Predictive Algorithmic Services

Problem

Even with the best data on what is happening and a deep understanding of how the environment works with controls in place, it is still a reactive model. How can we plan or forecast accurately when everything is constantly changing and we continue to pivot?

Solution

Take the Digital Platform analytic capabilities, built from the steps so far, and now introduce machine learning to apply operational intelligence to our operating models (supported by complex event patterns). Before looking deeply into the data science aspects, the solution here is to broadly apply inductive learning to the analytic models the platform already had. The outcomes will be a mix of predictions around volumes and capacity with relationship discovery (upstream and downstream dependencies). Other predictions are related to business behaviors and the likelihood of outcomes (used today to improve targeting). When put together, you can start modelling what you should do next with actions and implications already determined. Over time, experience builds into the models, which in turn improves the predictions. In short, it's strategic planning at scale and speed.

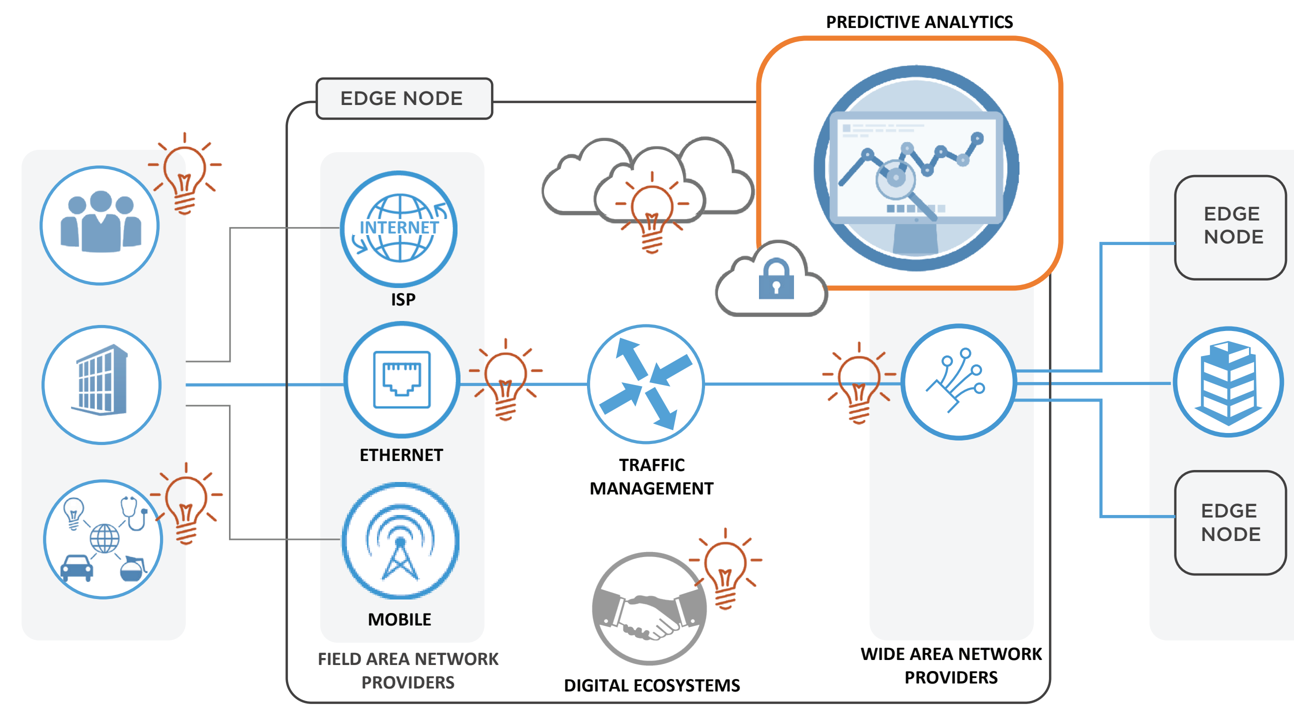
Constraints

1. Most services are myopic in nature, (e.g, watching utilization to flag for increasing capacity). The upstream change driving volatility in the service is unknown.
2. Likewise, downstream dependencies are notified when the service disrupts them. They in turn ask to be forewarned, but that information is still unknown.
3. How all these discrete services and upstream and downstream dependencies work is not effectively captured anywhere. Documenting them is even less practical when they are rapidly changing.
4. When constraints are hit, it requires some diligence to discover if the reasons are worth the investment to address. However, if you are using cloud services, you can automate capacity increases. Does anyone know the business reasons why that was done? Was it recorded anywhere?

Steps

1. Implement machine learning services or connect selected SaaS provider(s) respective services (defined here as an "engine").
2. Connect the engine to analytical models and gathered historical data.
3. Begin training the engine and modelling known good and bad conditions, improving accuracy using known historical information.
4. Compare the engine's model to the current model and dashboard and start testing predictions. If the predictions are wrong, there is more learning (or potentially other data) needed.
5. Move from binary decisions to detecting complex behaviors, to regression predictions about optimal pricing, or likely costs, etc.
6. Start applying this learning (predictive algorithms) more broadly to meaningful situations and strategic planning.

Reference View



Forces

- As business processes the shift to digital and IT services are virtualized, friction is being removed. At the time when it took too long to procure more capacity, friction was a bad thing. Today, and in the very near future, that may not be the case.
- One lesson distributed technology has taught, is that technology breeds more technology. More of this means more of that, which in turn requires more of something else (which uses the first thing) and the cycle continues. A self-fulfilling prophecy.
- Add these two concepts together and take into account that business and technology are combining.

Results

- Models can be continually refreshed with updated dependencies improving current situational awareness and understanding.
- Viewpoints on business activity, outages (failure predictions), utilization (waste), behaviors (fraud, etc.) and risk (operations heat map) can be built and continually refreshed.
- A digital dashboard and trends analysis to report on – which is mostly automated and less error-prone.
- Platform to evolve and incorporate external business events and activity with an eye toward digital business models and direction.
- Direction is really only limited by data.
- All aspects of the business have become digital, completing the digital platform.