IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

TRAFFIC PREDICTION AND FAST UPLINK FOR HIDDEN MARKOV IOT MODELS

Charly S, Sathyabalaji N
Department of Computer Science and Engineering, JKK Munirajah College of Technology, Erode, Tamilnadu - 638 506, India

Abstract: In this work, I present a novel traffic predictionand fast uplink (FU) framework for IoT networks controlledby binary Markovian events. First, I apply the forward algorithm with hidden Markov models (HMMs) in order to schedulethe available resources to the devices with maximum likelihood activation probabilities via the FU grant. In addition, I evaluate the regret metric as the number of wasted transmission slots to evaluate the performance of the prediction. Next,we formulate a fairness optimization problem to minimize the Age of Information (AoI) while keeping the regret as minimumas possible. Finally, I propose an iterative algorithm to estimate the model hyperparameters (activation probabilities) inareal-time application and apply an online-learning version of the proposed traffic prediction scheme. Simulation results show that the proposed algorithms out perform baseline models, suchas time-division multiple access (TDMA) and grant-free (GF) random-access in terms of regret, the efficiency of system usage, and AoI.

Keywords: Hidden Markov models, Active learning, wearable computing, machine learning, activity recognition, memory retention, cognitive factors, server monitoring. Time-division multiple access, Random-access, fast uplink, Age of Information

1. INTRODUCTION

RECENT advances in the Internet of Things (IoT) has led to the deployment of a large number of machinetype communication (MTC) devices to collect real-time information. The number of such IoT-MTC devices is rapidly growing to realize different use cases, such as environment monitoring, remote surgery, and autonomous vehicles [1]. In 5G, MTC service modes are massive MTC (mMTC) and ultrareliable low-latency communication (URLLC) [2]. The Quality-of-Service (QoS) demands vary among the service modes. In addition, many use cases have recently had more strict demands, which need extremely low end-to-end latency in a massive deployment of IoT devices to collect real-time information [2]. The behavior of the traffic of MTC devices (MTDs) differs from that of the traditional human-type communication devices (HTDs) [3]. The HTDs traffic tends

to be heterogeneous, whereas the traffic of MTDs is homogeneous and highly correlated. To elucidate traffic correlation in MTC, we consider the following road safety example as in Fig. 1: let event 1 and event 2 correspond to a vehicle moving down the street at normal speed, and a vehicle breaking the speed limit, respectively. Meanwhile, sensor 1 and sensor 2 are motion detectors, necessary to control the traffic lights, and speed limit alarm, respectively. In this scenario, event 1 will be detected by sensor 1 only. However, both sensors may likely detect event 2. Hence, we infer that sensor 2 will not likely be active except if sensor 1 is active. Moreover, if sensor 2 is active, sensor 1 will most probably be active but not vice versa. In such a scenario, it is essential to estimate the possible sensor activation pattern and allocate resources at low latency. If a human is crossing the street, a human detector or a road safety alarm could then transmit a signal to the base station (BS). The BS in turn sends a compulsory brake signal to a high-speed vehicle to enforce it to slow down the speed. This all should occur within a window of a few milliseconds to avoid an accident.

2. LITERATURE SURVEY

Six key features of machine type communication in 6G

While 5G is being rolled out in different parts of the globe, several research groups around the world have already started posing the question: What will the sixth generation (6G) be? The 6G vision is a data-driven society, enabled by near instant unlimited wireless connectivity. Driven by the impetus to provide vertical-specific wireless network solutions, machine type communication encompassing both its mission critical and massive connectivity aspects is foreseen to be an important cornerstone of 6G development. An over- arching vision for machine type communication in 6G networks is presented in paper. In this regard, some relevant performance indicators are first discussed, followed by a presentation of key enablers.

Traffic models for machine type communications

Machine-to-machine (M2M) or Machine-type Communication (MTC) is expected to significantly increase in future wireless networks. It exhibits considerably different traffic patterns than human-type communication, thus, claims for new traffic models and simulation scenarios. The challenge in designing such models is not only to accurately capture the behavior of single MTC devices but also to handle their enormous amount (e.g., up to 30 000 devices per cell) and their coordinated behavior. Source traffic models (i.e., each device is modeled as autonomous entity) are generally desirable for their precision and flexibility. However, their complexity is in general growing quadratically with the number of devices. Aggregated traffic models (i.e., all device are summarized to one stream) are far less precise but their complexity is mainly independent of the number of devices. In this work we propose an approach which is combining the advantages of both modeling paradigms, namely, the Coupled Markov Modulated Poisson Processes (CMMPP) framework. It demonstrates the feasibility of source traffic modeling for MTC, being enabled by only linearly growing complexity.

Compared to aggregated MTC traffic models, such as proposed by 3GPP TR37.868, CMMPP allows for enhanced accuracy and flexibility at the cost of moderate computational complexity.

Minimizing age of information in vehicular networks

Emerging applications rely on wireless broadcast to disseminate time-critical information. For example, vehicular networks may exchange vehicle position and velocity information to enable safety applications.

The number of nodes in one-hop communication range in such networks can be very large, leading to congestion and undesirable levels of packet collisions. Earlier work has examined such broadcasting protocols primarily from a MAC perspective and focused on selective aspects such as packet error rate. In this work, I propose a more comprehensive metric, the average system information age, which captures the requirement of such applications to maintain current state information from all other nearby nodes. I show that information age is minimized at an optimal operating point that lies between the extremes of maximum throughput and minimum delay. Further, while age can be minimized by saturating the MAC and setting the CW size to its throughput-optimal value, the same cannot be achieved without changes in existing hardware. Also, via simulations we show that simple contention window size adaptations like increasing or decreasing the window size are unsuitable for reducing age. This motivates my design of an application-layer broadcast rate adaptation algorithm. It uses local decisions at nodes in the network to adapt their messaging rate to keep the system age to a minimum. Our simulations and experiments with 300 ORBIT nodes show that the algorithm effectively adapts the messaging rates and minimizes the system age.

Minimizing the age of information through queues

In this poject, i investigate scheduling policies that minimize the age of information in single-hop queuing systems. i propose a Last-Generated, First-Serve (LGFS) scheduling policy, in which the packet with the earliest generation time is processed with the highest priority. If the service times are i.i.d. exponentially distributed, the preemptive LGFS policy is proven to be age-optimal in a stochastic ordering sense.4 If the service times are i.d. and satisfy a New-Better-than-Used (NBU) distributional property, the non-preemptive LGFS policy is shown to be within a constant gap from the optimum age performance.These age-optimal results are quite general: (i) they hold for arbitrary packet generation times and arrival times (including out-of-order packet arrivals); (ii) they hold for multi-server packet scheduling with the possibility of replicating a packet over multiple servers; (iii) and they hold for minimizing not only the time-average age and mean peak age, but also for minimizing the age stochastic process and any non-decreasing functional of the age stochastic process. If the packet generation time is equal to the packet arrival time, the LGFS policies reduce to the Last-Come, First-Serve (LCFS) policies. Hence, the age optimal results of
LCFS-type policies are also established.

3. PROPOSED SYSTEM

The contributions of this work are summarized as follows.

1) I formulate the device activation probabilities for the described HMM system model.

2) I apply the forward algorithm to predict the active devices and perform preemptive FU grant with low complexity.

3) I optimize an age parameter to compensate the AoI of the devices that have experienced high AoI while preserving the accuracy of the efficient forward algorithm.

4) For the case of unknown hyperparameters of the model, we apply an expectation–maximization algorithm to estimate the event

transition probabilities and the device activation probabilities-based only on the observations. Then, we apply the estimation procedure to present an offline-learning version of the FU algorithm.

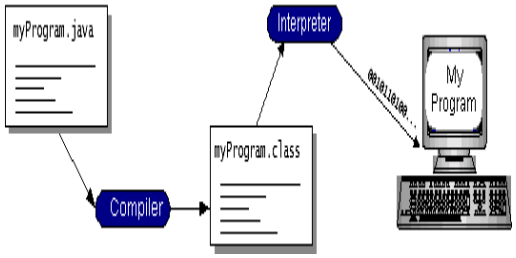
5) Finally, I rely on both the AoI compensation and the learned parameters to formulate an online-learning scheme that allows the BS to perform the FU algorithm in real-time applications, without the prior availability of large activation data sets.

6) The proposed online and offline schemes clearly outperform conventional GF and TDMA in terms of resource allocation efficiency while guaranteeing a favorable amount of fairness via age compensation.

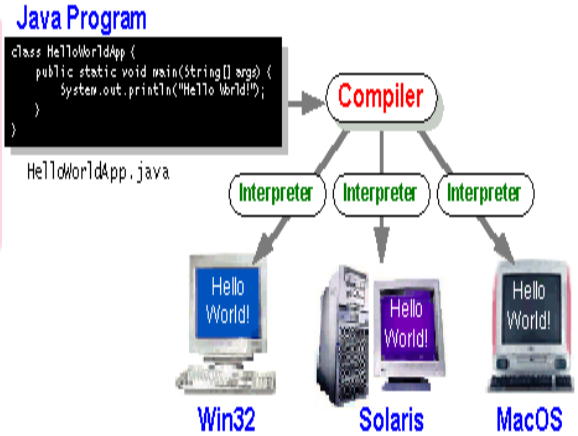
4. SYSTEM DESIGN

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



The Java Platform

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular

platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

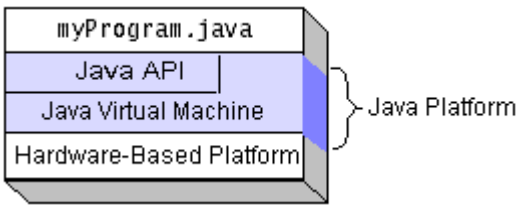
The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers,well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

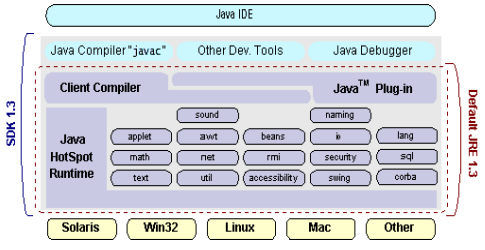
However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans™, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
- **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java**: You can keep

your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java™ Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.

- **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user
will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

5. RESULT AND DISCUSSION

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to
discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is
the process of exercising software with the intent of ensuring that the
Software system meets its requirements and user expectations and does not fail in an
unacceptable manner. There are various types of test. Each test type addresses a specific
testing requirement.

TYPES OF TESTING

Unit testing

Unit testing involves the design of test cases that validate that the internal program
logic is functioning properly, and that program inputs produce valid outputs. All decision
branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business
process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications

and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures: interfacing systems or procedures must be invoked.
Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing per-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

SYSTEM IMPLEMETATION

System implementation is the final stage of the project where the theoretical design is turn into working system. If the implementation system stage is not carefully

controlled and planned, it can cause confusion Thus, it can consider to be the mostcritical stage in achieving a successful of new system and in giving the user a confidence that the system will work and be effective. The implementation stage in the project involves, careful planning, investigationof the current system, checking constrains and the implementation in the software every menu is designed according to the user selection which has been implemented in user interface screen. It is done in the GUI development of user interface screen by using programming code. The application validations are made, taken into account of the entry levels available in various modules. Thus, all the aspects is charted out and complete project study is practically implemented successfully for the end user.

5. FUTURE WORK

In this paper, we focused on pool-based active learning. My ongoing work involves developing mindful active learning strategies that make query decisions on-the-fly as server monitoring data become available in real-time.

In my experiments i assumed that the time delay is equal to the difference between query time and sampling time. However, it is possible that the system may not respond to the query immediately. If the user prefers to proactively initiate the labeling process, the active learning process needs to recompute the queried observations by adding the time difference between query time and annotation time to reflect the time delay in our formulation. Furthermore , if there are multiple sessions in a day that the user intend to annotate the sensor data, this model can be used for each session separately. I also plan to investigate active learning solution s that take into account the possibility of delayed responses through context-sensitive active learning.

In this work, i simulated the memory strength of the end-user for validation purposes. My future work also focuses on conducting user studies that involve cognitive assessment of the user where i will assess the oracle's memory retention quantitatively.

6. CONCLUSION

This article considered the Markovian events which serve to model the activity of the massive deployment of IoT devices. I proposed an FU algorithm that efficiently predicts the activation pattern of the IoT devices based on the forward algorithm and grants the available resources to the devices with the highest likelihood of activation probabilities. I formulated an optimization problem that compromises a small value of the regret to minimize the AoI of the IoT devices and achieve a desirable degree of fairness. In addition, I formulated an expectation maximization algorithm based on the Baum–Welsh procedure to estimate the system hyper parameters.

Finally, i developed an online-learning version of the proposed scheme. Simulation results showed that the proposed algorithm outperforms the existing models, e.g., TDMA and GF, regarding regret, system usage efficiency, and AoI. The proposed algorithms were much simpler than machine learning-based predictors regarding the complexity of the computations. Therefore, the proposed algorithms could be used as traffic predictors in critical applications, e.g., predictive UAV positioning , road safety, and other applications with low-latency communication demands .

ACKNOWLEDGEMENT

The author would like to express their gratitude to everyone who helped them write this article and provided data for this research.

REFERENCE

[1] M. Latva-Aho and K. Leppanen, Key Drivers and Research Challenges for 6G

[2] Ubiquitous Wireless Intelligence, 6G Flagship, Univ. Oulu, Finland, Oulu, Finland, Sep. 2022. [2] N. H. Mahmood, H. Alves, O. A. López, M. Shehab, D. P. M. Osorio, and M. Latva-Aho, "Six key features of machine type communication in 6G," in Proc. 6G

[3] Summit, 2020, pp. 1–5. [3] M. Laner, P. Svoboda, N. Nikaein, and M. Rupp, "Traffic models for machine

[4] type communications," in Proc. 10th Int. Symp. Wireless Commun. Syst. (ISWCS), 2013, pp. 1–5. [4] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in

[5] vehicular networks," in Proc. 8th Annu. IEEE Commun. Soc. Conf. Sens. Mesh Ad

[6] Hoc Commun. Netw., 2011, pp. 350–358. [5] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Minimizing the age of information

[7] through queues," IEEE Trans. Inf. Theory, vol. 65, no. 8, pp. 5215–5232, Aug. 2019. [6] P. Popovski et al., "Wireless access in ultra-reliable low-latency communication

[8] (URLLC)," IEEE Trans. Commun., vol. 67, no. 8, pp. 5783–5801, Aug. 2019. [7] A. Laya, L. Alonso, and J. Alonso-Zarate, "Is the random access channel of LTE

[9] and LTE-A suitable for M2M communications? A survey of alternatives," IEEE

[10] Commun. Surveys Tuts., vol. 16, no. 1, pp. 4–16, 1st Quart., 2014. [8] N. H. Mahmood, R. Abreu, R. Böhnke, M. Schubert, G. Berardinelli, and T. H. Jacobsen, "Uplink grant-free access solutions for URLLC services in 5G new radio,"

[11] in Proc. 16th Int. Symp. Wireless Commun. Syst. (ISWCS), 2019, pp. 607–612. [9] "Study on RAN improvements for machine-type communications," 3rd Gener. Partnership Project (3GPP), Sophia Antipolis, France, Rep. (TR) 37.868, 2012. [10] R.-G. Cheng, J. Chen, D.-W. Chen, and C.-H. Wei, "Modeling and analysis of an

[12] extended access barring algorithm for machine-type communications in LTE-A

[13] networks," IEEE Trans. Wireless Commun., vol. 14, no. 6, pp. 2956–2968, Jun. 2015.Learn: A framework for sharing activity recognition models in wearable systems with context-varying sensors", ACM Trans. Des. Autom. Electron. Syst., vol. 24, 2019.