

```
pkt.new_hop = (pkt.sport * pkt.dport) % 10;  
if (pkt.arrival_time - last_time > 5) {  
    next_hop = pkt.new_hop;  
}  
last_time = pkt.arrival_time;  
pkt.next_hop = next_hop;
```

If conversion to straight-line code

```
pkt.new_hop = (1 ? ((pkt.sport * pkt.dport) % 10)  
                : pkt.new_hop);  
pkt.tmp0 = (1 ? (pkt.arrival_time - last_time > 5)  
            : pkt.tmp0);  
next_hop = ((1 && pkt.tmp0) ? (pkt.new_hop)  
           : pkt.next_hop);  
last_time = (1 ? (pkt.arrival_time) : last_time);  
pkt.next_hop = (1 ? (next_hop) : pkt.next_hop);
```

Predicate minimization

```
pkt.new_hop = ((pkt.sport * pkt.dport) % 10);  
pkt.tmp0 = (pkt.arrival_time - last_time > 5);  
next_hop = pkt.tmp0 ? (pkt.new_hop) : next_hop;  
last_time = (pkt.arrival_time);  
pkt.next_hop = next_hop;
```

Instruction Selection

```
pkt.tmp1 = pkt.sport * pkt.dport;  
pkt.new_hop = (pkt.tmp1 % 10);  
pkt.tmp2 = pkt.arrival_time - last_time;  
pkt.tmp0 = (pkt.tmp2 > 5);  
next_hop = pkt.tmp0 ? (pkt.new_hop) : next_hop;  
last_time = (pkt.arrival_time);  
pkt.next_hop = next_hop;
```

Instruction coalescing

```
pkt.tmp1 = pkt.sport * pkt.dport;  
pkt.new_hop = (pkt.tmp1 % 10);  
pkt.tmp2 = pkt.arrival_time - last_time;  
last_time = (pkt.arrival_time);  
pkt.tmp0 = (pkt.tmp2 > 5);  
next_hop = pkt.tmp0 ? (pkt.new_hop) : next_hop;  
pkt.next_hop = next_hop;
```

Partition based on dependencies

```
pkt.tmp1 = pkt.sport * pkt.dport
```

```
pkt.new_hop = (pkt.tmp1 % 10)
```