Probabilistic Event-Time Geometry: Formalization and Applications

Recap: Deterministic Event-Time Geometry Foundations

Events and Coordinates: In the deterministic formalism, an **event** is defined as a point $e = (l, t)$ with a spatial location l and a timestamp t [1]. The set of all locations L comes with a distance function $d(l_1, l_2)$ capturing spatial separation or communication latency between locations [2] [3]. The time domain T is typically the real numbers, so each event has a real-valued time coordinate [4] [5]. For example, in a distributed system or neuromorphic hardware, l might label a processor or neuron, and t the clock time it generates a spike or message.

Causal Order and Delay: A partial order of events is imposed by causality. **Event e_1 can cause event e_2** (written $e_1 \to e_2$) only if e_2 occurs *after* e_1 with enough time for a signal to travel from e_1's location to e_2's location [6]. Concretely, if $e_1=(l_1,t_1)$ and $e_2=(l_2,t_2)$ with a maximum signal speed c, then a necessary condition for $e_1 \to e_2$ is:

$$t_2 \ge t_1 + \frac{d(l_1,l_2)}{c} \,. $$

This captures the *causal delay constraint*: the time difference $\Delta t = t_2 - t_1$ must at least equal the propagation time between locations ${d(l_1,l_2)}/{c}$ [7]. If this inequality is violated, e_1 cannot influence e_2. If it holds with equality ($t_2 = t_1 + d/c$), the events are just on the edge of the causal limit (analogous to a light-speed signal arriving exactly on time).

Event-Time Interval: To quantify separation between events, deterministic event-time geometry defines an **interval** analogous to the spacetime interval in relativity. Given two events $e_1=(l_1,t_1)$ and $e_2=(l_2,t_2)$, define the time difference $\Delta t = t_2 - t_1$ and spatial distance $\Delta x = d(l_1,l_2)$. The **event-time interval squared** is [8] [9]:

$$ (\Delta s)^2 = c^2(\Delta t)^2 \;-\; (\Delta x)^2 \,. $$

This Minkowski-like metric categorizes event pairs. If $(\Delta s)^2 > 0$ with $\Delta t > 0$, the separation is *timelike* and e_1 lies inside e_2's future causal cone (meaning e_1 could potentially influence e_2) [10]. If $(\Delta s)^2 < 0$, the separation is *spacelike* – the events are too far apart in space relative to their time gap to be causally connected (outside each other's light-cone) [11]. The boundary case $(\Delta s)^2 = 0$ is *lightlike*, indicating e_2 arrives exactly on the fastest possible signal from e_1. This construction mirrors special relativity's invariant interval, and indeed an **Event-Time Invariance Theorem** holds: $(\Delta s)^2$ is the same in all inertial frames (under Lorentz-like transformations of x and t) [12] [13]. In other words, different observers (e.g. different clock synchronizations or reference frames) agree on the value of $c^2\Delta t^2 - \Delta x^2$, preserving the causal structure.

Deterministic Causal Cones: As in spacetime, one can define **causal cones**. Given an event e_1, its *future cone* consists of all events that could be influenced by e_1 (timelike or lightlike separated with later time), and its *past cone* consists of events that could influence e_1. Events outside these cones are spacelike-separated from e_1 and have no causal relation [10]. Deterministically, these cones have sharp boundaries at $\Delta s^2 = 0$ (the "speed-of-light" surface).

Incorporating Probability into Event-Time Geometry

To model real-world uncertainties, we **extend event-time geometry to a probabilistic domain**. In practice, event times and even locations might not be known exactly or may vary due to noise (e.g. clock jitter, sensor noise, variable delays). We replace fixed coordinates and intervals with random variables or distributions, introducing **stochastic definitions** for events, distances, and causal relationships. Below we develop each core concept in turn, infusing probability into the formalism:

Probabilistic Event Coordinates as Random Variables

In a probabilistic extension, an event is no longer a perfectly localized point in $L\times T$, but rather a **random variable or distribution over possible coordinates**. We can represent a single event E as a pair of random variables (L_E, T_E) taking values in L and T. For example, a neuromorphic sensor's spike might have a nominal timestamp t with some jitter, modeled as $T_E = t + \xi$ where ξ is a zero-mean random offset [14]. Likewise, if there is uncertainty in *where* an event occurred (e.g. a signal could originate from one of multiple nodes), L_E can be a random location. In general, an event becomes a **probability distribution ("cloud") in space-time** rather than a delta function at (l,t). We might denote an event's probability density as $P_E(l,t)$ over $L\times T$, or characterize it by parameters (such as a mean location and time and a covariance matrix capturing spread). This *event cloud* picture acknowledges that we only know an event's coordinates within some error bounds or distributions.

- **Example – Gaussian Event Cloud:** Suppose an event's timestamp is measured with error. We can model $T_E \sim \mathcal{N}(\mu_{t},\,\sigma_t^2)$ as a Gaussian centered at μ_t with variance σ_t^2, and its location as fixed $L_E=l$ (if location is known exactly) or also Gaussian around some mean position. Then the event is represented by a "blob" around (l,μ_t) in the space-time diagram, rather than an exact point. The smaller σ_t (and spatial variance), the tighter the cloud and the closer we are to the deterministic case.

By treating events as random coordinates, we can propagate this uncertainty through the geometry. Each quantity derived from event coordinates (time differences, distances, intervals, etc.) becomes a random variable derived from the underlying distributions. This lays the groundwork for *stochastic metrics and relationships* in the next sections.

Stochastic Distances and Separation Distributions

In deterministic geometry, the spatial separation $\Delta x = d(l_1,l_2)$ is a fixed number given two locations. In a probabilistic setting, if one or both location coordinates are uncertain, **distance itself can be a random variable**. For instance, if the location of e_1 or e_2 is not exact, $d(L_{e1}, L_{e2})$ will have a distribution. Even if locations are fixed, variability in signal propagation speed or path can introduce randomness into effective distance (think of network messages that sometimes take longer routes). We therefore consider a **distribution of spatial separation** $D_{12} = d(L_{e1}, L_{e2})$. Often the spatial

distance is known (e.g. two servers at fixed positions), but the *communication latency* may fluctuate around a base value. We can encapsulate such variability by treating the signal propagation speed c or the link latency as stochastic. For example, if the nominal propagation delay between two nodes is d/c, one might model the actual delay as $(d/c) + \chi$ where χ is a random delay (network jitter). This effectively means (L_{e1}, L_{e2}) have a distribution of *separation in time* beyond the deterministic minimum.

Distribution of Intervals: Given two probabilistic events, the time difference $\Delta T = T_{e2} - T_{e1}$ and distance $D_{12} = d(L_{e1}, L_{e2})$ are random, so the interval $(\Delta S)^2 = c^2 (\Delta T)^2 - (D_{12})^2$ becomes a random variable as well. Instead of a single invariant value, we get a *distribution* over interval values. We can characterize this distribution via its probability density or moments (mean, variance, etc.). Notably, the sign of $(\Delta S)^2$ (positive, zero, or negative) may not be certain – there will be some probability that two given events are timelike vs spacelike separated. This yields the concept of **fuzzy causal classes**: for example, *events e_1,e_2 might be timelike separated with 90% probability* (if most of the distribution of $(\Delta S)^2$ lies above zero) and spacelike 10% of the time. This contrasts with the crisp yes/no classification in the deterministic case.

Probabilistic Causal Delay Constraints

When events have uncertain times and distances, the causal ordering condition $t_2 \ge t_1 + d(l_1,l_2)/c$ can be enforced **in a probabilistic sense**. We define the **causal influence probability** as:

$$P(e_1 \to e_2) \;=\; P!\Big(T_{e2} \ge T_{e1} + \frac{d(L_{e1},L_{e2})}{c}\Big)\,. $$

This is the probability that e_2 occurs *after* the earliest possible signal from e_1. If $P(e_1 \to e_2)$ is high (near 1), we have strong confidence that e_1 could causally affect e_2. If it's low, then almost surely e_2 happened too soon for e_1 to matter. In a fully stochastic view, **causal relations become graded** – we speak of a degree of belief or confidence that one event precedes and can influence another, rather than an absolute certainty.

Such probabilistic causal constraints naturally arise in distributed systems with clock uncertainty. For example, consider two servers logging events with imperfectly synchronized clocks. We might only be able to say "event A happened before event B with 95% probability" given their timestamp distributions and known clock drift. Recent work formalizes this with relations like *"likely-happened-before"* which assigns a probability that one event precedes another, extending Lamport's happened-before relation [15]. This new relation acknowledges that two events considered concurrent under a deterministic view can often be ordered probabilistically if we incorporate clock offset distributions [15]. The trade-off is that this relation may be *intransitive* (A before B likely, B before C likely, but A before C not as likely) [16] [17], meaning the causal graph becomes a weighted directed acyclic graph with probabilities on edges rather than a strict partial order.

In practical modeling, we might set a threshold to decide a causal link: e.g. if $P(e_1 \to e_2) > 0.99$, we treat e_1 as a cause of e_2 with high confidence. Alternatively, we carry the probability weights and use them in further analyses (as discussed under causal confidence metrics). The key point is that **causal delay is no longer a single number $\delta(e_1,e_2)$, but a distribution** of possible delays (with some probability mass at "no valid causal link" if e_2 precedes e_1). We can talk about the expected causal delay $E[\delta]$ or quantiles of the delay distribution (e.g. "with 95% confidence, the causal delay is at least 2ms").

Stochastic Event-Time Intervals and Fuzzy Lightcones

With ΔT and ΔX as random variables, the **event-time interval** $(\Delta S)^2 = c^2(\Delta T)^2 - (\Delta X)^2$ inherits uncertainty. We get a **distribution of $(\Delta S)^2$** values rather than one invariant scalar. This leads to a *stochastic causal structure*: instead of absolutely timelike or spacelike separations, we have probabilities for each category. We define:

- $P_{\text{timelike}} = P((\Delta S)^2 > 0 \text{ and } \Delta T > 0)$, the probability that e_2 lies inside e_1's future lightcone.
- $P_{\text{spacelike}} = P((\Delta S)^2 < 0)$, the probability that e_1 and e_2 are effectively spacelike (no causal overlap).
- $P_{\text{lightlike}} = P((\Delta S)^2 = 0)$, which in a continuous model might be 0 unless we specifically allow a degenerate case; more realistically we talk about P being very close to the lightlike boundary.

These probabilities define a **fuzzy lightcone** boundary between causal and acausal regions. We can visualize a *stochastic causal cone*: instead of a sharp cone dividing "possible influence" from "impossible", we get a cone with blurry edges. For example, due to timing uncertainty a few events that would classically lie just outside the lightcone might still be influenced with a small probability. Conversely, some events inside the classical cone might occur too early (due to clock errors or unexpected delays) and thus fail to be influenced in some realizations. The concept of a **fuzzy lightcone** has analogies in physics (e.g. quantum or Planck-scale effects that blur the lightcone structure [18] [19]), but here it arises from classical uncertainty in event timing. As the uncertainty in timing/position shrinks, the fuzzy lightcone sharpens and approaches the deterministic case [19]. If the uncertainty is large compared to distances and times of interest, the causal structure becomes very smeared, and we must speak in terms of likelihoods of causal connection rather than definite cones.

In summary, by introducing probability into event coordinates, distances, and intervals, we transform the crisp geometry of events into a **stochastic geometry**. Every event is a cloud, every measurement an estimate, and every causal relation a probability. Next, we explore the new mathematical structures and invariants that emerge from this extension.

New Structures in Stochastic Event-Time Geometry

Introducing randomness into event-time geometry not only generalizes existing definitions, but also **enables new mathematical structures** and analogies. We highlight three key developments: stochastic causal cones, metrics on distributions (information geometry), and a generalized notion of Lorentz invariance for probabilistic laws.

Stochastic Causal Cones and Fuzzy Lightcones

In deterministic space-time, a light cone cleanly delineates the causal influence region of an event. In the probabilistic version, we define a **stochastic causal cone** for an event e_1 as the region of space-time where a future event e_2 falls with a given high probability p of being causally reachable from e_1. For instance, we might define the 90% causal cone of e_1 as ${(l,t): P((l,t) \text{ is influenced by } e_1) \ge 0.9}$. This region will typically be inside the classical lightcone of e_1 (if $p<1$), because some points near the edge have less than 100% chance of being reached due to uncertainty. As p is lowered, the cone

expands outward, approaching the classical lightcone in the limit $p \to 0$ (where even far points have a tiny but nonzero probability of influence if we allow extreme timing fluctuations).

Fuzzy Lightcone Visualization: Imagine drawing a lightcone but with a blurred boundary rather than a sharp line. The "blur" width corresponds to the uncertainty. One way to formalize this is via *level sets* of the influence probability function. If $f_{e_1}(l,t) = P(e_1 \to (l,t))$ is the probability that an event at (l,t) lies in e_1's future (i.e. e_1 can reach it), then for any threshold p, the set ${(l,t): f_{e_1}(l,t) = p}$ forms a fuzzy lightcone boundary at that confidence level. Closer to e_1 in time and space, $f_{e_1} \approx 1$ (inside the cone); far outside, $f_{e_1} \approx 0$. The transition region is where f_{e_1} drops from near 1 to near 0, indicating increasing uncertainty about causal connection. We could define an effective "speed" slightly above c for high-probability influence (say 99%), reflecting that with small probability a signal might arrive a bit sooner than the mean due to fluctuations (or more generally, to cover 99% of outcomes we might need to allow slightly faster-than-c propagation in rare cases, e.g. if c is mean speed and jitter sometimes shortens delay). This does **not** violate physical c – it's just acknowledging the distribution of delays.

Mathematically, one could model the fuzzy lightcone boundary by convolving the deterministic lightcone (a step function in $(\Delta s)^2$ space) with the distributions of ΔT and ΔX. The result is a smooth function rather than a step. For example, if $T_{e2}-T_{e1}$ has distribution with density $f_{\Delta t}$ and $d(L_{e1},L_{e2})$ is a constant for simplicity, then $P(e_1 \to e_2) = P(\Delta T \ge d/c) = 1 - F_{\Delta t}(d/c)$ (one minus the CDF at d/c). If ΔT is normally distributed with mean μ and stdev σ, this becomes $P(e_1 \to e_2) = 1 - \Phi!\Big(\frac{d/c - \mu}{\sigma}\Big)$ (where Φ is the standard normal CDF). This smoothly transitions from 1 (when d/c is far below μ) to 0 (when d/c is far above μ). Such a function can be interpreted as the fuzzy lightcone profile along that spatial direction. In more complex cases with uncertain distance, a double integral over both time and space distributions would be used.

In short, **stochastic causal cones generalize the idea of lightcones by assigning a probability to each point in space-time being causally connected**. They capture the "maybe yes, maybe no" nature of causality under uncertainty, and can be visualized as lightcones with uncertainty bands. These structures may be especially relevant in analyzing systems like wireless networks or neuroscience, where timings are inherently noisy, creating a probabilistic causal structure.

Metrics on Distributions in Event-Time Space (Information Geometry)

When events are represented as probability distributions ("clouds") rather than points, it becomes natural to ask: how do we measure the *distance* between two such fuzzy events? This calls for a **metric in the space of probability distributions** on event-time coordinates. A rich mathematical framework for this is *information geometry*, where one treats probability distributions as points on a manifold and uses measures like the Fisher information metric to quantify distances [20].

Fisher Information Metric: If an event E is described by a parametric family of distributions $p(l,t;\theta)$ (with θ a vector of parameters, such as mean time and variance), the Fisher information metric defines an infinitesimal distance between $p(\theta)$ and $p(\theta + d\theta)$ as:

$$ g_{ij}(\theta) = \mathbb{E}*{(l,t)\sim p(\theta)}!\Big[\partial* \ln p(l,t;\theta)\Big] \,. $$} \ln p(l,t;\theta)\; \partial_{\theta_j

This metric g_{ij} measures how sensitively the distribution changes with its parameters. Intuitively, if two event distributions are very similar (e.g. two spikes with almost the same uncertain time), the distance between them in the Fisher metric will be small; if they are easily distinguishable distributions, the distance will be larger. The Fisher–Rao distance between two distributions can then be obtained by integrating this metric along a path between their parameter points [20] .

In our context, consider two probabilistic events E_1 and E_2 with distributions $p_1(l,t)$ and $p_2(l,t)$ over $L\times T$. We could define a *distance* $d_{\text{dist}}(E_1, E_2)$ that reflects how different these distributions are. Several choices exist:

- **Fisher information distance:** If we restrict to a parametric form (say both are Gaussian clouds with known variance), we can compute the Fisher metric. For two Gaussians with means (l_1,t_1) and (l_2,t_2) and a common covariance, this yields a distance akin to the Mahalanobis distance between the mean vectors (normalized by the variance). In fact, if both E_1 and E_2 are Gaussian with identical covariance Σ, the Fisher distance corresponds to $\sqrt{\Delta \mu^\top \Sigma^{-1} \Delta \mu}$ for small differences, which resembles a Minkowski-like distance if time and space uncertainties are accounted appropriately.

- **Wasserstein (Earth-Mover) distance:** Another intuitive metric for distributions is the Wasserstein distance, which in the context of event coordinates would ask: what is the "cost" of transporting the probability mass of event E_1 to match E_2? If events are almost surely at specific points (deterministic events), Wasserstein distance reduces to ordinary metric distance (e.g. Minkowski distance in space-time). For fuzzy events, it measures the discrepancy taking into account geometry of $L\times T$. For example, if E_1 and E_2 have distributions centered a few milliseconds apart in time, the Wasserstein distance will roughly reflect that time difference (if spatial location distributions overlap).

- **KL-divergence or other divergences:** While not symmetric (hence not a true metric without symmetrization), measures like Kullback–Leibler divergence $D_{\text{KL}}(p_1 || p_2)$ quantify how one event's distribution differs from another's. These can be used to define *information-theoretic distances* or to optimize alignments of events. For instance, one might say an event E_2 is an "updated" version of E_1 if their distributions are close in KL-divergence.

The **choice of metric** depends on what aspects are important. The Fisher information metric is attractive because it is invariant under reparameterizations and has connections to the Cramér–Rao bound (it relates to how well we could estimate parameters of an event given observations). In practical terms, if we consider the *space of all possible event coordinates distributions*, the Fisher metric endows it with a Riemannian geometry. Interestingly, some research has modeled entire event universes with information geometry; e.g., representing events in a high-dimensional space and finding that only a few parameters capture the shape complexity of their distribution, embedding them effectively in a 4D manifold [21] [22] . This hints at deeper connections between probabilistic event-time geometry and the fundamental geometric structures of general relativity or cosmology (where one often deals with distributions of matter and events in space-time).

In summary, **metrics over distributions** allow us to compare uncertain events quantitatively. Two events might be far apart in classical terms but if each is very uncertain, their distributional overlap might be significant, making their *statistical distance* smaller than the raw distance. Such metrics can be used to

cluster events, detect outliers (events that do not fit the expected timing/location distribution), or to define *geodesics* (paths of gradually changing distributions, possibly relevant for tracking how an event's uncertainty evolves over time or through transformations).

Stochastic Lorentz Invariance and Distributional Laws

In deterministic event-time geometry, Lorentz-like transformations (shifting frames at constant relative velocity) preserve the interval $(\Delta s)^2$ exactly [12]. In the probabilistic setting, since $(\Delta s)^2$ is now a random variable, we cannot demand a single value to remain fixed. Instead, we require that **the statistical law of the interval is preserved**. This means that if we have two events with certain distributions in one frame, and we transform to another frame, the *distribution* of the interval and other invariant quantities remains the same (even though individual coordinates change).

Transforming Distributions: Consider an event E with distribution $p(l,t)$ in frame F. Under a frame transformation (e.g. a stochastic Lorentz boost) to frame F', an event originally at deterministic coordinates (x,t) goes to (x',t') given by the usual formulas $t' = \gamma(t - v x/c^2)$, $x' = \gamma(x - v t)$ [13]. If (L_E,T_E) is random in F, we can derive the distribution in F' by a change of variables: $p'(x',t') = p(x,t)\,|J|^{-1}$ where $|J|=1$ for this linear transform (so essentially $p'(x',t') = p(x(x',t'),\,t(x',t'))$). Now, what does *stochastic Lorentz invariance* entail? At a minimum, it means that any physically meaningful probability statement is frame-independent. For example, if in frame F we say "the probability that e_2 lies in e_1's causal future is 0.95", this should also hold true in frame F' – the event of e_2 being causally after e_1 is coordinate-independent. Indeed, since causality itself is invariant (no frame can reorder causally related events), the probability of $e_1 \to e_2$ should be an invariant. In our formalism, $P(e_1 \to e_2) = P(T_{e2} \ge T_{e1} + d(L_{e1},L_{e2})/c)$ should be the same no matter which inertial frame we use to evaluate it. This is a consistency check: it *must* hold if our probabilistic model is deriving from physical reality, otherwise two observers would disagree on causal likelihoods which would be strange (aside from issues of prior knowledge).

A stronger condition of stochastic invariance is that the *full distribution* of the interval $(\Delta S)^2$ is invariant under frame changes. In deterministic relativity, Δs^2 is a scalar invariant. In the probabilistic extension, Δs^2 is a random scalar; under transformations, each realization's value stays the same (since for each sample of the random variables, $(\Delta s)^2$ computes to the same number regardless of frame). This implies that the *multivariate distribution* of $(\Delta t, \Delta x)$ is such that $c^2 (\Delta t)^2 - (\Delta x)^2$ yields the same set of values in all frames. In practice, if we generate many realizations of two events according to their distribution in frame F and compute $(\Delta s)^2$ for each, we'd get a histogram of values. Doing the same in frame F' will produce an identical histogram. This is true trivially if we directly transform each realization (since each pair's Minkowski interval is invariant per relativity), but it also constrains how we specify distributions. **Stochastic Lorentz invariance essentially means we impose that our prior distribution for events does not single out a preferred frame** beyond what deterministic invariance allowed. For instance, if we assume the uncertainty in time and space of events is isotropic (no preferred direction), then the family of distributions should be such that any transformed version (boost or rotation) still falls in the same family.

Another way to view it: our probabilistic laws (like "events occur with certain mean rate and jitter") should hold the same form in any inertial frame. If in one frame event times are Gaussian with mean μ and variance σ^2, in a moving frame they might not remain exactly Gaussian in form (because a linear combination of t and x is involved), but the underlying physical randomness doesn't create a new

effect. If one accounts for the mixing of time and space, the statistical description is preserved in the sense that all physical predictions (like probabilities of causality, correlations between events) are the same.

Distributional Invariants: We might identify new invariants such as the distribution of proper time along a chain of events (the "proper time" being the time measured in a frame where events are back-to-back in space). In a stochastic scenario, each realization has a proper time, and that distribution might be invariant. For example, if an event e_1 causes e_2, one can define the *proper delay* τ such that $c^2\tau^2 = c^2\Delta t^2 - \Delta x^2$ (if $\Delta s^2>0$). This τ is like the time measured in the rest frame moving with velocity that made these two events coincident in space. If events have some randomness, τ will vary, but its distribution might be simpler or more fundamental than distributions of Δt or Δx separately. Requiring that distribution of τ is invariant (which it should be, as τ is a physical observable akin to proper time between events) provides a check on any models we design for the randomness. In analogy, in relativity any physical process duration (like particle decay time) should yield the same distribution of proper times regardless of the frame in which you measure the decay events.

In summary, **stochastic Lorentz invariance** means that our probabilistic event-time geometry respects relativity's symmetry at the level of distributions. Transformations preserve the *laws* (the form of distributional relationships) even if individual outcomes vary. Practically, this guides us to design models where noise or uncertainty is added in a covariant way. For example, if we add a random time offset to events to simulate jitter, that offset should be drawn from a distribution that does not depend on an observer's motion (no adding more jitter in one frame than another). It's a consistency principle ensuring the extended geometry remains physically meaningful and doesn't introduce frame-dependent randomness that would violate relativity's core tenets.

Modeling and Simulation Strategies for Probabilistic Event-Time Geometry

Having established the theoretical framework, we consider how to **model, simulate, and compute** within probabilistic event-time geometry. Implementing these ideas requires practical strategies to represent uncertain events and to calculate probabilistic intervals and causal relationships. Key approaches include representing events as ensembles of points (clouds), using Monte Carlo simulations or analytical methods to propagate uncertainties, and defining metrics of causal confidence in event networks.

Representing Probabilistic Events as Clouds of Points

One straightforward representation of an uncertain event is as a **cloud of sample points** in space-time. Each sample (l, t) in the cloud represents one possible realization of the event's coordinates. This could be obtained by random draws from the distribution of (L_E, T_E). For example, if an event's time is $T_E \sim N(\mu_t,\sigma_t^2)$, we sample a set of times from that normal distribution; if location is certain, all samples have the same l, otherwise we sample from a spatial distribution as well. The set of sample points ${(l_i, t_i)}_{i=1}^N$ visualizes the **probability cloud** of the event. The density of points in a region reflects how likely the event is to occur there. With enough samples, any needed probability (like $P(e_1 \to e_2)$ or distribution of Δs^2) can be approximated by counting occurrences in the samples.

Advantages of point clouds:

- They are simple and simulation-friendly: one can just propagate each sample through transformations or into calculations without requiring closed-form mathematics.
- They naturally capture any correlation between time and space uncertainties (each sample is a joint draw, preserving correlations).
- They can be visualized directly on a space-time plot, giving intuition about the "spread" of an event.

Alternate representations: Instead of raw samples, one might keep a **parametric form** for the distribution (like mean and covariance for a Gaussian). This is memory-efficient (just store a few parameters instead of many points) and sometimes allows analytical calculations of probabilities. For instance, if we assume normal distributions, $P(e_1 \to e_2)$ can often be expressed in terms of error functions or Q-functions. Another representation is an **interval or bounding region**: e.g., an event might be specified to occur in the time window $[t_{\min}, t_{\max}]$ with equal likelihood (uniform distribution) and at a certain location. This interval representation (used in systems like Google's TrueTime, which gives timestamps as an interval of confidence [23]) is coarser but provides guaranteed bounds on uncertainty. It's essentially treating the event as a "fuzzy interval" rather than a full distribution.

In practice, the representation chosen often depends on the domain: - In neuromorphic hardware, one might have statistical models of neuron firing jitter (perhaps Poisson processes or Gaussians around a mean inter-spike interval). - In distributed systems, clock offset uncertainty might be modeled with bounded intervals or Gaussian mixtures learned from synchronization data [15]. - In event-driven AI, one might use belief distributions for event times (like in a Bayesian network node).

The **event cloud model** is a unifying notion: whether via explicit samples, analytical distributions, or bounded regions, we treat events as "clouds" rather than points. This allows the geometry to be simulated or computed with by essentially doing "fuzzy geometry" – overlapping clouds, intersections (for coincidence of events with some probability), etc., analogous to how one would handle error ellipses in tracking problems.

Monte Carlo Simulation vs. Analytical Estimation

When dealing with probabilistic intervals and causal probabilities, two broad approaches arise:

- **Monte Carlo simulation:** Generate many random realizations of the events (using the clouds as described), and then compute deterministic relations for each realization. For example, to estimate $P(e_1 \to e_2)$, we can draw N samples of $(T_{e1}, L_{e1}, T_{e2}, L_{e2})$ from the joint uncertainty distribution. For each sample, check if $t_{2} \ge t_{1} + d(l_1,l_2)/c$. The fraction of samples that satisfy the inequality is an empirical estimate of $P(e_1 \to e_2)$. Similarly, by computing $(\Delta s)^2$ for each sample pair, we get a sample of the interval distribution from which we can plot a histogram or compute statistics (mean, variance, quantiles). Monte Carlo is easy to implement and accommodates any complexity (nonlinearities, arbitrary distributions). Its downside is that it can be computationally intensive for very low probability events (many samples needed to estimate tiny probabilities accurately) or high-dimensional uncertainties. However, with modern computing, Monte Carlo is a staple for uncertainty propagation.

- **Analytical estimation:** When possible, derive closed-form or semi-closed-form expressions for the probabilities or distributions of interest. For instance, if T_{e1} and T_{e2} are independent Gaussians (with known means and variances) and clocks are synchronized (so no correlation in time), then $T_{e2}-T_{e1}$ is also Gaussian with mean $\mu_{2}-\mu_{1}$ and variance $\sigma_1^2+\sigma_2^2$. Then $P(e_1 \to e_2) = P(T_{e2}-T_{e1} \ge d/c)$ can be computed by a simple Gaussian tail formula. If $X = d(L_{e1},L_{e2})$ is random (say L_{e1} has a distribution or d depends on random path delays), one might need to convolve distributions. There are known results for the distribution of a difference or sum of random variables, and for the difference minus a constant, etc. Analytical formulas can also leverage integration: for example, $P(e_1 \to e_2) = \iint I{t_2 \ge t_1 + d/c}\; p_{T1,T2}(t_1,t_2)\;p_{L1,L2}(l_1,l_2)\;dt_1\,dt_2\,dl_1\,dl_2$ (where $I{}$ is an indicator function). Solving this integral explicitly might be hard in general, but in some cases it factorizes or reduces with change of variables.

Often a hybrid approach is used: approximate an integral via numerical methods, or use importance sampling to Monte Carlo efficiently for rare events. Another analytical tool is to use **moment-based approximations**: e.g., approximate the distribution of $(\Delta s)^2$ by a normal using its mean and variance (via central limit theorem if sums are involved). One must be cautious, as probabilities near boundaries (like $P((\Delta S)^2>0)$) can be sensitive to distribution tails.

Example: Suppose two events have times $T_{e1} = t$ (exact) and $T_{e2} = t + \delta$ where δ is an uncertain delay due to processing, say $\delta \sim \text{Exponential}(\lambda)$ (a positive exponential distribution). Let their locations be distance d apart, fixed. Then $P(e_1 \to e_2) = P(\delta \ge d/c) = \exp(-\lambda\,d/c)$ (one minus the CDF of exponential). This is an analytic formula giving a clear relationship: the longer the distance d, the smaller the probability (exponentially decaying) that the second event falls after the first given the random delay. A Monte Carlo simulation would approximate this by sampling many δ values and counting, which should converge to the same result.

In summary, **Monte Carlo vs analytical** is a trade-off between generality and closed-form insight. Monte Carlo is like performing experiments on a simulated multitude of event-timelines to empirically measure geometric quantities, whereas analytical methods aim to solve the probability geometry with mathematics. In practice, one may use Monte Carlo to validate analytic results or to handle cases where analysis is intractable. Analytical results, when obtainable, are valuable for providing formulas that can guide understanding (for example, seeing that a certain probability decays with distance squared vs exponentially might inform design decisions in a system). For real-time systems analysis, fast computation of these probabilities is important, so finding simplifications (like assuming certain distributions) can reduce the need for brute force simulation.

Causal Confidence Metrics for Event Graphs

In complex systems, we often deal with many events forming a network or graph (e.g. nodes as events, directed edges as “could have caused” relationships). With uncertainty, a deterministic event graph (like a Lamport causal graph) turns into a **probabilistic event graph**. Each potential edge $e_i \to e_j$ can be assigned a weight or confidence value reflecting how likely e_i precedes and influences e_j. We need metrics to summarize and work with these confidence values.

Edge-wise confidence: The most direct metric is the probability of a causal relation, as defined earlier, $P(e_i \to e_j)$. We might call this the **causal link confidence**. It can serve as a weight on the directed edge

from e_i to e_j. A value of 1 means a guaranteed causal ordering (within model limits), and 0 means definitely not causally related (likely e_j happened before e_i or too soon after). Values in between indicate uncertainty. For example, if $P(e_i \to e_j) = 0.8$, we have 80% confidence i happened before and could affect j. This could be visualized by color intensity or line thickness in a graph diagram of events.

Path confidence and transitivity: In a graph, if event A likely influences B, and B likely influences C, what is the confidence that A influences C (directly or indirectly)? One might be tempted to multiply probabilities (0.9 * 0.8 = 0.72) as an estimate, but because of intransitivity issues, it's not always straightforward [16]. If uncertainties are independent and small, multiplication might approximate the confidence of a causal chain. However, there could be correlations: uncertainty that makes $A \to B$ false might also make $B \to C$ false, etc. One could instead compute $P(A$ precedes C) explicitly from their time distributions (noting that B is an intermediate that might not perfectly synchronize the uncertainties). Some research suggests constructing a *Bayesian network* or a *graphical model* for event ordering [24] [25]. Each event's timestamp is a random variable, and causal links impose inequality constraints. We can then infer probabilities of various partial orders. In general, inferring the probability of a complex partial order (like an entire ordering of many events) is combinatorially hard, but edge-wise and small subgraph confidences can be computed.

Metrics for entire graphs: Beyond individual edges, we might want a single measure of how "causally consistent" an event graph is under uncertainty. For instance, if we have a set of events that should be totally ordered in time, the confidence that no causality violations occur in the set could be a metric. Another could be the *expected number of causality violations* (events that end up happening out of order). In distributed systems, one could define a **fairness metric** or **causal fidelity** metric: e.g., "the fraction of event pairs that are ordered correctly (cause before effect) in the observed execution." If clocks are very uncertain, this fraction could be low; if they are precise, it will be high. This relates to the concept of *consistency* in causal delivery protocols – sometimes protocols allow a bounded probability of delivering messages out-of-order, and one can measure that probability.

To compute such metrics, Monte Carlo can again be used: simulate many possible realizations of the entire event set (drawing random clock skews, delays, etc.), then count how often the event graph respects all causal relations or how many edges are inverted. This gives an empirical distribution of graph consistency.

Confidence-based event ordering: One interesting application of causal confidence is *probabilistic ordering algorithms*. If two events are concurrent in Lamport sense, but one has an 70% chance of being earlier than the other, some systems might choose to order them accordingly to break ties (this is suggested in the "likely-happened-before" approach [15]). That introduces a small chance of error (30% the order might be wrong), but could be worthwhile for fairness or performance. The design of such systems needs metrics for how often they get the order wrong (which is exactly the probability of the opposite order). By assigning confidences and perhaps requiring a threshold (e.g. only enforce an order if $P(A \text{ before } B) > 0.99$), one can balance correctness and determinism.

In summary, **causal confidence metrics** provide a quantitative handle on event graphs under uncertainty. They extend the binary logic of causality to a graded spectrum, enabling nuanced decisions and analyses. Such metrics are crucial in systems like distributed databases (to decide if one transaction happened before another under uncertainty), in causal inference on noisy data (to gauge confidence in cause-effect hypotheses), and even in debugging/log analysis (to highlight event relations that are ambiguous vs those that are reliable). They connect naturally to **probabilistic graphical models**, as we discuss next, where

events and their causal links can be nodes and edges in a Bayesian network or factor graph representing temporal uncertainty.

Connections to Broader Mathematical Frameworks

The probabilistic event-time geometry we've outlined doesn't exist in isolation – it intersects with several established mathematical and computational frameworks. By recognizing these connections, we can leverage existing theories and tools to deepen understanding and solve problems in this domain.

Stochastic Differential Geometry Perspective

If we view event-time space as a manifold (akin to space-time), introducing probability suggests that events might follow stochastic processes on this manifold. **Stochastic differential geometry** is the field that merges differential geometry (the mathematics of smooth spaces and curves) with stochastic analysis (random processes, diffusion, etc.) [26]. One way this connection appears is in considering how an event's coordinates might *evolve* with time or under some random influences. For instance, imagine a sequence of events (perhaps representing a moving sensor's readings over time). In deterministic geometry, we could have a world-line (trajectory) for the sensor. In probabilistic terms, the sensor's reported events could jitter around an underlying smooth trajectory. We might model this as a **stochastic curve** in event-time space, satisfying a stochastic differential equation (SDE).

- A simple example: $dT = ds$ (time moves forward at rate 1) and $dL = v\,ds + \sigma\,dW_s$, where dW_s is a Wiener process (Brownian motion) term. This would model a location that moves with average velocity v but has random Brownian perturbations – a **random walk in space-time**. The geometry of such a path can be analyzed with tools from stochastic calculus [27]. Questions like "what is the probability that this path stays inside a lightcone?" become accessible via SDE theory.

- More formally, one could consider a Riemannian or pseudo-Riemannian manifold equipped with a metric (like $ds^2 = c^2 dt^2 - dx^2$) and then study stochastic processes on it (diffusions that respect the metric). This ties into **stochastic calculus on manifolds** [26] – e.g., defining Brownian motion on a curved space. In our case, the manifold is flat Minkowski (if no gravity or curvature is assumed), but the presence of boundaries (lightcones) and domains (causal vs acausal regions) makes it interesting to consider reflecting or absorbing conditions ("particle cannot go faster than c on average" etc., perhaps enforced by drift/diffusion constraints).

- Stochastic differential geometry can also provide *local* descriptions of uncertainty. Instead of specifying full distributions, we might specify that at small scales, an event's coordinates have a certain diffusion behavior. For example, clock drift could be treated as a slow random drift in the time coordinate relative to a standard clock – this can be written as an SDE for the time coordinate of each node's clock: $dT_{\text{node}} = (1+\epsilon)\,dt$ with ϵ evolving randomly (perhaps as an Ornstein-Uhlenbeck process if drifts tend to revert) [28]. Then the difference between two clocks is a stochastic process whose distribution we can solve for (often it yields something like a Gaussian whose variance grows with time, if drift is unbiased).

In summary, thinking in terms of **stochastic differential geometry** allows us to describe *dynamic uncertainty* in event-time geometry – how uncertainty accumulates or propagates as events progress. It provides a bridge to physics (where stochastic processes in space-time are studied in statistical mechanics,

diffusion processes, etc.) and to engineering (where random vibrations or jitters might be modeled as differential equations with noise). This connection could be particularly useful for understanding systems where timing noise accumulates over long sequences of events (e.g., successive spikes in a neuron with random drift in inter-spike interval, or cascading delays in a network).

Information Geometry and Statistical Manifolds

We already touched on information geometry in the context of metrics for distributions. To expand, **information geometry** views the set of all probability distributions of interest as a manifold, where each point is a distribution and smooth paths correspond to changing distribution parameters [20]. In event-time geometry, one could consider the *entire configuration* of events as a point in a high-dimensional space, and then consider distributions over those configurations. For example, imagine we have n events in a system. Deterministically, specifying all their coordinates $(l_1,t_1), (l_2,t_2),\dots,(l_n,t_n)$ gives one configuration point in $(L\times T)^n$. Now if there is uncertainty, we have a distribution over $(L\times T)^n$. That distribution lives in a space of dimension related to n and the uncertainty per event. Information geometry can be used to analyze these distributions.

Application of information geometry:

- **Fisher metric and Cramér–Rao:** The Fisher information metric can be used to understand the *estimation of event parameters*. If we treat the event times/locations as parameters to be estimated from noisy observations (or from each other), the Fisher information sets bounds on how precisely we can do so. For example, if we have a probabilistic causal relation, the Fisher information might tell us how many samples or how accurate clocks need to be to confidently distinguish the order of two events. A high Fisher information means small changes in timing lead to big changes in likelihood of outcomes, implying easier estimation.

- **Geodesics = simplest changes:** On the manifold of distributions, geodesics (shortest paths under the information metric) represent the most natural way to morph one event's distribution into another. This could theoretically model how one event's probabilistic description might evolve into another's. For instance, in a system that gradually increases synchronization, the distributions of relative clock offsets might move along a geodesic in distribution space from a broad distribution (unsynchronized) to a tight one (synchronized).

- **Divergence as energy:** Kullback–Leibler divergence in information geometry can act like a potential or energy. Minimizing KL between a predicted distribution and an observed one is akin to updating beliefs (Bayesian update). In event-time context, if new information arrives (say, we get a time-stamp correction), one could formalize the update to event time distributions as a projection in information space. This connects to filtering and probabilistic graphical model inference (e.g., updating the distribution of an event time given some evidence is a geometric operation).

Recent research even suggests that with certain assumptions, **event-universe distributions can be parametrized by a small number of dimensions** – for example, embedding all event relationships in a four-dimensional manifold using information-geometric reasoning [29] [30]. This resonates with the idea that perhaps the combination of space-time and probability still yields a manageable structure, not an explosion of complexity. Information geometry provides the mathematical underpinning to explore such

conjectures and to possibly unify deterministic and probabilistic invariants (e.g., a four-dimensional info-geometric space that encodes both the Minkowski structure and the uncertainty structure).

Probabilistic Graphical Models with Temporal Structure

Probabilistic graphical models (PGMs) are a cornerstone of AI for representing complex joint distributions using graphs. **Dynamic Bayesian networks**, **hidden Markov models**, and **temporal factor graphs** are all examples that incorporate time in some way. Our probabilistic event-time geometry can naturally be cast as a form of graphical model: each event is a random node (with variables for time and possibly location), and causal influences create directed edges between nodes.

Bayesian network interpretation: If $e_1 \to e_2$ causally with some delay distribution, we can treat T_{e2} as a random variable conditional on T_{e1}. For example, $T_{e2} = T_{e1} + \Delta + \text{(noise)}$ where $\Delta = d(L_{e1},L_{e2})/c$ may be a deterministic offset (if distance fixed) and "noise" accounts for extra delay. This is essentially a conditional probability distribution $P(T_{e2} | T_{e1})$. All such relations among events form a **graphical model of event times**. If the graph is a DAG (no causal loops, as is typically the case if we only allow forward-in-time causation), it constitutes a Bayesian network. We could then perform **inference** on this network: e.g., given some observed times (maybe some events were directly measured), infer the distributions of others, or query the probability that a certain ordering holds. The framework of PGMs gives algorithms like belief propagation or sampling methods to do these calculations systematically.

Temporal point processes and PGMs: In some cases, events occur in continuous time and one models them via point processes (Poisson processes, Hawkes processes for self-exciting events, etc.). These can be related to PGMs by discretizing time or by specialized continuous-time graphical models (where conditional intensity functions play the role of conditional probabilities). For example, a **Hawkes process** is one where each event increases the rate of future events (a kind of causal influence modelled in time). One can interpret a Hawkes process as a graphical model where each event has children events with some probability rate. If we add spatial structure (like location), we get spatio-temporal point process models, which are essentially probabilistic event geometries albeit often without an explicit metric like we defined. By incorporating the metric constraints (like the lightcone limit on influence), we can refine such models to respect causal cones (no influence outside the lightcone).

Constraint networks: Another angle is using factor graphs or constraint networks to enforce the causal inequalities. One could create a factor for each pair (e_i, e_j) that enforces $t_j \ge t_i + d(l_i,l_j)/c$ in a soft way (as a probability weight rather than a hard constraint). The resulting probabilistic model assigns higher probability to configurations of event times that respect causality, and lower (or zero) to those that grossly violate it. Then, sampling from this model effectively simulates random event scenarios consistent with our probabilistic geometry. This is reminiscent of **probabilistic graphical models with inequality constraints**, which can be handled with Gibbs sampling or specialized algorithms.

Learning and reasoning: If one collects data of many events (e.g., logs from distributed systems, spike trains from neural recordings), one could fit a probabilistic event-time model to this data. Graphical model structure learning could identify likely causal links (this connects to causality inference, where algorithms try to deduce cause-effect from correlations, but here respecting time ordering). The temporal structure means we might use algorithms for **dynamic PGMs** or **chronological Bayesian networks**. Also, frameworks like **Allen's interval algebra** (commonly used for temporal reasoning in AI) can be extended

with probabilities – e.g., each temporal relation (before, after, overlaps, etc.) gets a probability. The geometry perspective adds the metric notion, which is not in classical interval algebra (that algebra treats time qualitatively). By mixing metric (interval size distribution) with logic (temporal relations), we get a powerful hybrid model.

In conclusion, probabilistic event-time geometry can be viewed through the lens of **PGMs as a unifying language**: events are random variables, causal constraints are dependencies. This connection means we can apply a vast array of existing AI algorithms to do inference (answer queries like "what's the probability event A occurred before B given all data?") or learning (estimating the typical delay distributions from data). It also means that our approach can benefit from the rich theory of PGMs, such as understanding conditional independence: e.g., if $e1 \to e2 \to e3$ is a chain, then T_{e1} and T_{e3} might be independent given T_{e2}, etc. This could simplify computations by exploiting conditional independencies defined by the causal graph – very analogous to how relativistic causality implies certain factorizations in multi-event probabilities (no influence propagates outside the cone means some independence beyond a certain separation).

Applications to Real-World Systems

The formalism of probabilistic event-time geometry is not just abstract; it has direct implications for analyzing and designing real-world systems that deal with asynchronous events and uncertain timing. We highlight three domains where this enriched model can be applied to great benefit: neuromorphic hardware, distributed computing systems, and event-driven AI frameworks.

Neuromorphic Hardware and Spiking Neural Systems

Context: Neuromorphic systems like spiking neural networks (SNNs) operate via discrete events (spikes) in time, much like the brain. Neurons fire spikes when thresholds are reached, and these spikes travel to other neurons with some delay. In hardware or in vivo, there is inherent *timing jitter* – variability in when a neuron actually fires or when a spike arrives due to noisy channel transmission [31]. Even the threshold crossing can vary with slight noise in the membrane potential. Traditional analysis of SNNs often assumes precise spike times, but as hardware clocks or analog components introduce randomness, a probabilistic event-time geometry is needed.

Applying the model: Each spike can be treated as a probabilistic event. For example, if a neuron usually fires 10ms after its input, but sometimes it's 9ms or 12ms, we model the firing event's time as a distribution (perhaps Gaussian around 10ms). The location coordinate is the neuron ID which is typically fixed (no uncertainty in which neuron fired, unless noise causes a different neuron to erroneously fire – that could be considered as well, but that's more like a different event entirely). With these event clouds, one can analyze:

- **Causal inference in spike trains:** Determine the probability that a spike from neuron A caused a subsequent spike in neuron B. Because of jitter, some spikes of B might appear to precede A's spike or come too early for A to be the cause. Our causal probability formula $P(A \to B)$ gives a quantitative handle. For instance, if neuron B fires 5ms after A on average but with ±2ms jitter, and axonal propagation takes 3ms, we can compute the probability B's spike was caused by A's spike vs it was spontaneous or caused by some other input arriving slightly later. This helps in **spike train analysis**, a common task in neuroscience to understand functional connectivity. It moves beyond

binary “did fire before” checks to a confidence measure of causal influence per spike [15] (though that reference was distributed systems, the idea carries to neural events).

- **Robustness to timing noise:** Designers of neuromorphic hardware want to ensure the computation (which is often encoded in spike patterns and their timings) is robust to jitter. Using probabilistic event-time geometry, one can formally analyze how much jitter can be tolerated before computation fails. For example, consider a sequence of neurons meant to fire in order (a polychronous chain). With each link having some probability of mis-timing, one can calculate the probability the whole chain fires in the correct order (that would be a path confidence in our event graph). If it drops below a threshold, the chain might not be reliable. This guides how precise the hardware timing needs to be, or it suggests adding error-correcting features (like majority voting neurons) to mitigate timing errors.

- **Temporal coding and decoding:** In neuromorphic sensors like Dynamic Vision Sensors (event cameras), events encode pixel changes with microsecond timestamps [32]. If we want downstream processing to interpret these events (say to compute motion or recognize patterns), it must handle uncertainty in those timestamps (due to sensor noise or clock drift between sensor and processor). By using event-time geometry with distributions, one can propagate the uncertainty from sensor input through to actuator output. For instance, an event-driven camera might inform a spike-based controller; using our framework, the controller can weigh recent spikes by their timing uncertainty to avoid reacting to what might be noise. Essentially, it could compute a **confidence of causation** for each sensor spike influencing a motor neuron spike, similar to how we computed $P(e_1 \to e_2)$, and use a threshold to filter out unlikely causal signals.

Real-world neuromorphic devices have shown that a small amount of jitter doesn’t wreck performance, but too much does [33]. Our probabilistic geometric viewpoint provides a language to quantify this transition. It also connects to *information geometry*: one could ask how much information (in the Fisher sense) a spike’s timing carries, or how distinguishable two spiking patterns are given jitter. This might help in designing codes for SNNs that are more robust – e.g., using relative timing of groups of spikes (which might have lower variance due to common noise cancellation) rather than absolute timing of single spikes.

Distributed Systems: Clock Drift, Delay Compensation, and Scheduling

Context: Distributed computing systems consist of multiple nodes (machines, processors) that communicate and coordinate by sending messages (events) to each other. These systems face the fundamental challenge of **clock synchronization** – no two nodes share exactly the same clock reading, and clocks drift over time due to hardware differences [34]. Network message delays are variable (jitter on the network). As a result, determining the order of events (especially from different nodes) is non-trivial. Classic solutions like Lamport timestamps provide a partial order, and more advanced protocols (Vector clocks, etc.) capture causality but not time *magnitudes*. Google’s Spanner TrueTime API uses bounded intervals for time uncertainty to ensure external consistency [35]. However, bounding worst-case can be conservative. A probabilistic approach allows more nuance: rather than assume the worst possible drift or delay, assume a distribution and work with probabilities of ordering.

Applying the model: Each node's event (like "transaction committed" or "sensor reading") can be seen as an event with uncertain time relative to a global reference. The uncertainty stems from unknown clock offset and network delay in logging the event. Probabilistic event-time geometry helps in several ways:

- **Clock drift modeling:** We can assign each node a clock offset θ as a random variable that drifts slowly. For a short interval, one might treat θ as roughly constant but unknown, drawn from some distribution (e.g., uniform in [-5,5]ms if that's the sync uncertainty after NTP sync). Over longer periods, θ might itself change – an SDE model as discussed: $d\theta = \alpha dt + \sigma dW$ could capture a combination of steady drift α and random wander [28]. Using these distributions, we can transform event timestamps into a global frame with uncertainty. An event at node A with local time t_A corresponds to a distribution for global time $t = t_A + \theta_A$. Node B similarly $t_B + \theta_B$. Now comparing times becomes comparing two distributions. The result is $P(t_A+\theta_A < t_B+\theta_B)$ which is exactly the $P(e_1 \to e_2)$ we've described. Recent work like *Tommy* system explicitly computes such probabilities by learning per-clock offset distributions [15]. By doing so, one can implement **probabilistic ordering**: e.g., schedule transactions in order of their real occurrence with some confidence, improving fairness [15].

- **Delay compensation:** In distributed protocols (e.g., consensus algorithms, or time-sensitive networking), one often needs to set timeouts or wait times that accommodate network delay variability. Instead of a single worst-case bound, using a distribution of network delay can optimize performance. For instance, if message delays are on average 50ms with a standard deviation of 10ms, waiting a fixed 100ms is safe but maybe too conservative for 99% of cases. Using probabilistic geometry, a coordinator can decide to proceed after 70ms with a small risk that a slow message arrives later. Quantitatively, $P(\text{no message arrives after } t)$ can be computed from delay distributions, and one can pick t to balance risk. Essentially this is using the tail distribution of delays – which is akin to our $P(e_1 \not\to e_2)$ if we consider e_1 as send and e_2 as receive events.

- **Robust scheduling:** In real-time distributed scheduling, tasks are assigned considering communication latencies. Our geometry can be used to reason about **probabilistic schedulability**: the chance that all deadlines are met given the variability. Each event (start of a task, finish of a task) is uncertain; one can propagate these uncertainties through a task graph. For example, if a task must start after two prerequisite events from other nodes, we find the distribution of the start time (the max of the two finish times). With that distribution, we can compute the probability it starts by a certain time. System designers can then choose to add buffers or adjust priorities to ensure high probability of meeting requirements. This is a quantitative approach to what-if scenarios: "What is the probability our system misses a deadline if clock sync uncertainty increases by 10%?" can be answered by plugging into the model. It goes beyond yes/no schedulability to a risk assessment.

- **Monitoring and anomaly detection:** By expecting events to obey the probabilistic geometry, we can flag cases that deviate. For example, if two events are supposedly causally unrelated, the probability they violate causal order (one arriving too early) should be near zero. If in logs we find a timestamp that implies a violation far outside expected distribution (like a message appearing 1 second before it was sent, when drift was supposed to be <10ms), that's an anomaly indicating a clock failure or log error. So the model provides a baseline for normal operation.

In summary, distributed systems benefit from **uncertainty-aware reasoning** about time. By embracing probability (as advocated in some recent research [36]), one can achieve fairer ordering, better resource utilization, and more graceful handling of unpredictability, compared to strictly worst-case or purely logical time methods. Probabilistic event-time geometry gives the formal underpinning to design and analyze these improvements, ensuring that even under uncertainty the causal structure (safety properties like no causality violation) is maintained with quantifiable confidence.

Event-Driven AI Systems: Uncertainty-Aware Learning and Reasoning

Context: Many AI systems are event-driven, meaning they react to inputs or triggers that occur asynchronously. Examples include complex event processing systems (e.g., for stock trading or sensor networks), autonomous robots responding to sensor events, or user-interaction driven systems. These systems often use rules or learning models that assume certain event orderings or timing. Incorporating uncertainty is crucial when events can be missing, delayed, or spurious.

Knowledge representation with uncertainty: In AI, *knowledge graphs* or *rule systems* might include temporal relations ("Event A happens before B", "if X occurs within 5 minutes after Y, then do Z"). A probabilistic event-time model allows these rules to be evaluated in a flexible way. For instance, instead of a rule triggering only if X occurs within exactly 5 minutes of Y, we could have a confidence that X and Y are related given a distribution of delays. There's research on reasoning with uncertain temporal rules [37] [38] – essentially adding probability to temporal logic. Our framework provides a semantics: the probability of a temporal pattern can be derived from integrals over the joint distributions of events.

Machine learning on event streams: Many learning algorithms for event streams (like predicting future events or classifying patterns) can benefit from probabilistic timing features. For example, in predictive maintenance, an AI might watch events (sensor readings) and learn that usually Event A is followed by Event B after about 10 seconds. With a probabilistic geometry view, the AI can represent that relation with a distribution (10s ± 2s). If suddenly B comes 20s after A, it might flag it as abnormal. Traditional sequence learning might implicitly capture this as well, but having an explicit probabilistic time model can improve interpretability and allow combination with rule-based systems. Models like **Temporal Bayesian Networks** or **Time-sensitive neural networks** can incorporate these interval distributions.

Probabilistic graphical models for events: As previously noted, using PGMs, one can create AI systems that reason about uncertain events. For instance, a robotic planning system might have nodes representing "obstacle detected" event and "braking action" event with uncertain times. It can infer "did I brake in time?" as a probability. If the confidence is low, it might take a safer action (like stop entirely). This is essentially reasoning under uncertainty in the temporal domain.

Decision making with temporal uncertainty: Consider an event-driven AI that has to decide on an action based on events (e.g., a security system deciding if an alarm is real based on sensor events). If sensor events have time jitter, the sequence might appear out of order sometimes. A deterministic system might misinterpret that. A probabilistic system would maintain beliefs: e.g., "Sensor A triggered before Sensor B with 80% probability; if A before B, likely intrusion from front door, else from window." Then it can weigh actions (check front door vs window) by these probabilities. This resembles a POMDP (partially observable Markov decision process) but with continuous time uncertainty as part of the state.

Learning from data: If we have logs of an AI system's events with time, we can learn the distributions of delays and use those to improve the model. For example, an AI assistant might learn that user pauses between certain spoken commands average 1 second. If a pause is much longer, maybe the user is done, and it shouldn't wait indefinitely. The assistant's event model (the end-of-command event) can thus be probabilistic, with a threshold that adapts (like using the distribution to decide cutoff).

Bridging to human-time perception: Interestingly, humans also have uncertainty in judging time and causal relations (especially in asynchronous communication). An event-driven AI interacting with humans might benefit from aligning with the human's probabilistic model of events. For instance, people often reason "if I don't get a reply in 5 minutes, they likely didn't see my message." That's a probabilistic inference on an event (reply) given no event has happened yet. An AI could similarly use a learned distribution of reply times to decide when to notify the sender of a delay. This is essentially using the absence of an event as information in a probabilistic way.

In summary, event-driven AI systems become more robust and intelligent when they account for timing uncertainty. By explicitly modeling the distribution of event times and delays, they can make smarter decisions, avoid brittle assumptions of exact timing, and communicate probabilities to users or other systems. The probabilistic event-time geometry gives a coherent way to fuse temporal logic (events and causality) with probabilistic reasoning. Frameworks from probabilistic programming could even be employed: one could write a probabilistic program that defines event generation and use inference to answer queries like "what's the probability that scenario X is happening given the event timestamps seen so far?". This elevates event-driven AI to be **uncertainty-aware by design**, reducing errors in complex, real-world environments where timing is never perfectly reliable.

Conclusion

We have expanded the deterministic event-time geometry into a **probabilistic event-time geometry**, redefining events as distributions ("clouds") in space-time and recasting distances, intervals, and causal relations in stochastic terms. This formalization introduced fuzzy causal cones that reflect uncertainty, metrics on event distributions that allow measuring differences and information content, and a notion of stochastic invariance ensuring that physical laws hold in distributional form across reference frames. We also outlined how to model and simulate such geometries, using Monte Carlo sampling or analytical methods to estimate probabilistic intervals and causal confidence, and how to construct metrics that quantify the reliability of causal links in complex event graphs [15].

Furthermore, we connected this framework to **stochastic differential geometry** (modeling random motion and drift in event coordinates), **information geometry** (viewing event distributions through the lens of statistical manifolds and metrics), and **probabilistic graphical models** (which naturally represent uncertain causal structures). These connections provide powerful mathematical tools and computational algorithms for working with probabilistic event-time geometry.

Finally, we looked at a range of real-world applications: - In **neuromorphic systems**, where spike timing is key, our probabilistic geometry helps in analyzing and ensuring reliable computation in the presence of spike-time jitter and variability. - In **distributed computing**, it offers a principled way to handle clock drift and network latency by quantifying the likelihood of ordering and causal relations [15], leading to fairer and more robust protocols. - In **event-driven AI**, it enhances temporal reasoning by treating time and causality

in a probabilistic manner, thereby improving decision-making under uncertainty and enabling learning algorithms to capture temporal patterns more flexibly.

By unifying these ideas, probabilistic event-time geometry forms a comprehensive framework for reasoning about events in time and space when determinism gives way to uncertainty. It extends the elegant structure of relativistic space-time to the domain of stochastic systems, ensuring that even as we embrace randomness, we retain a consistent geometry of cause and effect – now equipped with probabilities and confidence levels. This enriched model stands to improve the design and analysis of any system where timing matters but isn't perfectly predictable, grounding such improvements in a firm theoretical foundation.

Sources:

- D. Eventine et al., *"Event-Time Geometry: Motivation and Core Concepts,"* (2025) – defines deterministic event-time coordinates, intervals, and invariance [6] [8] [9].
- M. H. J. Geng et al., *"Beyond Lamport, Towards Probabilistic Fair Ordering,"* arXiv:2510.13664 (2025) – introduces probabilistic "likely-happened-before" relations for distributed events [15].
- L. Babai, *"Clock Synchronization in Distributed Systems,"* ETH Zürich Lecture Notes (2015) – discusses clock drift as random deviation (e.g. 30–50 ppm) in networked nodes [34].
- R. T. Smith, *"Robustness of STDP to Spike Timing Jitter," Sci. Reports* **8**, 12820 (2018) – models neural spike time jitter as random variables added to mean timing [14].
- O. Shor et al., *"Rao-Fisher information geometry and dynamics of the event-universe," Heliyon* **9**(9): e19863 (2023) – explores representing events as distributions and embedding event-universe in a low-dimensional manifold via information geometry [21] [22].

---

1 2 3 4 5 6 7 8 9 10 11 12 13 32 Event-Time Geometry_ Motivation.pdf
file://file_000000002ed061f7ae10eb0f3790584d

14 Robustness of STDP to spike timing jitter | Scientific Reports
https://www.nature.com/articles/s41598-018-26436-y?error=cookies_not_supported&code=ff0b2273-539a-4029-81a2-cee38093679d

15 16 17 23 35 36 Beyond Lamport, Towards Probabilistic Fair Ordering
https://arxiv.org/html/2510.13664v1

18 19 arxiv.org
https://arxiv.org/pdf/1012.2393

20 Information geometry (part 1/3) - Ro's blog
https://rojefferson.blog/2018/08/12/information-geometry-part-1-2/

21 22 29 30 Rao-Fisher information geometry and dynamics of the event-universe views distributions - PubMed
https://pubmed.ncbi.nlm.nih.gov/37809879/

24 25 37 38 [1207.1427] A Model for Reasoning with Uncertain Rules in Event Composition Systems
https://arxiv.org/abs/1207.1427

26 PhD Seminar An Introduction to Stochastic Differential Geometry ...
https://math.uni.lu/docsem/talks/2019/03/01/baumgarth.html

27 Stochastic Differential Geometry: An Introduction
https://link.springer.com/content/pdf/10.1007/978-94-009-3921-9_3.pdf

28 34 [PDF] Clock Synchronization - DISCO
https://disco.ethz.ch/courses/hs15/distsys/lecture/chapter9.pdf

31 Quantitative analyses of how optimally heterogeneous neural codes ...
https://www.nature.com/articles/s41598-024-81029-2

33 On the relationship between synaptic input and spike output jitter in ...
https://pmc.ncbi.nlm.nih.gov/articles/PMC19583/