* P4 stands for **P**rogramming **P**rotocol-independent **P**acket **P**rocessors.
* P4 is a language for expressing how packets are processed by the data plane of a programmable forwarding element such as a hardware or software switch, network interface card, router, or network appliance.
* Sole aim of P4 is to program the data plane of a target and if the target has control plane application as well P4 is not concerned with it.
* In a traditional switch the data plane implementation or functionality is defined by the manufacturer and based on that we must develop application which is not a promising way in a dynamic industry, and it incurs huge costs for companies.
* A P4-programmable switch differs from a traditional switch in following 2 ways:

1. **Data plane functionality** is not fixed in advance rather it is defined by a P4 program which does the job of configuring switches which does not have any prior knowledge of network protocols.
2. **Control Plane** communicates with the data plane using the same channel as fixed function chips, but the set of tables and other objects in the data plane are no longer fixed as they are defined by P4 program.

* Compiling a set of P4 programs produces 2 artifacts:

1. A **data plane configuration** that implements the forwarding logic described in the input program.
2. An **API** for managing the state of the data plane objects from the control plane.

* The computational complexity of P4 program is linear in the total size of all headers.

**BENEFITS OF P4:**

* **Flexibility**: P4 makes forwarding policies expressible in contrast to fixed function chips which expose fixed function forwarding engines to their use.
* **Expressiveness:** P4 can express hardware agnostic sophisticated algorithms along with portability among the devices which has P4 compatibility and support
* **Resource mapping and management:** P4 programs describe storage resources abstractly (e.g., IPv4 source address); compilers map such user-defined fields to available hardware resources and manage low-level details such as allocation and scheduling.
* **Software engineering:** It provides benefits such as type checking, information hiding and software reuse.
* **Component Libraries:** The underlying hardware has its own indigenous architecture and compiler wrapped around hardware specific functions, to be used by high level P4 program.
* **Decoupling Hardware and Software Evolution:** Target manufactures can use abstract architecture to further decouple the low-level architectural details from high level processing.
* **Debugging:** Target manufactures can provide software environment where the designs can be tested before being deployed into the field.

**ARCHITECTURE:**

Architecture is an important aspect of P4 language because it acts as a contract between the program and the hardware.

The hardware provider provides compiler and the architecture on which the compiler is built.

From this we can say that P4 programs are not portable across different architectures.

The behavior of a P4 program can be fully described in terms of transformations that map vectors of bits to vectors of bits.

**P4 PROGRAM STRUCTURE:**

1. Headers
2. Parser
3. Check-Sum Verification
4. Ingress Processing
5. Egress Processing
6. Check-Sum update
7. Deparser
8. Switch

**EXPLAINING THE STRUCTURE:**

1. **HEADER:** This part describes the headers that should be present in the packet. It basically instructs the parser what headers are to be parsed and what information it should contain based on that the parser automaton can be constructed. The keyword in P4 to define a header is **header,** and it is actually a structure which is similar to a structure in C language. All such headers are to assembled into a single structure where only the type is to be declared and no need to instantiate it.
2. **Parser:** It parses the incoming header and depending on the parsing conditions it decides whether to drop the packet or to pass it further for ingress processing. It is same as an Automata system where the parser advances to a new state depending on the condition of the data whether it contains appropriate information required to advance to the next step. The parser method takes input the **packet**, along with **header outline declared as headers**, **metadata** and a special struct called **standard\_metadata\_t.** The states are defined using the keyword **state** followed by the name of the state and they are to be arranged in chronological order of their occurrence.
3. **CheckSum Verification:** This part is important as per the fact that there are chances of the packet getting corrupted during transit because of the external noise which it encounters while passing through physical link. So after the packet is parsed and is accepted and before ingress processing a checksum verification is performed to verify that the packet is without any external attenuation.
4. **Ingress Processing:** In this phase all the packet processing that is to be needed like updating TTL value, changing the source and destination address and many more. This phase also contains the match action pipeline containing actions and tables which act as switch case as in C. Hence, it contains a key and based on that the actions are called. There are different kinds of tables which can be looked up in the documentation available on the official P4 language repository.
5. **Egress Processing:** This phase refers to the processing that is to be done before the packet goes out of the switch and into the external network.
6. **CheckSum Computation:** As we all know that once the packet passes through ingress and egress processing pipeline, there will be change in data in the packet and it is manifested that the checksum will change and hence a new checksum is to be computed otherwise the other switches and routers in the network would understand that the packet is corrupted and will drop it as they are unaware of the fact that the packet may not be corrupted and there must be some change in data during ingress or egress processing at a previous switch or router.
7. **Deparser:** This phase actually builds the packet and then pushes it out into the external network.