# IceCube Global Trigger

**Draft V2R0**

S. H. Seo [1]

Penn State University

December 14, 2005

[1]email: shseo@phys.psu.edu, phone: 814-863-2015

# 1 Introduction

IceCube trigger system is purely software-based (using JAVA programing language): there is no hardware trigger in IceCube Data Acquisition System (DAQ) [2]. This is possible due to relatively low data rate (∼kHz; most of them are background downgoing muon) which can be handled by IceCube online DAQ software trigger system.

IceCube trigger system consists of three components depending on level of triggering. They are, in hierarchial order (low to high), String Processors (SP)/IceTop Data Handlers (IDH), subdetector trigger systems (InIce/IceTop/AMANDA etc...), Global Trigger (GT).

Right after the trigger system there is a Event Builder (EB). The purpose of EB is to request raw data in SPs/IDHs for specific time intervals using final trigger information from GT, and then to send the triggered raw data (with a specific format) to DAQ Dispatch. The DAQ Dispatch will send the triggered data to online data filtering system which will select data which needs to be sent to northern hemisphere via satellite. The maximum buffer time of SP/IDH to hold raw data until EB requests them is about 30 sec.

The trigger system operates on light-weight data (time stamp of each hit and DOM IDs or String ID). Full waveform information is never used for triggering but charge (i.e., energy) information may be used for near future as well for more advanced triggering. The trigger information added in each trigger level will be kept and propagate from the lowest to highest level trigger system. Fig. 1 shows the structure of the IceCube DAQ and trigger system.

As shown in the Fig. 1 GT sits in the highest level of IceCube trigger system. That allows GT perform inter-detector triggering, which is the reason why GT exists because in subdetector level, their own triggering information is not shared.

In following sections, GT requirements, funtionality, structure and relationship with DAQ are discussed more in detail.

# 2 Global Trigger Requirements

To have a fully functional GT certain requirements should be satisfied. Those requirements come in two parts: input/output requirements and GT configuration. In following subsections it will be discussed what they are.

## 2.1 Requirements of Input to GT

The requirements of input triggers from subdetectors to GT come in three aspects:

- Format of input triggers (to GT) must be that of “trigger-request payload”.

[2] AMANDA has both hardware and software triggers.

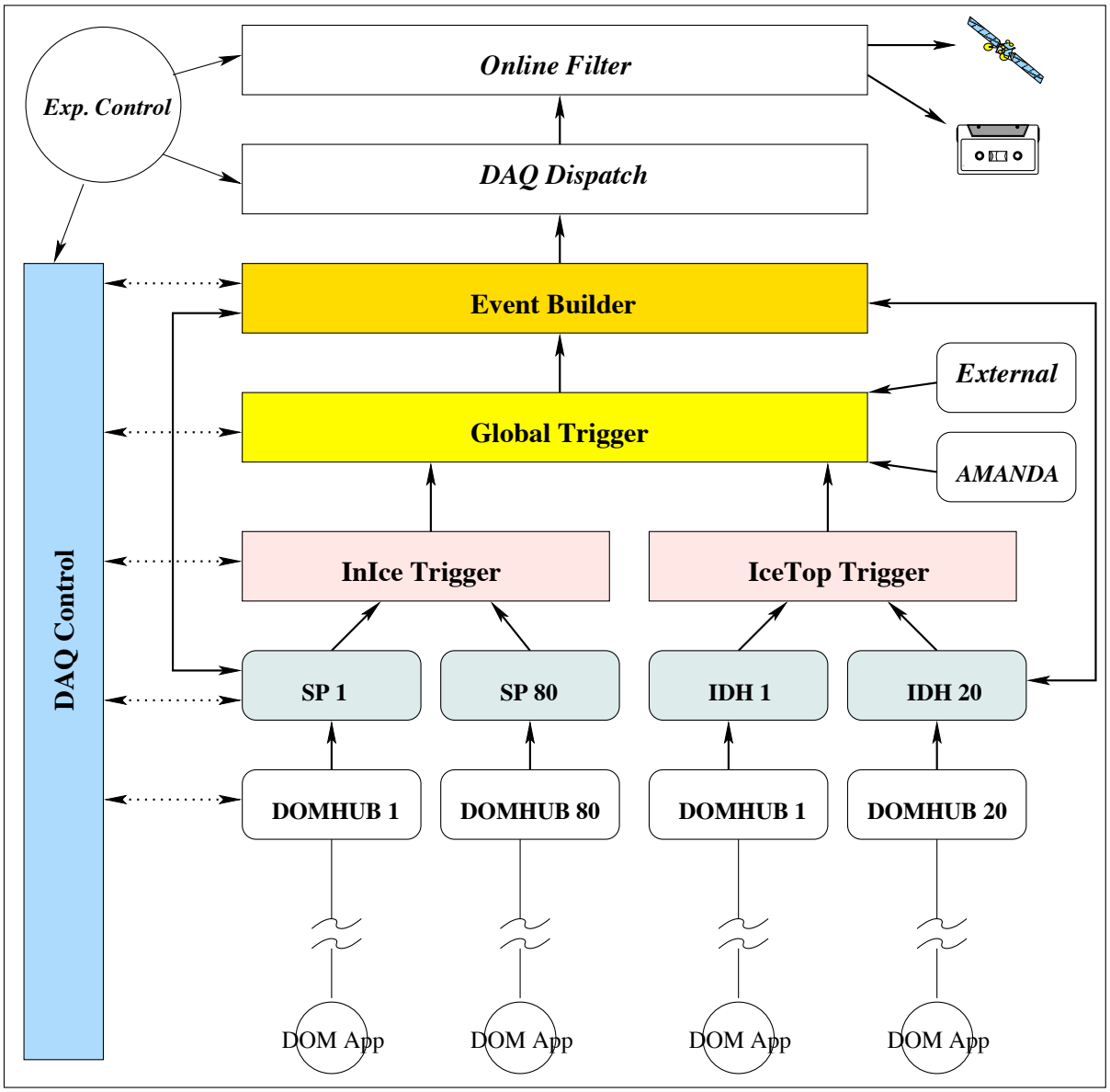


Figure 1: IceCube DAQ and trigger system (colored ones).

- Input Trigger Request Payloads (to GT) should contain meaningful readout elements.
- Input triggers (to GT) must be time-ordered.

Let's take a look at one by one more in detail.

### 2.1.1 Input/output trigger format: TriggerRequestPayload

In IceCube trigger system, each trigger system sends and receives trigger information via (IceCube) JAVA object called "payload"s. There are a few types of payloads depending on the level of trigger information they contain. Among them "trigger-request payload" is what GT expects as input and creates as output. Thus all sub-detectors and external detectors are obliged to send trigger-request payloads to GT. Table 1 lists some payload types relavent to trigger system.

Following shows a list of content in a trigger-request payload and its binary representation (triggerUtil V02-06-03).

```
------------------------------------------------------------------

Trigger Request Payload Header:

  1. Payload length: 4B
  2. Payload   type: 4B (a fixed integer value of 9)
  3. Payload   time: 8B (this payload time should be the same as
                         following 6. First time)

Trigger Request Payload Record:

  1. Record type: 2B (a fixed value of 4)
  2. Unique ID: 4B (e.g., trigger counter)
  3. Trigger type: 4B
  4. Trigger Configuration ID: 4B
  5. Source ID: 4B (a fixed value: InIce = 4000, IceTop =  5000,
                                     GT = 6000, AMANDA = 10000)

  6. First time: 8B
  7. Last time: 8B
  8. ReadoutRequestPayload:

    1) Request type: 2B (a fixed value of 255)
    2) Unique ID: 4B
    3) Source ID: 4B
    4) Number of Readout elements: 4B
```

```
      List of Readout-request Elements (N x 32B, N = 0, 1, 2, ...)
        --Each Readout-request Element Information (total 32B)
          (a) Readout Type: 4B
          (b) First time: 8B
          (c) Last time: 8B
          (d) SourceID: 8B
          (e) DOMID: 4B

  9. Vector of sub-payloads:

    1) Composite length: 4B (a fixed value of 8)
    2) Composite type: 2B (a fixed value of 1)
    3) Number of payloads: 2B
       List of sub-payloads
         --Each sub-payload Information (various size)
           Trigger Request Payload

-----------------------------------------------------------------------
```

Total size of a trigger-request payload varies depending on the number of sub-payloads and readout-request elements. More details of the payload system in general is well described in Pat's document [1].

### 2.1.2 Meaningful Readout Elements

When each subdetector is configured it gets configuration information for each trigger type and trigger configuration ID. A pair of trigger type-trigger configuration ID of a subdetector should match configuration information related to readout time: offset time; time window to past, time window to future. The offset time was introduced to compensate for overall geometrical distance between subdetectors. Table 2 shows an example of readout time configuration for some readout types. In subtrigger system, time interval of each readout request element is obtained by extending its original trigger time using readout time window obtained from configuration (see Table 3). This readout time information in each readout element in a final GT event is very important for GT to check time overlap of subtriggers.

### 2.1.3 Time-ordered Input Triggers

When each sub/external detector sends its output triggers to GT, those triggers are required to be time-ordered. The failure of that can cause severe error in GT splicer.

### 2.1.4 Global Trigger Output

Global Trigger output format should be that of a trigger request payload. Fig. 2

Table 1: Payload types and the level of information contents.

| Payload Type | Trigger Info | Readout Request Info | Raw Data |
|---|---|---|---|
| HitPayload | Yes | No | No |
| TriggerRequestPayload | Yes | Yes | No |
| HitDataPayload | Yes | Yes | Yes |

Table 2: Example of readout time configuration associated with trigger type and trigger config ID (unit: $\mu$sec).

| | | | Readout Time Window (offset; past, future) | |
|---|---|---|---|---|
| Source ID | Trigger Type | Trigger Config ID | InIce-Global | IceTop-Global |
| InIce | SM | 0 | (0; -2, 2) | (-8; -2, 2) |
| InIce | Calib | 0 | (0; -1, 1) | No readout |
| IceTop | SM | 0 | (+8; -2, 2) | (0; -2, 2) |

Table 3: Example of readout elements in a sub-trigger (unit: $\mu$sec): readout time window were obtained using configuration in Table 2.

| | | | | Readout Elements | |
|---|---|---|---|---|---|
| Source ID | Trigger Type | Config ID | Trigger Time | InIce-Global | IceTop-Global |
| InIce | SM | 0 | (20, 23) | (18, 25) | (10, 14) |
| InIce | Calib | 0 | (30, 30) | (29, 31) | No readout |
| IceTop | SM | 0 | (35, 37) | (25, 31) | (33, 39) |

shows an example of contents in a final GT event.

## 2.2 Global Trigger Configuration

To run a global trigger algorithm, the following configuration information should be provided: trigger type and trigger configuration ID for each trigger, maximum readout time window of sub-detector-trigger, and time gap information. Trigger type and trigger configuration ID will be carried in each GT event. Let's take a look at each more in detail.

### 2.2.1 Trigger Type

Trigger type has an one-to-one relation with each GT algorithm. For example, "ThroughputTrigger" and "TwoCoincidenceTrigger" are names of trigger types and at the same time global trigger algorithms. In configuration database, trigger types are represented as integer. However in DAQ software there is a map which connects an integer value of a trigger type to a name (string) of the trigger type.

### 2.2.2 Trigger Configuration ID

Trigger type alone, however, is not sufficient information to run a specific trigger algorithm. For example, to run "TwoCoincidenceTrigger" we need to provide two trigger types with trigger configuration IDs we are interested: i.e., coincidence between InIce SMT (trigger type) with multiplicity 10 (trigger configuration ID) and IceTop SMT (trigger type) with multiplicity 8 (trigger configuration ID), or coincidence between InIce SMT (trigger type) with multiplicity 5 (trigger configuration ID) and AMANDA SMT (trigger type) with multiplicity 10 (trigger configuration ID) etc.... Trigger configuration ID provides a specification to a certain trigger type. In that way we can run several, for example, "TwoCoincidenceTrigger" with different trigger configuration IDs.

### 2.2.3 Maximum Readout Time Window

The maximum readout time window will be used to set a "time-gate" which will determine whether it's safe to release the GT events contained in GlobalTriggerBag or not. The maximum readout time window (the earliest one) is added to the start time of the earliest payload of interest informed by GlobalTrigHandler. This information does not need to be in GT configuration data base (DB): it is, in fact, calculated from configuration of active subdetector triggers. Fig. 3 illustrates how final GT events are decided to be released safely, i.e., to be send to EB.

### 2.2.4 Time-Gap Option

One of the fundemental functionalities of GT, which will be discussed more later, is to manage readout-request elements in a final GT event collected from subdetector-

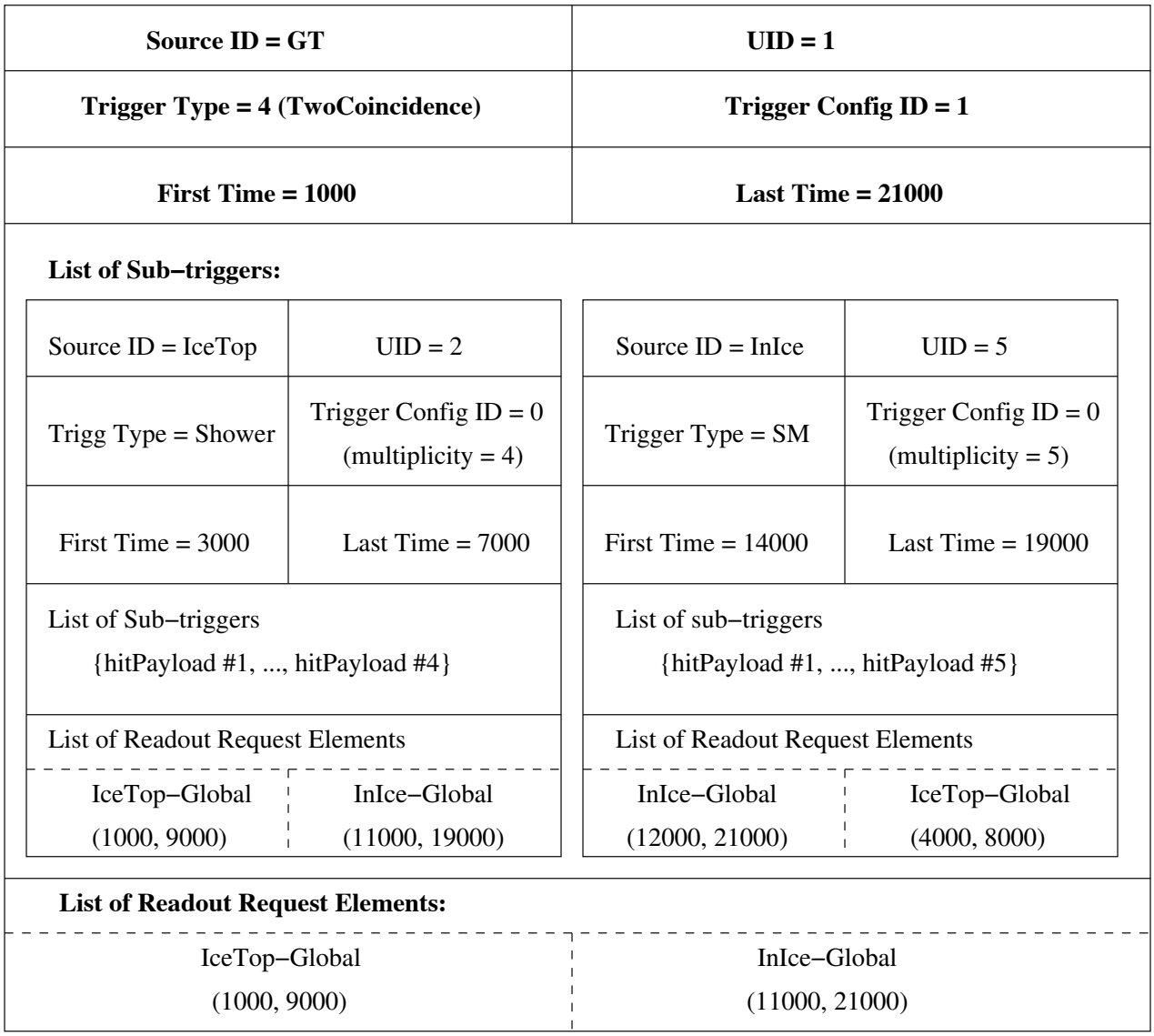


Figure 2: An example of contents in a final GT event.

triggers. Since time-gap option is directly related to managing readout-request elements, I would like to mention readout-request elements. A final GT event contains one or more readout-request elements. Each readout-request element consists of four fields: readout type, readout time, string ID and DOM ID. There are a total of five readout types: Global, InIce Global, IceTop Global, InIce String, InIce Module, and IceTop Module.

Depending on readout types, string ID and/or DOM ID can be null as shown in Table 4.

Time-gap option information is used when there are more than two readout-request elements in a final GT event with the same string ID (or DOM ID) but with different time intervals. If the time-gap option is set to be "no time gap" then only one readout element with the same source ID will exist in a list of readout-request elements in a final GT event. Table 5 shows an example of readout-request elements with (top table) and without (bottom table) time-gap option, respectively, in a final GT event.

For more details about configuration, please read configuration document [2].

# 3 Global Trigger Functionalities

The functionalities of GT can be classified as two aspects:

- First, GT performs triggering, i.e., selects input triggers according to trigger configuration.
- Second, GT merges triggers when they overlap in their readout time and issues final GT events.

It's very natural to think that GT performs triggering. I just want to emphasize here again that GT can perform inter-detector triggering by virtue of the fact that GT sits in the highest level of the IceCube trigger system. To perform triggering, GT needs to know selection conditions when it is configured.

Another fundemental functionalities of GT is to merge triggers if they overlap in time. For merged triggers, GT checks redundant readout elements and removes them if there are. These GT functionalities are explained more in following section.

# 4 Global Trigger Project Structure

To make GT system flexible it was designed to be as much modular as possible. That will result in easy debugging in development side and also brings economic effect in the sense that a particular module (class) can be used in any other classes where only that particlar functionality is needed. All of this is possible because IceCube DAQ adopts one of object-oriented languages, JAVA.

Classes with similar funcitonalities are grouped in the same package. GT project consists of 3 packages: framework, triggers and tools. Framework package is in charge

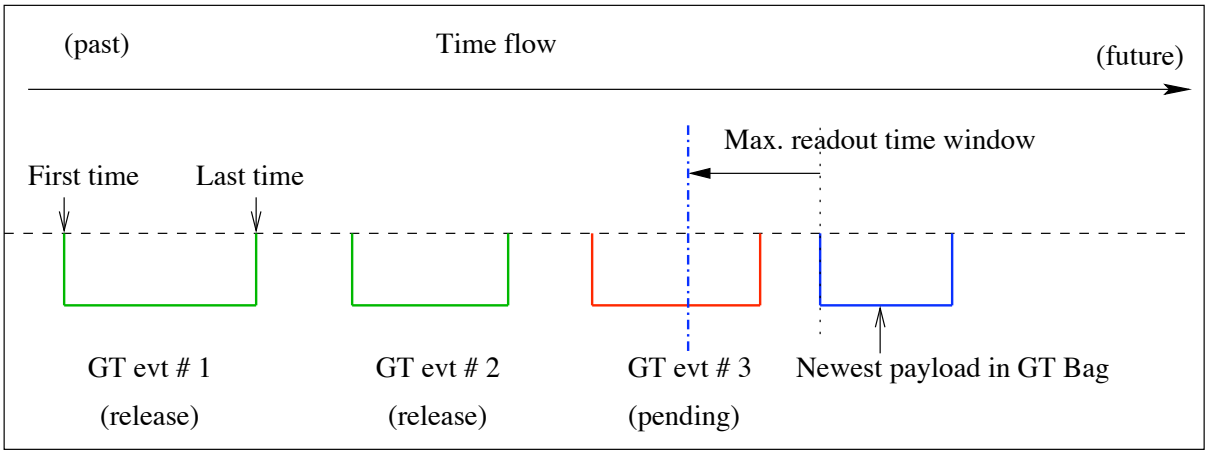


Figure 3: Mechanism of releasing final GT events. In this example GT event # 3 is not releasable because it's still within time-gate.

Table 4: Readout types and their meaning

| Readout Type | Meaning | String ID | DOM ID |
|---|---|---|---|
| Global | Readout all DOMs in InIce and IceTop | Null | Null |
| InIce Global | Reaout all DOMs in InIce | Null | Null |
| IceTop Global | Readout all DOMs in IceTop | Null | Null |
| InIce String | Readout specified string(s) in InIce | Valid IDs | Null |
| InIce Module | Readout specified DOM(s) in InIce | Valid IDs | Valid IDs |
| IceTop Module | Rreadout specified DOM(s) in IceTop | Valid IDs | Valid IDs |

Table 5: An example of readout-request elements in a final GT event with time-gap (Top) and without time-gap (Bottom). $t_1 < t_2 < t_3 < t_4 < t_5 < t_6 < t_7 < t_8 < t_9$

| List of Readout-request Elements (with time-gap) | | | | |
|---|---|---|---|---|
| | # 1 | # 2 | # 3 | # 4 |
| Readout Type | InIce String | IceTop Global | InIce Global | InIce String |
| String ID | 21 | null | null | 21 |
| DOM ID | null | null | null | null |
| Readout Time | ($t_1$, $t_2$) | ($t_2$, $t_3$) | ($t_4$, $t_7$) | ($t_8$, $t_9$) |

| List of Readout-request Elements (without time-gap) | | | |
|---|---|---|---|
| | # 1 | # 2 | # 3 |
| Readout Type | InIce String | IceTop Global | InIce Global |
| String ID | 21 | null | null |
| DOM ID | null | null | null |
| Readout Time | ($t_1$, $t_9$) | ($t_2$, $t_3$) | ($t_4$, $t_7$) |

of managing GT system. Trigger package contains all GT algorithms. Tools package is to provide common GT functionalities. Figure 4 shows skematics of GT structure. Let's take a look at each pacakge more in detail. To help undestanding, framework package will be discussed last.

## 4.1 Triggers

In this package all global trigger algorithms live. Currently (Dec. 2005) three trigger algorithms are implemented in GT level: throughput, two-coincidence and three-coincidence trigger algorithms. Any additional trigger algorithm can be easily implemented in this package by using already developed environment, i.e., trigger framework and tools.

Each trigger algorithm should create a new trigger request payload when its trigger condition is met. The first time and last time of a newly created payload are determined by time of its readout elements: The firstTime (lastTime) is the earliest (latest) time of the readout elements. A new trigger-request-payload contains original input trigger(s) as its subset. See Fig. 2.

### 4.1.1 Throughput trigger

As the name of this algorithm implies it does apply no selection algorithm: total number of input triggers = total number of triggers created by this algorithm. Even though the total input and output numbers are the same, contens of each trigger can be different because first and last time of a newly created tirgger is the earliest and latest readout time which usually are not the same as the first and last time of input trigger. It's needless to say that sourceID, trigger type and trigger configuration ID of a newly created trigger are different from those of input trigger. Fig. 5 shows how throughput trigger algorithm works.

### 4.1.2 Coincidence triggers:

To run a conditional trigger three input trigger information are required inside its algorithm: trigger type, trigger configuration ID and subdetector ID. Those information are contained in incoming payloads. Inside coincidence-trigger algorithm, the three information is represented/interpreted as one concept called "trigger ID".

So far two types of coincidence trigger algorithms are implemented: TwoCoincidenceTrigger and ThreeCoincidenceTrigger. In any coincidence trigger algorithm, it checks time-overlap of incoming payloads with different trigger ID which we want to select as coincidence.

In coincidence-trigger algorithm, we need a certain time indicator which guarantees that a newly created coincidence-trigger won't be associated with following triggers. The time indicator is called time-gate for coincidence trigger. Time-gate is a dynamic variable and is updated in CoincidenceTriggerBag which holds incoming triggers selected by coincidence-trigger algorithm until they are guaranteed not to associated

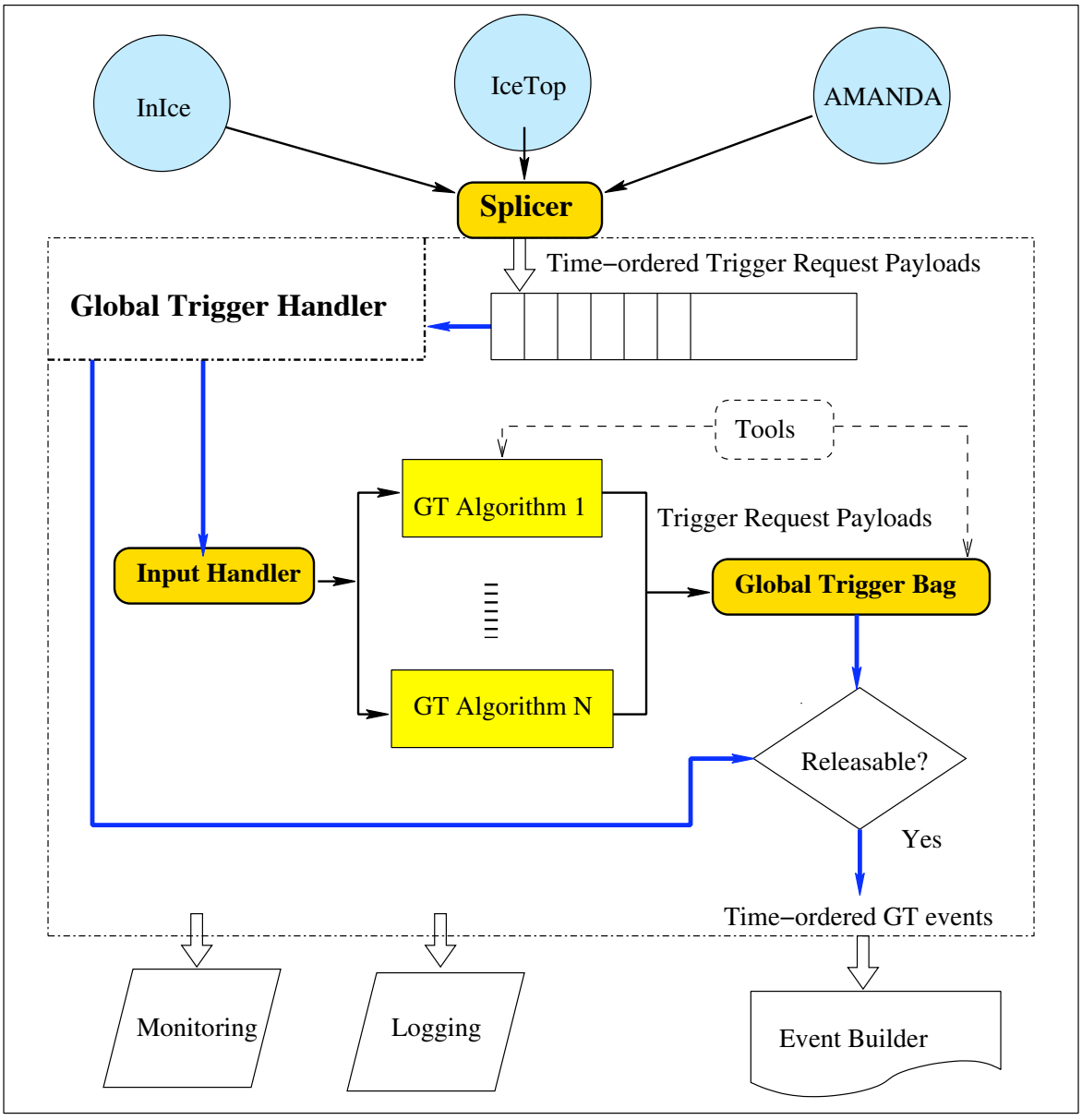


Figure 4: Skematics of Global Trigger structure.

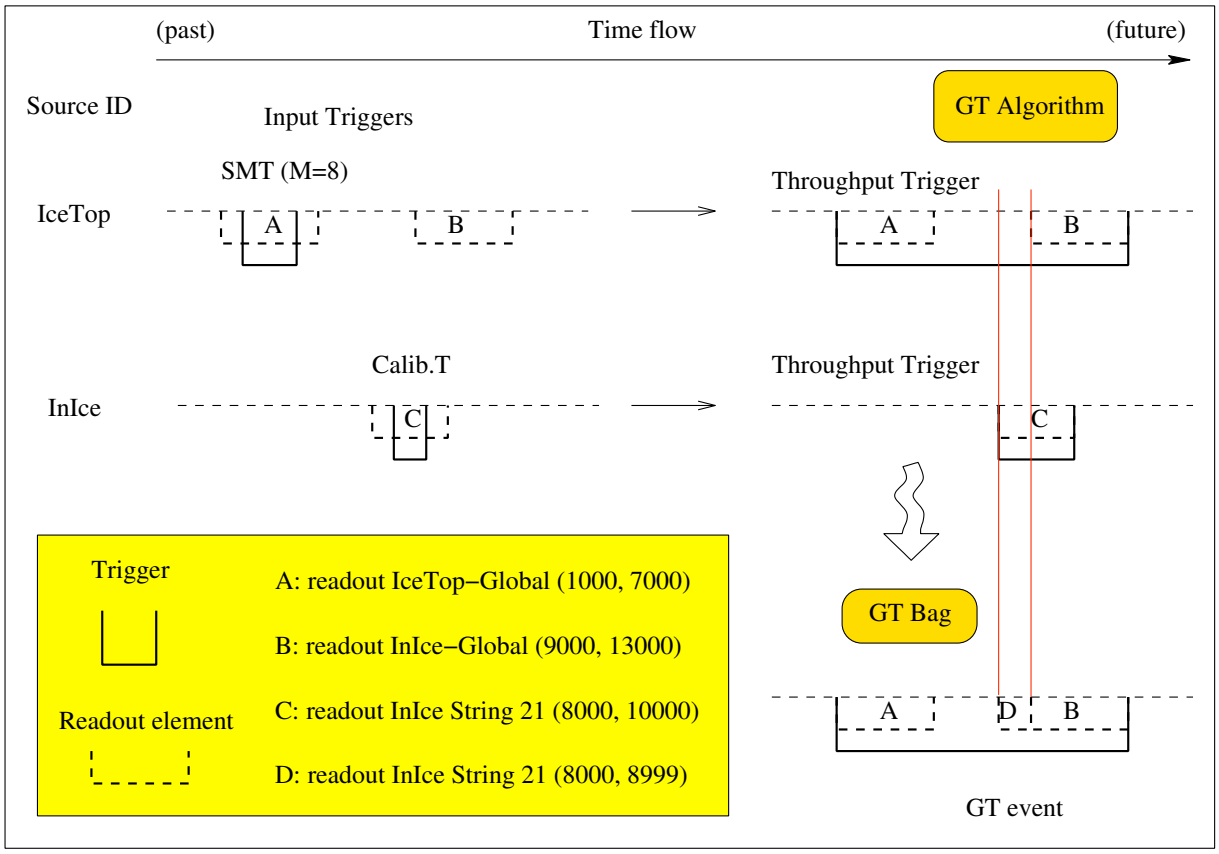


Figure 5: A schematic of "throughput trigger" process.

with following triggers. The guarantee is effective when the last time of a coincidence trigger candidate is smaller than the time-gate at the moment. The time-gate is set as the first time of the configured incoming trigger minus the maximum readout time window to the past.

In two-coincidence-trigger algorithm, if two (or more) configured payloads with different trigger IDs overlap in time, then those payloads are selected to be a two-coincidence trigger candidate until the last time of the candidate is smaller the the time-gate at that moment, which is when the candidate trigger becomes a final two-coincidence trigger event. Fig. 6 shows how this trigger algorithm works. The mechanism of three-coincidence-trigger algorithm is basically the same as that of two-coincidence-trigger algorithm but three different incoming "trigger ID"s are required instead of two to form a new trigger. Both two- and three-coincidence-trigger algorithms inherit CoincidenceTrigger class which contains common functionalities to any kind of coincidence-trigger algorithm. Thus any N-coincidence-trigger algorithm can be easily implemented by inheriting CoincidenceTrigger class.

## 4.2 Tools

All classes related to common GT functionalities live in this package. Those common GT functionalities are to merge triggers if they overlap in time, to handle redundant readout elements and to creat readout-request-payloads. This package is used in each trigger algorithm and trigger framework.

Currenly four classes exist to support those common GT functionalities: GlobalTrigEventWrapper, GlobalTrigReadoutElement, SimpleMerger and SmartMerger.

GlobalTrigEventWrapper object should be used whenever GT creates a new trigger-request-payload. That situation exists in each trigger algorithm and GlobalTrigBag.

When it's called in each trigger algorithm, readout-request elements will not be modified but just collected (when more than two triggers form a new trigger). Readout-request elements can be modified only in the final stage of creating trigger-request-payloads, i.e., in GlobalTrigBag where final GT event is determined and then created.

To decide final readout-request elements GlobalTrigEventWrapper calls GlobalTrigEventReadoutElement where readout elements of which time and/or space overlap are managed by two consecutive mechanisms of merging: SimpleMerger and SmartMerger objects in order of complexity.

In SimpleMerger readout-request elements with same readout type are merged if their readout time overlaps.

After readout elements are managed within the same readout type in SimpleMerger, they are further managed in SmartMerger.

In SmartMerger time-overlap and space-overlap are checked between different readout types in the same subdetector. Once two readout elements with different readout types overlap in time, SmartMerger check space-overlap to manage redundant readout. Table 6, 7 and 8 show how unmanaged elements (in Table 6) are processed in SimpleMerger (Table 7) and SmartMerger (Table 8).

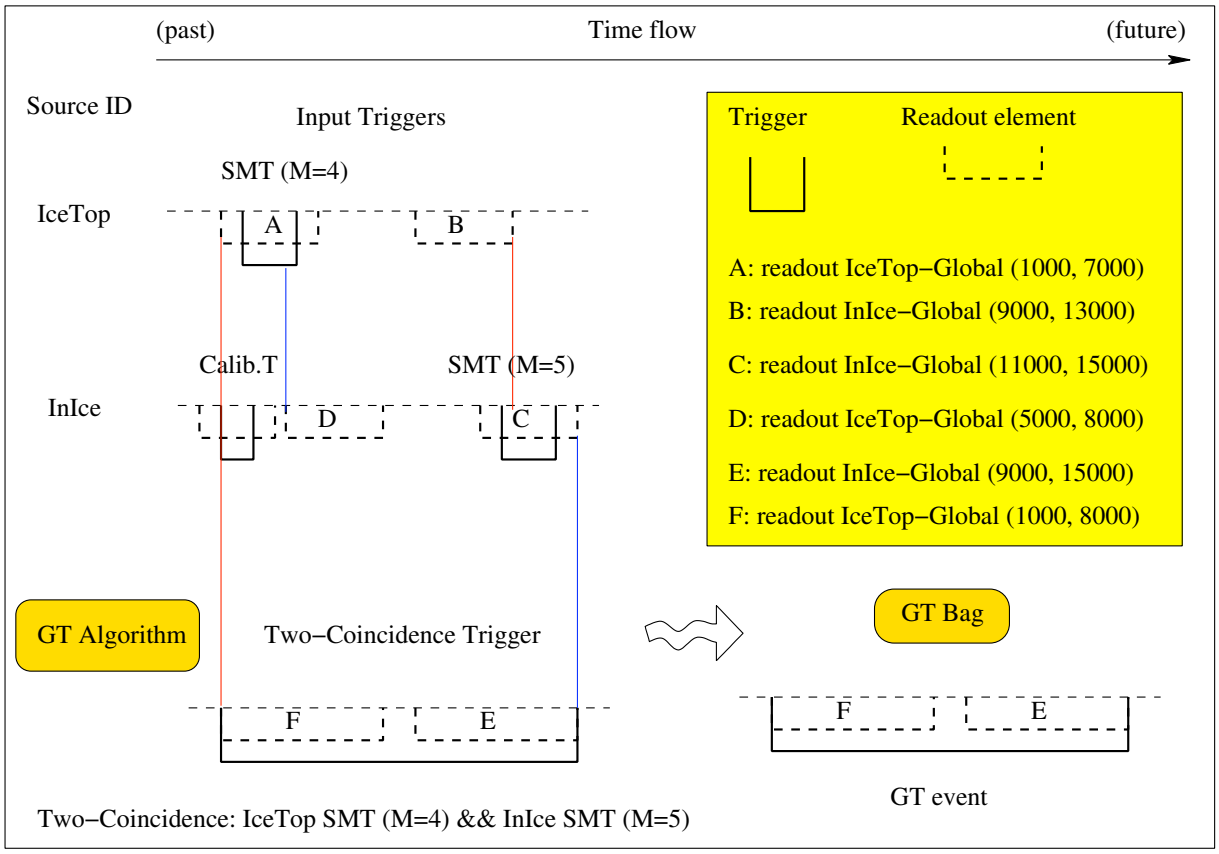


Figure 6: A schematic of "two coincidence trigger" process.

Table 6: An example of collected readout-request elements (before management by GT).

| List of Readout-request elements | | | | | |
|---|---|---|---|---|---|
| | # 1 | # 2 | # 3 | # 4 | # 5 |
| Readout Type | IceTop Global | InIce Global | InIce Global | InIce Module | InIce String |
| String ID | null | null | null | 1 | 21 |
| DOM ID | null | null | null | 33 | null |
| Readout Time (0.1 nsec) | (1000, 3000) | (5000, 8000) | (4000, 9000) | (8000, 9000) | (8000, 11000) |

Table 7: An example of collected readout request elements (after simple-merger in GT).

| List of Readout-request Elements | | | | |
|---|---|---|---|---|
| | # 1 | # 2 | # 3 | # 4 |
| Readout Type | IceTop Global | InIce Global | InIce Module | InIce String |
| String ID | null | null | 1 | 21 |
| DOM ID | null | null | 33 | null |
| Readout Time (0.1 nsec) | (1000, 3000) | (4000, 9000) | (8000, 9000) | (8000, 11000) |

### 4.2.1 Boundary Treatement

When SmartMerger merges readout elements, sometimes there is a situation to deal with boundary of readout time. In other words, how to split readout time when more than two readout elements overlap and lower level readout element is part of higher level readout element: i.e., a DOM readout and a string (which contains the DOM) readout. This time-boundary needs to be properly treated to avoid redundant readout on the bounday.

Current way to deal with time-boundary is that when SmartMerger splits the boundary of readout time, one of the readout time will be incremented in one unit, i.e., tenth of nano-second. An example of this situation (after treatement of time-boundary) is shown between readout element # 2 and # 3 in Table 8. Figure 7 illustrates the example.

## 4.3 Trigger Framework

The purpose of this package is to manage whole GT system.; Managing input, output and trigger algorithms. It consists of three classes: GlobalTrigManager, GlobalTrigHandler and GlobalTrigBag. GlobalTrigManager and GlobalTrigHandler are identical in functionality but the former is used in conjunction with the splicer, i.e., inside DAQ framework. The execute() method in GlobalTrigManager is called when GT splicer is ready to release a batch of spliced (i.e., time-ordered) trigger-request-payloads from subdetectors. Before any GT trigger algorithm is applied, those GT input payloads need to be handled using input-handler object. The purpose of input-handler is to compensate the functionality of splicer by handling readout-request-payloads, i.e., payloads which have both start time and end time: Currently splicer takes care of only start time of payloads. That does not guarantee that current batch of payloads are not associated with following batch of payloads. For example, one of the end time of the payloads can be long enough to overlap with one of the payloads in the following batch: Splicer alone can not handle this situation. The input-handler guarantees that current batch of payloads are not associated with following batch of payloads.

In conjunction with GlobalTrigBag, GlobalTrigManager (or GlobalTrigHandler) handles also all outputs (trigger-request payloads) created by each global trigger algorithm.

When GlobalTrigManager receives spliced objects it starts each trigger algorithm and then each trigger algorithm starts creating new trigger-request-payloads when trigger condition is met. These new payloads generated by all active trigger algorithms are then sent to GlobalTrigBag. where final GT events are determined after merging the collected payloads if they overlap in time and managing readout elements properly. Then GlobalTrigManager (or GlobalTrigHandler) releases to EB final GT events in GlobalTrigBag if they are safe to release, i.e., outside time-gate. Fig. 8 shows schematics of how GlobalTrigBag operates. There are three cases in the figure. In case 1, both throughput and two-coincidence trigger algorithms are active, in case 2, only throughput trigger is active, and in case 3, only two-coincidence trigger is active. Note

Table 8: An example of collected readout-request elements (after SmartMerger in GT).

| List of Readout-request Elements | | | |
|---|---|---|---|
| | # 1 | # 2 | # 3 |
| Readout Type | IceTop Global | InIce Global | InIce String |
| String ID | null | null | 21 |
| DOM ID | null | null | null |
| Readout Time (0.1 nsec) | (1000, 3000) | (4000, 9000) | (9001, 11000) |

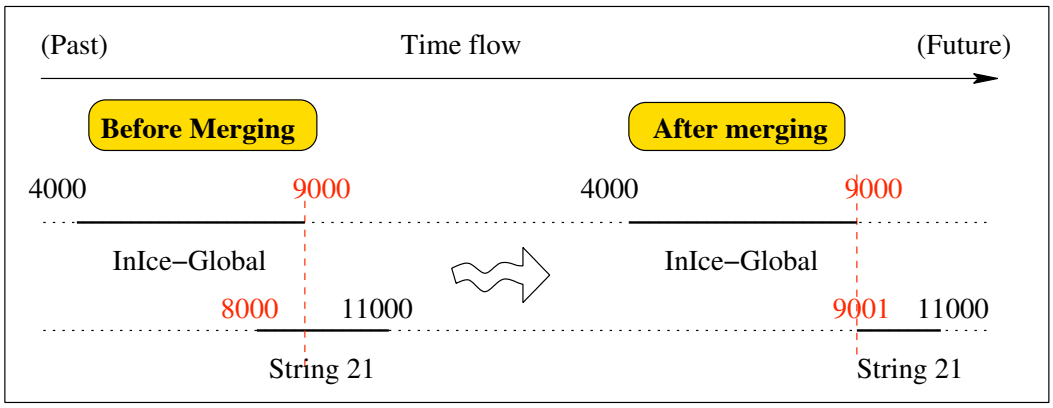


Figure 7: Merging mechanism of readout-request elements and treatment of boundary. The numbers are readout time in tenth of nsec unit.

that the results (i.e., final GT events) of case 1 and 2 are the same.

# 5 Global Trigger and DAQ

IceCube DAQ consists of DOMHUB application (software) and trigger system (software) as shown in Fig. 1. DAQ Control manages all the DAQ components in DAQ framework which are to handle state change and to set global configuration to all components.

In this section some aspects of GT inside the DAQ framework are discussed.

## 5.1 GT State Machine

The behavior (i.e., configure, start, stop, etc...) of GT is managed by GT state machine which can be globally controlled by the state machine of DAQ control. Fig. 9 shows GT state machine which is the same for InIce and IceTop triggers.

## 5.2 GT Configuration

In DAQ framework GT configuration and GT configuration data base are managed in globalTrig-prod and globalTrig-prod-conf projects, respectively. GT configuration consists of two parts: global and local (or internal) configurations. Global configuration is to set up input and ouput channels, payload factory, cache management, splicer, global trigger manager. Local (or internal) configuration is to set up specific values for each active triggers: trigger type, trigger configuration ID, and some configuration values required by a certain trigger, i.e., two-coincidence-trigger information.

## 5.3 GT Dependencies

Currently (Dec. 2005, DAQ-PROD V00-03-02) IceCube DAQ consists of 34 projects and globalTrig project depends on 7 projects among them as shown in Table 9.

# 6 Summary and Prospects

GT is the highest level software trigger system in IceCube. The functionalities of GT are to perform differnt trigger algorithms at the top level, manages merging triggers and creates final GT events which will be used by EB to fetch raw data from SPs and IDHs. Currently (Dec. 2005) three trigger algorithms are ready to be run in S.Pole: ThroughputTrigger, TwoCoincidencTrigger and ThreeCoincidenceTrigger. Those algorithms are fully tested in both functionality and reliability. To run those algorithms GT requires certain trigger configuration information: trigger type, trigger configuration ID, maximum readout time window and time-gap option. GT is ready to process any input triggers from InIce, IceTop and AMANDA triggers.

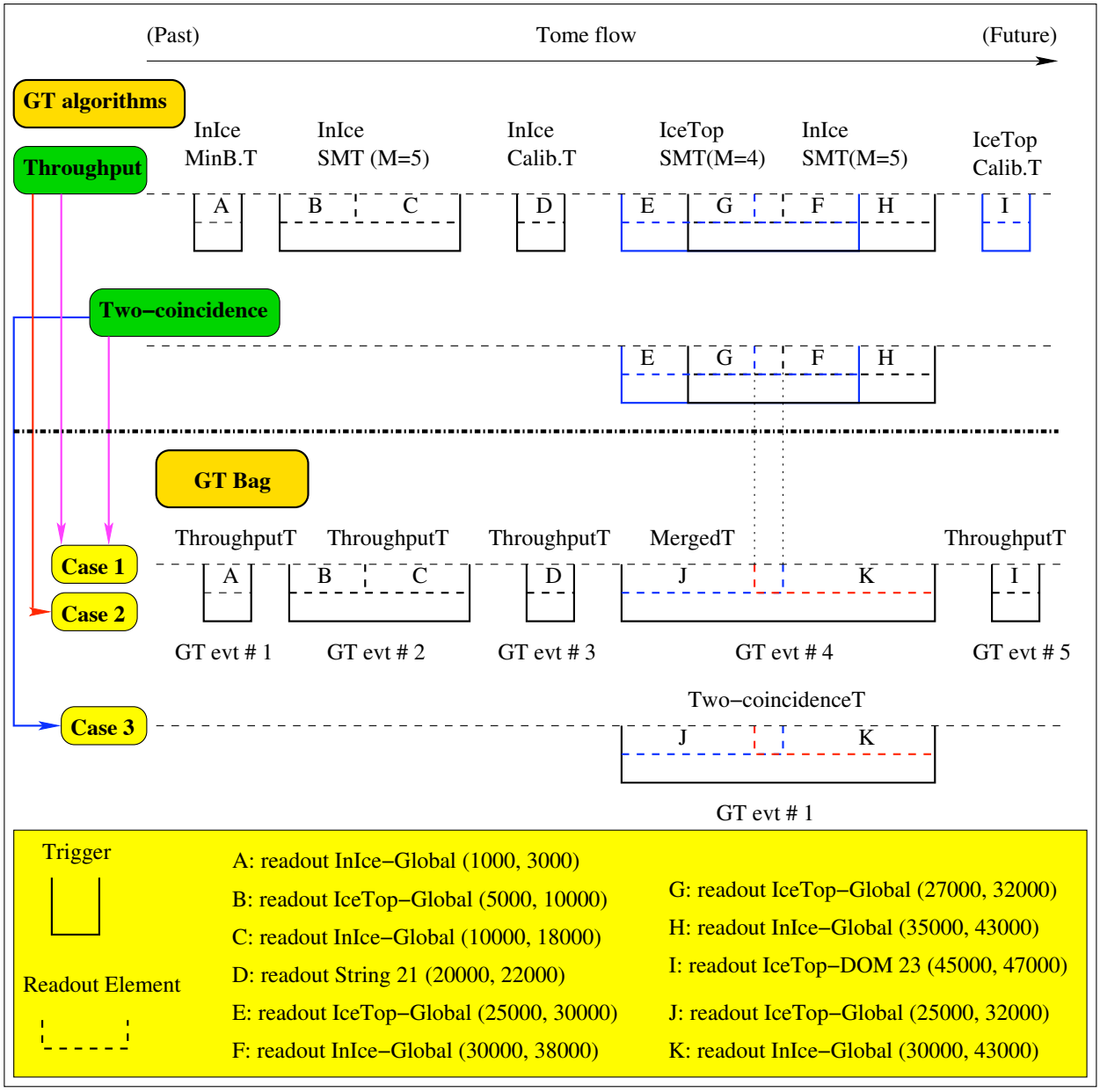


Figure 8: Skematic of "Global Trigger Bag". Case 1: both throughput and two-coincidence triggers are active, Case 2: only throughput trigger is active Case 3: only two-coincidence trigger is active. Note: the results of case 1 and 2 are the same. The numbers are (first time, last time) in nsec unit.

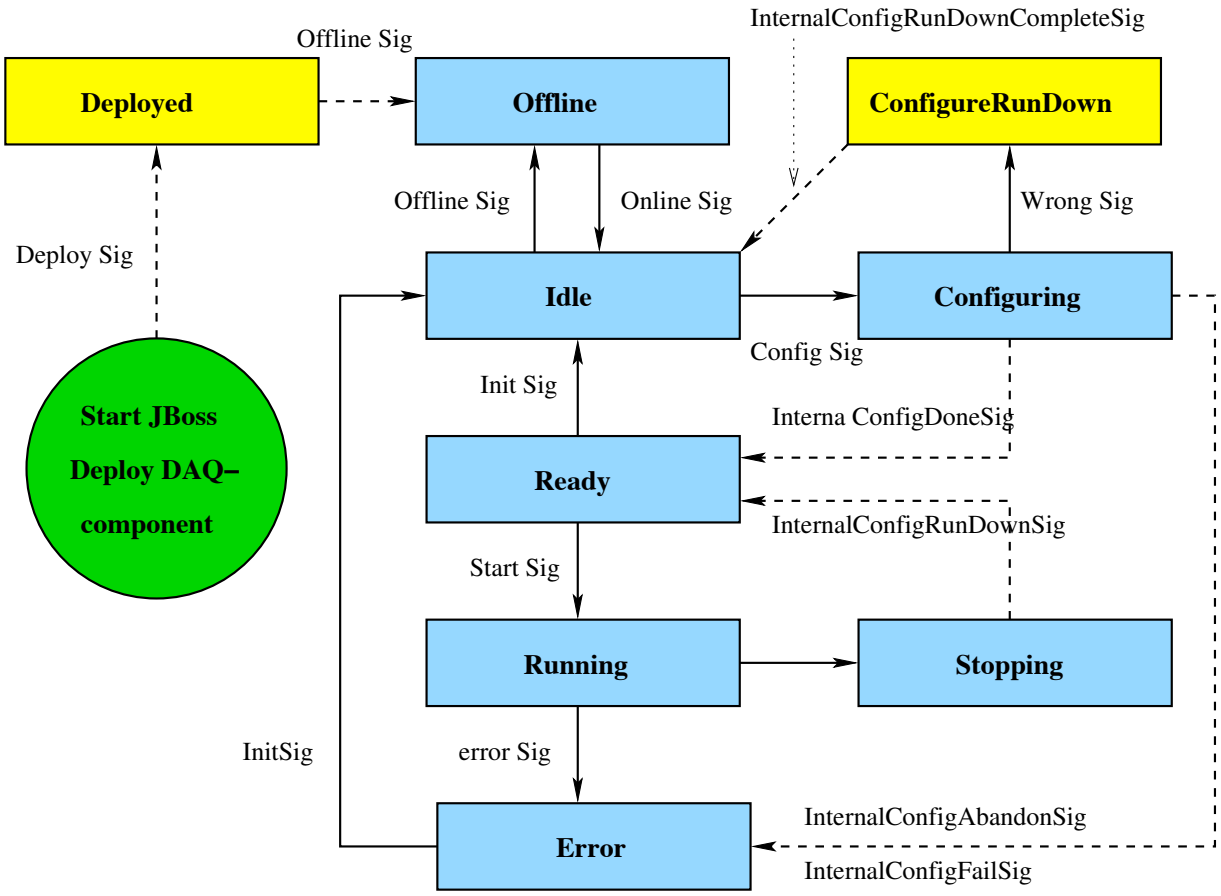


Figure 9: State machine for InIce, IceTop and Global triggers.

Table 9: DAQ Projects and GT dependency (Yes → the projects globalTrig depends on).

| DAQ Project Name | Contents | GT Dependency |
|---|---|---|
| DAQ Utility Projects | | |
| daq-common | | Yes |
| icebucket | | Yes |
| iceboss-services | | – |
| splicer | time-ordering | Yes |
| triggerUtil | | Yes |
| Test Utility Projects | | |
| daq-testframe | trigger test frame outside JBoss | – |
| daq-test-util | trigger test util outside JBoss | Yes |
| payload-generator | | Yes |
| Trigger Projects | | |
| globalTrig | | Self |
| icetopTrig | | – |
| iniceTrig | | – |
| triggerConfig | trigger framework | Yes |
| Control/Configuration Projects | | |
| daq-config-util | | – |
| daqcontrol-prod | | – |
| daqcontrol-prod-conf | | – |
| daq-db-init | | – |
| daq-prod-conf | | – |
| domhub-prod | | – |
| domhub-prod-conf | | – |
| eventBuilder-prod | | – |
| eventBuilder-prod-conf | | |
| globalTrig-prod | | – |
| globalTrig-prod-conf | | – |
| icetopTrig-prod | | – |
| icetopTrig-prod-conf | | – |
| iniceTrig-prod | | – |
| iniceTrig-prod-conf | | – |
| monitorBuiler-prod | | – |
| monitorBuilder-prod-conf | | – |
| stringProc-prod | | – |
| stringProc-prod-conf | | – |
| tcalBuilder-prod | | – |
| tcalBuilder-prod-conf | | – |
| Misc. Projects | | |
| daq-dispatch | | – |

GT system uses trigger framework and that allows easier maintenance of all triggers (InIce, IceTop, Global, AMANDA, and any other external triggers) for a long run. GT was designed to be object-oriented, which allows adding new trigger algorithm relatively easy. It is also easy to add any new external detector trigger as input to GT.

# References

[1] P. Toale, "Payload", 2005 .

[2] P. Toale, "Trigger Configuration", 2005 .