18-842: Distributed Systems **Spring 2015**
Lab 1: Clocks

Objective: Develop facilities to track events using Logical and Vector Clocks.

Specification: You will develop a "clock" infrastructure and add it to your communication infrastructure. You will then develop two demonstration applications, built on that infrastructure: a capability demo and a logging application.

First, let's discuss the infrastructure. ClockService should be an abstract type with Logical and Vector subclasses. You'll need to be able to do the following things:

- Only one type of ClockService (i.e. Logical or Vector) will be used system-wide at any time. Easily allow applications to specify which subclass you want to use. This suggests an "Object Factory" design pattern.

- The VectorClock size is based on the number of names in the configuration file.

- Messages need to include a timestamp. This suggests that a "TimeStampedMessage" subclass be developed, which includes a field of a "TimeStamp" class.

- MessagePasser should be able to automatically timestamp all message transmission and reception. This means you'll have to add facilities to MessagePasser so it can find the ClockService instantiation. The application simply uses the same MessagePasser.send and MessagePasser.receive methods, it doesn't do anything special to get timestamps — the MessagePasser takes care of that. Your Lab0 application should (theoretically) be able to use timestamped messages with fairly minimal changes *to the application code.*

- Events other than message transmission may need timestamps. Your application code should be able to call ClockService to get a timestamp for any arbitrary use.

- Timestamps will need to be able to be inspected and compared. Your application code needs to be able to determine from timestamps if an event "happened before" or is "concurrent with" some other event. Note carefully the limits on Logical clocks in this regard. Those limits are inherent in the use of Logical clocks — don't try to tack on any other facilities to overcome such limits. In this case, simply inform the application via an exception or similar means.

Demonstrate: To demonstrate the full capabilities of your ClockService, update your Lab0 application so that you can, at the TA's direction, do the following:

- Initialize your system to use Logical or Vector clocks,
- Display the timestamp of a message (either for sending or receiving)
- Have a timestamp issued (for arbitrary, non-message related reasons)

Then, to explore application development with distributed system infrastructure, create and demonstrate a centralized logging facility for a distributed system. This may or may

not be a subclass of “Logger.” The idea is that the Logger is an additional member of the distributed system (and thus needs to be initialized using the config file). All processes in the system can send messages to your Logger process. The actual content of the messages to be logged are application defined. **Don't log every event and message**, just the ones your application explicitly decides to log.

The Logger then will order all log messages in timestamp order, clearly identifying any messages that are concurrent. Obviously, all messages to the Logger need to be of type "TimeStampedMessage" and the comparisons done by the Logger should be via the comparison methods of the Timestamps. Demonstrate your system with at least 4 processes (one of which can be the Logger process) on at least 2 different nodes communicating and timestamping events.

You’ll note that I haven’t been explicit about what the output looks like. I want you to develop something that makes sense. The log output should clearly show as many messages in the proper order as is possible. Concurrent messages should be noted, as well as showing the limits of this concurrency (i.e. just exactly which messages it is concurrent with).

During your demonstration, be prepared to discuss the architecture and design decisions you made. Prepare a simple drawing (legible, not handwritten!) showing the interaction among the objects and classes in your system. Be able to show the interactions of the objects for startup as well as for send and receipt of a message.

Also discuss what happens in your system in the presence of errors. Show that you have explored the drop / delay / duplicate capability of MessagePasser. I’m not suggesting you need to implement any extra facilities to make the system reliable. What I am suggesting is that you explore the limitations of your system in an empirical fashion. You should be making very extensive use of delay to ensure your system runs properly.

Further Notes:

Logger has to work within the use of the clock service. Logger discovers stuff only via messages that also are sent to it via the MessagePasser.

Logger is a different application, using MessagePasser. Don't try to make all your nodes run the same application.

Logger’s output can be made at a single time in a non-dynamic fashion. The receipt of a message may change the ordering (i.e. it might get inserted in the sorted list at a far-past point. I don’t want you to have to figure out how to update a user interface dynamically). You can just wait until a user says “Let me know right now what the log file looks like” and the logger will dump it’s current knowledge.

Logger doesn’t have to work with both Logical and Vector clocks. Once you have developed both capacities, then you as the designer will make a choice about which to use.

Administrative details: The lab is due at 9:30 AM on 9 February. At that time, an archive of your source code needs to be deposited on ALE. Each team will give a quick demo, using the identical code, to a TA before 9:30 AM on 11 February.

18-842 Lab Teams

Teams 1-9 demo to Krish

Team 1: Jialiang Tan (jialiant) and Joel Krebs (jkrebs)
Team 2: Advaya Krishna (advayak) and Yvonne Yuan Yuan (yyuan1)
Team 3: Joyce Chung (joycechu) and Rachita Chandra (rchandra)
Team 4: Harshad Shirwadkar (hshirwad) and Jackie Jiang (xiaotiaj)
Team 5: Qing Zhou (qzhou) and Yang Wu (yangwu)
Team 6: Brandon Wolfe (blwolfe) and Suril Dhruv (sjdhruv)
Team 7: Ding Zhao (dingz) and Wallace Wang (xinyuwan)
Team 8: Jing Yu (jingyu) and Norman Wu (luow)
Team 9: Qinyu Tong (qtong) and Jian Wang (jianwan3)

Teams 10-18 demo to Utsav

Team 10: Zhengyang Zuo (zzuo) and Nicolas Mellis (nmellis)
Team 11: Omkar Gawde (ohg) and Lawrence Tsang (ltsang)
Team 12: Congshan Lv (congshal) and Prabhanjan Batni (pbatni)
Team 13: Kenny Sung (tsung) and Ruei-Min Lin (rueiminl)
Team 14: Wenhan Lu (wenhanl) and Subramanian Natarajan (snatara1)
Team 15: Lixun Mao (lmao) and Chanjuan Zheng (chanjuaz)
Team 16: Saud Almansour (salmanso) and Mohan Yang (mohany)
Team 17: Xing Wei (xingw1) and Wanchao Liang (wanchaol)

Teams 18-25 demo to Jeremy

Team 18: Keane Lucas (kjlucas) and Bujar Tagani (btagani)
Team 19: Will Snavely (wsnavely) and Cassie Urmano (curmano)
Team 20: Joe Vessella (jvessell) and Hongyi Zhang (hongyiz)
Team 21: Xuan Zhang (xuanz1) and Xinkai Wang (xinkaiw)
Team 22: Jeff Brandon (jdbrando) and Ryan Cutler (rcutler)
Team 23: Anish Jain (anishj) and Yujing Zhang (yujingz1)
Team 24: Hailei Yu (haileiy) and Mayur Sharma (mayurs)
Team 25: Hailun Zhu (hailunz)

Teams 26 - 33 demo to Akshay

Team 26: Yachen Wang (yachenw) and Yibin Yan (yibiny)
Team 27: Xingchi Jin (xingchij) and Siqi Wang (siqiw)
Team 28: Rohan Sehgal (rohanseh) and Sean Klein (smklein)
Team 29: Vinay Bhat (vbhat) and Behrouz Rabiee (brabiee)
Team 30: Eryue Chen (eryuec) and Kaidi Yan (kaidiy)
Team 31: Xiaokai Sun (xiaokais) and Hsueh-Hung Cheng (hsuehhuc)
Team 32: Gaurav Jain (gmjain) and Chun-Ning Chang (chunninc)

18-842 Lab Teams

Team 33: Jonathan Lim (jlim2) and Shan Gao (shang)

Teams 34-41 demo to Vishvesh

Team 34: Shuo Chen (shuoc) and Samantha Allen (slallen)

Team 35: Pranav Bagree (pbagree) and Huacong Cai (hcai)

Team 36: Rohith Jagannathan (rjaganna) and Huiyuan Wang (huiyuanw)

Team 37: Nick Winski (nwinski) and Ara Macasaet (lmacasae)

Team 38: Ke Wang (kewang1) and Haoyu Chen (haoyuche)

Team 39: Pushen Gao (pusheng) and Minghan Chen (minghan2)

Team 40: Dominik Wienand (dwienand) and Mayank Singh Shishodia (mshishod)

Team 41: Jialiang Lin (jialianl) and Yin Lin (yinlin)