. Each of these can be constructed in time $O(N)$ (Cormen 2009, Schoenmakers 1993).

4.1 Cost of a Transmission Event

Consider a transmission event at some point in time. Due to the network size-invariant station density assumption (Section 3.1), we presume that the total number of stations within range $r + r_D$ is bounded by some integer k. The event has two parts – an update of the MMH and an update of the k-d tree. We will show that each of these can be performed in time $k \times O(\log N)$, where N is the total number of simulated agents.

4.1.1 Update of MMH

For each of the k stations, we incur at most the cost of a min-heap update. The per-station computation cost is $O(\log N)$. If the updated t_{cross} is a finite value, then it percolates down the min-heap from the root to its correct position in $O(\log N)$ (*heap bubble-down* operation). If the updated value of t_{cross} is infinity, then it gets deleted from the root of the min-heap which takes time $O(\log N)$ and added to the hash table in $O(1)$.

4.1.2 Update of k-d tree

We use the old k-d tree position key (p_k) to delete the stations within a k-d tree. We reinsert these stations with the updated position p_k. Since the k-d tree is balanced, then this has a per-station cost of $O(\log N)$.

4.2 Cost of Move, Deletion, and Add Events

A station move-event could potentially result in the lowering of t_{cross} and lead to an update of the min-heap. The station position within the min-heap, i_m, is used to compare its key with its parent and percolate the station up the tree (*heap bubble-up* operation). The cost of this operation is $O(\log N)$.

For a deletion event, if the station is in the min-heap, then its position i_m is used to swap with the rightmost leaf of the min-heap, which is subsequently deleted. The station at i_m is then appropriately moved up or down the min-heap to its correct position using its key t_{cross}. The cost of this operation is $O(\log N)$. The deletion to a balanced k-d tree is a $O(\log N)$ operation.

Adding a station to MMH of new LP is an $O(\log N)$ operation: it is the standard heap insertion of adding to the heap bottom and bubbling up. The addition to a balanced k-d tree is an $O(\log N)$ operation.

5 SELECTION OF r_D AND r

The clipping radius for transmission (r) is the distance beyond which we ignore the interference caused from a transmission. It depends on the transmit power, receiver gain and channel conditions. In order to attain high accuracy in simulations, one needs this r to be a multiple of the maximum per-antenna range for a successful simulation of the underlying protocol. The concept of clipping range also assumes that the agents beyond r have a total interference that is negligible. This needs to be verified by analyzing the underlying network applications. The simulation experiments that we have ran, described in Section 6, suggest that $r = 4r_{TX}$ is sufficient.

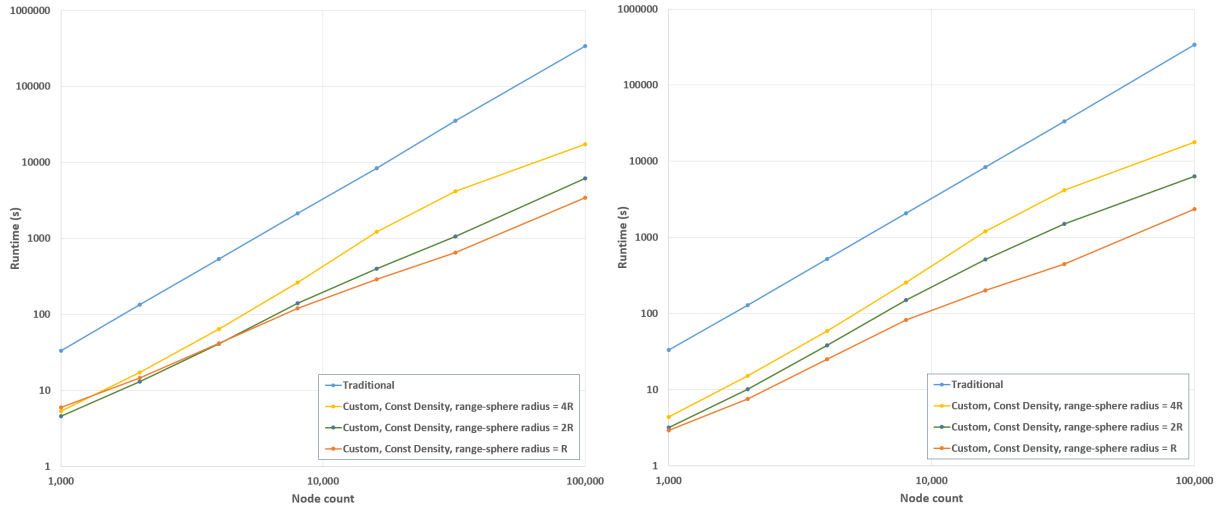
The returned list size from an MMH query is indirectly proportional to r_D, while the returned list size from the k-d tree is directly proportional to r_D. This means that r_D needs to balance the computational cost of querying these two data structures. We want the two lists to be small (constant size) compared to the number of simulated agents in a LP. We have found that setting r_D to be a small multiple of r yields good results. In our simulations we found that, $r_D = r$ outperformed the $r_D = 2r$ and $r_D = 4r$ configurations (see Figure 2).

6 SIMULATION RESULTS

We discuss multiple simulations that show the superior performance of our approach (KMsim) over the standard approach (BFsim). Two of the most commonly used network discrete-event simulators, ns-3 and OPNET, are also used in our simulation result studies. ns-3 simulations demonstrate that KMsim is orders of magnitude faster than BFsim at large network sizes (Figure 3a). We compute the relative error of the end-to-end delay profile between KMsim and BFsim, which shows that KMsim maintains high simulation fidelity with respect to BFsim (Table 1). Simulations are performed on an Intel(R) Xeon(R) CPU E5-2697 @ 2.70 GHz processor and are single-core DES.

6.1 Comparing Runtime of KMsim and BFsim

In this setup, we increase the number of agents in the simulation from 1,000 stations to 100,000 stations. Each agent uses the exponential distribution for time of occurrence of mobility events and communication events. The mean time for communication events per station is set at 1 tick. Two sets of mean time for mobility events (waypoint changes) are studied – (a) mean time for waypoint change is $10\times$ mean time for communication events and (b) mean time for waypoint change is $100\times$ mean time for communication events. In both cases, waypoint positions change slower than the time for a data communication. We use a random walk for the actual waypoints and stop simulation after 1,000 ticks in time. The total wall clock time is proportional to the number of simulation ticks. This means that if one has to estimate the wall clock time taken to simulate a million ticks of the same number of nodes, then one can simply multiply the runtime for 1,000 ticks by 1,000. The simulation world is 3-D with dimensions of 10,000 $\times$ 10,000 $\times$ 100. For the KMsim comparison plots, we run three sets of experiments under the network size-invariant density specification: (i) $r_D = 4r$ (ii) $r_D = 2r$, and (iii) $r_D = r$. The runtime comparison of BFsim and KMsim in the two scenarios is shown in Figure 2.



(a) 10 communication events per waypoint change. (b) 100 communication events per waypoint change.

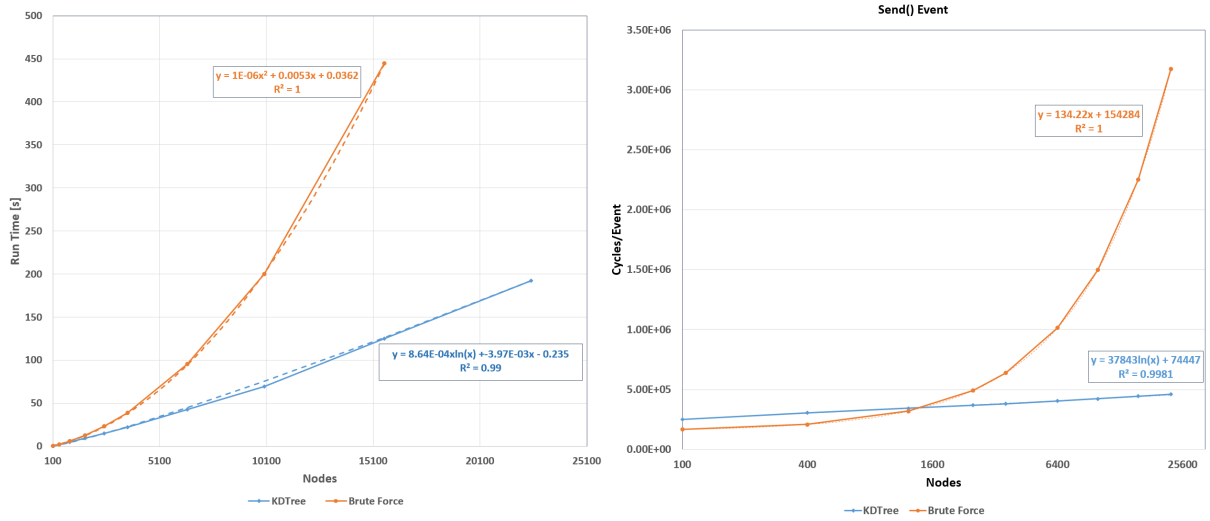
Figure 2: Comparing traditional approaches to our custom approach for simulating mobility and communication events.

As can be seen, the KMsim approach significantly outperforms BFsim approach for all node counts shown. The superiority is even better as the number of communication events per mobility event increases. It is important to reiterate that lower mobility event rate does not imply lack of mobility but rather the frequency of changing direction and speed of movements. Realistic simulations of wireless networks are expected to tend towards higher communication event per mobility. From Fig 2b, we see that at 100,000

nodes, the BFsim takes 340,440 seconds (or 3 days 22 hrs 34 min) to complete the same simulation that only runs in wall clock time of 2358 seconds (or 39 min 18 seconds) for the KMsim with $r_D = r$. Note that BFsim's performance is essentially identical in these setups. However, average density of the network has a significant effect on KMsim as its efficiency is best achieved when the average number of agents within a neighborhood is constant, i.e., independent of the total count of agents in the simulation world. We did not perform a parameter sweep for optimal choices of the range-sphere, but the plots in Figure 2 already show superior performance for the choices of $r_D = r$, $r_D = 2r$, and $r_D = 4r$.

6.2 Comparing Performance on a Typical Network Simulator, ns-3

In this setup, ns-3 was configured to use the range propagation loss model for wireless communications. The network was a *static* grid meaning that all agents were stationary. The reason for choosing a static grid was for simpler verification of simulation correctness and to reduce focus on runtime variations between runs. Every agent has an exponentially distributed communication time with a mean of 10 minutes and a period of 1 day was simulated. Every agent broadcasts a status packet that is received by it neighbors within the transmission range and which leads to no further communication. The runtime results are averaged from 10 independent monte carlo runs. The simulations were performed with and without full-fledged reception event logic. In the simulations, the k-d tree result consists of half of the idea in KMsim; the other half of KMsim is the MMH which was not implemented since the agents are stationary. In both the traditional approach and the k-d tree approach, clipping is used to eliminate scheduling of reception events where the signal-to-interference-plus-noise ratio is insignificant. The comparison of the runtime and cycle counts per event is shown in Figure 3. As seen from Figure 3b, we have an exponential speedup in the

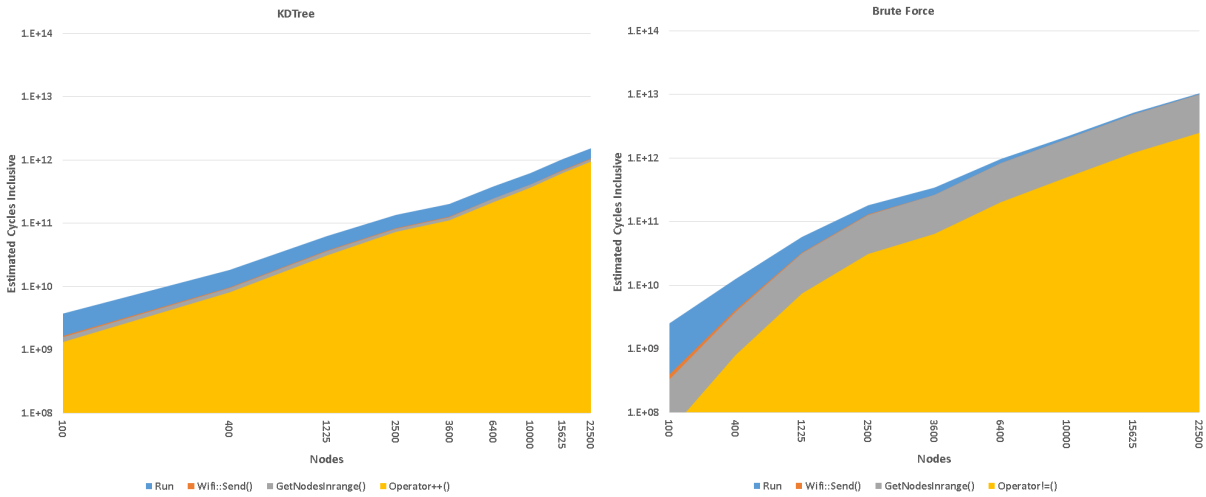


(a) Runtime comparison.

(b) Cycles per event comparison.

Figure 3: ns-3 simulations comparing traditional approach to k-d tree approach for scheduling of reception events (reception events are *no-op* operations).

cycles spent per transmission event. In Figure 3a, this speedup manifests as a runtime of $O(N \log N)$ as opposed to $O(N^2)$, which translates to a speedup of 3.5X at a grid size of 125×125 nodes. For a validation of how this speedup is achieved, we use *Valgrind's KCachegrind* module to get accurate cycle counts for the functional decomposition of the two implementations (see Figure 4). The profile result show that the `GetNodesInrange()` function is predominantly expensive when comparing the traditional approach against the k-d tree approach (i.e., $O(N^2)$ versus $O(N \log N)$), which explains the curve-fits in Fig 3. In the previous results, our emphasis was on the gain in performance strictly based on transmission events,



(a) k-d tree approach.

(b) traditional approach.

Figure 4: Cycles breakdown view of basic operations in the ns-3 simulation.

where clipping has been employed and reception event logic are replaced by no-op operations. To give a better treatment of the true performance gain due to clipping, we use the traditional approach to DES and show the speedup gained from clipping. As seen from Fig 5, clipping provides a high simulation speedup

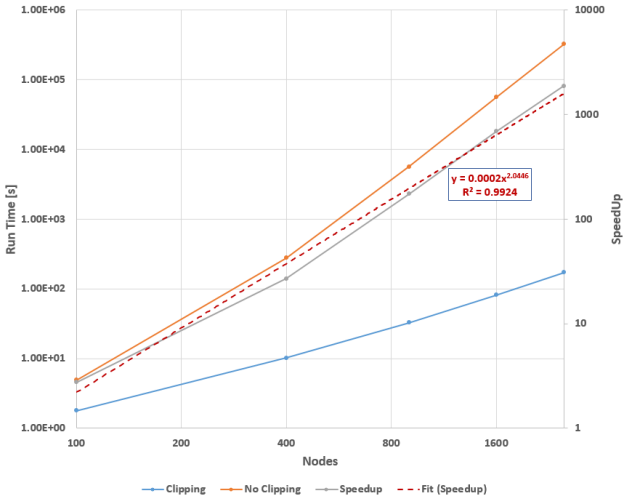


Figure 5: ns-3 wireless network simulation speedup due to clipping.

reaching about 2,000X at 50×50 nodes. Of course, this speedup will be meaningless if the simulation fidelity is lost; however, as we will demonstrate in Section 6.3, the fidelity of the simulations tend to stay true at moderate clipping range when compared to the default no clipping.

6.3 Simulation Fidelity after Clipping

Up to this point, we have shown the superior runtime performance that the techniques of KMsim achieves when compared to traditional (aka BFsim) methods. However, it is important to investigate the potential effects on the simulation fidelity when using the KMsim approach. In this subsection, we analyze the effect of clipping the transmissions in a wireless setting on the fidelity of a wireless network simulation.

We have a wireless grid topology identical to what was described in Section 6.2 but we use OPNET as the simulator, where the transmission range is set to 400 meters and the grid is only of size 20×20. The nodes are stationary and use Dynamic Source Routing (Johnson 1994, Johnson and Maltz 1996). The metric being measured is the end-to-end delay between packets originating from the top-left corner (source) to the bottom-right corner (destination) of the grid. The source node transmits every ten seconds. There is additional background traffic generated by the rest of the nodes to random destinations (not including the two nodes of interest). This background traffic has an inter-packet generation time that is uniformly distributed between [9,10] seconds. The packet size is constant and slightly above 200 bits and the wireless data rate is 1 Mbits/s. We explored various clipping ranges ranging from 500 m to 9,000 m. The simulation runtime ranged from approximately five minutes (500 m) to about two hours (9,000 m).

We periodically collect end-to-end delay time. We compare the delay profile results with and without clipping, noting again that the default behavior of most wireless network simulators is no clipping. The metric used for this comparison is the cumulative distribution function (CDF) of the end-to-end delay. Figure 6 shows the delay profile distribution for the baseline (no clipping) and various clipping regimes.

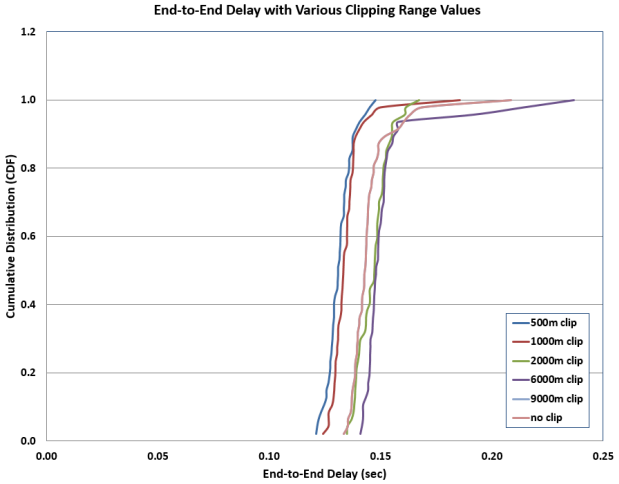


Figure 6: Effect of clipping on wireless network simulation fidelity.

We also computed the relative error (RE) defined as:

$$\frac{\sqrt{\frac{1}{n}\sum_i (\hat{x}_i - \hat{y}_i)^2}}{\sum_i x_i / n} \tag{2}$$

between a clipped simulation ($\{y_i\}$) and the baseline ($\{x_i\}$) for various clippings where the 'hat' ($\hat{}$) denotes the sorted vector of size, n. The relative error is expected to monotonically decrease on average with respect to the clipping range. In these simulations, we used 10 independent samples to generate the average values. Notice that even at 500 m, the RE is already below 25%. Table 1 compares the end-to-end delay at various clipping ranges, which is a subset of the clipping ranges we examined. Table 1 also shows agreeable range

Clipping	Baseline(∞)	500 m	1,000 m	2,000 m
Mean	1.45E-01 s	1.32E-01 s	1.35E-01 s	1.43E-01 s
Stdev	5.86E-02 s	2.91E-02 s	4.43E-02 s	3.05E-02 s
Rel.Err.	–	2.44E-01	1.80E-01	2.35E-01

Table 1: Comparison of end-to-end delay at various clipping ranges with baseline (no clipping)

of values for the end-to-end delay profile and the relative errors are negligible as comparison between

different set of pseudo-random seeds within the "no clipping" can yield similar variations. For this study, OPNET was used due to the ease of generating the delay profile CDFs; however, in Section 6.2 we used ns-3 because of our familiarity with its source code.

7 CONCLUSION AND FUTURE WORK

We have described a new approach for rendering previously impractical wireless network simulations practical. The runtime complexity of current approaches for simulating large wireless networks limits the size of current studies. Using our approach, we have demonstrated exponential speedup of communication events. This translates to a performance improvement that grows with the count of simulated agents, which is two orders of magnitude improvement at 100,000 agent count. In general, our approach has a $O(N \log N)$ runtime as opposed to $O(N^2)$ runtime of traditional approaches. The key data structures used in our approach are: (i) a min-heap used to identify when station positions need to be updated and (ii) a k-d tree used to compute the neighbors of a data transmitting agent.

We are currently considering alternatives to these data structures in order to further lower the runtime cost. The alternative data structures are a Weighted Radix Priority Queue (WRPQ) for the min-heap and a Cell-Array for the k-d tree. The WRPQ stores the range-sphere crossing times in a collection of coarsely sorted buckets that we expect to have an amortized processing cost of $O(1)$ per communication and move event. The Cell-Array divides the entire simulation region under an LP into cells and stores the stations within a cell in an array of linked lists. The area of a cell is close to the effective area of a transmission. Hence transmission events within a cell affect only neighboring cells and if the station density is constant then the amortized cost of computing the list of affected stations will be $O(1)$. We hope to thoroughly analyze this approach in the future.

This work focuses on the simulation of wireless networks and we predict that our approach has immediate applicability to other simulations where agents are mobile and frequent interactions exist between agents on the basis of neighborhood as seen in disease propagation and other models.

Many wireless networks have humans as the underlying agents. Various research studies on human motion have demonstrated that human motion does not fit the random walk that is typically used in simulations. Human motion is fairly predictable, has strong correlations, and is often distributed in clusters (as opposed to uniformly), where these clusters can be work and home locations. This information can be leveraged by PDES to partition the simulation region in order to reduce inter-LP communication thereby improving simulation runtime. Thus, research in graph-partitioning algorithms can inform good load-balancing of mobile wireless networks PDES.

REFERENCES

Sylvain Bougerel 2017. "Spatial C++ Library: Generic Multi-Dimensional Containers and Spatial Operations". Accessed: 2017-04-02, URL: http://spatial.sourceforge.net/.

Cormen, T. H. 2009. *Introduction to algorithms*. MIT press.

Gupta, P., and P. R. Kumar. 2000. "The capacity of wireless networks". *IEEE Transactions on information theory* 46 (2): 388–404.

Johnson, D. B. 1994. "Routing in Ad Hoc Networks of Mobile Hosts". In *IEEE Workshop on Mobile Computing Systems and Applications*, 158–163. IEEE Computer Society.

Johnson, D. B., and D. A. Maltz. 1996. *Mobile Computing*. Kluwer Academic Publishers.

Kriegel, H.-P., T. Brinkhoff, and R. Schneider. 1993. "Efficient Spatial Query Processing in Geographic Databse Systems". *IEEE Data Engineering Bulletin* (3): 10–15.

Ooi, B. C. 1987. "Spatial kd-tree: A data structure for geographic database". In *Datenbanksysteme in Büro, Technik und Wissenschaft*, 247–258. Springer.

Schoenmakers, B. 1993. "Inorder traversal of a binary heap and its inversion in optimal time and space". In *Mathematics of Program Construction*, 291–301. Springer.

Wald, I., and V. Havran. 2006. "On building fast kd-trees for ray tracing, and on doing that in $O(N \log N)$". In *Interactive Ray Tracing 2006, IEEE Symposium on*, 61–69. IEEE.

AUTHOR BIOGRAPHIES

ANAND GANTI is a member of technical staff in Advanced Systems at Sandia National Laboratories in Albuquerque, NM (USA). He works in the areas of algorithms, information systems, and mathematical modeling. His e-mail address is aganti@sandia.gov.

UZOMA ONUNKWO is a member of technical staff in Advanced Systems at Sandia National Laboratories in Albuquerque, NM (USA). He has years of experience in network simulations, high performance computing, and information processing systems. He was a co-author on a conference paper for the 2013 IEEE International Conference on Computing, Networking, and Communications, which won an IEEE Best Paper award. His email address is uonunkw@sandia.gov.

RICHARD SCHROEPPEL is an expert in Cybersecurity R&D. His email address is rschroe@sandia.gov.

MICHAEL SCOGGIN is a computer engineer with expertise in scalable architectures and algorithms. His email address is mpscogg@sandia.gov.

BRIAN VAN LEEUWEN is an expert in Cybersecurity R&D. He has led many high fidelity simulation projects for high consequence systems, with emphasis on cyber security. His email address is bpvanle@sandia.gov.